From Circle Placements to Rectangle Placements

—

Nonlinear Optimization in Electronic Design Automation

Vom Fachbereich Mathematik
der Technischen Universität Kaiserslautern
zur Verleihung des akademischen Grades
Doktor der Naturwissenschaften
(Doctor rerum naturalium, Dr. rer. nat.)
genehmigte

Dissertation
von
Uwe Nowak

1. Gutachter: Prof. Dr. Karl-Heinz Küfer
2. Gutachter: Prof. Dr. Christiane Tammer

Disputation: 28. Oktober 2011

D 386

Acknowledgments

This work was done with the financial support of the Department of Optimization at the Fraunhofer Institute for Industrial Mathematics (ITWM).

I am deeply grateful to my supervisor Prof. Dr. Karl-Heinz Küfer for his guidance throughout my studies and for giving me the opportunity to work on a challenging and very interesting topic.

Also, I would like to thank all the members of the Optimization department for an awesome working atmosphere with many fruitful discussions, for a lot of fun at many social events and their mental support.

For proof-reading and many helpful comments, I specially thank Agnes Dittel, Bastian Bludau, Christina Erben, Hendrik Ewe, Ingmar Schüle, Jan Schwientek, Martin Berger, Martin Pieper, Michael Bortz, Richard Welke, Tabea Grebe and Volker Maag.

Finally, I thank my wife Miriam for her patience and love.

Kaiserslautern, June 2011

Contents

1 Introduction

1.1 Electronic Design Automation (EDA)

In the modern world, electronic devices surround us in our everyday life. Most recent advances are the numerous mobile communication devices providing more comfort and flexibility. During the last decades, both the number and the complexity of these devices grew rapidly.

Till the 1970s, electronic devices were manually designed. Due to the increasing complexity of the electronic circuits, the need for automation of the design process raised significantly. Since the mid 1970s computers and algorithms are powerful enough to support the engineers in the development of electronic systems. This computer aided engineering of electronic circuits is called *electronic design automation* (EDA).

A driving force for EDA was the design of *integrated circuits* (IC). In ICs, semiconductor based electronic components, mainly transistors and diodes, are integrated into a single chip. Due to the increasing miniaturization in this area the amount of components per chip exponentially increased, up to more than a billion transistors per chip today (Intel Core i7). Due to the rapid increase in the number of circuit components, computer aided support is required. This area of EDA is called *very large scale integration* (VLSI).

Another type of electronic circuits are *printed circuit boards* (PCB). In this technology, heterogeneous components such as resistors, capacities or chips are placed on a substrate. The components are connected by copper included in the substrate. Compared to VLSI, the number of components in a PCB is much smaller, usually only up to a few thousand. However, on PCBs the involved components are much more heterogeneous. Hence, there is less automation in the design of PCBs. Usually the placement is done manually, while there are elaborated *auto routers* to layout the wiring between the components.

A relatively new technique to integrate an electronic system is *2.5D System-in-Package* (SiP). In SiPs, PCB substrates are vertically stacked over each other to gain additional miniaturization. The lack of EDA tools for SiPs gave raise to this thesis. System-in-Packages are described in detail in Section 1.2.

The process of electronic circuit design passes through several steps. We summarize the main steps here, for details see [Sait; Youssef 1999].

- In the first step, called *logical design*, the schematic of the electronic circuit is created. The engineer defines the electronic components of the circuit and how their pins are connected. Although EDA can support this step by simulating the logical circuit, this step has to be done manually by an engineer.
- The next step is the *physical design*. This step includes the choice of the concrete packages for each component and the layout of the circuit. It is complex and very time consuming but, in contrast to the architectural design, can be better automatized. Therefore, most effort in EDA was put on this step. After the assignment of the packages, the layout of a circuit is defined by the coordinates of the components and the routing of the nets. This problem is almost computational intractable and therefore separated into subproblems that are itself NP-hard (see [Lengauer 1990]).

  In the *placement*, the components are assigned to geometric coordinates. In this process, no exact routing of the nets is performed. The aim of this step is to make the circuit small and to create some routing-aware placement. While in manual placement the engineer tries to keep connected components close together, automatic placer usually use abstract wire length models that can be easily computed (see Section 1.3). The amount of software support for the placement varies for different technologies.

  In the *routing*, the positions of the components and thus of the pins are fixed and the detailed routing of the nets is performed. This area of EDA is well analyzed and sophisticated commercial auto routers exist.
- The last *validation step* is to verify a placed circuit regarding its thermical and signal properties. This can be done by measuring a physical prototype. However, EDA tools also support this step by detailed physical simulations.

1.2 System-in-Package

2.5D System-in-Package (SiP) is a relatively new integration concept for miniaturization of electronic circuits. In this concept, discrete components are placed onto substrates that are vertically stacked into a single package. While keeping the flexibility and techniques of standard PCB layouts, the use of the third dimension in SiPs offers a higher level of miniaturization. SiPs also require less advanced production techniques than ICs and are thus more cost-effective. Furthermore, SiPs can be used as a single component within standard PCB placements.

SiPs have recently been studied in the literature. Engineering aspects are analyzed in [Polityko 2008], whereas modeling aspects can be found in [Richter et al. 2007]. For recent advances in placement automation for SiPs, see [Berger 2010].

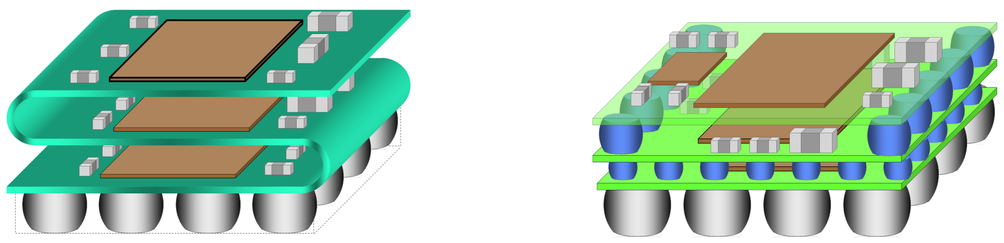

(a) Folded flex. (b) Solder balls.

Figure 1.1: Different technologies for stacking modules vertically.

The most common techniques for vertically stacking substrate modules are folded flex and solder ball connections. Folded flex means to place the components on a flexible substrate and this substrate is folded in s-shape, see Figure 1.1a. Solder balls mean to place the components on standard PCB boards that are vertically connected with solder balls, see Figure 1.1b.

SiPs are related to the previous technologies as PCB, but nevertheless considerably different. In contrast to the sophisticated support tools for PCB and VLSI design, there is a lack of tools for SiP-layout, see [Polityko 2008]. The concept of such a support tool has been shown in [Richter et al. 2007]. Different requirements for the layout of System-in-Packages have to be regarded.

- In SiPs, there are vertical interconnections (VIC) between the vertically stacked modules. Typically these interconnections are solder balls or a copper wire within a folded substrate, but also other types exist. In particular, solder balls have to be treated in a special way, as they are discrete components that do not exist in logical design, but appear during the physical placement step and change their lateral size depending on their height.
- SiPs use the third dimension. Thus, in the physical design step, the components have to be assigned to the different module sides before the two-dimensional placement on each substrate can be done. In the additional *partitioning step*, the algorithm or the engineer aims to optimize contradicting objectives. The lateral size and the height of the SiP should be small. However, as there is no detailed placement in this step, the lateral size can only be estimated. The number of VICs is also to be minimized. This is due to the fact that vertical interconnections require space, but also that electrical signals should not be routed through VICs.
- SiPs usually contain only up to few hundred components and thus significantly less components than VLSI designs and still less components than typical PCB layouts. However, in contrast to VLSI, the components in SiPs are heterogeneous and often of very different size. In contrast to typical PCB layouts, there is a stronger focus on miniaturization. Therefore, beside the location also the rotation of the components is crucial in the SiP placement.

In this thesis we focus on the placement step after the partitioning was performed. This step is the *rectangle placement problem* and described in Section 1.3.

1.3 Rectangle Placement Problem in EDA

The *rectangle placement problem* is the placing of the components on one module side. The components are a set of fixed sized, rectangular objects. Each component has a set of pins with fixed positions relative to a reference point of the component. The pins are connected by nets. The goal is to find a placement of the components on the module that has a short total weighted length of all nets and a small bounding box of all components. Of course the components are not allowed to overlap. The components are also allowed to be rotated. But in practice, usually not all rotations are possible, instead they often can only be rotated by a multiple of 45° or even a multiple of 90°.

In the final placement, a complex routing of the nets is necessary. For routing, very sophisticated commercial tools exist. However, the complete routing process is too time consuming to be done within the placement algorithm. Instead, surrogate net models are used to estimate the net length of a placement. The detailed routing is finally done in a post processing step.

The most common models are the half perimeter wire length [Nam; Cong 2007], the clique model and the star model [Viswanathan; Chu 2004].

Half Perimeter The half perimeter wire length is defined as the half perimeter of the rectangular container containing all pins of the net. The main advantage of this model is that it can be modeled as a linear program.

Clique Model In the clique model, a net is represented by the pairwise connections of its pins. The net length is the sum of all weighted pairwise connection lengths in squared Euclidean distance. This model can be formulated as a quadratic positive semidefinite function of the pin positions.

Star Model In the star model, a net with more than three pins is represented by connecting all pins to an additional introduced connection point, denoted as the net center. The net length is the sum of the squared Euclidean distances of the pin positions to the net center.

In [Viswanathan; Chu 2004] it is shown that the clique model and the star model are essentially equivalent (see Lemma 3.11). We mainly use the clique model in this thesis. By stronger penalizing of larger nets this model balances the length of different nets. Furthermore, this model yields a smooth, convex objective function in non-linear programs.

1.3.1 Approaches to Auto Placers

The placement problem is of a high practical relevance and thus it is extensively studied in the literature. Different approaches have been proposed and have been used in academic and commercial software tools. A recent survey is in [Nam; Cong 2007].

Algorithms for the placement problem are called auto placers. There are two main approaches, discrete and continuous placers.

Discrete placers work on a discrete representation of the placement. The advantage of discrete techniques is that the non-overlapping of the components can be inherently encoded in the placement.

Common discrete techniques are, e.g., metaheuristics as local search, simulated annealing or evolutionary algorithms. Starting from randomly generated solutions, they iteratively try to improve the objective by changing the solutions. These placers work well for small instances and can even tackle larger instances with several hundreds of components. However, iterative solvers have a local view on the problem. They tend to make big decisions as the placement of large components almost randomly at the beginning and, usually, are not able to revise them during the algorithm. While achieving a good packing density, such solvers generate solutions with poor wire length.

Other discrete techniques are constraint programming or mixed integer programming. They lead to high quality placements and, for small instances, can even proof their optimality. However, due to their memory consumption and their running time, these algorithms are impractical for instances with larger number of components.

Continuous or analytical placer usually formulate the placement problem as a constrained non-linear program. Non-linear solvers can handle significantly larger instances than discrete solvers. However, by enforcing the non-overlapping of components, the problem becomes highly non-convex and has many local optima. Furthermore, allowing components to be rotated significantly complicates the non-linear program. Hence, analytical placers are usually applied for VLSI design, where the rotation is not considered and non-overlapping is replaced by the concept of density and later heuristically created.

For very large instances, the placement problem usually cannot be tackled directly. Instead, hierarchical algorithms are applied. They separate the problem into subproblems, solve them independently and unify the results.

1.3.2 Lack of Algorithms for Large SiP-Instances

SiP-Instances lead to placement problems with up to a few hundred components. Consequently after the partitioning step usually no more than 250 components have to be placed on one module side. On the one hand, these instances are too large for constraint programming and mixed integer programming solvers. On the other hand, metaheuristics generate solutions of poor wire length.

Analytical algorithms handle global objectives as the wire length more naturally. However, as the components are too heterogeneous, their non-overlapping cannot be enforced heuristically in a post-processing step as for VLSI placement. Modeling the non-overlapping constraints within the non-linear model yields highly non-convex problems. Hence, standard local non-linear solvers most likely converge to a poor local optimum.

There exist global optimization techniques as branch and bound or interval arithmetic that find the provable optimal solution to non-convex optimization problems. However, these approaches have unacceptable running times for instances appearing in practice. Another approach are multiple shooting algorithms. They do not guarantee global optimality, but try to overcome poor local optima by doing several local optimization runs from different starting points. However, the selection of good starting points is crucial for their success.

To solve practical problems, the structure of the placement problem has to be exploited. In this thesis we present an algorithm that uses a combination of smoothing techniques, global optimization techniques and further methods to achieve good placements with respect to the wire length in reasonable time for typical electronic circuits of System-in-Packages. A survey of the *rounded rectangle algorithm* is given in Section 1.4.

1.4 Rounded Rectangle Algorithm

In this chapter we give a brief survey of the rounded rectangle algorithm and its main principles. Technical details are presented in later chapters of this thesis.

1.4.1 Abstraction as Circles

Our rounded rectangle algorithm represents an analytical placer for the rectangle placement problem. Like all successful solvers, it has to exploit the special structure of the problem.

For System-in-Packages there are up to a few hundred components of very different size. Hence, the rotation and the non-overlapping is important and we have to deal with these aspects explicitly. Unfortunately, the rotation of rectangles leads to many local optima, see Figure 1.2. However, the rotation and non-overlapping of the components can be decoupled by modeling the rectangles as circles. If we then enforce the non-overlapping of the circles, the rectangles can freely rotate inside these circles.

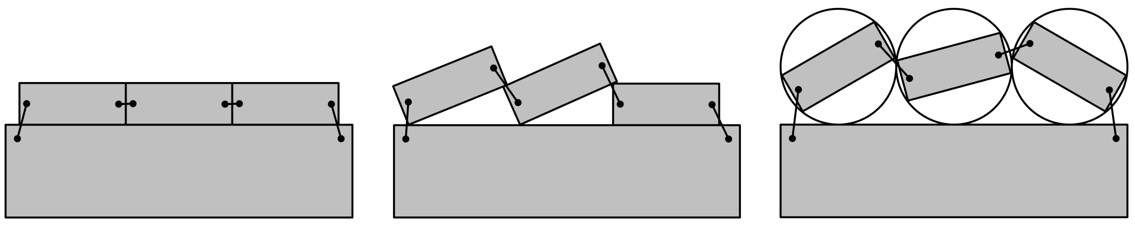

(a) Global optimum. (b) Local optimum. (c) No local optimum if the rectangles are enclosed by circles.

Figure 1.2: Enclosing rectangles by circles decouples the rotation from the non-overlapping. This reduces the number of local optima.

By this decoupling, the number of local optima is reduced, as displayed in Figure 1.2. Circles are less likely to interlock than rectangles. In Figure 1.3 for the rectangles the algorithm gets stuck in a local optimum. However, if the connection is strong enough, for circles this is no local optimum.

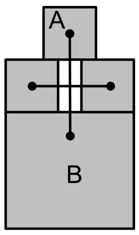
The algorithm starts with an abstract model, where we enclose all components by circles. In the initial phase, we try to find a good placement of these circles without overlap. This phase is very important, as it has a big impact on the final placement. Related problems were already analyzed in the literature. While there are few theoretical results for these problems, several heuristics were developed, e.g. [Anjos 2001] and [Anjos; Vannelli 2006]. We give a survey of these techniques in Chapter 7 and apply them to the rectangle placement problem for System-in-Packages.

1.4.2 Transformation from Circles to Rectangles

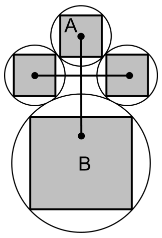
After the initial phase, we have an overlapping free placement of the enclosing circles of the components. However, at the end of the algorithm, a placement of rectangular components is desired. Hence, we have to transform the circles into rectangles.

Therefore, we introduce the concept of rounded rectangles. Rounded rectangles are rectangles whose corners are replaced by circle quarters. Circles and rectangles can be modeled as rounded rectangles. A circle is a rounded square with maximal corner radius, while a standard rectangle is a rectangle with corner radius zero. Figure 1.4 shows the transformation of a circle to a rectangle via rounded rectangles.

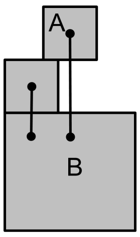
An important property of electronic circuits is that the rotation of the components is only allowed to be a multiple of 45° or 90°. In System-in-Packages, usually only multiples of 90° are allowed. During the transformation from circles to rectangles, we also increase a penalty for the deviation of the component rotation from an orthogonal rotation. At the end of the transformation, a component is modeled as a rounded rectangle with edges parallel to the coordinate axes.



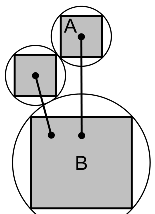
(a) Rectangles can interlock. Independent of the strength of the connection between A and B, the algorithm gets stuck.



(b) Circles do not interlock. If the connection between A and B is strong enough, they can overcome the interlock.



(c) Rectangles can interlock. It is easy to visualize that the connection length could be improved by moving A to the right. However, with fixed rotations the algorithm gets stuck.



(d) Circles do not interlock. The connection between A and B pushes the other circle to the side to come closer to B.

Figure 1.3: The circle placement problem has less local optima than the rectangle placement problem.

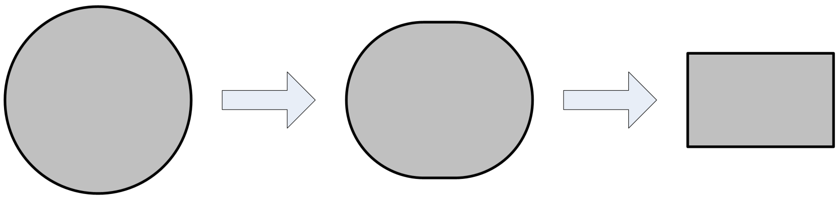


Figure 1.4: Transformation from circles to rectangles via rounded rectangles.

In System-in-Packages, there are few large components and many small components of approximately equal size. The relative placement of these large components is important for the placement of the small components, while a rough placement of the smaller components is sufficient for a good placement of the larger ones. In particular, while refining larger components from circles to rectangles, the rotation and relative placements of the smaller components can be necessary to chance. Hence, it is advantageous to keep the smaller components as circles while transforming the larger ones from circles to rectangles. After this transformation, the rotations of the large components are kept fixed and the smaller components are transformed from circles to rectangles.

1.5 Thesis Outline

In this section we give a brief overview of this thesis. In the different chapters we analyze different aspects of the rounded rectangle algorithm surveyed in Section 1.4.

In Chapter 2 some basic mathematical concepts used throughout this thesis are introduced. Furthermore, we introduce notation and describe the problem instances and the computational environment that we use for our numerical evaluations.

In Chapter 3 we address the circle rotation problem. This problem is to rotate circles that have fixed center positions in order to minimize their wire length. We show that this problem is essentially equivalent to the Hermitian minimization problem already studied in literature. We proof that it is NP-hard. We further develop different models and a custom solution algorithm for the circle rotation problem. In a detailed numerical analysis, we show that the custom solution algorithm generates high quality solutions computationally efficient in order to use it as an essential subroutine within the rounded rectangle algorithm. Furthermore, by convexification, we develop an algorithm to generate solutions with an absolute a-priori and a-posteriori approximation quality.

In Chapter 4 we develop a branch and bound algorithm for the circle rotation problem. Applying computational geometry algorithms, we develop a novel domain reduction algorithm for the circle rotation problem. Based on results of Chapter 3, we derive problem adapted fast lower and upper bound evaluations. With these ingredients we show in an extensive numerical evaluation that the algorithm is able to solve even large practical problem instances up to global optimality in short running time.

In Chapter 5 we consider the circle rotation problem for one net. We show that this problem is equivalent to placing a point with minimal squared Euclidean distance to a set of circles in the plane. We state a convex approximation of the problem and analyze the relations of their optimal solutions. Based on this convexification, we identify instances for which the problem can be solved to optimality. Finally, we state a global branch and bound algorithm for this problem.

In Chapter 6 we consider the problem to place identical circles of one net. We do an asymptotic analysis for increasing number of circles and show asymptotically that wire length optimal packings are also area optimal. However, area optimal packings are not wire length optimal. This analysis supports to focus on the wire length optimization in this thesis.

In Chapter 7 we analyze the circle placement problem. This problem is to place connected circles without overlapping in the plane such that the wire length is minimized. This problem is solved as an initial step in the rounded rectangle algorithm. In a first step, we apply the attractor repeller model known for facility layout. We create a novel scaling invariant version of this attractor repeller model and show in a numerical evaluation that it is significantly superior to previously known models. Furthermore, based on results of the circle rotation problem, we apply local search and monotonic basin hopping methods to overcome local optima.

In Chapter 8 the rounded rectangle algorithm is stated. Based on the results of Chapter 7, we generate a good initial solution for circular component approximations. The circles of this initial solution are successively refined to rectangles by solving a sequence of non-linear programs. We show that the non-linear formulation satisfies the important Mangasarian-Fromovitz constraint qualification. Furthermore, we evaluate the algorithm numerically and show that it generates high quality solutions for practical problem instances in short time.

2 Notation and Conventions

In this chapter we introduce the basic mathematical concepts used in this thesis. Furthermore, we describe the problem instances and the computational environment that we use for the numerical evaluations.

In Section 2.1 we summarize the main properties of complex numbers. In this thesis we state most functions as real valued functions on a complex vector space. To consider real derivatives of such functions, the Wirtinger calculus is introduced in Section 2.2. In Section 2.3 we state properties of univariate and multivariate complex quadratic functions.

In Section 2.4 we survey the theory of non-linear optimization applied in this thesis. We give a survey of the Karush-Kuhn-Tucker conditions, constraint qualifications and theory of duality. In Section 2.5 we summarize the concepts and results for the exterior penalty function approach to constrained optimization. In Section 2.6 the block coordinate descent method for non-linear programs is introduced. We show under some assumptions, how the Lagrange multipliers of the optimal solution are computed.

The clique wire length model we use in this thesis is defined in Section 2.7.

In Section 2.8 we describe the problem instances used for numerical evaluations. In Section 2.9 the computational environment is described.

2.1 Complex Numbers

We give a brief introduction to complex numbers.

Definition 2.1 (Complex Number). *By introducing a value $\imath$ with $\imath^2 = -1$ the field of complex numbers is*

$$\mathbb{C} := \{a + \imath b : a, b \in \mathbb{R}\}$$

with addition $(a+\imath b)+(c+\imath d) = (a+c)+\imath(b+d)$ and multiplication $(a+\imath b)\cdot(c+\imath d) = (ac - bd) + \imath(ad + bc)$.

The representation $a+\imath b$ is denoted as Cartesian form. The real numbers are embedded in the complex numbers by representing $a \in \mathbb{R}$ as $a+\imath 0 \in \mathbb{C}$. Division of two complex numbers can be done with non-zero nominator by

$$\frac{a+\imath b}{c+\imath d} = \frac{(a+\imath b)(c-\imath d)}{c^2+d^2} = \frac{ac+bd}{c^2+d^2} + \imath\frac{bc-ad}{c^2+d^2}.$$

Definition 2.2. *The following notations are defined:*

- *The norm or magnitude is $|a+\imath b| = \sqrt{a^2+b^2}$.*
- *The real part is $\Re(a+\imath b) = a$.*
- *The imaginary part is $\Im(a+\imath b) = b$.*
- *The conjugate of a complex number is $\overline{a+\imath b} = a-\imath b$.*

The set $\mathbb{C}$ is isomorphic to $\mathbb{R}^2$ by the natural isomorphism of identifying the complex number $a+\imath b \in \mathbb{C}$ with the point $(a,b) \in \mathbb{R}^2$. Then adding a complex number $a+\imath b$ geometrically is a translation by (a,b) and conjugation of a complex number means reflection on the y-axis.

As each point in the plane can be represented by polar coordinates, the same holds for complex numbers.

Definition 2.3 (Polar Form)**.** *The complex number $z = a+\imath b \in \mathbb{C}$ can be written in polar form as*

$$z = r\cos(\varphi) + \imath r\sin(\varphi) = r\exp(\imath\varphi)$$

where $r = |z| \geq 0$ and the angle is defined as $\varphi = \arg z$.

Theorem 2.4. *Multiplication and division in polar form can be done by*

$$\begin{aligned}(r\exp(\imath\varphi)) \cdot (s\exp(\imath\psi)) &= rs\exp(\imath(\varphi+\psi)),\\ \frac{r\exp(\imath\varphi)}{s\exp(\imath\psi)} &= \frac{r}{s}\exp(\imath(\varphi-\psi)) \quad \text{if } s>0.\end{aligned}$$

By the multiplication in polar form it can be seen that multiplication with a complex number $r\exp(\imath\varphi)$ is a rotation by φ and scaling by r.

To state some computation rules, let now be $a \in \mathbb{R}$ and $w, z \in \mathbb{C}$. Then $\Re(az) = a\Re(z)$, $\Re(z+w) = \Re(z) + \Re(w)$, $\Im(az) = a\Im(z)$, $\Im(z+w) = \Im(z) + \Im(w)$. Furthermore, $\overline{z \cdot w} = \overline{z} \cdot \overline{w}$, $\overline{z+w} = \overline{z} + \overline{w}$, $2\Re(z) = z + \overline{z}$, $2\imath\Im(z) = z - \overline{z}$, $z\overline{z} = |z|^2$ and $z = \overline{z}$ if and only if $z \in \mathbb{R}$.

We now consider the set $\mathbb{C}^n$.

- $\mathbb{C}^n$ is a n-dimensional $\mathbb{C}$ vector space.
- $\mathbb{C}^n$ is a $2n$-dimensional $\mathbb{R}$ vector space and isomorphic to $\mathbb{R}^{2n}$ by $\theta : \mathbb{C}^n \to \mathbb{R}^{2n}$, $\theta(w_1, \ldots, w_n) = (\Re(w_1), \Im(w_1), \ldots, \Re(w_n), \Im(w_n))$.

We now consider scalar products.

- The scalar product of the $\mathbb{R}$ vector space $\mathbb{R}^n$ is $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{y}^T \boldsymbol{x} = \sum_{i=1}^n x_i y_i$.
- The scalar product of the $\mathbb{C}$ vector space $\mathbb{C}^n$ is $\langle \boldsymbol{w}, \boldsymbol{z} \rangle_{\mathbb{C}} = \boldsymbol{z}^H \boldsymbol{w} = \sum_{i=1}^n w_i \overline{z}_i$.
- The scalar product of the $\mathbb{R}$ vector space $\mathbb{C}^n$ is implied by the isomorphism θ as

$$\begin{aligned}\langle \boldsymbol{w}, \boldsymbol{z} \rangle_{\mathbb{R}} &= \sum_{i=1}^n \Re(w_i)\Re(z_i) + \Im(w_i)\Im(z_i) \\ &= \sum_{i=1}^n \Re(w_i \overline{z_i}) = \Re\left(\sum_{i=1}^n w_i \overline{z_i}\right) \\ &= \Re(\boldsymbol{z}^H \boldsymbol{w}) = \Re(\langle \boldsymbol{w}, \boldsymbol{z} \rangle_{\mathbb{C}})\end{aligned}$$

Note that concepts as linear dependency depend on the underlying field. So two vectors in $\mathbb{C}^n$ can be linear independent if we consider the $\mathbb{R}$ vector space and linear dependent if we consider the $\mathbb{C}$ vector space. To distinguish these concepts, where important we say that vectors are $\mathbb{R}$ linear independent resp. $\mathbb{C}$ linear independent.

Sometimes we need a vector where one coordinate is removed. This notation is introduced in Definition 2.5.

Definition 2.5. *For a vector $\boldsymbol{z} \in \mathbb{C}^n$ denote $\pi_k(\boldsymbol{z}) = (z_1, \dots, z_{k-1}, z_{k+1}, \dots, z_n)$ the vector with removed k-th coordinate.*

In this thesis we often require complex numbers to lie within the unit circle or on the boundary of the unit circle.

Definition 2.6. *We define the following sets*

$$\mathcal{U} := \{z \in \mathbb{C} : |z| \leq 1\}, \qquad \partial\mathcal{U} := \{z \in \mathbb{C} : |z| = 1\}.$$

Definition 2.7. *We define the normalization of a complex number $z \in \mathbb{C}$ as*

$$\mathit{Normed}[v] = \begin{cases} \{v/|v|\} & \text{if } v \neq 0, \\ \partial\mathcal{U} & \text{if } v = 0. \end{cases}$$

If $v \neq 0$, we sometimes consider $\mathit{Normed}[v] \in \mathbb{C}$ as complex number.

By the real scalar product we can define hyperplanes in the complex space.

Definition 2.8. *With $v \in \partial\mathcal{U}$ denoting the normal vector and d the distance, a hyperplane in $\mathbb{C}$ is the set*

$$H(v,d) = \{z : \langle z, v \rangle_{\mathbb{R}} = d\} = \{z : \Re(z\overline{v}) = d\}.$$

The corresponding closed half spaces are given by

$$H_{\leq}(v,d) := \{z : \langle z, v \rangle_{\mathbb{R}} \leq d\} \qquad \text{and} \qquad H_{\geq}(v,d) := \{z : \langle z, v \rangle_{\mathbb{R}} \geq d\}.$$

The normal vector v can also be expressed by its argument, i.e, $v = \exp(\imath\gamma)$. Then γ is called normal vector angle and we define $H(\gamma, d) = H(v, d)$.

2.2 Wirtinger Calculus

For a function $f : \mathbb{C} \to \mathbb{C}$ the property of being complex differentiable is very strong. By separating real and imaginary part, we can consider f as a function $f : \mathbb{R}^2 \to \mathbb{R}^2$. The requirement of being real differentiable is much weaker. However, computations of easy to write complex functions become quite complicated when separated into real and imaginary parts. The Wirtinger calculus greatly simplifies the real differentiation of complex functions.

Definition 2.9 (Complex Differentiable Function). *Let $U \subset \mathbb{C}$ be an open set and $f : U \to \mathbb{C}$ a function. Then f is complex differentiable (or holomorphic) on U if for each $z_0 \in U$ the limit exists:*

$$f'(z_0) := \lim_{z \to z_0} \frac{f(z) - f(z_0)}{z - z_0}.$$

Definition 2.10 (Real Differentiable Function). *Let $U \subset \mathbb{C}$ be an open set and $f : U \to \mathbb{C}$ a function, $f(x + \imath y) = u(x, y) + \imath v(x, y)$ for $x, y \in \mathbb{R}$. Then f is real differentiable on U if the real functions u and v are differentiable on U.*

The following theorem shows that complex differentiability is much stronger then real differentiability.

Theorem 2.11 (Cauchy-Riemann Differential Equations). *Let $X \subset \mathbb{C}$ be an open set and $f : X \to \mathbb{C}$ a continuous function. Then f is complex differentiable, if and only if it is real differentiable and the Cauchy-Riemann-Equations are satisfied:*

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \qquad \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}.$$

In Wirtinger Calculus a function $f(z) = f(x, y)$ is now written as $f(z, \overline{z})$ in terms of $z = x + \imath y$ and $\overline{z} = x - \imath y$. By considering the variables z and $\overline{z}$ as being independent, one can compute the partial derivatives of f with respect to z or $\overline{z}$. The observation is that this idea can made rigorous.

Definition 2.12 (Real and Conjugate Real Derivative). *Let $X \subset \mathbb{C}$ be an open set and $f : X \to \mathbb{C}$ a real differentiable function with $f(z) = f(x, y)$. Then the real derivative and the conjugate real derivative are defined by*

$$\frac{\partial}{\partial z} := \frac{1}{2}\left(\frac{\partial}{\partial x} - \imath\frac{\partial}{\partial y}\right), \qquad \frac{\partial}{\partial \overline{z}} := \frac{1}{2}\left(\frac{\partial}{\partial x} + \imath\frac{\partial}{\partial y}\right).$$

There are several facts about this operators.

- The Cauchy-Riemann-Equations for f are equivalent to $\frac{\partial f}{\partial \overline{z}} = 0$.
- There is a set of derivative identities:

$$\begin{aligned} \frac{\partial z}{\partial z} &= \frac{\partial \overline{z}}{\partial \overline{z}} = 1, & \frac{\partial \overline{z}}{\partial z} &= \frac{\partial z}{\partial \overline{z}} = 0, \\ \frac{\partial \overline{f}}{\partial \overline{z}} &= \overline{\left(\frac{\partial f}{\partial z}\right)}, & \frac{\partial \overline{f}}{\partial z} &= \overline{\left(\frac{\partial f}{\partial \overline{z}}\right)}, \\ \frac{\partial f \circ g}{\partial z} &= \frac{\partial f}{\partial g}\frac{\partial g}{\partial z} + \frac{\partial f}{\partial \overline{g}}\frac{\partial \overline{g}}{\partial z}, & \frac{\partial f \circ g}{\partial \overline{z}} &= \frac{\partial f}{\partial g}\frac{\partial g}{\partial \overline{z}} + \frac{\partial f}{\partial \overline{g}}\frac{\partial \overline{g}}{\partial \overline{z}}. \end{aligned}$$

Most of the theory for univariate calculus can be transferred to multivariate calculus.

Definition 2.13 (Real and Conjugate Real Derivative)**.** *Let $X \subset \mathbb{C}^n$ be an open set and $f : X \to \mathbb{C}^m$ a real differentiable function with $f(\boldsymbol{z}) = f(\boldsymbol{x}, \boldsymbol{y})$. Then the cogradient and the conjugate cogradient operators are defined by*

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{z}} &:= \frac{1}{2}\left(\frac{\partial}{\partial \boldsymbol{x}} - \imath \frac{\partial}{\partial \boldsymbol{y}}\right) = \left(\frac{\partial}{\partial z_1}, \dots, \frac{\partial}{\partial z_n}\right), \\ \frac{\partial}{\partial \overline{\boldsymbol{z}}} &:= \frac{1}{2}\left(\frac{\partial}{\partial \boldsymbol{x}} + \imath \frac{\partial}{\partial \boldsymbol{y}}\right) = \left(\frac{\partial}{\partial \overline{z_1}}, \dots, \frac{\partial}{\partial \overline{z_n}}\right). \end{aligned}$$

The $m \times n$-Matrix $\frac{\partial f}{\partial \boldsymbol{z}}(\boldsymbol{z}, \overline{\boldsymbol{z}})$ is the Jacobian, $\frac{\partial f}{\partial \overline{\boldsymbol{z}}}(\boldsymbol{z}, \overline{\boldsymbol{z}})$ the conjugate Jacobian. As for the univariate case the common differential rules are valid.

We now consider the special case of real valued functions. Then we write the gradient $\nabla_{\boldsymbol{z}} := \frac{\partial}{\partial \boldsymbol{z}}$ and conjugate gradient $\nabla_{\overline{\boldsymbol{z}}} := \frac{\partial}{\partial \overline{\boldsymbol{z}}}$.

Theorem 2.14. *Let $X \subset \mathbb{C}^n$ be an open set and $f : X \to \mathbb{R}$ a real differentiable function $f(\boldsymbol{z}) = f(\boldsymbol{x}, \boldsymbol{y})$. Then the following statements are equivalent:*

1. *f has a stationary point in $\boldsymbol{z}$.*
2. *$\nabla_{\boldsymbol{z}} f(\boldsymbol{z}, \overline{\boldsymbol{z}}) = 0$.*
3. *$\nabla_{\overline{\boldsymbol{z}}} f(\boldsymbol{z}, \overline{\boldsymbol{z}}) = 0$.*

Sometimes functions depend on complex and real variables. We want to embed such functions in the current framework. It would be convenient to use the Wirtinger gradient with respect to the complex variables and the standard gradient with respect to the real variables.

So let $X \subset \mathbb{C}^n$ and $U \subset \mathbb{R}^m$ be open sets, $f : X \times U \to \mathbb{R}$ a real differentiable function. Then define $Y := U + \imath\mathbb{R}$,

$$g : X \times Y \to \mathbb{R}, \qquad g(\boldsymbol{z}, \boldsymbol{u} + \imath \boldsymbol{v}) = f(\boldsymbol{z}, \boldsymbol{u}).$$

Then g is real differentiable. Denote by ∇ the Wirtinger gradient and by ∇' the real gradient. Then with $\boldsymbol{w} = (\boldsymbol{z}, \boldsymbol{u} + \imath\boldsymbol{v})$ it is

$$\begin{aligned}\nabla_{\overline{\boldsymbol{w}}} g(\boldsymbol{w}) &= \left(\nabla_{\overline{\boldsymbol{z}}} g(\boldsymbol{z}, \boldsymbol{u} + \imath\boldsymbol{v}), \frac{1}{2}\nabla'_{\boldsymbol{u}} g(\boldsymbol{z}, \boldsymbol{u} + \imath\boldsymbol{v}) + \frac{1}{2}\imath\nabla'_{\boldsymbol{v}} g(\boldsymbol{z}, \boldsymbol{u} + \imath\boldsymbol{v})\right) \\ &= \left(\nabla_{\overline{\boldsymbol{z}}} f(\boldsymbol{z}, \boldsymbol{u}), \frac{1}{2}\nabla'_{\boldsymbol{u}} f(\boldsymbol{z}, \boldsymbol{u})\right), \\ \nabla_{\boldsymbol{w}} g(\boldsymbol{w}) &= \left(\nabla_{\boldsymbol{z}} f(\boldsymbol{z}, \boldsymbol{u}), \frac{1}{2}\nabla'_{\boldsymbol{u}} f(\boldsymbol{z}, \boldsymbol{u})\right),\end{aligned}$$

Remark 2.15. *We can consider functions depending on complex and real variables as functions with domain $\mathbb{C}^n$ and when calculating the gradient we can use the standard Wirtinger gradient and the standard real gradient. However, we have to consider the issue to multiply the real derivatives with the factor $\frac{1}{2}$.*

2.3 Complex Quadratic Function

2.3.1 Univariate Function

We now consider a univariate quadratic function ψ that is used in the remaining part of the thesis.

Definition 2.16. *Let $a \in \mathbb{R}^{\geq 0}$, $u \in \mathbb{C}$, $b \in \mathbb{R}$. Define*

$$\psi[a, u, b] : \mathbb{C} \to \mathbb{R}, \qquad \psi[a, u, b](z) = a\,|z|^2 + 2\Re(z\overline{u}) + b.$$

To shorten notation, we sometimes define $\psi := \psi[a, u, b]$ and just use $\psi(z)$.

Remark 2.17. *If $a > 0$ then $\psi[a, u, b]$ is strictly convex and it is*

$$\psi[a, u, b](z) = a\left|z + \frac{u}{a}\right|^2 - \frac{|u|^2}{a} + b.$$

Lemma 2.18. *Let $\psi := \psi[a, u, b]$. Then the set of minimizers of ψ on $\partial\mathcal{U}$ is equal to $-$Normed$[u]$, i.e.*

- *If $u = 0$, then every $z \in \partial\mathcal{U}$ is a minimum.*
- *Otherwise the minimum is taken for $z = -$Normed$[u]$.*

For the minimum of ψ on $\mathcal{U}$ it holds:

- *If $a = 0$ and $u = 0$, the minimum is taken for arbitrary $z \in \mathcal{U}$.*
- *If $a = 0$ and $u \neq 0$, the minimum is taken for $z = -$Normed$[u]$.*
- *If $a > 0$ and $|u| \leq a$, the minimum is taken for $z = -\frac{u}{a}$.*
- *If $a > 0$ and $|u| \geq a$, the minimum is taken for $z = -$Normed$[u]$.*

Proof. Consider the minimization problem on $\partial\mathcal{U}$. Then it is $\psi(z) = 2\Re(z\overline{u}) + a + b$. If $u = 0$ then ψ is constant and the claim follows. Otherwise the minimum of ψ is taken for the minimum of $\Re(z\overline{u})$, which is taken for $z = -\text{Normed}\,[u]$.

Consider now the minimization problem on $\mathcal{U}$. For $a = u = 0$ is ψ a constant function, so the claim follows. If $a = 0$ and $u \neq 0$ again the minimum of ψ is taken for the minimum of $\Re(z\overline{u})$, which is taken for $z = -\text{Normed}\,[u]$. If $a > 0$ by Remark 2.17 the other statements follow. □

Lemma 2.19. *Let $\psi = \psi[a, u, b]$. Let ψ_r^* be the minimal value of ψ on $\mathcal{U}$ and ψ_s^* be the minimum of ψ on $\partial\mathcal{U}$. Then*

$$\psi_s^* = a + b - 2\,|u|\,,$$
$$\psi_r^* = \begin{cases} a + b - 2\,|u| & \text{if } |u| \geq a, \\ -\frac{|u|^2}{a} + b & \text{if } |u| < a. \end{cases}$$

Especially

- *If $|u| \geq a$, then $\psi_r^* = \psi_s^*$.*
- *If $|u| < a$, then*

$$\psi_s^* - \psi_r^* = a\left(1 - \frac{|u|}{a}\right)^2 \leq a.$$

Proof. If $|u| \geq a$ the statement follows immediately from Lemma 2.18.

If $|u| < a$ it is $a > 0$. So by Lemma 2.18 the minimum of ψ over $\partial\mathcal{U}$ is taken for $z_s^* = -\text{Normed}\,[u]$ and a minimum of ψ over $\mathcal{U}$ is taken for $z_r^* = -\frac{u}{a}$. So it follows by Remark 2.17

$$\begin{aligned}\psi(z_s^*) - \psi(z_r^*) &= \left(a\left|-\frac{u}{|u|} + \frac{u}{a}\right|^2 - \frac{|u|^2}{a} + b\right) - \left(a\left|-\frac{u}{a} + \frac{u}{a}\right|^2 - \frac{|u|^2}{a} + b\right) \\ &= a\left|\frac{u}{a} - \frac{u}{|u|}\right|^2 = a\left(1 - \frac{|u|}{a}\right)^2 \leq a\end{aligned}$$

where the last inequality holds, as $0 \leq \frac{|u|}{a} \leq 1$. □

2.3.2 Multivariate Function

We now consider a multivariate quadratic function Ψ.

Definition 2.20. *Let $A \in \mathbb{C}^{n\times n}$ be a Hermitian positive semidefinite form, $\boldsymbol{u} \in \mathbb{C}^n$ and $b \in \mathbb{R}$. Define*

$$\Psi[A, \boldsymbol{u}, b] : \mathbb{C}^n \to \mathbb{R}, \qquad \Psi[A, \boldsymbol{u}, b](\boldsymbol{z}) = \boldsymbol{z}^H A\boldsymbol{z} + 2\Re(\boldsymbol{u}^H\boldsymbol{z}) + b.$$

Definition 2.21. *Let $\Psi = \Psi[A, u, b]$. The k-th coordinate function Ψ_k is the function Ψ in the variable z_k, when all other variables are fixed. With Definition 2.5 we get*

$$\Psi_k(\cdot, \pi_k(\boldsymbol{z})) : \mathbb{C} \to \mathbb{R}, \qquad \Psi_k(\xi, \pi_k(\boldsymbol{z})) = \Psi(z_1, \ldots, z_{k-1}, \xi, z_{k+1}, \ldots, z_n).$$

We now show that the k-th coordinate function Ψ_k of the multivariate function Ψ is of the form of the univariate quadratic function ψ.

Lemma 2.22. *Let $\Psi = \Psi[A, u, b]$. Then*

$$\Psi_k(\cdot, \pi_k(\boldsymbol{z})) = \psi \left[a_{kk}, \sum_{j \neq k} a_{ki} z_i + u_k, \sum_{i,j \neq k} a_{ij} \overline{z_i} z_j + 2\Re \left(\sum_{j \neq k} z_j \overline{u_j} \right) + b \right]$$

Proof. Calculation with $A = (a_{ij})_{1 \leq i,j \leq n}$ and $\boldsymbol{u} = (u_1, \ldots, u_n)$ yields

$$\begin{aligned}
&\Psi_k(\cdot, \pi_k(\boldsymbol{z})) \\
&= \Psi(z_1, \ldots, z_{k-1}, \xi, z_{k+1}, \ldots, z_n) \\
&= a_{kk} \left|\xi\right|^2 + \sum_{i \neq k} a_{ik} \overline{z_i} \xi + \sum_{j \neq k} a_{kj} \bar{\xi} z_j + \sum_{i,j \neq k} a_{ij} \overline{z_i} z_j + 2\Re \left(\xi \overline{u_k} + \sum_{j \neq k} z_j \overline{u_j} \right) + b \\
&= a_{kk} \left|\xi\right|^2 + 2\Re \left(\xi \cdot \overline{\sum_{j \neq k} a_{ki} z_i + u_k} \right) + \sum_{i,j \neq k} a_{ij} \overline{z_i} z_j + 2\Re \left(\sum_{j \neq k} z_j \overline{u_j} \right) + b.
\end{aligned}$$

□

2.4 Non-Linear Optimization

We summarize some basic results for non-linear optimization which are used throughout the thesis. For a more complete introduction containing proofs see [Bertsekas 1999], [Bazaraa; Sherali; Shetty 1993] or [Nocedal; Wright 2006]. Furthermore, we do not need all statements in its full generality. Where appropriate, we simplify the statements to the case required for this thesis.

In most text books these results are stated for the domain $\mathbb{R}^n$. By Section 2.2 they can easily be transferred to the $\mathbb{R}$ vector space $\mathbb{C}^n$. So let $\mathbb{F}$ be either $\mathbb{R}$ or $\mathbb{C}$. If $\mathbb{F} = \mathbb{R}$, then $\mathbb{F}^n$ is the standard Euclidean vector space, $\langle \cdot, \cdot \rangle$ the standard scalar product and ∇ the standard gradient operator. If $\mathbb{F} = \mathbb{C}$, then $\mathbb{F}^n$ denotes the $\mathbb{R}$ vector space $\mathbb{C}^n$ and thus $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_{\mathbb{R}}$. Furthermore, with ∇ we mean the Wirtinger gradient to either the variable or its conjugate.

In this section let $X \subset \mathbb{F}^n$ be an open set, $f : X \to \mathbb{R}$, $g_i : X \to \mathbb{R}$, $i \in I = \{1, \ldots, m\}$ and $h_j : X \to \mathbb{R}$, $j \in J = \{1, \ldots, k\}$. Denote $g = (g_1, \ldots, g_m)$ and $h = (h_1, \ldots, h_k)$. Let f, g and h be continuously differentiable.

Definition 2.23 (Non-Linear Problem)**.** *A non-linear problem in standard formulation is*

$$\begin{aligned} \min_{\boldsymbol{x}\in X} \quad & f(\boldsymbol{x}) \\ \text{s. t.} \quad & g(\boldsymbol{x}) \leq 0 \\ & h(\boldsymbol{x}) = 0. \end{aligned} \tag{P}$$

We call the problem convex, if X, f and g are convex and h is affine.

Definition 2.24. *For the problem (P) the Lagrange function is*

$$L : \mathbb{F}^n \times \mathbb{R}^m \times \mathbb{R}^k \to \mathbb{R}, \qquad (\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\eta}) \mapsto f(\boldsymbol{x}) + \boldsymbol{\mu}^T g(\boldsymbol{x}) + \boldsymbol{\eta}^T h(\boldsymbol{x}).$$

In the following we state some optimality conditions for non-linear programs. However, they contain some regularity assumptions to problem or the optimal point. A point satisfying these assumptions is called regular.

There are various constraint qualifications that assure regularity. We now state some of these. Let $\boldsymbol{x}^* \in X$ be a feasible point. Denote by $I(\boldsymbol{x}^*) = \{i \in I : g_i(\boldsymbol{x}^*) = 0\}$ the set of active inequality constraints.

Slater's Constraint Qualification Let g be convex at $\boldsymbol{x}^*$. Let there be a point $\boldsymbol{x}$ such that $g_i(\boldsymbol{x}) < 0$ for all $i \in I$. Let $\nabla h_j(\boldsymbol{x}^*)$, $j \in J$ be linear independent. Then $\boldsymbol{x}^*$ is regular.

Linear Independent Constraint Qualification (LICQ) If the equality constraint gradients $\nabla h_j(\boldsymbol{x}^*)$, $j \in J$ and the active inequality constraint gradients $\nabla g_i(\boldsymbol{x}^*)$, $i \in I(\boldsymbol{x}^*)$ are linearly independent, then $\boldsymbol{x}^*$ is regular.

Mangasarian-Fromovitz Constraint Qualification (MFCQ) Let $\nabla h_j(\boldsymbol{x}^*)$, $j \in J$ be linearly independent. Let there be a $\boldsymbol{d} \in \mathbb{R}^n$ such that $\langle \boldsymbol{d}, \nabla g_i(\boldsymbol{x}^*)\rangle < 0$, $i \in I(\boldsymbol{x}^*)$ and $\langle \boldsymbol{d}, \nabla h_j(\boldsymbol{x}^*)\rangle = 0$ for $j \in J$. Then $\boldsymbol{x}^*$ is regular.

Remark 2.25. *Equivalent to MFCQ is the condition: For $\eta_j \in \mathbb{R}$, $j \in J$ and $\mu_i \in \mathbb{R}^{\geq 0}$, $i \in I(\boldsymbol{x}^*)$ it is*

$$\sum_{i\in I(\boldsymbol{x}^*)}^{k} \mu_i \nabla g_i(\boldsymbol{x}^*) + \sum_{j\in J}^{k} \eta_j \nabla h_j(\boldsymbol{x}^*) = 0 \qquad \Longrightarrow \qquad \eta_j = 0, j \in J \wedge \mu_i = 0, i \in I(\boldsymbol{x}^*)$$

Theorem 2.26 (Karush-Kuhn-Tucker Necessary Conditions)**.** *Let $\boldsymbol{x}^* \in X$ be feasible to (P) and regular. Let f and g and h be continuously differentiable at $\boldsymbol{x}^*$. If $\boldsymbol{x}^*$ solves the problem locally, there are vectors $\boldsymbol{\mu}^* \in \mathbb{R}^m$ and $\boldsymbol{\eta}^* \in \mathbb{R}^k$ such that*

$$\begin{aligned} \nabla_{\boldsymbol{x}} L(\boldsymbol{x}^*, \boldsymbol{\mu}^*, \boldsymbol{\eta}^*) &= 0, \\ \mu_i^* \cdot g_i(\boldsymbol{x}^*) &= 0, \quad i \in I, \\ \boldsymbol{\mu}^* &\geq 0. \end{aligned}$$

If Slater's condition or LICQ holds, then $\boldsymbol{\mu}^*$ and $\boldsymbol{\eta}^*$ are unique.

Theorem 2.27 (Karush-Kuhn-Tucker Sufficient Conditions)**.** *Let* (P) *be convex and* $\boldsymbol{x}^* \in X$ *a regular feasible solution and assume that the KKT conditions hold. Then* $\boldsymbol{x}^*$ *is a global optimum to* (P)*. If the convexity assumptions hold only in a neighborhood of* $\boldsymbol{x}^*$*, then* $\boldsymbol{x}^*$ *is a local optimum to* (P)*.*

Definition 2.28 (Lagrangian Dual Problem)**.** *For* (P) *the dual function* $\theta : \mathbb{R}^m \times \mathbb{R}^k \to \mathbb{R}$ *is defined by*

$$\theta(\boldsymbol{\mu}, \boldsymbol{\eta}) = \inf\{L(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\eta}) : \boldsymbol{x} \in X\}.$$

Then the dual problem is

$$\begin{aligned} \max \quad & \theta(\boldsymbol{\mu}, \boldsymbol{\eta}) \\ \text{s. t.} \quad & \boldsymbol{\mu} \geq 0. \end{aligned} \tag{D}$$

The dual problem gives a lower bound for the primal problem. However, in the non-convex case this bound might not be tight.

Theorem 2.29 (Weak Duality Theorem)**.** *Let* $\boldsymbol{x}$ *be feasible to* (P) *and* $(\boldsymbol{\mu}, \boldsymbol{\eta})$ *be feasible to* (D)*, i.e.* $\boldsymbol{\mu} \geq 0$*. Then* $f(\boldsymbol{x}) \geq \theta(\boldsymbol{\mu}, \boldsymbol{\eta})$*.*

Theorem 2.30 (Strong Duality Theorem)**.** *Let the problem* (P) *be convex and have a finite optimal solution. Let the Slater's condition hold. Then there is no duality gap, i.e. there is* $\boldsymbol{x}^*$ *feasible to* (P) *and* $(\boldsymbol{\mu}^*, \boldsymbol{\eta}^*)$ *be feasible to* (D) *such that*

$$f(\boldsymbol{x}^*) = \theta(\boldsymbol{\mu}^*, \boldsymbol{\eta}^*).$$

Definition 2.31 (Saddle Point)**.** *A solution* $(\boldsymbol{x}^*, \boldsymbol{\mu}^*, \boldsymbol{\eta}^*)$ *with* $\boldsymbol{x}^* \in X$ *and* $\boldsymbol{\mu}^* \geq 0$ *is called a saddle point of the Lagrangian function, if for all* $\boldsymbol{x} \in X$ *and* $(\boldsymbol{\mu}, \boldsymbol{\eta})$ *with* $\boldsymbol{\mu} \geq 0$

$$L(\boldsymbol{x}^*, \boldsymbol{\mu}, \boldsymbol{\eta}) \leq L(\boldsymbol{x}^*, \boldsymbol{\mu}^*, \boldsymbol{\eta}^*) \leq L(\boldsymbol{x}, \boldsymbol{\mu}^*, \boldsymbol{\eta}^*).$$

Theorem 2.32 (Saddle Point Theorem)**.** *Let* $\boldsymbol{x}^* \in X$ *and* $(\boldsymbol{\mu}^*, \boldsymbol{\eta}^*)$ *with* $\boldsymbol{\mu}^* \geq 0$*.*

Then $(\boldsymbol{x}^*, \boldsymbol{\mu}^*, \boldsymbol{\eta}^*)$ *is a saddle point of the Lagrangian if and only if* $\boldsymbol{x}^*$ *is optimal for* (P) *and* $(\boldsymbol{\mu}^*, \boldsymbol{\eta}^*)$ *is optimal for* (D) *and there is no duality gap, i.e.* $f(\boldsymbol{x}^*) = \theta(\boldsymbol{\mu}^*, \boldsymbol{\eta}^*)$*.*

If (D) *is convex, then* $(\boldsymbol{x}^*, \boldsymbol{\mu}^*, \boldsymbol{\eta}^*)$ *is a saddle point of the Lagrangian if and only if* $(\boldsymbol{x}^*, \boldsymbol{\mu}^*, \boldsymbol{\eta}^*)$ *satisfies the KKT-conditions in Theorem 2.26.*

2.5 Exterior Penalty Function

In this section we survey results for the quadratic penalty function approach to equality constrained optimization problems. For a detailed analysis and extensions see [Bazaraa; Sherali; Shetty 1993] or [Nocedal; Wright 2006].

Consider the minimization problem

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s. t.} \quad & h(\boldsymbol{x}) = 0 \\ & \boldsymbol{x} \in X. \end{aligned} \tag{P}$$

The idea of the penalty approach is, to add the violation of the constraint to the objective. Then a sequence of problems for increasing penalty weight μ is solved:

$$\begin{aligned} \min \quad & g_\mu(\boldsymbol{x}) := f(\boldsymbol{x}) + \mu \left|\left|h(\boldsymbol{x})\right|\right|^2 \\ \text{s. t.} \quad & \boldsymbol{x} \in X. \end{aligned} \tag{Q_μ}$$

One expects that for increasing $\mu \to \infty$ the solution of (Q_μ) should converge to an optimal and hence feasible solution of P. And such results exists indeed.

Theorem 2.33. *Assume a sequence $\mu_k \to \infty$ for $k \to \infty$ and assume that there is a global minimizer $\boldsymbol{x}_k$ of (Q_{μ_k}). Then every limit point $\boldsymbol{x}^*$ of the sequence $\boldsymbol{x}_k$ is a global minimizer of (P). In particular, if $\boldsymbol{x}_k$ is contained in a compact set, such a limit point exists and $h(\boldsymbol{x}_k) \to 0$ for $k \to \infty$.*

Despite yielding a strong theoretical result, Theorem 2.33 is of little practical value. Usually, it is not possible to determine an exact global minimizer $\boldsymbol{x}_k$ of (Q_{μ_k}). A more practical result is given in Theorem 2.34.

Theorem 2.34. *Assume $X = \mathbb{R}^n$ and $\mu_k \to \infty$ for $k \to \infty$ and assume that $\boldsymbol{x}_k \in X$ satisfies $\nabla g_{\mu_k}(\boldsymbol{x}_k) \leq \tau_k$ with $\tau_k \to 0$ for $k \to \infty$. Denote by $\boldsymbol{x}^*$ a limit point of the sequence $\boldsymbol{x}_k$.*

- *If $\boldsymbol{x}^*$ is infeasible to (P) then it is a stationary point of $\left|\left|h(\boldsymbol{x})\right|\right|^2$.*
- *If $\boldsymbol{x}^*$ is feasible to (P) and the equality constraint gradients are linearly independent at $\boldsymbol{x}^*$, then $\boldsymbol{x}^*$ is a KKT point for (P). In this case for any subsequence $\boldsymbol{x}_{k_j}$ converging to $\boldsymbol{x}^*$ the Lagrange multiplier λ_i^* for h_i at $\boldsymbol{x}^*$ in P is*

$$\lambda_i^* = \lim_{j \to \infty} -\mu_{k_j} h_i(\boldsymbol{x}_{k_j}).$$

2.6 Block Coordinate Descent Method

The block coordinate descent method, also known as non-linear Gauss-Seidel method or alternating optimization, is a simple but practically efficient optimization method. A short description relying on strong assumptions is in [Bertsekas 1999, Section 2.7], for a more general and rather complete survey see [Bezdek; Hathaway 2003].

Let $D := \mathbb{R}^{m_1} \times \cdots \times \mathbb{R}^{m_n}$ and $f : D \to \mathbb{R}$ continuously differentiable. Let $X = X_1 \times \cdots \times X_k$ with $X_k \subset \mathbb{R}^{m_k}$. Consider the problem

$$\min_{\boldsymbol{x} \in X} f(\boldsymbol{x}). \tag{2.1}$$

Denote $\boldsymbol{y}_k = \pi_k(\boldsymbol{x}) = (x_1, \dots, x_{k-1}, x_{k+1}, \dots, x_n)$. Assume that for all k and fixed $\boldsymbol{y}_k$ we can solve the problem

$$\min_{\xi \in X_k} f_k(\xi, \boldsymbol{y}_k) = f(x_1, \dots, x_{k-1}, \xi, x_{k+1}, \dots, x_n). \tag{2.2}$$

The block coordinate descent method now starts with an initial $\boldsymbol{x}^0$ and iteratively computes $\boldsymbol{x}^{t+1} = (x_1^{t+1}, \dots, x_n^{t+1})$ from the current iteration $\boldsymbol{x}^t = (x_1^t, \dots, x_n^t)$ by

$$\begin{aligned} \boldsymbol{y}_k^t &= (x_1^{t+1}, \dots, x_{k-1}^{t+1}, x_{k+1}^t, \dots, x_n^t), \\ x_k^{t+1} &\in \operatorname*{argmin}_{\xi \in X_k} f_k(\xi, \boldsymbol{y}_k^t), \quad k = 1, \dots, n. \end{aligned}$$

Theorem 2.35 ([Bertsekas 1999, Theorem 2.7.1])**.** *Assume all X_k to be closed and convex. If in each step the minimum is uniquely attained, then every limit point of a sequence generated by the block coordinate descent method is a stationary point of* (2.1).

Note that the assumptions in Theorem 2.35 can be relaxed, but always some kind of convexity assumptions have to be made for theoretical convergence statements. However, in practice even if these conditions are not satisfied, the block coordinate descent method works well.

Remark 2.36. *In any case the sequence $f(\boldsymbol{x}^t)$ is decreasing. So if f is bounded below, $f(\boldsymbol{x}^t)$ converges from above to some f_0. In practice usually the algorithm is stopped when the decrease becomes to small, e.g. when $f(\boldsymbol{x}^t) - f(\boldsymbol{x}^{t+1}) \leq \varepsilon$ for some fixed $\varepsilon > 0$. The the algorithm is assured to terminate after a finite number of steps, as $f(\boldsymbol{x}^t) \to f_0$ implies $f(\boldsymbol{x}^t) - f(\boldsymbol{x}^{t+1}) \to 0$.*

2.6.1 Dual Variables in Block Coordinate Descend Method

The Block coordinate descent method is a feasible method, i.e, it approaches the optimal solution from above. However, in this thesis we apply the method in cases where we need a lower bound for the optimum. We did not find any result in literature about the estimation of the optimality gap in block coordinate descent method. However, in our case the problem satisfies very strong assumptions. We now prove convergence results of the dual variables of the block coordinate descent method for these strong assumptions.

Assumption 2.37. *We now make the following assumptions.*

- *The function f is strictly convex.*
- *The set X and thus X_k are compact and convex.*
- *There are continuously differentiable convex functions $g_k^i : \mathbb{R}^{m_k} \to \mathbb{R}$, $k = 1, \ldots, n$, $i = 1, \ldots, c_k$ such that with $g_k = (g_k^1, \ldots, g_k^{c_k})$ it is*

$$X_k = \{\xi \in \mathbb{R}^{m_k} : g_k(\xi) \leq 0\}.$$

- *Slater's Condition is satisfied, i.e. there is an x such that $g_k(x) < 0$ for all k.*

Corollary 2.38. *Under Assumption 2.37 the block coordinate descent method converges to the unique global optimum of f on X.*

Proof. Each X_k and so X is convex. Furthermore, f is strictly convex, so each subproblem (2.2) is strictly convex. In each step the minimum is uniquely attained. By Theorem 2.35 every limit point of the sequence $\boldsymbol{x}^t$ is a stationary point. As (P) is strictly convex on a compact domain, there is a unique global optimum $\tilde{\boldsymbol{x}}$ which is the only stationary point. So every limit point of $\boldsymbol{x}^t$ is $\tilde{\boldsymbol{x}}$, i.e. $\boldsymbol{x}^t$ converges to $\tilde{\boldsymbol{x}}$. □

We now consider the Lagrange multipliers. With Assumption 2.37 the problem (2.1) can be stated as

$$\begin{array}{ll} \min & f(x_1, \ldots, x_m) \\ \text{s. t.} & g_k(x_k) \leq 0, \quad k = 1, \ldots, m. \end{array} \tag{P}$$

Then (P) has a unique global optimum $\tilde{\boldsymbol{x}} = (\tilde{x}_1, \ldots, \tilde{x}_n)$ with unique Lagrange multiplier $\tilde{\boldsymbol{\mu}} = (\tilde{\mu}_1, \ldots, \tilde{\mu}_n)$. The problem (2.2) becomes

$$\begin{array}{ll} \min & f_k(\xi, \boldsymbol{y}_k) = f(x_1, \ldots, x_{k-1}, \xi, x_{k+1}, \ldots, x_n) \\ \text{s. t.} & g_k(\xi) \leq 0, \quad k = 1, \ldots, m. \end{array} \tag{$P_k(\boldsymbol{y}_k)$}$$

As the problem is also strictly convex the solution and its Lagrange multipliers are unique. They depend on the vector $\boldsymbol{y}_k$. Denote them $x_k^*(\boldsymbol{y}_k)$ and $\mu_k^*(\boldsymbol{y}_k)$.

We already know that the primal solutions of $(P_k(\boldsymbol{y}_k))$ converge to the primal solution of (P). We now show that the same holds for the Lagrange multipliers. Therefore, we first need the following theorem.

Theorem 2.39 ([Semple; Zlobec 1986, Corollary 3.3])**.** *Consider the convex mathematical programming model $P(\boldsymbol{y})$*

$$\begin{array}{ll} \min & f(\boldsymbol{x}, \boldsymbol{y}) \\ \text{s. t.} & g(\boldsymbol{x}, \boldsymbol{y}) \leq 0 \end{array}$$

where $\boldsymbol{y} \in Y \subset \mathbb{R}^p$ is a data vector, $\boldsymbol{x} \in X \subset \mathbb{R}^n$ is the variable, $f : X \times Y \to \mathbb{R}$ is continuous and $f(\cdot, \boldsymbol{y})$ and $g(\cdot, \boldsymbol{y})$ are convex for every $\boldsymbol{y}$.

Assume that the set of optimal solutions of $P(\boldsymbol{y}^*)$ is bounded and not empty. Assume that Slater's condition holds for $P(\boldsymbol{y}^*)$. Furthermore, assume that for all $\boldsymbol{y}$ in some neighborhood of $\boldsymbol{y}^*$ there exists unique Lagrangian multiplier $\boldsymbol{\mu}(\boldsymbol{y})$ for $P(\boldsymbol{y})$. Then $\boldsymbol{\mu}(\boldsymbol{y})$ is continuous in $\boldsymbol{y}^*$.

Theorem 2.40. *Under Assumption 2.37 in the block coordinate descent method the Lagrange multiplier of the subproblems $(P_k(\boldsymbol{y}_k))$ converge to the Lagrange multiplier of the global optimum of (P), i.e.*

$$\lim_{t\to\infty} \mu_k^*(\boldsymbol{y}_k^t) = \tilde{\mu}_k.$$

Proof. By Corollary 2.38 it is $\boldsymbol{x}^t \to \tilde{\boldsymbol{x}}$, so $\boldsymbol{y}_k^t \to (\tilde{x}_1, \ldots, \tilde{x}_{k-1}, \tilde{x}_{k+1}, \tilde{x}_n) =: \tilde{\boldsymbol{y}}_k$. Slater's condition is satisfied, further by strict convexity for each $\boldsymbol{y}_k$ there is a unique solution of $(P_k(\boldsymbol{y}_k))$. So we can apply Theorem 2.39 and conclude that

$$\lim_{t\to\infty} \mu_k^*(\boldsymbol{y}_k^t) = \mu_k^*(\tilde{\boldsymbol{y}}_k).$$

Set $\mu_k^* := \mu_k^*(\boldsymbol{y}_k^*)$ and $\boldsymbol{\mu}^* = (\mu_1^*, \ldots, \mu_n^*)$. Then by definition for each k

$$\nabla f_k(\tilde{x}_k, \tilde{\boldsymbol{y}}_k) + (\boldsymbol{\mu}_k^*)^T \nabla g_k(\tilde{x}_k) = 0$$

and complementary slackness $(\mu_k^i)^* g_k^i(\tilde{x}_k) = 0$ holds.

We now show that $\boldsymbol{\mu}^*$ are the unique Lagrange multiplier satisfying the KKT conditions for the global problem (P), and so $\boldsymbol{\mu}^* = \tilde{\boldsymbol{\mu}}$. We denote the natural extension of g_k to the domain D by ignoring all but the k-th element vector by g_k again. Then complementary slackness $\boldsymbol{\mu}_i^* g_k^i(\tilde{\boldsymbol{x}})$ holds and

$$\begin{aligned}
&\nabla f(\tilde{\boldsymbol{x}}) + (\boldsymbol{\mu}^*)^T \nabla g(\tilde{\boldsymbol{x}}) \\
=& \Big(\nabla_{x_k} f(\tilde{\boldsymbol{x}}) + (\boldsymbol{\mu}^*)^T \nabla_{x_k} g(\tilde{\boldsymbol{x}})\Big)_{k=1,\ldots,n} \\
=& \Big(\nabla f_k(\tilde{x}_k, \tilde{\boldsymbol{y}}_k) + (\boldsymbol{\mu}_k^*)^T \nabla g_k(\tilde{x}_k)\Big)_{k=1,\ldots,n} \\
=& (0, \ldots, 0).
\end{aligned}$$

So $\boldsymbol{\mu}^*$ are the unique Lagrange multiplier to (P), i.e. $\boldsymbol{\mu}^* = \tilde{\boldsymbol{\mu}}$. □

2.7 Wire Length

In most parts of this thesis we try to find a placement with a short connection length. While the shape of the objects and the decision variables can vary, the wire length models remain the same. There are several abstractions of the wire length described in Section 1.3, in this thesis we use the clique model unless another model is explicitly stated.

Definition 2.41 (Clique Net Length)**.** *Let $\boldsymbol{q} = (q_1, \dots, q_m) \in \mathbb{C}^m$ be the pin positions of a net. The clique net length is*

$$\operatorname{nl}(\boldsymbol{q}) = \frac{1}{2} \sum_{k,l=1}^{m} |q_k - q_l|^2 .$$

Lemma 2.42. *The clique net length is a positive semidefinite Hermitian form. More precise with I_m denoting the $m \times m$ identity matrix and 1_m the $m \times m$ matrix of ones it is*

$$\operatorname{nl}(\boldsymbol{q}) = \boldsymbol{q}^H (m I_m - 1_m) \boldsymbol{q}.$$

Proof. The stated identity holds, as

$$\begin{aligned} \operatorname{nl}(\boldsymbol{q}) &= \frac{1}{2} \sum_{k,l=1}^{m} |q_k - q_l|^2 \\ &= \frac{1}{2} \sum_{k,l=1}^{m} \left(|q_k|^2 - \overline{q_k} q_l - \overline{q_l} q_k + |q_l|^2 \right) \\ &= \sum_{k=1}^{m} m\, |q_k|^2 - \sum_{k,l=1}^{m} \overline{q_k} q_l \\ &= \boldsymbol{q}^H (m I_m - 1_m) \boldsymbol{q}. \end{aligned}$$

Furthermore, the net length is obviously non-negative, so it is positive semidefinite. □

However, in general not all pins are connected with each other but they are in different nets. Then the wire length is the sum of the net length of the different nets. These nets can also be weighted by its importance. The nets are indexed by $k = 1, \dots, o$.

Definition 2.43 (Total Wire Length)**.** *Let $\boldsymbol{q} \in \mathbb{C}^m$ be the vector of all pin positions. Let μ_k the weight of net k and N_k its pin selection matrix, i.e. $N_k \boldsymbol{q}$ are the pin positions of net k. The total wire length is*

$$\operatorname{wl}(\boldsymbol{q}) = \sum_{k \in \mathcal{N}} \mu_k \cdot \operatorname{nl}(N_k \boldsymbol{q}) = \boldsymbol{q}^H \left(\sum_{k=1}^{o} \mu_k N_k^T (m_k I_{m_k} - 1_{m_k}) N_k \right) \boldsymbol{q} =: \boldsymbol{q}^H Q \boldsymbol{q}.$$

As each net length and all weights are non-negative, the wire length is non-negative and thus a positive semidefinite form of the pin positions.

Lemma 2.44 (Wire Length Scales Linearly with the Net Weights)**.** *Let $\alpha \geq 0$. In the settings of Definition 2.43 denote by wl_α the total wire length if all net weights are scaled by α, i.e. the net weight of net k is are $\alpha \mu_k$. Then*

$$\operatorname{wl}_\alpha(\boldsymbol{q}) = \boldsymbol{q}^H (\alpha Q) \boldsymbol{q} = \alpha \cdot \boldsymbol{q}^H Q \boldsymbol{q} = \alpha \cdot \operatorname{wl}(\boldsymbol{q}).$$

Proof. Simple computation yields

$$\begin{aligned}\mathrm{wl}_\alpha(\boldsymbol{q}) &= \boldsymbol{q}^H \left(\alpha \sum_{k=1}^{o} \mu_k N_k^T (m_k I_{m_k} - 1_{m_k}) N_k\right) \boldsymbol{q}\\ &= \boldsymbol{q}^H (\alpha Q)\boldsymbol{q} = \alpha \cdot \boldsymbol{q}^H Q \boldsymbol{q} = \alpha \cdot \mathrm{wl}(\boldsymbol{q}).\end{aligned}$$

□

The following statement is a generalization of the computation done in Lemma 2.42 and is used later in this thesis.

Lemma 2.45. *Let $w_{kl} \geq 0$ be the weight of the connection between q_k and q_l. Then*

$$\sum_{k<l} w_{kl} |q_k - q_l| = \frac{1}{2} \sum_{k,l=1}^{m} w_{kl} |q_k - q_l|^2 = \boldsymbol{q}^H Q \boldsymbol{q}$$

where w_{kk} is arbitrary and $Q = (q_{kl})$ is given by

$$\begin{aligned} q_{kk} &= \sum_{l \neq k} w_{kl},\\ q_{kl} &= -w_{kl} \quad \text{for } k \neq l.\end{aligned}$$

Proof. Computations yield

$$\begin{aligned}\sum_{1\leq k<l\leq m} w_{kl} |q_k - q_l|^2 &= \frac{1}{2} \sum_{k,l=1}^{m} w_{kl} |q_k - q_l|^2\\ &= \frac{1}{2} \sum_{k,l=1}^{m} w_{kl} \left(|q_k|^2 - \overline{q_k} q_l - \overline{q_l} q_k + |q_l|^2\right)\\ &= \sum_{k=1}^{m} \left(\sum_{l=1}^{m} w_{kl}\right) |q_k|^2 - \sum_{k,l=1}^{m} w_{kl} \overline{q_k} q_l\\ &= \boldsymbol{q}^H Q \boldsymbol{q}.\end{aligned}$$

□

2.8 Problem Instances

Our numerical evaluation was done based on 5 real world problem instances. These are Egrain (E), Versiplektor (V), Mekas (M), SapKit (S) and Dill (D). As all circuits except Dill have less than 100 components and Dill has 1308 components, we generated additional circuit instances by partitioning of Dill. In fact, this is realistic as for System-in-Packages the components are partitioned onto the different modules and thus on each module side there is always a subset of the components. These instances are summarized in Table 2.1 on page 28.

2.9 Computational Environment

Computations were performed on the Hercules Cluster of the Fraunhofer Institute for Industrial Mathematics, Kaiserslautern. The tests ran on a dedicated dual Intel Xeon LV 5148 Woodcrest node with 2.33 GHz, 4 cores and 8GB RAM per node. On the machines ran Suse Linux Enterprise Edition 11, the compiler was GCC 4.1.2.

As non-linear solver we used Ipopt 3.9.2. This is a primal-dual interior point algorithm, see [Wächter; Biegler 2006]. It was compiled with the sparse linear solver MA27 and the Intel Math Kernel Library 10.2.2. without parallelization.

The decision to use Ipopt was done for the following reason. We compared SQP, active set and interior point methods in Matlab 2010a for the circles facility layout problem of Chapter 7. For smaller instances all solvers were fast but SQP performed best. However, for larger instances where running time becomes more important, the interior point method was significantly superior to the other solvers. As the facility layout problem has similar structure to the placement problems in this thesis, we decided to use an interior point method. Ipopt is a widely used free interior point solver with an easy-to-use C++ interface.

Our implemented algorithms are not parallelized and thus single threaded. Nevertheless to avoid memory and cache conflicts, each process ran on a dedicated note. Especially computation times are meaningful, as the computation process was not influenced by other processes.

Circuit	Nr. Components	Nr. Nets	Nr. Pins
D0019	19	385	792
D0034	34	412	849
D0059	59	443	938
D0078	78	465	1006
D0104	104	487	1070
D0138	138	515	1151
D0183	183	541	1241
D0243	243	582	1363
D0323	323	657	1565
D0380	380	709	1699
D0448	448	773	1872
D0529	529	805	1995
D0625	625	854	2164
D0736	736	910	2351
D0868	868	990	2590
D0943	943	1039	2729
D1027	1027	1093	2885
D1116	1116	1151	3047
D1217	1217	1217	3252
D1308	1308	1275	3410
E0031	31	29	75
S0041	41	112	274
M0057	57	38	117
V0093	93	144	414

Table 2.1: Real world circuits used for numerical evaluations.

3 Circle Rotation Problem

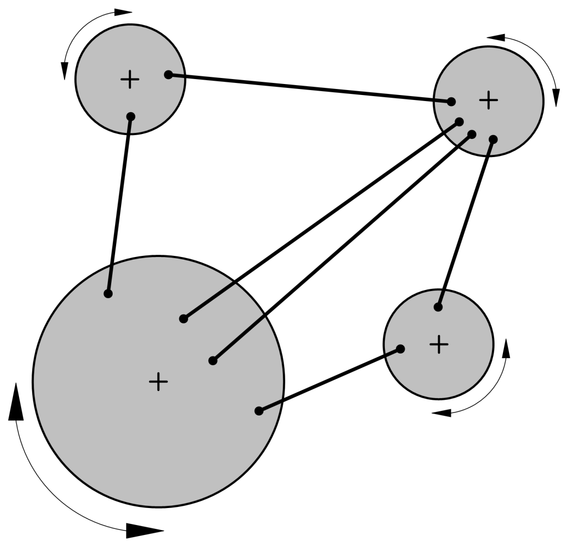


Figure 3.1: The circle rotation problem is to rotate fixed centered, connected circles such that the wire length is minimized.

In this chapter we investigate the circle rotation problem. In this problem a set of fixed centered, connected circles is given. The connection points of the circles are not in the circle center. Therefore, the rotation of the circles influences the connection length. The circle rotation problem is to find rotations of the circles such that the connection length is minimized.

This problem has to be solved in a local search strategy of our algorithm for the circle placement problem studied in Chapter 7 and the rounded rectangle algorithm stated in Chapter 8. In these algorithms, numerous circle rotation problems are solved to local optimality. Therefore, it is important to solve the circle rotation problem fast and to achieve a good solution quality.

We show that it is equivalent to the NP-hard Hermitian minimization problem. For this problem exist semidefinite programming techniques in the literature that guarantee an approximation ratio. However, to our best knowledge, the quality of fast, local optimization algorithms for this problem has not been studied in the literature.

We state and compare different models and algorithms for the circle rotation problem. In particular, we show that local optimization algorithms yield good results in short running times.

In Section 3.1 we define the circle rotation problem and formulate it as Hermitian minimization problem. Based on this formulation, we show in Section 3.3 that it is NP-hard. In Section 3.2 we give a survey of the literature results for the Hermitian minimization problem.

In the remaining part of the chapter we develop different algorithms to solve the circle rotation problem. In Section 3.4 we present local solution algorithms to the circle rotation problem and prove absolute approximation results. In Section 3.5 we consider the problem in the star net model. In Section 3.6 we make a detailed numerical comparison of the algorithms and show that local optimization algorithms for the circle placement problem yield high quality solutions in short running times.

3.1 Problem Statement

We now formalize the circle rotation problem. Given is:

- A set $\mathcal{C}$ of n circles with fixed center positions $c_j \in \mathbb{C}$. The vector of the circle centers is $\boldsymbol{c} = (c_1, \dots, c_n)^T$.
- A set $\mathcal{P}$ of m pins. Pin l is connected to circle $j(l)$ and has offset $p_l \in \mathbb{C}$ from its center. The vector of pin offsets is $\boldsymbol{p} = (p_1, \dots, p_m)^T$.
- A set $\mathcal{N}$ of o nets. Net k connects the pins $\mathcal{P}_k$ and has weight μ_k.

We want to rotate the circles such that the wire length is minimized. As a relaxation of this *strict* model we also consider the *relaxed* model, where the circles are also allowed to be shrinked (but not stretched).

We encode the rotation of circle j by a complex number z_j. So the vector of decision variables is $\boldsymbol{z} = (z_1, \dots, z_n)^T \in \mathbb{C}^n$. Recall that by Definition 2.6

$$\mathcal{U}^n := \{z \in \mathbb{C} : |z| \leq 1\}^n, \qquad \partial\mathcal{U}^n := \{z \in \mathbb{C} : |z| = 1\}^n.$$

As in the strict model the circles can only be rotated but not stretched or shrinked, we have the constraint $\boldsymbol{z} \in \partial\mathcal{U}^n$. In contrast to this in the relaxed model the constraint is $\boldsymbol{z} \in \mathcal{U}^n$.

For given rotations $\boldsymbol{z}$ the absolute pin positions can be computed. Then pin l with offset p_l from the center c_j of circle j has the position $c_j + z_j p_l$. Define the circle-to-pin matrix $\Phi = (\phi_{lj}) \in \{0,1\}^{m \times n}$ with $\phi_{lj} = 1$ if and only if pin l is on circle j. Furthermore, set $P = \operatorname{diag}(\boldsymbol{p})$. Then the vector $\boldsymbol{q}$ of absolute pin positions is

$$\boldsymbol{q} := \Phi\boldsymbol{c} + P\Phi\boldsymbol{z}. \tag{3.1}$$

The wire length is defined as in Definition 2.43 as the weighted sum of connections of the distance of pins in a net. It can be expressed as $\mathrm{wl}(\boldsymbol{z}) = \boldsymbol{q}^H Q \boldsymbol{q}$ with a positive semidefinite Hermitian matrix Q. So we get the following statement.

Lemma 3.1. *The total wire length in the circle rotation problem is a positive semidefinite Hermitian form of the circle rotations $\boldsymbol{z}$, i.e.*

$$\mathrm{wl}(\boldsymbol{z}) = b + \boldsymbol{u}^H \boldsymbol{z} + \boldsymbol{z}^H \boldsymbol{u} + \boldsymbol{z}^H A \boldsymbol{z}.$$

Proof. The total wire length in the circle rotation problem is $\mathrm{wl}(\boldsymbol{z}) = \boldsymbol{q}^H Q \boldsymbol{q}$. With the pin positions being $\boldsymbol{q} = \Phi \boldsymbol{c} + P \Phi \boldsymbol{z}$ it follows

$$\begin{aligned}\mathrm{wl}(\boldsymbol{z}) = \boldsymbol{q}^H Q \boldsymbol{q} &= (\Phi \boldsymbol{c} + P\Phi \boldsymbol{z})^H Q (\Phi \boldsymbol{c} + P \Phi \boldsymbol{z}) \\ &= b + \boldsymbol{u}^H \boldsymbol{z} + \boldsymbol{z}^H \boldsymbol{u} + \boldsymbol{z}^H A \boldsymbol{z}\end{aligned}$$

with $b := \boldsymbol{c}^H \Phi^T Q \Phi \boldsymbol{c}$, $\boldsymbol{u} := \Phi^T P^H Q \Phi \boldsymbol{c}$ and $A := \Phi^T P^H Q P \Phi$. □

Definition 3.2. *The circle rotation problem CR is* $\min_{z \in \partial \mathcal{U}^n} \mathrm{wl}(\boldsymbol{z})$ *or, formulated in standard NLP form*

$$\begin{aligned} \min \quad & b + \boldsymbol{u}^H \boldsymbol{z} + \boldsymbol{z}^H \boldsymbol{u} + \boldsymbol{z}^H A \boldsymbol{z} \\ \text{s. t.} \quad & |z_j|^2 = 1, \quad j = 1, \ldots, n. \end{aligned} \tag{3.2}$$

In Lemma 3.1 we showed that the wire length is positive semidefinite and therefore convex. However, the feasible set $\partial \mathcal{U}^n$ of the strict model is not convex. Recall that in the relaxed model we allowed the circles to be shrinked, so the feasible set is the convex set $\mathcal{U}^n$. In particular, the rotation problem in the relaxed model is a convex problem and can be solved to optimality by standard NLP solvers.

In the following we state the Hermitian minimization problem HM. This problem has been analyzed in literature and is closely related to the strict circle rotation problem. In Section 3.3 we show that the strict circle rotation problem is essentially equivalent to the Hermitian minimization problem.

Definition 3.3 (Hermitian Minimization Problem (HM))**.** *For a Hermitian positive semidefinite matrix H find* $\min_{z \in \partial \mathcal{U}^n} \boldsymbol{z}^H H \boldsymbol{z}$.

3.2 Literature Survey

Problem HM of Definition 3.3 has been analyzed in the literature for general Hermitian matrices H under different assumptions. Beside the minimization problem $\min_{z \in \partial \mathcal{U}} \boldsymbol{z}^H H \boldsymbol{z}$, also the maximization problem $\max_{z \in \partial \mathcal{U}} \boldsymbol{z}^H H \boldsymbol{z}$ has been considered. This problem is the natural extension of the maximum cut problem to complex numbers. As the approximation results to both the max cut problem and its extension are achieved by semidefinite programming (SDP), we briefly sketch SDP here. For a detailed, self contained introduction see e.g. [Todd 2001], [Vandenberghe; Boyd 1996] or [Alizadeh 1995]

An SDP can be seen as a linear program where the variables are ordered in square matrix form. Additionally to linear programs, this matrix has to be positive semidefinite. Denote variables by $X \in \mathbb{R}^{n \times n}$ and assume $C, A_i \in \mathbb{R}^{n \times n}$ and $b_i \in \mathbb{R}$. So with $C \bullet X$ denoting the component wise multiplication and stated as a maximization problem an SDP is

$$\begin{array}{lll} \max & C \bullet X & \\ \text{s. t.} & A_i \bullet X = b_i, & i = 1, \ldots, n \\ & X \succeq 0. & \end{array}$$

There are algorithms (e.g. interior point methods or augmented Lagrangian methods) that solve an SDP up to predefined precision in polynomial time.

Definition 3.4 (Max Cut Problem)**.** *Let G be a graph with positive edge weights. A cut in G is a partition of the vertices into disjoint subsets. The cut-set is the edges with end points in different partitions. The cut weight is the sum of weights of the edges in the cut set.*

The maximum cut problem is to find a cut with maximal cut weight.

The max cut problem is known to be NP hard, see [Garey; Johnson 1979]. It is even known to be NP-hard to approximate the max cut value to better than $16/17 = 0.941$, see [Håstad 2001]. In [Goemans; Williamson 1995] a polynomial algorithm to approximate max cut by 0.8786 is presented. The approximation is done by semidefinite programming and randomized rounding. They formulate max cut as a quadratic integer program to maximize $\sum_{i<j} w_{ij}(1 - y_i y_j)$ with $y_i \in \{-1, 1\}$, where w_{ij} denotes the edge weights between vertex i and j.

Note that the constraints $y_i \in \{-1, 1\}$ enforce real variables to lay on the boundary of the unit circle. So a natural extension to complex numbers is $y_i \in \partial\mathcal{U}$. Similar semidefinite programming approaches as those in [Goemans; Williamson 1995] are applied to the complex problem HM.

The main idea of this approach is as follows: Write $\boldsymbol{z}^H H \boldsymbol{z} = H \bullet \boldsymbol{z}\boldsymbol{z}^H = H \bullet Z$ where $Z = \boldsymbol{z}\boldsymbol{z}^H$ is a rank 1 matrix. Then the Hermitian minimization problem is equivalent to finding $\min\{H \bullet Z \,:\, Z \text{ has rank } 1\}$. Now by relaxing the rank 1 constraint to the constraint that Z is positive semidefinite with diagonal entries 1, the relaxed semidefinite program is obtained:

$$\begin{array}{lll} \max & H \bullet Z & \\ \text{s. t.} & Z_{ii} = 1, & i = 1, \ldots, n \\ & Z \succeq 0. & \end{array}$$

The solution to this program gives an upper bound for the original problem. The subtle part is, to recover a good solution vector $\boldsymbol{z}$ for the original problem from the relaxed SDP solution Z.

In [Zhang; Huang 2006] a polynomial time $\frac{\pi}{4}$ approximation algorithm for the maximization problem with positive semidefinite matrices is proved by the above SDP relaxation ideas.

In [So; Zhang; Ye 2007] the same approximation ratio $\frac{\pi}{4}$ is achieved by a similar approach. They additionally present a $c/\log(n)$ approximation result for the maximization problem with arbitrary matrices (i.e. also negative definite or indefinite) with zero diagonal entries. The constant c depends on the matrix entries. As the maximization problem with matrix H is equivalent to the minimization problem with $-H$, this result gives useful insights to the minimization problem.

In [Luo et al. 2007] the related problem to find $\min\{\boldsymbol{z}^H H\boldsymbol{z} : \boldsymbol{z} \in \mathbb{C}^n, |z_i| \geq 1, i = 1, \dots, n\}$ with a positive semidefinite matrix H is considered. By SDP relaxation they prove an $8n$ approximation algorithm for this problem.

Denote $f(\boldsymbol{z}) = \boldsymbol{z}^H H\boldsymbol{z}$ and by f_{min} and f_{max} the minimum and maximum of f over $\partial\mathcal{U}^n$, respectively. Then it is known from [Luo; Luo; Kisialiou 2003] that for every local minimum $\boldsymbol{z}^*$ it is

$$\frac{f(\boldsymbol{z}^*) - f_{min}}{f_{max} - f_{min}} \leq \frac{1}{2}.$$

All these approaches are interesting from a theoretical perspective. However, the results for the minimization problem are too weak to be relevant in practice. Hence, in this chapter we consider local solution algorithms and show that they solve the problem up to good solution quality in very short time.

3.3 NP-Hardness

We prove the NP-hardness of the strict circle rotation problem by reduction from the NP-hard matrix partition problem:

Matrix Partition Decision Problem (MPD): Given a matrix $G = (\boldsymbol{g}_1, \dots, \boldsymbol{g}_n) \in \mathbb{R}^{m\times n}$ where $\boldsymbol{g}_i$ denotes column i, is there a subset I of $\{1, \dots, n\}$ such that

$$\sum_{k\in I} \boldsymbol{g}_k = \frac{1}{2}\sum_{k=1}^{n} \boldsymbol{g}_k.$$

Hermitian Minimization Decision Problem (HMD): Given a positive semidefinite Hermitian matrix $H \in \mathbb{C}^{n\times n}$, is there $\boldsymbol{z} \in \partial\mathcal{U}^n$ such that $\boldsymbol{z}^H H\boldsymbol{z} = 0$.

Circle Rotation Decision Problem (CRD): Given a circuit and a constant K, is there $\boldsymbol{z} \in \partial\mathcal{U}^n$ such that $\mathrm{wl}(\boldsymbol{z}) \leq K$.

In [Zhang; Huang 2006] the following theorem is shown. For completeness, we summarize the proof here.

Theorem 3.5. *The Hermitian minimization decision problem (HMD) is NP-hard.*

Proof. Decision variables are $\boldsymbol{z} = (z_0, z_1, \ldots, z_{2n})^T$. Denote the identity matrix by I_n and by $\mathbf{1}_n$ the all one vector with n elements . Define $H = D^T D$ with

$$D = \begin{pmatrix} -\mathbf{1}_n & I_n & I_n \\ -\frac{1}{2}G\mathbf{1}_n & G & 0 \end{pmatrix} \in \mathbb{R}^{(m+n)\times(2n+1)}.$$

We show that a solution of the (MPD) exists if and only if the (HMD) has a solution for H. It follows

$$\begin{aligned} & \boldsymbol{z}^H H \boldsymbol{z} = 0 \\ \iff & D\boldsymbol{z} = 0 \\ \iff & z_k + z_{n+k} = z_0 \qquad \text{for } k = 1, \ldots, n & (3.3) \\ & \wedge \sum_{k=1}^{n} z_k \boldsymbol{g}_k = \frac{z_0}{2} \sum_{k=1}^{n} \boldsymbol{g}_k. & (3.4) \end{aligned}$$

Let $\frac{z_k}{z_0} = \exp(\imath\varphi_k)$, then for $k = 1, \ldots, n$ (3.3) is equivalent to $\cos\varphi_k + \cos\varphi_{n+k} = 1$ and $\sin\varphi_k + \sin\varphi_{n+k} = 0$. It follows $\varphi_k = \pm\frac{\pi}{3}$ and especially $\cos\varphi_k = \cos\varphi_{n+k} = \frac{1}{2}$ and $\sin\varphi_k = \pm\frac{1}{2}$ for $k = 1, \ldots, 2n$. Now (3.4) is equivalent to

$$\begin{aligned} & \sum_{k=1}^{n} \frac{z_k}{z_0} \boldsymbol{g}_k = \frac{1}{2} \sum_{k=1}^{n} \boldsymbol{g}_k \\ \iff & \Re\left(\sum_{k=1}^{n} \frac{z_k}{z_0} \boldsymbol{g}_k\right) = \Re\left(\frac{1}{2}\sum_{k=1}^{n} \boldsymbol{g}_k\right) \qquad \wedge \qquad \Im\left(\sum_{k=1}^{n} \frac{z_k}{z_0} \boldsymbol{g}_k\right) = \Im\left(\frac{1}{2}\sum_{k=1}^{n} \boldsymbol{g}_k\right) \\ \iff & \sum_{k=1}^{n} \cos(\varphi_k) \boldsymbol{g}_k = \frac{1}{2}\sum_{k=1}^{n} \boldsymbol{g}_k \qquad \wedge \qquad \sum_{k=1}^{n} \sin(\varphi_k) \boldsymbol{g}_k = 0. \end{aligned}$$

As $\cos\varphi_k = \frac{1}{2}$ the first equation is always true. As $\sin\varphi_k \in \left\{-\frac{1}{2}, +\frac{1}{2}\right\}$ the second equation is equivalent to the existence of a matrix partition of G. □

Theorem 3.6. *Let CR*$[n]$ *denote the problem CR with n circles. Let HM*$[n]$ *denote the Problem HM with a* $n \times n$ *matrix.*

(i) Each instance of CR$[n]$ *can be formulated as instance of HM*$[n+1]$*.*

(ii) Each instance of HM$[n]$ *can be formulated as instance of CR*$[n]$ *up to an additive constant.*

Proof. Proof of (i): Consider an instance of CR[n]. Then the wire length is

$$\mathrm{wl}(\boldsymbol{z}) = b + \boldsymbol{u}^H \boldsymbol{z} + \boldsymbol{z}^H \boldsymbol{u} + \boldsymbol{z}^H A \boldsymbol{z} = \begin{bmatrix} \boldsymbol{z}^H & 1 \end{bmatrix} \begin{bmatrix} A & \boldsymbol{u} \\ \boldsymbol{u}^H & b \end{bmatrix} \begin{bmatrix} \boldsymbol{z} \\ 1 \end{bmatrix} =: \begin{bmatrix} \boldsymbol{z}^H & 1 \end{bmatrix} H \begin{bmatrix} \boldsymbol{z} \\ 1 \end{bmatrix}.$$

Let $\tilde{\boldsymbol{w}}$ be an optimal solution of the instance of HM[$n+1$]:

$$\tilde{\boldsymbol{w}} \in \operatorname*{argmin}_{\boldsymbol{w} \in \partial \mathcal{U}^{n+1}} \boldsymbol{w}^H H \boldsymbol{w}.$$

As $\tilde{w}_{n+1} \neq 0$, we can define $\tilde{\boldsymbol{z}}$ by $\tilde{z}_i = \frac{\tilde{w}_i}{\tilde{w}_{n+1}}$, $i = 1, \dots, n$. Then $\tilde{\boldsymbol{z}}$ is an optimal solution of the circle rotation problem instance.

Proof of (ii): Now consider an instance of HM[n]. Denote the Hermitian positive semidefinite matrix by $H = (h_{ij}) \in \mathbb{C}^{n \times n}$. Define an instance of C[n] as follows: There are n circles centered in the origin. There are nets $\{i, j\}$ for $1 \leq i < j \leq n$ with weight 1. Net $\{i, j\}$ is connected to pin (i, j) on circle i and to pin (j, i) on circle j. The offset of pin (i, j) is $p_{ij} \in \mathbb{C}$. Set

$$p_{ij} := \begin{cases} 1 & \text{for } i < j, \\ 0 & \text{for } i = j, \\ -\overline{h_{ij}} & \text{for } i > j. \end{cases}$$

Furthermore, set $K = \sum_{i,j=1}^n |p_{ij}|^2 - \sum_{i=1}^n h_{ii}$. It is

$$\begin{aligned} &\text{for } i < j: && -\overline{p_{ij}} p_{ji} = -1 \cdot (-\overline{h_{ji}}) = \overline{h_{ji}} = h_{ij}, \\ &\text{for } i > j: && -\overline{p_{ij}} p_{ji} = -(-h_{ij}) \cdot 1 = h_{ij}, \\ &\text{for } i = j: && -\overline{p_{ij}} p_{ji} = 0. \end{aligned}$$

The pin position q_{ij} of pin (i, j) is $z_i p_{ij}$. As $|z_i| = 1$ for $i = 1, \dots, n$ we have

$$\begin{aligned} \mathrm{wl}(\boldsymbol{z}) &= \frac{1}{2} \sum_{i,j=1}^n |z_i p_{ij} - z_j p_{ji}|^2 \\ &= \frac{1}{2} \sum_{i,j=1}^n \left(|p_{ij}|^2 + |p_{ji}|^2 - \overline{z_i p_{ij}} z_j p_{ji} - z_i p_{ij} \overline{z_j p_{ji}} \right) \\ &= \sum_{i,j=1}^n |p_{ij}|^2 - \sum_{i,j=1}^n \overline{p_{ij}} p_{ji} \overline{z_i} z_j \\ &= \sum_{i,j=1}^n |p_{ij}|^2 + \sum_{i,j=1}^n (-\overline{p_{ij}} p_{ji}) \overline{z_i} z_j + \sum_{i=1}^n h_{ii} \overline{z_i} z_i - \sum_{i=1}^n h_{ii} \overline{z_i} z_i \\ &= \sum_{i,j=1}^n h_{ij} \overline{z_i} z_j + \sum_{i,j=1}^n |p_{ij}|^2 - \sum_{i=1}^n h_{ii} \\ &= \boldsymbol{z}^H H \boldsymbol{z} + K. \end{aligned}$$

It follows $\boldsymbol{z}^H H \boldsymbol{z} = \mathrm{wl}(\boldsymbol{z}) - K$. Thus the Hermitian minimization problem is formulated as circle rotation problem up to an additive constant. □

Corollary 3.7 (Strict Circle Rotation Problem is NP-Hard)**.** *The circle rotation decision problem (CRD) is NP-hard, even if all circles are centered in the origin and all nets have weight 1 and contain only two pins.*

Proof. We show a reduction from the NP-hard problem (HMD) to (CRD). Let H be a positive semidefinite Hermitian matrix $H \in \mathbb{C}^{n\times n}$. HMD is satisfied if and only if $\min_{z\in\partial\mathcal{U}} \boldsymbol{z}^H H\boldsymbol{z} \leq 0$.

By Theorem 3.6 there is an instance I of CR and a constant K such that

$$\min_{z\in\partial\mathcal{U}} \boldsymbol{z}^H H\boldsymbol{z} = \min_{z\in\partial\mathcal{U}} \mathrm{wl}(\boldsymbol{z}) + K.$$

So HMD is satisfied for this H if and only if (CRD) is satisfied for this instance I and the constant K.

Furthermore, obviously (CRD) is in NP, as for a solution $\boldsymbol{z}$ it can be checked if $\mathrm{wl}(\boldsymbol{z}) \leq K$ by evaluating a Hermitian form. □

3.4 Problem in Clique Net Model

3.4.1 Absolute Approximation Error in Euclidean Encoding

We know by Lemma 3.1 that the wire length is of the form

$$\mathrm{wl}(\boldsymbol{z}) = b + \boldsymbol{u}^H\boldsymbol{z} + \boldsymbol{z}^H\boldsymbol{u} + \boldsymbol{z}^H A\boldsymbol{z}$$

with $b = \boldsymbol{c}^H\Phi^T Q\Phi\boldsymbol{c}$, $\boldsymbol{u} = \Phi^T P^H Q\Phi\boldsymbol{c}$ and $A = \Phi^T P^H QP\Phi$ positive semidefinite.

Theorem 3.8. *Let* wl^r *be the minimum in the relaxed model and* wl^s *the minimum in the strict model, i.e.*

$$\mathrm{wl}^r = \min_{z\in\mathcal{U}^n} \mathrm{wl}(\boldsymbol{z}), \qquad \mathrm{wl}^s = \min_{z\in\partial\mathcal{U}^n} \mathrm{wl}(\boldsymbol{z}).$$

If $Q \neq 0$ *it is*

$$0 \leq \mathrm{wl}^s - \mathrm{wl}^r \leq ||A||_1 = \sum_{i,j=1}^{n} |a_{ij}|$$

and for $\min_{i,j} |c_i - c_j| \to \infty$ *we get*

$$\frac{|\mathrm{wl}^s - \mathrm{wl}^r|}{|\mathrm{wl}^r|} \to 0.$$

Proof. Define

$$\boldsymbol{u}' \in -\text{Normed}\,[\boldsymbol{u}] = -\left(\text{Normed}\,[u_1]\,,\dots,\text{Normed}\,[u_n]\right)^T.$$

Then it is $\boldsymbol{u}' \in \partial\mathcal{U}$ and for the wire length we get

$$\begin{aligned}&\text{wl}^r \geq b - 2\,||\boldsymbol{u}||_1\\&\text{wl}^s \leq \text{wl}(\boldsymbol{u}') = b - 2\,||\boldsymbol{u}||_1 + (\boldsymbol{u}')^H A\boldsymbol{u}' = b - 2\,||\boldsymbol{u}||_1 + ||A||_1\end{aligned}$$

and so the first statement follows.

Take a net (with weight μ) connecting at least two circles i and j. Denote the offset of the pins of this net by p^i and p^j. The length of this net in the restricted model is greater or equal to $|c_i - c_j| - p^i - p^j$. Thus we get

$$\text{wl}^r \geq \mu(|c_i - c_j| - p^i - p^j)^2 \to \infty$$

and together with the first statement the second statement follows. □

3.4.2 Circle Rotation Problem with Angle Encoding

In Definition 3.2 we encoded the circle rotation problem as a Hermitian form over the domain $\partial\mathcal{U}^n$. Another natural encoding is to encode each circle rotation by its angle. So the rotation of circle j is given by its angle φ_j and the decision variables are $\boldsymbol{\varphi} = (\varphi_1,\dots,\varphi_n)$. This is the same as encoding the rotation z_j of circle j in polar coordinates $z_j = \exp(\imath\varphi_j)$ where the constraints $|z_j| = 1$ are inherently satisfied.

Denoting $\exp(\imath\boldsymbol{\varphi}) = (\exp(\imath\varphi_1),\dots,\exp(\imath\varphi_n))^T$, for the wire length depending on $\boldsymbol{\varphi}$ we get with the notations b, $\boldsymbol{u}$ and A as in Definition 3.2

$$\text{wl}(\boldsymbol{\varphi}) = b + \boldsymbol{u}^H \exp(\imath\boldsymbol{\varphi}) + \exp(\imath\boldsymbol{\varphi})^H\boldsymbol{u} + \exp(\imath\boldsymbol{\varphi})^H A \exp(\imath\boldsymbol{\varphi}). \tag{3.5}$$

The advantage of this formulation for non-linear solvers is that it is an unconstrained optimization problem. However, the disadvantage is that the objective function is non-convex and involves trigonometric functions.

3.4.3 Solution by Block Coordinate Descent Method

We know by Theorem 3.6 that with Ψ as in Definition 2.20

$$\text{wl}(\boldsymbol{z}) = b + 2\Re(\boldsymbol{u}^H\boldsymbol{z}) + \boldsymbol{z}^H A\boldsymbol{z} = \Psi[A,\boldsymbol{u},b].$$

Recall that Definition 2.5 denotes $\pi_k(\boldsymbol{z})$ to be the vector with removed k-th coordinate. We denote by $\mathrm{wl}_k(\cdot, \pi_k(\boldsymbol{z}))$ the wire length when only rotating the k-th circle for fixed other circles. Then it is

$$\begin{aligned}\mathrm{wl}_k(\cdot, \pi_k(\boldsymbol{z})) &= \Psi_k(\cdot, \pi_k(\boldsymbol{z})) \\ &= \psi\left[a_{kk}, \sum_{i\neq k} a_{ki} z_i + u_k, \sum_{i,j\neq k} a_{ij}\overline{z_i} z_j + 2\Re\left(\sum_{i\neq k} z_i \overline{u_i}\right) + b\right].\end{aligned}$$

By Lemma 2.18 we compute the minimum of $\mathrm{wl}_k(\cdot, \pi_k(\boldsymbol{z}))$ as

$$\operatorname*{argmin}_{\xi\in\partial\mathcal{U}} \mathrm{wl}_k(\xi, \pi_k(\boldsymbol{z})) = -\mathrm{Normed}\left[\sum_{i\neq k} a_{ki} z_i + u_k\right].$$

Now we can apply the block coordinate descent method from Section 2.6. Geometrically this method means that, starting from an initial solution, each circle is optimally rotated subsequently while keeping the other circles fixed. This is done repeatedly for circles $1, \dots, n$ until the progress during the last n circle rotations is too small.

Note that, while the objective is convex, the prerequisites of Theorem 2.35 are not satisfied, as neither the domains $\partial\mathcal{U}$ of the subproblems are convex, nor the optimum is unique in all cases. A non-unique optimum for the rotation of a single circle k can occur, as for $\sum_{i\neq k} a_{ki} z_i + u_k = 0$ the function wl_k is constant on $\partial\mathcal{U}$ and we get

$$\operatorname*{argmin}_{\xi\in\partial\mathcal{U}} \mathrm{wl}_k(\xi, \pi_k(\boldsymbol{z})) = -\mathrm{Normed}\left[\sum_{i\neq k} a_{ki} z_i + u_k\right] = -\mathrm{Normed}\,[0] = \partial\mathcal{U}.$$

However, by Remark 2.36 the algorithm is guaranteed to terminate, also the sequence of solutions might not converge to a local optimum.

3.5 Problem in Star Net Model

In this section we formulate the circle rotation problem in the star model. First we show that clique and star model are equivalent. This means that if the net weights in the models are chosen appropriately, the net length in both models is equal. However, the star model gives additional insights into the problem. In particular, it is possible to formulate the optimization problem as problem in the net centers and analytically compute the rotations.

3.5.1 General Equivalence of Star Model and Clique Model

Definition 3.9 (Star Model)**.** *Let $\boldsymbol{q} = (q_1, \dots, q_m)^T \in \mathbb{C}^m$ be the pin positions of a net and $s \in \mathbb{C}$ be an arbitrary point called* net center*. Then the star net length of the net is*

$$\mathrm{nl}_{star}(\boldsymbol{q}, s) = \sum_{l=1}^{m} |q_l - s|^2 .$$

In [Viswanathan; Chu 2004] the following results are shown.

Lemma 3.10 (Optimal Net Center is in the Gravity Center)**.** *Let $\boldsymbol{q} \in \mathbb{C}^m$ be the pin positions of a net. Then the star net length $\mathrm{nl}_{star}(\boldsymbol{q}, s)$ is minimal, if and only if the net center s is the gravity center of $\boldsymbol{q}$.*

Proof. For the gradient we have

$$0 = \nabla_{\overline{s}}\, \mathrm{nl}_{star}(\boldsymbol{q}, s) = \sum_{l=1}^{m}(q_l - s) = -ms + \sum_{l=1}^{m} q_l \quad \Longleftrightarrow \quad s = \frac{1}{m}\sum_{l=1}^{m} q_l,$$

i.e. the gravity center is the only stationary point and turns out to be the unique global minimum. □

Lemma 3.11. *Let $\boldsymbol{q} \in \mathbb{C}^m$ be pin positions and $s \in \mathbb{C}$ the gravity center of $\boldsymbol{q}$. Then it is*

$$\mathrm{nl}(\boldsymbol{q}) = m \cdot \mathrm{nl}_{star}(\boldsymbol{q}, s).$$

Proof. We compute

$$\begin{aligned}
\mathrm{nl}_{star}(\boldsymbol{q}, s) &= \sum_{l=1}^{m} |q_l - s|^2 = \sum_{l=1}^{m} |q_l|^2 + m\,|s|^2 - \overline{s}\sum_{l=1}^{m} q_l - s\sum_{l=1}^{m}\overline{q_l} \\
&= \sum_{l=1}^{m} |q_l|^2 + \frac{1}{m}\left|\sum_{k=1}^{m} q_k\right|^2 - \frac{1}{m}\sum_{k=1}^{m}\overline{q_k}\sum_{l=1}^{m} q_l - \frac{1}{m}\sum_{k=1}^{m} q_k\sum_{l=1}^{m}\overline{q_l} \\
&= \sum_{l=1}^{m} |q_l|^2 - \frac{1}{m}\sum_{k,l=1}^{m}\overline{q_k}q_l = \frac{1}{m}\boldsymbol{q}^H(mI_m - 1_m)\boldsymbol{q} = \frac{1}{m}\,\mathrm{nl}(\boldsymbol{q}).
\end{aligned}$$

□

In general there is more than one net. As in Definition 2.43 we define the total wire length to be the weighted sum of the lengths of the different nets. In Lemma 3.11 the ratio between star and clique wire length of a net depends on the number of pins in the net. So for the total wire length to be equal in both models, we have to rescale the net weights. With m_k denoting the number of pins of net k and μ_k the weight of net k in the clique model, the weight of net k in the star model is defined to be $\mu'_k := m_k\mu_k$.

Definition 3.12 (Total Wire Length in the Star Model)**.** *Let $\boldsymbol{q} \in \mathbb{C}^m$ be the vector of all pin positions. Let N_k be the pin selection matrix of net k, i.e. $\boldsymbol{q}_k = N_k\boldsymbol{q}$ are the pin positions of net k. Let μ'_k be the weight of net k in the star model. Let $\boldsymbol{s} \in \mathbb{C}^m$ be the net centers. The total wire length is*

$$\mathrm{wl}_{star}(\boldsymbol{q}, \boldsymbol{s}) = \sum_{k=1}^{o} \mu'_k \cdot \mathrm{nl}_{star}(\boldsymbol{q}_k, s_k).$$

An immediate consequence of Lemma 3.11 is the equivalence of the total wire length in the clique and in the star model. In fact, Definition 3.12 was just stated such that both values are equal for the optimal choice of the net centers.

Corollary 3.13 (Clique and Star Model are Equivalent)**.** *If the net centers $\boldsymbol{s}^*$ are optimal chosen (i.e. as the gravity center of the corresponding nets), it is*

$$\mathrm{wl}_{star}(\boldsymbol{q}, \boldsymbol{s}^*) = \mathrm{wl}(\boldsymbol{q}).$$

Proof. By Definition 3.12, Lemma 3.11 and the definition $\mu'_k := m_k\mu_k$ it follows

$$\mathrm{wl}_{star}(\boldsymbol{q}, \boldsymbol{s}^*) = \sum_{k=1}^{o} \mu'_k \cdot \mathrm{nl}_{star}(\boldsymbol{q}_k, s^*_k) = \sum_{k=1}^{o} \mu_k \cdot \mathrm{nl}(\boldsymbol{q}_k) = \mathrm{wl}(\boldsymbol{q}).$$

□

3.5.2 Star Model for the Rotation Problem

The absolute pin positions $\boldsymbol{q}$ in the circle rotation problem are defined by the circle rotations by (3.1). So the total wire length $\mathrm{wl}_{star}(\boldsymbol{q}, \boldsymbol{s})$ depends on the circle rotations and the net centers.

In the following let $\mathcal{P}_k$ denote the pins of net k and $\mathcal{P}^j$ denote the pins of circle j. Furthermore, $k(l)$ is the net of pin l and $j(l)$ is the circle of pin l. Recall that with Φ being the circle-to-pin matrix and pins and P the diagonal matrix of the pin offsets as in (3.1), it is $\boldsymbol{q} = \Phi\boldsymbol{c} + P\Phi\boldsymbol{z}$. Then we get

$$\begin{aligned}\mathrm{wl}_{star}(\boldsymbol{z}, \boldsymbol{s}) :=& \mathrm{wl}_{star}(\boldsymbol{q}, \boldsymbol{s}) = \sum_{k=1}^{o} \mu'_k \, \mathrm{nl}(\boldsymbol{q}_k, s_k) \\ =& \sum_{k=1}^{o} \sum_{l \in \mathcal{P}_k} \mu'_k \left| c_{j(l)} + z_{j(l)} p_l - s_k \right|^2 .\end{aligned} \tag{3.6}$$

We already know by Lemma 3.10, how to choose the net centers based on the pin positions or, equivalently, on the circle rotations, such that the wire length becomes optimal. In the following we are going to show that for given net centers we can also analytically compute the circle rotations such that the wire length becomes optimal.

We first find a different representation of the wire length in the star model. In (3.6), first the wire length of all nets are computed and then summed up. We can also first compute all weighted distances of the pins of a circle to their corresponding net centers, and then sum of these weighted connection lengths of each circle.

Theorem 3.14. *The weighted connection length* $\mathrm{wl}^{j}_{star}(z_j, \boldsymbol{s})$ *of the pins of circle* j *to the corresponding net centers only depends on the rotation of circle* j *and it is*

$$\mathrm{wl}^{j}_{star}(z_j, \boldsymbol{s}) = \sum_{l \in \mathcal{P}^j} \mu'_{k(l)} \left| c_j + z_j p_l - s_{k(l)} \right|^2 = \psi[a_j, u_j(\boldsymbol{s}), b_j(\boldsymbol{s})] \qquad (*)$$

with

$$\begin{aligned} a_j &= \sum_{l \in \mathcal{P}^j}^{m} \mu'_{k(l)} \left| p_l \right|^2 , \\ u_j(\boldsymbol{s}) &= \sum_{l \in \mathcal{P}^j}^{m} \mu'_{k(l)} \overline{p_l} \left(s_{k(l)} - c_j \right) , \\ b_j(\boldsymbol{s}) &= \sum_{l \in \mathcal{P}^j}^{m} \mu'_{k(l)} \left| s_{k(l)} - c_j \right|^2 . \end{aligned}$$

For the wire length the following identity holds

$$\mathrm{wl}_{star}(\boldsymbol{z}, \boldsymbol{s}) = \sum_{j=1}^{n} \mathrm{wl}^{j}_{star}(z_j, \boldsymbol{s}). \qquad (**)$$

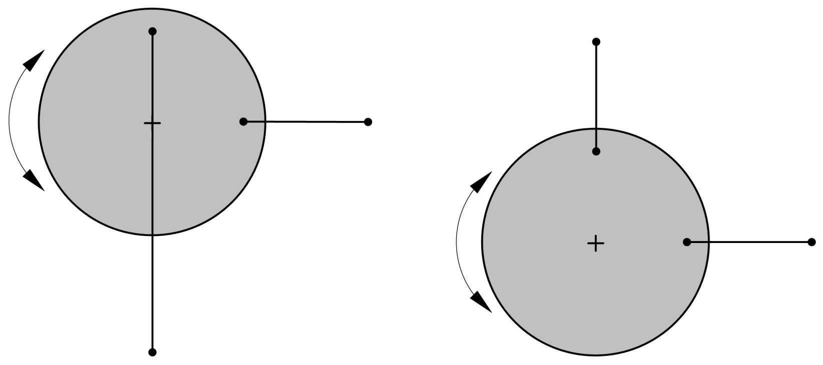
Proof. Identity $(*)$ can be seen by

$$\sum_{l \in \mathcal{P}^j}^{m} \mu'_{k(l)} \left| c_j + z_j p_l - s_{k(l)} \right|^2 = \left| z_j \right|^2 a_j + 2\Re(z_j \overline{u_j(\boldsymbol{s})}) + b_j(\boldsymbol{s}) = \psi[a_j, u_j(\boldsymbol{s}), b_j(\boldsymbol{s})].$$

Then we get identity $(**)$:

$$\begin{aligned} \mathrm{wl}_{star}(\boldsymbol{z}, \boldsymbol{s}) &= \sum_{k=1}^{o} \sum_{l \in \mathcal{P}_k} \mu'_k \left| c_{j(l)} + z_{j(l)} p_l - s_k \right|^2 = \sum_{l=1}^{m} \mu'_k \left| c_{j(l)} + z_{j(l)} p_l - s_{k(l)} \right|^2 \\ &= \sum_{j=1}^{n} \sum_{l \in \mathcal{P}^j} \mu'_{k(l)} \left| c_j + z_j p_l - s_{k(l)} \right|^2 = \sum_{j=1}^{n} \mathrm{wl}^{j}_{star}(z_j, \boldsymbol{s}). \end{aligned}$$

□



(a) Orthogonality of the net centers and the pin offsets.

(b) Linear dependence of the net centers and the pin offsets.

Figure 3.2: A circle is stretched, if the angle between the net centers and the pin offsets is not too big. For linear dependent pin offsets, the circles are stretched, as all net lengths decrease with growing circle. For orthogonal pin offsets, whenever one net decreases its length another net increases the length.

These values a_j, u_j and b_j have a geometric interpretation.

The value a_j is the weighted sum of pin offsets of component j. This is geometrically intuitive, as this coefficient is related to $|z_j|^2$ and thus related to the size of the circle. However, if the pin offsets are further away from the circle center, shrinking and stretching the circle has a bigger impact on the wire length.

The value u_j is the (weighted) scalar product of the pin offsets and the location of the corresponding connection points relative to the circle center. So u_j has large magnitude, if the vector of pin offsets (p_l) and the vector of corresponding net center positions relative to the circle center $(s_{k(l)} - c_j)$ are nearly linearly dependent. It has a small magnitude, if they are nearly orthogonal. This is visualized in Figure 3.2. If both vectors are linear dependent, for the perfect rotation every pin position is on the connection from the net center to the circle center and the difference between the best rotation and the worst rotation is large. If both vectors are orthogonal, there is no particular good rotation and the difference between the best and the worst rotation is small.

The value b_j is the weighted distance from the circle center to the net centers. This is independent of the rotation and, of course, influences the wire length.

We observe that if we assume the net centers to be fixed, the wire length $\mathrm{wl}^j_{star}(z_j, \boldsymbol{s})$ only depends on the rotation of circle j. So we have decoupled the rotations from each other. Furthermore, we already know for given circle rotations and thus for given pin positions, how to choose the net centers for optimal wire length. Conversely, based on Lemma 2.18 we show that for given net centers we can analytically compute the optimal circle rotations in the strict and in the relaxed model.

Definition 3.15 (Wire Length of the Net Centers)**.** *We define the wire length for each circle and the total wire length depending on the net centers as*

$$\mathrm{wl}^{s,j}_{star}(\boldsymbol{s}) := \min_{z_j \in \partial\mathcal{U}} \mathrm{wl}^{j}_{star}(z_j, \boldsymbol{s}), \qquad \mathrm{wl}^{s}_{star}(\boldsymbol{s}) := \sum_{j=1}^{n} \mathrm{wl}^{s,j}_{star}(\boldsymbol{s}),$$

$$\mathrm{wl}^{r,j}_{star}(\boldsymbol{s}) := \min_{z_j \in \mathcal{U}} \mathrm{wl}^{j}_{star}(z_j, \boldsymbol{s}), \qquad \mathrm{wl}^{r}_{star}(\boldsymbol{s}) := \sum_{j=1}^{n} \mathrm{wl}^{r,j}_{star}(\boldsymbol{s}).$$

Lemma 3.16. *The function* $\mathrm{wl}^{s}_{star}(\boldsymbol{s})$ *is*

$$\mathrm{wl}^{s}_{star}(\boldsymbol{s}) = \boldsymbol{s}^H D \boldsymbol{s} + 2\Re(\boldsymbol{v}^T \boldsymbol{s} + v_0) + b - 2\sum_{j=1}^{n} \left| \boldsymbol{w}_j^H \boldsymbol{s} + d_j \right|$$

with

$$\boldsymbol{v} = (m_k \mu'_k)_{k=1,\dots,o}, \qquad v_0 = -\sum_{l=1}^{m} \mu'_{k(l)} \overline{c_{j(l)}},$$

$$D = \mathrm{diag}(\boldsymbol{v}), \qquad b = \sum_{l=1}^{m} \mu'_{k(l)} \left(|p_l|^2 + \left| c_{j(l)} \right|^2 \right),$$

$$\boldsymbol{w}_j = (\mu'_k \sum_{l \in \mathcal{P}^j \cap \mathcal{P}_k} \overline{p_l})_{k=1,\dots,o}, \qquad d_j = -\sum_{l \in \mathcal{P}^j} \mu'_{k(l)} \overline{p_l} c_j.$$

Proof. We compute the strict optimal rotation of circle j by Lemma 2.18 and get $\mathrm{wl}^{s,j}_{star}(\boldsymbol{s}) = a_j + b_j(\boldsymbol{s}) - 2\,|u_j(\boldsymbol{s})|$. With $m_k = |\mathcal{P}_k|$ being the number of pins in net k:

$$\begin{aligned}
\mathrm{wl}^{s}_{star}(\boldsymbol{s}) &= \sum_{j=1}^{n} \mathrm{wl}^{s,j}_{star}(\boldsymbol{s}) \\
&= \sum_{j=1}^{n} \left(\sum_{l \in \mathcal{P}^j} \mu'_{k(l)} (|p_l|^2 + \left| s_{k(l)} - c_j \right|^2) - 2 \left| \sum_{l \in \mathcal{P}^j} \mu'_{k(l)} \overline{p_l} \left(s_{k(l)} - c_j \right) \right| \right) \\
&= \sum_{l=1}^{m} \mu'_{k(l)} \,|p_l|^2 \quad + \sum_{l=1}^{m} \mu'_{k(l)} \left| s_{k(l)} \right|^2 - 2\sum_{l=1}^{m} \mu'_{k(l)} \Re(s_{k(l)} \overline{c_{j(l)}}) + \sum_{l=1}^{m} \mu'_{k(l)} \left| c_{j(l)} \right|^2 \\
&\quad - 2\sum_{j=1}^{n} \left| \sum_{l \in \mathcal{P}^j} \mu'_{k(l)} \overline{p_l} (s_{k(l)} - c_j) \right| \\
&= \sum_{k=1}^{o} m_k \mu'_k \,|s_k|^2 - 2\Re \left(\sum_{k=1}^{o} m_k \mu'_k s_k - \sum_{l=1}^{m} \mu'_{k(l)} \overline{c_{j(l)}} \right) + \sum_{l=1}^{m} \mu'_{k(l)} (|p_l|^2 + \left| c_{j(l)} \right|^2) \\
&\quad - 2\sum_{j=1}^{n} \left| \sum_{k=1}^{o} s_k \mu'_k \sum_{l \in \mathcal{P}^j \cap \mathcal{P}_k} \overline{p_l} - \sum_{l \in \mathcal{P}^j} \mu'_{k(l)} \overline{p_l} c_j \right| \\
&= \boldsymbol{s}^H D \boldsymbol{s} + 2\Re(\boldsymbol{v}^T \boldsymbol{s} + v_0) + b - 2\sum_{j=1}^{n} \left| \boldsymbol{w}_j^H \boldsymbol{s} + d_j \right|.
\end{aligned}$$

□

We can now evaluate the wire length depending only on the net centers, depending only on the circle rotation, or depending on both. The following Lemma 3.17 shows the intuitive fact that all three optimization methods are equivalent.

Lemma 3.17. *Let $X \subset \mathbb{F}^n$, $Y \subset \mathbb{F}^m$ and $f : X \times Y \to \mathbb{R}$. Set*

$$\begin{aligned} y(x) \in Y(x) &:= \operatorname*{argmin}_{y \in Y} f(x,y), & f_x &: X \to \mathbb{R}, & f_x(x) &:= f(x, y(x)) \\ x(y) \in X(y) &:= \operatorname*{argmin}_{x \in X} f(x,y), & f_y &: Y \to \mathbb{R}, & f_y(y) &:= f(x(y), y) \end{aligned}$$

where the sets $Y(x)$ and $X(y)$ are never empty. Then the following statements are equivalent

1. *f has a global optimum in (x_0, y_0).*
2. *f_x has a global optimum in x_0 and $y_0 \in Y(x_0)$.*
3. *f_y has a global optimum in y_0 and $x_0 \in X(y_0)$.*

Proof. We show $1 \iff 2$, as $1 \iff 3$ is similar.

As for $y_1, y_2 \in Y(x)$ it is $f(x, y_1) = \min_{y \in Y} f(x,y) = f(x, y_2)$, the function f_x is well defined.

Assume f_x has a global optimum in x_0 and $y_0 \in Y(x_0)$. Then it is for all $x \in X, y \in Y$:

$$f(x,y) \geq f(x, y(x)) = f_x(x) \geq f_x(x_0) = f(x_0, y(x_0)) = f(x_0, y_0).$$

Assume f has a global optimum in (x_0, y_0). Then by definition $y_0 \in Y(x_0)$ and for all $x \in X$ we have

$$f_x(x) = f(x, y(x)) \geq f(x_0, y_0) = f(x_0, y(x_0)) = f_x(x_0).$$

□

Recall that the rotation problem is denoted CR in the strict model and CRR in the relaxed model. Then we get the set of equivalent problems:

$$\begin{aligned} (CR) &\iff \min_{\boldsymbol{z} \in \partial \mathcal{U}^n} \operatorname{wl}(\boldsymbol{z}) \iff \min_{\substack{\boldsymbol{z} \in \partial \mathcal{U}^n \\ \boldsymbol{s} \in \mathbb{C}^o}} \operatorname{wl}_{star}(\boldsymbol{z}, \boldsymbol{s}) \iff \min_{\boldsymbol{s} \in \mathbb{C}^o} \operatorname{wl}^s_{star}(\boldsymbol{s}), \\ (CRR) &\iff \min_{\boldsymbol{z} \in \mathcal{U}^n} \operatorname{wl}(\boldsymbol{z}) \iff \min_{\substack{\boldsymbol{z} \in \mathcal{U}^n \\ \boldsymbol{s} \in \mathbb{C}^o}} \operatorname{wl}_{star}(\boldsymbol{z}, \boldsymbol{s}) \iff \min_{\boldsymbol{s} \in \mathbb{C}^o} \operatorname{wl}^r_{star}(\boldsymbol{s}). \end{aligned}$$

CRR is a convex problem and can be solved to optimality by standard non-linear solvers. We now compute the error of this relaxation.

Definition 3.18. *For given net centers $\boldsymbol{s}$, the circle j is* stretched, *if $|u_j(\boldsymbol{s})| \geq a_j$.*

Definition 3.19 (Relaxation Error)**.** *For net centers $\boldsymbol{s} \in \mathbb{C}^o$ the relaxation error is defined as $E(\boldsymbol{s}) := \mathrm{wl}^s_{star}(\boldsymbol{s}) - \mathrm{wl}^r_{star}(\boldsymbol{s})$.*

Theorem 3.20. *Assume a circle rotation setting with n circles, m pins and o nets. Denote for net centers $\boldsymbol{s} \in \mathbb{C}^o$ the circles that are not stretched by $\check{\mathcal{C}}$. Recall the definition of a_j and $u_j(\boldsymbol{s})$ from Theorem 3.14. Then for the relaxation error $E(\boldsymbol{s})$ we get*

$$E(\boldsymbol{s}) = \sum_{j \in \check{\mathcal{C}}} a_j \left(1 - \frac{|u_j(\boldsymbol{s})|}{a_j}\right)^2 \leq \sum_{j \in \check{\mathcal{C}}} a_j \leq \sum_{j \in \mathcal{C}} a_j.$$

Proof. By Lemma 2.19 it holds that

$$\begin{aligned} &\text{for } j \in \check{\mathcal{C}}: \qquad && \mathrm{wl}^{s,j}_{star}(\boldsymbol{s}) - \mathrm{wl}^{r,j}_{star}(\boldsymbol{s}) = a_j \left(1 - \frac{|u_j(\boldsymbol{s})|}{a_j}\right)^2 \leq a_j, \\ &\text{for } j \notin \check{\mathcal{C}}: \qquad && \mathrm{wl}^{s,j}_{star}(\boldsymbol{s}) - \mathrm{wl}^{r,j}_{star}(\boldsymbol{s}) = 0, \end{aligned}$$

and so the first equality and inequality follows. The last inequality is obvious. □

Corollary 3.21 (Approximation Result in the Star Model)**.** *Denote the optimal net centers for the relaxed model by $\boldsymbol{s}_r$ and for the strict model by $\boldsymbol{s}_s$. As (CRR) is a convex problem in the rotations, the optimal relaxed solution $\boldsymbol{s}_r$ can be computed by convex optimization and we have*

$$\mathrm{wl}^s(\boldsymbol{s}_s) \leq \mathrm{wl}^s(\boldsymbol{s}_r) = \mathrm{wl}^r(\boldsymbol{s}_r) + E(\boldsymbol{s}_r) \leq \mathrm{wl}^s(\boldsymbol{s}_s) + E(\boldsymbol{s}_r).$$

3.6 Numerical Results

3.6.1 Evaluation Settings

3.6.1.1 Algorithms

We implemented four algorithms. The non-linear algorithms were implemented in Ipopt (see Section 2.9). Complex numbers were split into real and imaginary part. The stop criteria were set to *tol* $= 10^{-5}$ and *acceptable_tol* $= 5 \cdot 10^{-5}$ and *max_iter* $=$ 50000. All other options were left at default.

Euclid The Euclid algorithm is the implementation of Definition 3.2 in real coordinates for Ipopt. Exact Jacobian and Hessian evaluation functions are implemented.

Angle The angle algorithm is the Ipopt implementation of the unconstrained minimization problem to minimize (3.5). Exact Jacobian and Hessian evaluation functions are implemented.

Star The star algorithm is the implementation of the unconstrained minimization problem to minimize wl^s_{star} defined in Lemma 3.16 in real coordinates for Ipopt. Exact Jacobian and Hessian evaluation functions are implemented. Note that at points with $u_j(\boldsymbol{s}) = 0$ this model is not differentiable. However, these points are no local optima and due to the numerical inaccuracies they do not occur in the practical algorithms.

BCD The block coordinate descent algorithm is described in Section 3.4.3. The stopping criterion is that during the last n single circle rotations the total improvement is less than 10^{-4}.

3.6.1.2 Problem Instances

For each of the Dill circuits in Section 2.8, random and possibly overlapping placements of the circles were generated. The generation algorithm was as follows.

For each circuit instance denote as A the sum of the area of all circles. Furthermore, choose an area factor $a_f \geq 0$. Then all circle centers c_i are chosen independently uniformly distributed from the square with area $a_f A$ located in the positive quadrant with one corner in the origin, i.e. $\Re(c_i)$ and $\Im(c_i)$ were chosen independently uniformly distributed from $[0, \sqrt{a_f A}]$. So by the area factor a_f the amount of overlapping is controlled. While for small a_f the circles are very likely to overlap, for larger a_f the circles are less likely to overlap. Such a randomly placed circuit instance is denoted as problem instance.

For our numerical evaluation we chose the area factors

$$a_f \in \{0,\ 0.001,\ 0.002,\ 0.005,\ 0.01,\ 0.02,\ 0.05,\ 0.1,\ 0.2,\ 0.5,\ 1,\ 2,\ 5,\ 10\}.$$

We could also think of using rectangles with different aspect ratios instead of squares. We compared the algorithm for different aspect ratios (from 0.1 to 1), however there was only little, if any, influence on the results and performance of all analyzed algorithms. So we omit this degree of freedom in the problem instances for the numerical evaluation.

3.6.1.3 Evaluation Method

For each of the problem instances we generated 100 feasible, random starting points by taking the circle rotation angles φ_j, $j \in \mathcal{C}$ independently uniformly distributed in $[0, 2\pi)$. Based on this, for Euclid and BCD the starting vectors were defined by setting the initial rotations to $\exp(\imath\varphi_j)$, $j \in \mathcal{C}$. For the star algorithm, the optimal net centers were computed based on the randomly chosen circle rotations and taken as starting point.

Then each of the algorithms was run for each of the 100 starting points. We measured CPU time, number of iterations, the objective value and the final solution $\boldsymbol{\varphi}$ as vector of rotation angles. Some algorithms have very short running times for small problem instances. To avoid inaccuracies in the CPU time, we ran an algorithm multiple times per problem instance and initial solution, until at least 1 CPU second was used. By dividing the total CPU time through the number of runs, the average time per run could be measured with higher precision. As the algorithms are not randomized, all these repeated runs yield the same number of iterations and equal objective values and solution vectors.

As we do not compute lower bounds in this chapter, we compare the solutions with the best known solutions for the problem instance. In Chapter 4 we show, that except for very small area factors, these best known solutions are globally optimal. We sometimes refer to the best known solution as optimal solution, although for small area factors it is not proved to be globally optimal.

3.6.1.4 Symmetry Breaking

For $a_f = 0$ the problem is rotation symmetric, i.e. rotating all circles by the same angle does not change the solution. When analyzing the number of local optima, we want to avoid this symmetry.

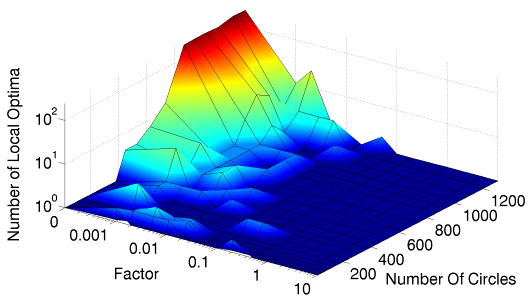
The first idea is to rotate all circles, such that the rotation angle of the first circle is zero. However, it turns out that some circle rotations do not influence some other rotations, although the circuit is connected. This is due to the fact that some pins have zero offset, i.e. their position is independent of the circle rotation.

Because of this, the connectivity of the circles is analyzed, and circles that influence each other are identified. After the algorithm run, the rotations are normalized for each connectivity group, i.e. the circles of each group are rotated such that the first circle of each group of connected circles has rotation angle zero. For a detailed analysis of the symmetry breaking see Section 4.3.

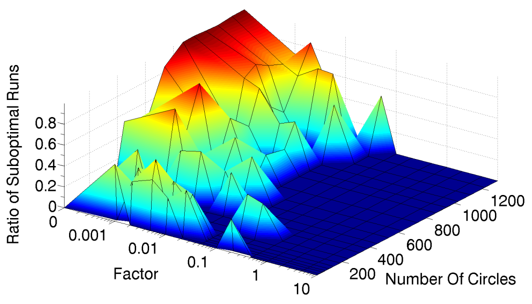
3.6.2 Analyzing the Problem Structure

First we present a numerical evaluation of the problem structure. As we show later, the algorithms do behave not too differently, so we unified the results of all algorithms for showing the problem structure. This means, for each problem instance there were 400 optimization runs (each of the 4 algorithms for 100 random start points).

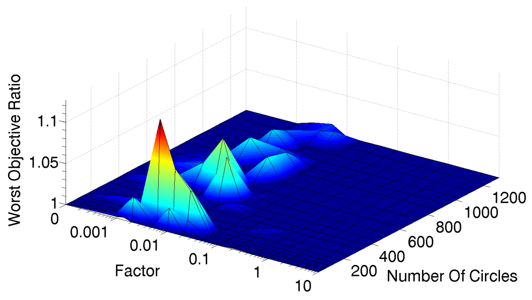
In Figure 3.3 we show for each problem instance the number of discovered local optima, the ratio of suboptimal solutions found and the ratio of the worst objective divided by the best objective. Note that the axis of the area factors is a regularized logarithmic scale (such that $a_f = 0$ could be shown).



(a) Number of optimal solutions.



(b) Ratio of runs yielding suboptimal results.



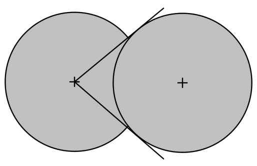
(c) Worst objective ratio.

Figure 3.3: Structure of the circle rotation problem.

Figure 3.3a shows the number of optimal solutions found during the 400 runs. The decision, if two solutions are equal, cannot be made based on the objective function value. So the solution angle vectors were clustered, such that in no cluster the maximum norm distance of any two solution vectors is greater than 0.01. The number of clusters then was declared as the number of local optima. Note that the axis of the local optima is also a logarithmic scale.

Figure 3.3b presents the ratio of suboptimal solutions. Based on the clustering described above, the minimal clusters are the clusters which contain a solution with minimal objective. We then say a solution is optimal if it belongs to such a cluster, even if, due to numerical issues, the objective value of this solution is slightly worse than the optimal solution. Subsequently, the number of suboptimal solutions is the number of solutions not contained in these clusters.

In Figure 3.3c the ratio of the worst solution divided by the best solution for each problem instance is shown. This is likely to be (very close to) the ratio between the worst found solution and the optimal solution.


(a) Small distance.

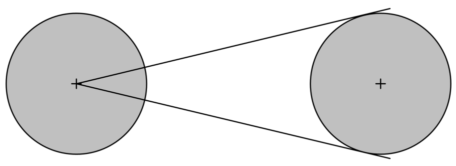

(b) Large distance.

Figure 3.4: Great distance removes the impact of the circle rotation. If the connected circle is far away, the interval of possible optimal rotations becomes small.

To analyze these results, recall that by Lemma 3.1 the wire length is

$$\begin{aligned} \mathrm{wl}(\boldsymbol{z}) &= b + 2\Re(\boldsymbol{u}^H \boldsymbol{z}) + \boldsymbol{z}^H A \boldsymbol{z}, \\ \text{with} \quad b &:= \boldsymbol{c}^H \Phi^T Q \Phi \boldsymbol{c}, \\ \boldsymbol{u} &:= \Phi^T P^H Q \Phi \boldsymbol{c}, \\ A &:= \Phi^T P^H Q P \Phi. \end{aligned}$$

We recognize that for small area factors the number of local optima is large. In fact, for $a_f = 0$ and a large number of circles, almost every run converged to a different solution. For a small number of circles the number of local optima for small positive values of a_f is larger than for $a_f = 0$. This seams plausible, as for small area factors all circles overlap. However, when the area factor is large, the circles are spread very far. This means, each circle is connected to other circles far away. The further away another connected circle is, the less important is its rotation. This is visualized in Figure 3.4 and can also be seen in the formula for the wire length. The greater the area factor a_f is, the larger is the magnitude of the components of $\boldsymbol{c}$ and $\boldsymbol{u}$ and the value of b. However, if the entries of $\boldsymbol{u}$ are large compared to the entries of A, the optimal solution is close to the minimum of $\Re(\boldsymbol{u}^H \boldsymbol{z})$. So the number of local optima reduces.

The ratio of suboptimal solutions is strongly related to the number of local optima. This seems obvious, as if there are more local optima, the algorithm is more likely to converge to a non-global optimum.

Quality of the Worst Local Optimum

Very interesting is the ratio of the worst objective divided by the best objective shown in Figure 3.3c. For small area factors, especially for $a_f = 0$, there are many local optima, but the objective values of these local optima are very close to the best objective value found. This can be seen in more detail in Figure 3.5. Also for large area factors, this quotient is either exactly one (if only one local optimal solution was found) or close to one, i.e. the non-global optima were almost as good as the best found solution. However, for area factors close to $a_f = 0.005$ this ratio becomes up to 1.13. While the exact value of this ratio depends on the random circle distribution, the tendency is the same for all number of circles.

Considering problem instances with large area factor (e.g. greater than 0.1), we observe that there are few local optima. And if there are local optima, their objective is nearly as good as the best known solution.

First let us explain this behavior geometrically. As the circles are spread, the optimal rotation of a circle does not depend heavily on the rotation of its connected circles, see Figure 3.4. So it is likely that there is no non-global local optimum. However, even if there is a non-global optimum, comparing the solutions revealed that most of the circle rotations are almost equal. In fact, there are only few circles that have a very different rotation from the global optimum. This means, a small subgroup of circles is rotated poorly while most of the circles are rotated nearly optimal. However, then the impact of the poorly rotated circles on the total objective is small.

This can also be seen by looking at the wire length function. The greater the area factor is, the larger become the entries of $\boldsymbol{c}$ and thus the larger becomes the value b. However, the entries of A do not change at all, while the entries of $\boldsymbol{u}$ do not increase as fast as b. So the constant part of the wire length is dominating, e.g. the total impact of the rotation is less than for smaller area factors. This explains the small values in Figure 3.3c even if there are local optima.

Now consider problem instances with small area factor (e.g. smaller than 0.002). There are many local optima, but their objective value is almost equal. Further analysis of the local optimal solution shows that for different local optimal solutions most of the circle rotations are almost equal. Only some rotations are considerably different. However, their impact on the total wire length is small.

Finally, we look at instances with area factor in between, e.g. $a_f = 0.005$. There are some local optima and the worst of them is significantly worse than the best solution. Analyzing different optima shows that many rotations are considerable different. It seems that the diversification of the centers has enough impact such that bad rotations significantly influence other circle rotations. But the impact is too small to completely determine most of the circle rotations. Furthermore, the constant part of the wire length is not large enough to absorb the impact of bad circle rotations.

3.6.3 Algorithm Comparison

Objective Values

First, in Figure 3.5 we compare the results of the algorithms. For all problem instances the objectives were divided by the best objective value found by any of the algorithms. On the y-axis, this ratio is shown. On the x-axis, the percentage of runs is stated. So if at 75% for an algorithm the graph shows a value of 1.1 this means that in 75% of the runs the algorithm is better then 1.1 times the best known solution.

Again we recognize the pattern described in Section 3.6.2. The algorithms tend to find suboptimal solutions more often for small area factors, but the objective value of the suboptimal solution is worst for $a_f = 0.005$. Note that there are suboptimal solutions not visible in Figure 3.5, as there objective is too close to the best known objective.

In most cases (except $a_f = 0$) the star algorithm is slightly inferior to the other algorithms, which are very close to each other. Also the block coordinate descent method seems to be slightly worse than Euclid and star algorithm for $a_f > 0$. However, all algorithms find optimal or close-to-optimal solutions very often.

CPU Time and Number of Iterations

We now consider the running time and the number of iterations required by the algorithms to converge to a local optimum.

While for the angle, Euclid and star algorithm the definition of an iteration is obvious, this is more complicated for the block coordinate descent method. We define an iteration as the step from $\boldsymbol{x}^t$ to $\boldsymbol{x}^{t+1}$ as described in Section 2.6. So in each iteration, each circle is rotated once. However, the algorithm is stopped, when the improvement during the last n single circle rotations is smaller than some predefined threshold. This can be in the middle of an iteration. So we define the number of iterations as the number of single circle rotations divided by the number of circles. This might be a fractional number.

In Figure 3.7 the median of the number of iterations and the median of the CPU time in milliseconds are shown for exemplarily chosen area factors. For $a_f > 1$ the diagrams remain similar to $a_f = 1$. In Figure 3.6 the median of the number of iterations and the median of the CPU time in milliseconds are shown with respect to all problem instances with the specified area factor.

In Figures 3.8 to 3.11 detailed profiles of each of the algorithms are given. For each algorithm the median CPU time and the median number of iterations are shown as a diagram over the number of circles and area factors. Furthermore, quantiles for the CPU time over the number of circles for some area factors are shown. Finally, the quantiles of the CPU time over the area factor are shown.

We first look at the CPU time in the Figures 3.6 and 3.7. Obviously, the star algorithm is by far the worst of the four algorithms. Especially note the break in the scale on the y-axis. Nevertheless, the performance gap to the other algorithms becomes smaller, as the area factor increases. The best algorithm in terms of CPU time is the block coordinate descent method, followed by the angle algorithm and the Euclid algorithm. As expected, the running time of all algorithms increases with the number of circles. The absolute distance between these algorithms increases with the number of circles,

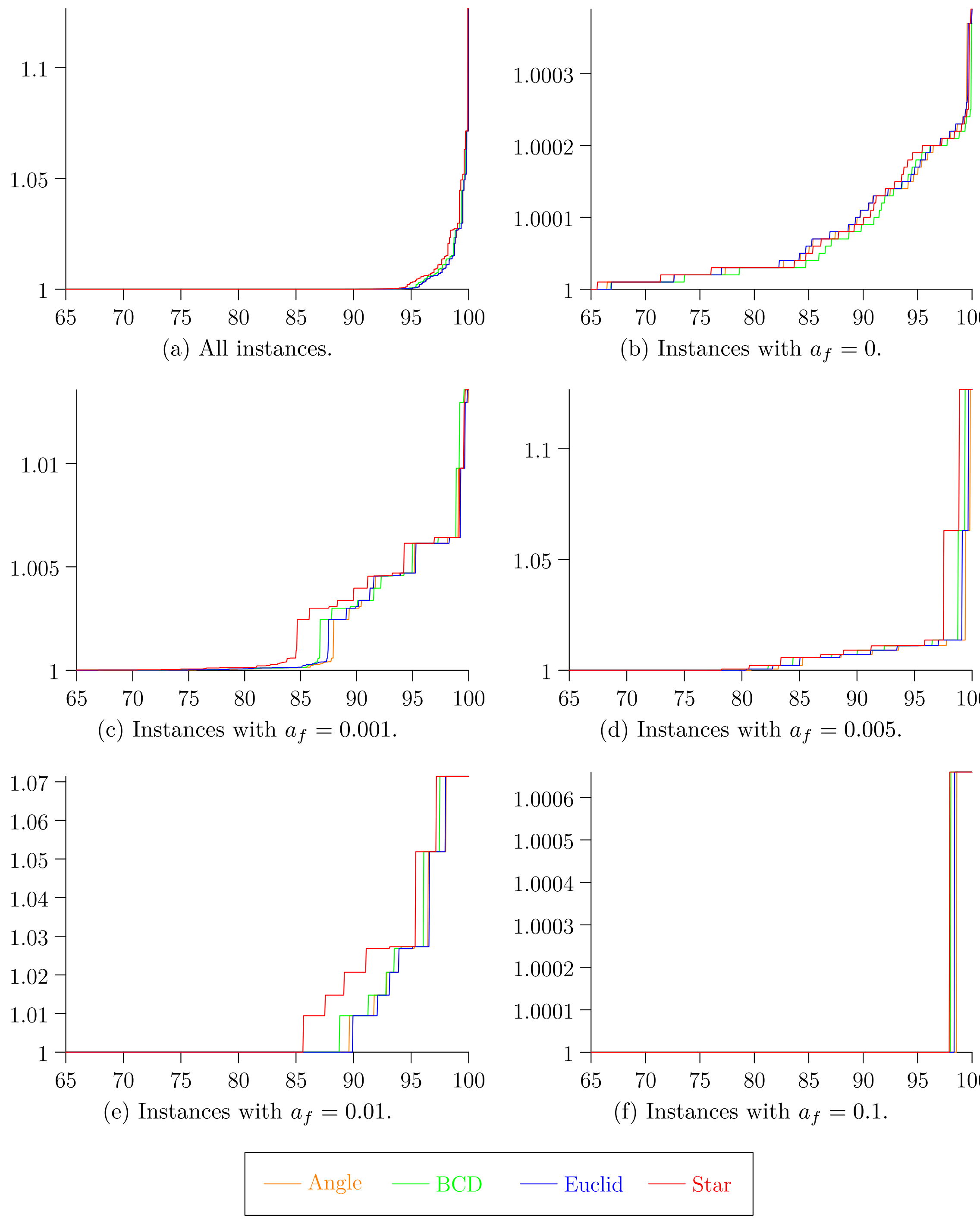


Figure 3.5: Objective distribution for different area factors. On the x-axis the percentage of the runs is stated, on the y-axis the objective ratio between the ratio of the objective and the best objective is shown.

but remains approximately equal for different area factors. But this is only the case as the block coordinate descent method is very fast compared to the other algorithms. In the Figures 3.9a and 3.9f it can be seen that its running time increases, if the area factor gets small.

We now consider the number of iterations in the Figures 3.6 and 3.7. For $a_f = 0$, the block coordinate descent method takes many iterations and the number of iterations increases with the number of circles. All other algorithms need less iterations and the increase depending on the number of circles is smaller. The Euclid algorithm requires more iterations than the angle algorithm, which requires more iterations than the star algorithm. For greater area factors, the behavior of the block coordinate descent method changes: For $a_f = 0.002$ BCD requires less iterations, but the number of iterations is very volatile. The number of iterations of the other algorithms are similar to the case $a_f = 0$, although the gap between angle and star algorithm increases. Finally, for $a_f = 1$ BCD takes by far the fewest iterations, followed by the star algorithm. Euclid and Angle algorithm require more iterations. Furthermore, their number of iterations increases with the number of circles, while this is constant for BCD and the star algorithm.

Comparing these diagrams, we observe that the iterations of the block coordinate descent method are very fast. Even for $a_f = 0$ where the algorithm takes more iterations than the other algorithms, it is much faster. Conversely, an iteration of the star algorithm is very slow. Even for $a_f = 1$ where it requires less than 10 iterations, it takes much more time to converge than the other algorithms. In fact, the BCD is always superior to all other algorithms in running time. The angle algorithm seems to be a pretty good non-linear model, as it is always second fastest after BCD.

In the following paragraphs we have a closer look at the different algorithms.

Consider the star algorithm in Figure 3.8. For small area factors, the CPU time and the number of iterations is growing approximately linear depending on the number of circles. For larger area factors the CPU time does not increase that fast. In fact, in Figure 3.6b it can be seen that the number of iterations remains almost constant. Hence, the gap to the other algorithms gets smaller for increasing area factors, see Figure 3.6. The long running time is due to the wire length formula. Especially computing the Jacobin and Hessian is more expensive for the star wire length model than for the clique wire length used in the Euclid or angle algorithm. Furthermore, Table 2.1 on page 28 shows that in particular for small instances the number of nets is significantly larger than the number of components. Hence, the number of variables in the star algorithm is larger than the number of variables in the other algorithms.

We now analyze the block coordinate descent method visualized in Figure 3.9. Its CPU time increase depending on the number of circles is approximately linear. For larger area factors, the running time dependency on the starting point decreases. So in Figure 3.9e the difference between the best and the worst running time is very small. Remarkable is the poor worst case behavior for $a_f = 0$ as shown in Figure 3.9c and

Figure 3.9f. Recall that the block coordinate descent method can jump between local optima. It is likely that for $a_f = 0$, as there are many local optima, the algorithm jumps very often. However, after such a jump the rotation of the other circles has to be adapted, thus the algorithm takes longer to converge. As the area factor increases, the number of local optima decreases, thus such a jump is less likely and the worst case performance improves.

Now we consider the angle algorithm in Figure 3.10. This algorithm is reasonable fast and very robust. Its running time grows approximately linearly with the number of circles for all area factors. Indeed, Figures 3.10a, 3.10b and 3.10f show that neither running time nor iterations heavily depend on the area factor. Also the difference between worst case behavior and average or best case behavior is small. The performance independence of the area factor is a clear difference to the BCD, but can be easily explained. While the BCD jumps between local optima, the angle algorithm just converges to the “closest” local optimum. It does not jump but behaves continuously. So it is not influenced by possibly many other local optima.

Finally, look at the Euclid algorithm in Figure 3.11. This algorithm is similar to the angle algorithm. Its objective is a quadratic form, however it includes one nonlinear equality constraint per circle. For most of the cases it behaves similar to the angle algorithm. However, sometimes it has convergence problems and takes orders of magnitude more time to converge. In such runs the algorithm does not make progress for 10000 iterations ore more, until it suddenly converges again. Although we cannot really explain this behavior, it seems to be some artificial behavior of Ipopt. However, also for cases where the algorithm converges normally, it is slower and takes more iteration than the angle algorithm.

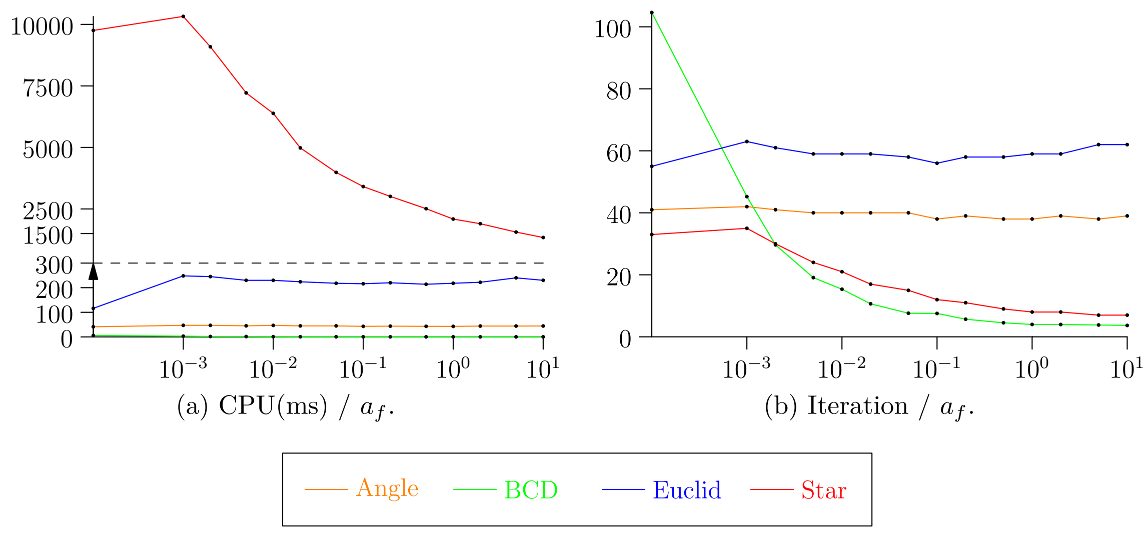


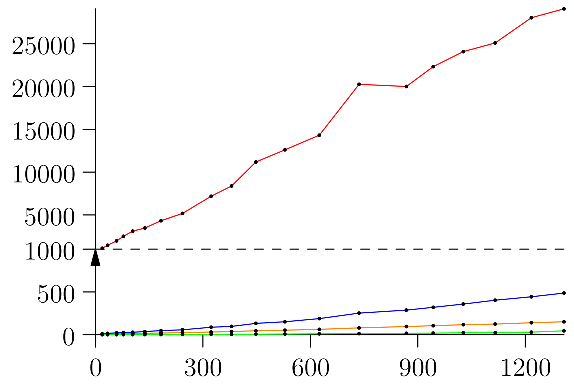
Figure 3.6: Median of CPU time and number of iterations. In both graphs on the x-axis the area factor is stated. In (a) on the y-axis the median of the CPU time in milliseconds for one run is stated. Note that the scale of the y-axis changes for 300ms. The median is taken over all number of circles, as the basic behavior remains approximately equal for all number of circles. In (b) on the y-axis the median number of iterations is shown.

3.6.4 Conclusion

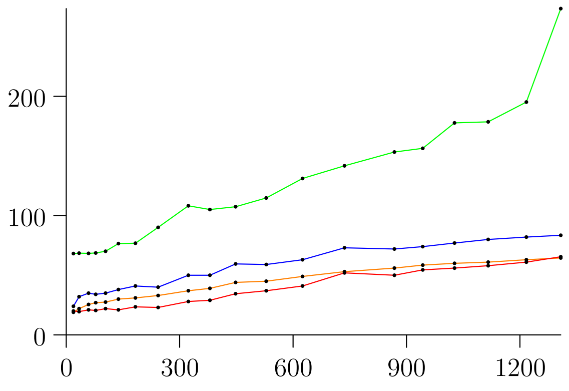
In this section we analyzed the local algorithms for the circle rotation problem numerically. We showed that the solution quality of all algorithms is similar and that the local algorithms yield close to optimal solutions in most cases.

For instances with small area factors, the algorithms rarely reach the global optimum exactly, however all local optimal objective values are close to the best known objective. For large area factors, the algorithms always find the optimal solution.

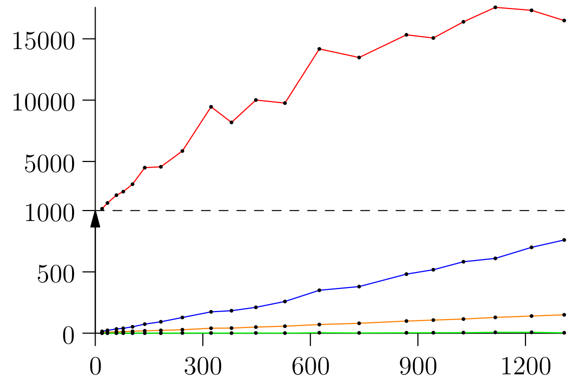
In terms of running time, the block coordinate descent method outperforms the other algorithms by orders of magnitudes. The best formulation to be used in non-linear programs is the angle rotation, as it is significantly better than the Euclidean rotation. The star model is substantially worse than all other models.



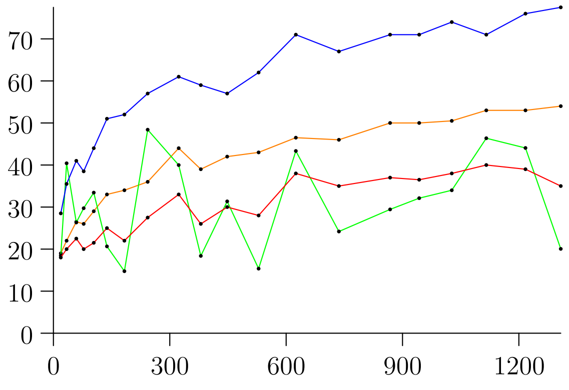
(a) CPU(ms) / #circles, $a_f = 0$.



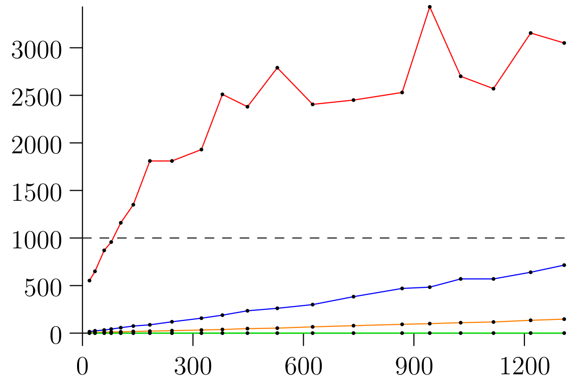
(b) Iteration / #circles, $a_f = 0$.



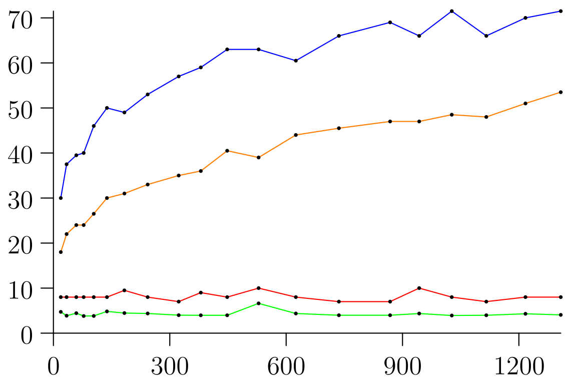
(c) CPU(ms) / #circles, $a_f = 0.002$.



(d) Iteration / #circles, $a_f = 0.002$.



(e) CPU(ms) / #circles, $a_f = 1$.

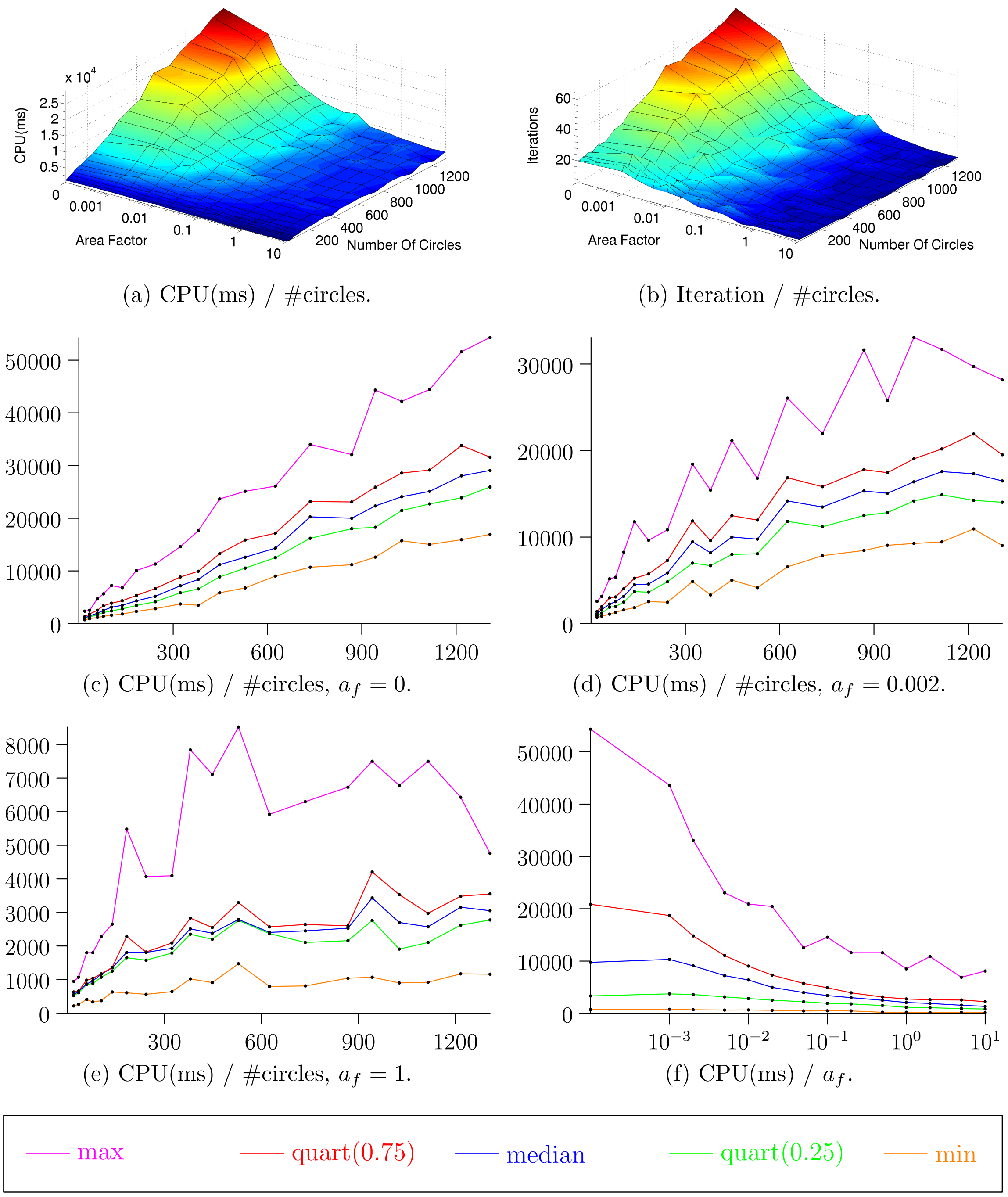


(f) Iteration / #circles, $a_f = 1$.

— Angle — BCD — Euclid — Star

Figure 3.7: Median of CPU time and number of iterations. On the x-axis the number of circles is stated. On the y-axis the median of the CPU time in milliseconds for one run is stated. Note that the scale of the y-axis changes in (a) to (c) at 1000ms. These plots are chosen examples for the typical behavior, for $a_f > 1$ they remain similar to $a_f = 1$.

Star Algorithm



(a) CPU(ms) / #circles.

(b) Iteration / #circles.

(c) CPU(ms) / #circles, $a_f = 0$.

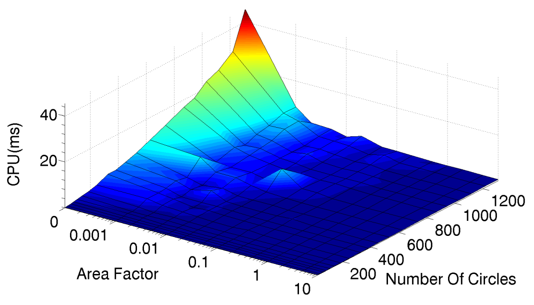
(d) CPU(ms) / #circles, $a_f = 0.002$.

(e) CPU(ms) / #circles, $a_f = 1$.

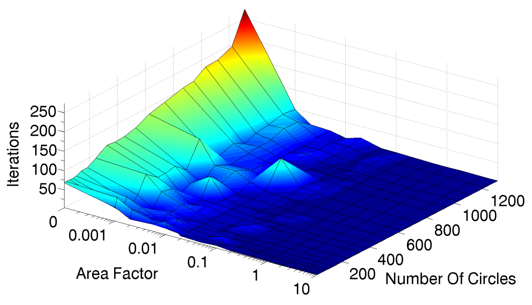
(f) CPU(ms) / a_f.

Figure 3.8: Performance profile of the star algorithm. In (c), (d) and (e) on the x-axis the number of circles is stated. In (f) on the x-axis the area factor is shown. On the y-axis the quantiles of the CPU time in milliseconds are stated.

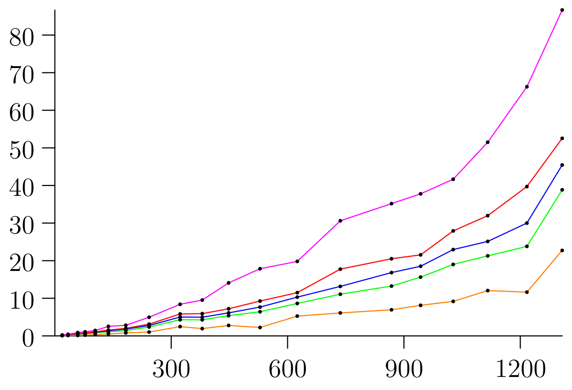
Block Coordinate Descend Method



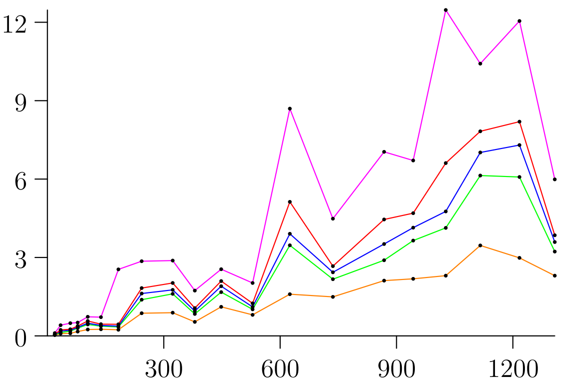
(a) CPU(ms) / #circles.



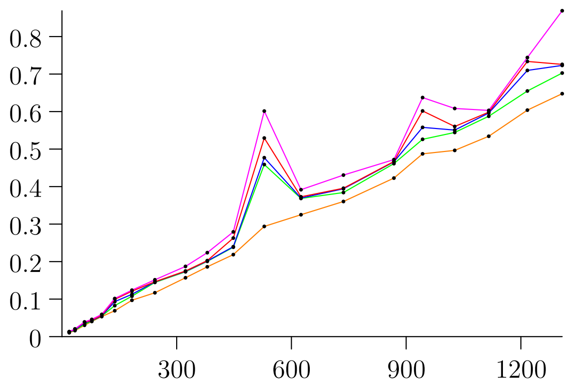
(b) Iteration / #circles.



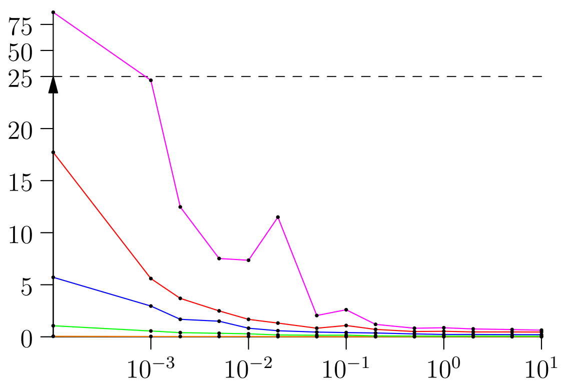
(c) CPU(ms) / #circles, $a_f = 0$.



(d) CPU(ms) / #circles, $a_f = 0.002$.



(e) CPU(ms) / #circles, $a_f = 1$.



(f) CPU(ms) / a_f.

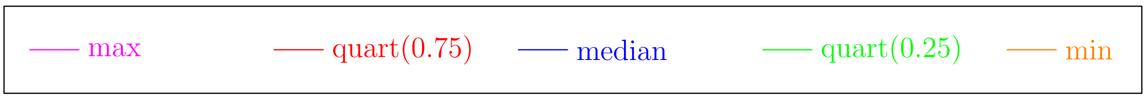
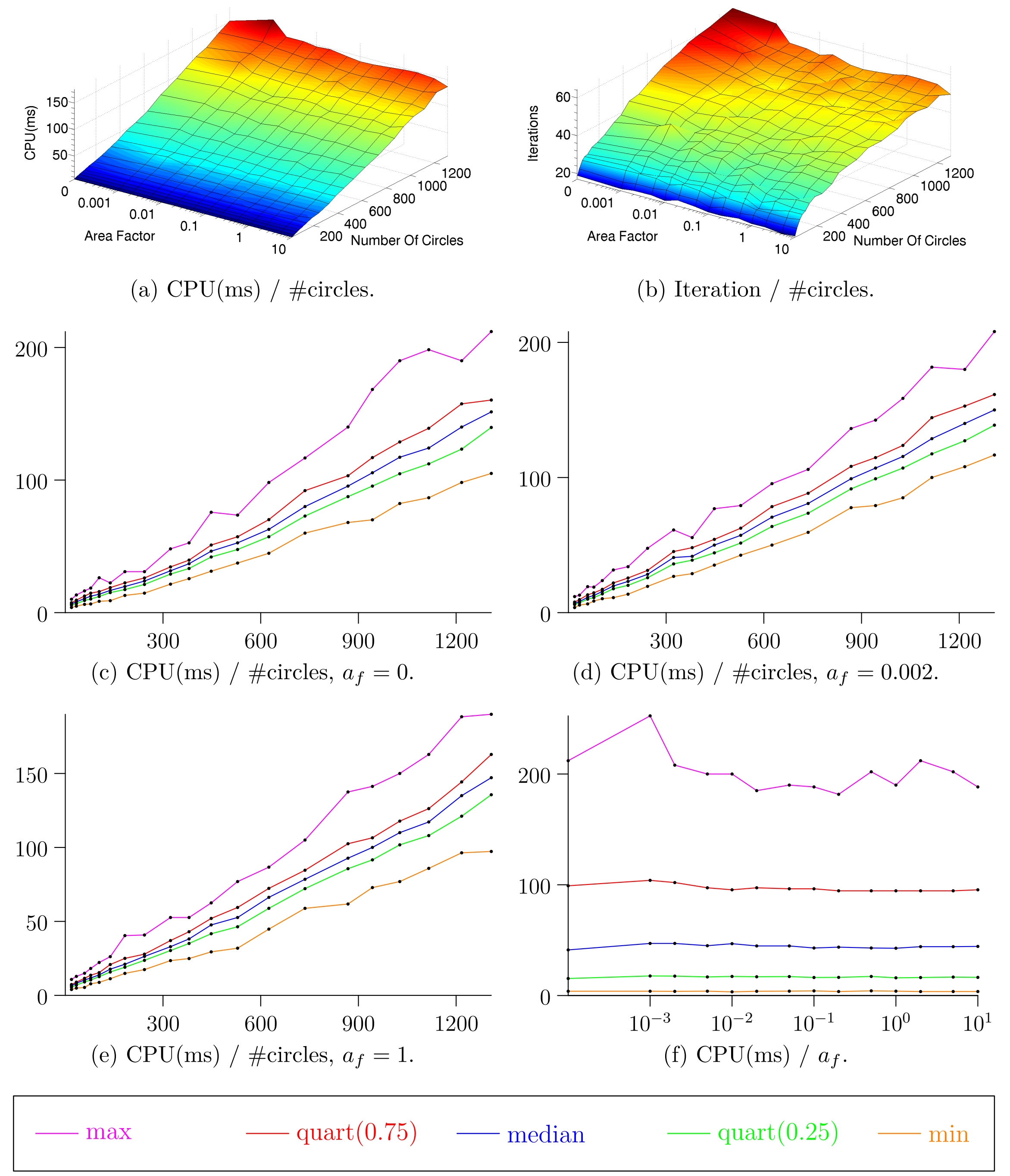


Figure 3.9: Performance profile of the block coordinate descent method. In (c), (d) and (e) on the x-axis the number of circles is stated. In (f) on the x-axis the area factor is shown. On the y-axis the quantiles of the CPU time in milliseconds are stated. Note the scale of the y-axis changes at 25 in (f).

Angle Algorithm



(a) CPU(ms) / #circles.

(b) Iteration / #circles.

(c) CPU(ms) / #circles, $a_f = 0$.

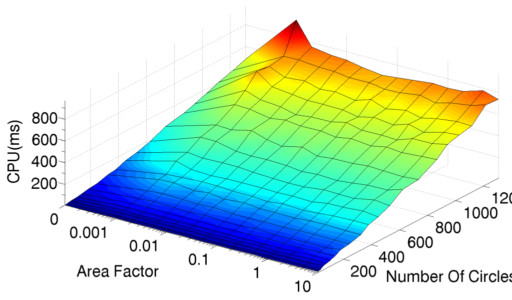
(d) CPU(ms) / #circles, $a_f = 0.002$.

(e) CPU(ms) / #circles, $a_f = 1$.

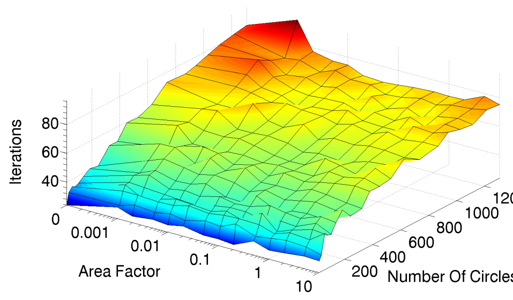
(f) CPU(ms) / a_f.

Figure 3.10: Performance profile of the angle algorithm. In (c), (d) and (e) on the x-axis the number of circles is stated. In (f) on the x-axis the area factor is shown. On the y-axis the quantiles of the CPU time in milliseconds are stated.

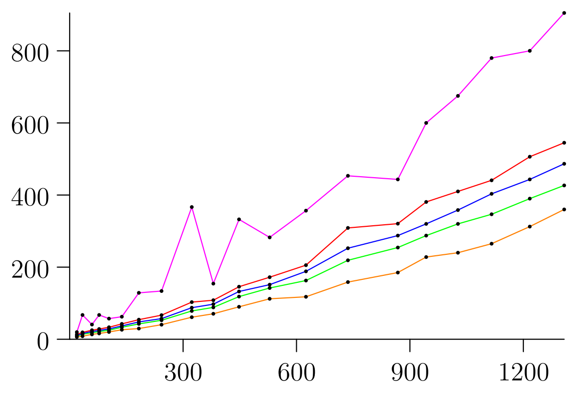
Euclid Algorithm



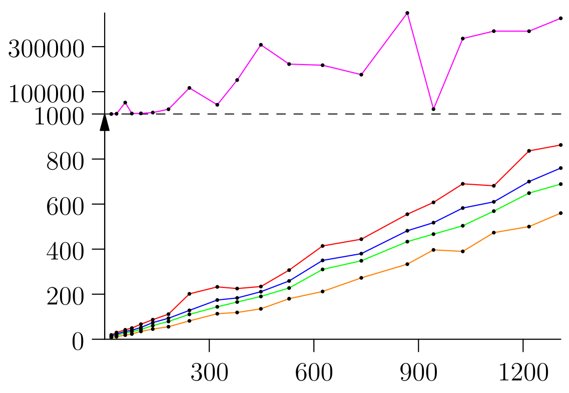
(a) CPU(ms) / #circles.



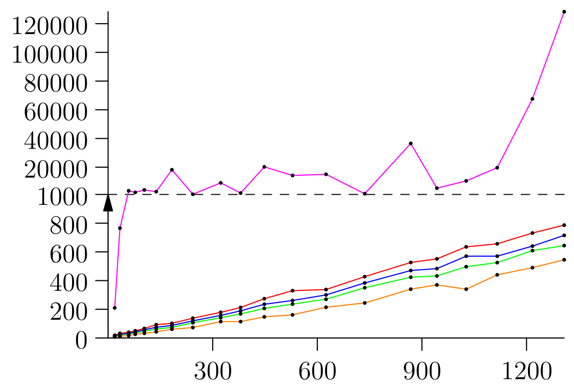
(b) Iteration / #circles.



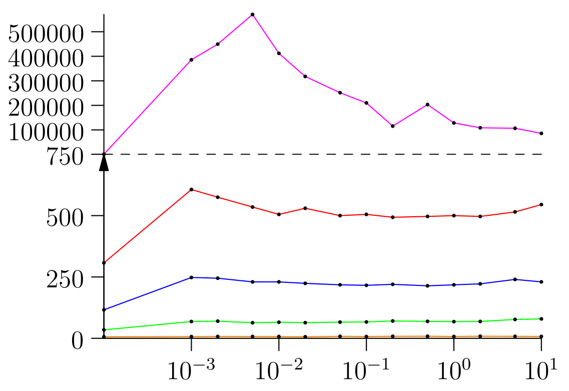
(c) CPU(ms) / #circles, $a_f = 0$.



(d) CPU(ms) / #circles, $a_f = 0.002$.



(e) CPU(ms) / #circles, $a_f = 1$.



(f) CPU(ms) / a_f.

max — quart(0.75) — median — quart(0.25) — min

Figure 3.11: Performance profile of the Euclid algorithm. In (c), (d) and (e) on the x-axis the number of circles is stated. In (f) on the x-axis the area factor is shown. On the y-axis the quantiles of the CPU time in milliseconds are stated. Note the break of the y-axis-scale in (d), (e) and (f).

4 Global Optimization of the Circle Rotation Problem

In this chapter we consider global solution algorithms for the circle rotation problem stated in Chapter 3. For a set of fixed centered connected circles, the circles have to be rotated such that the wire length is minimized.

Semidefinite programming approaches for the Hermitian minimization problem equivalent to the circle rotation problem have been studied in the literature. We proposed and analyzed local optimization algorithms for the circle rotation problem in Chapter 3. However, we have not presented an algorithm to solve this problem to global optimality.

In Section 4.1 we reformulate the wire length minimization problem as Hermitian minimization problem. Furthermore, we show that it is also equivalent to an indefinite quadratic optimization problem on a convex domain. We survey solution methods known in the literature for this problem.

In Section 4.2 we introduce a branch and bound algorithm for the circle rotation problem. In the remaining part of the chapter, the main elements of this algorithm are explained and analyzed. In Section 4.3 we explain symmetry breaking techniques to reduce the number of branches. In Section 4.4 we state the branching schemes used within the algorithm.

Based on computational geometry, we develop a novel domain reduction algorithm in Section 4.5. This algorithm reduces the number of branches that correspond to areas not containing an optimal solution.

In Section 4.6 and Section 4.7 we show how to compute the lower and upper bounds efficiently by the block coordinate descent method. For the lower bound evaluation, we prove that the Lagrange multipliers of the subproblems of the block coordinate descent method converge to the Lagrange multipliers of the global optimum.

In a detailed analysis in Section 4.8 we show that our branch and bound algorithm efficiently solves practical problem instances to global optimality. In particular, we show that our domain reduction algorithm often assures global optimality without branching and improves the running time of the algorithm by orders of magnitudes.

4.1 Quadratic Optimization Problem

We know by Theorem 3.6 that minimizing the wire length in the clique model is equivalent to computing $\min\{\boldsymbol{z}^H H \boldsymbol{z} : \boldsymbol{z} \in \partial\mathcal{U}^n\}$ with some positive semidefinite Hermitian matrix $H \in \mathbb{C}^{n\times n}$ and $\partial\mathcal{U} = \{z \in \mathbb{C} : |z| = 1\}$ as defined in Definition 2.6. Define

$$f(\boldsymbol{z}) = \boldsymbol{z}^H H \boldsymbol{z} = \sum_{i,j} h_{ij}\overline{z_i}z.$$

Definition 4.1. *We define f_k as the function f in the variable z_k, when all other variables are fixed. With $\pi_k(\boldsymbol{z}) = (z_1, \dots, z_{k-1}, z_{k+1}, \dots, z_n)^T$ as in Definition 2.5 we get*

$$f_k(\cdot, \pi_k(\boldsymbol{z})) : \mathbb{C} \to \mathbb{R}, \qquad f_k(\xi, \pi_k(\boldsymbol{z})) = f(z_1, \dots, z_{k-1}, \xi, z_{k+1}, \dots, z_n).$$

As $f(\boldsymbol{z}) = \Psi[H, 0, 0]$ with Φ from Definition 2.20 by Lemma 2.22 it is

$$f_k(\cdot, \pi_k(\boldsymbol{z})) = \psi\left[h_{kk}, \sum_{j\neq k} h_{kj}z_j, \sum_{i,j\neq k} h_{ij}\overline{z_i}z_j\right]. \tag{4.1}$$

4.1.1 Formulation as Indefinite Quadratic Optimization Problem

Recall that CR from Definition 3.2 denotes the circle rotation problem and HM from Definition 3.3 denotes the problem with positive semidefinite matrix H to find $\min_{z\in\partial\mathcal{U}} \boldsymbol{z}^H H \boldsymbol{z}$. By Theorem 3.6 we have shown that these problems are essentially equivalent, i.e. equivalent up to a constant. We now construct another formulation that can be used to solve HM or CR.

4.1.2 Problem Reformulation

Definition 4.2 (Indefinite Hermitian Minimization (IHM))**.** *The indefinite Hermitian minimization problem IHM is: For a (possibly indefinite) Hermitian matrix H find $\min_{z\in\mathcal{U}^n} \boldsymbol{z}^H H \boldsymbol{z}$.*

In contrast to HM, here the objective function is not necessarily convex while the domain is convex. We now show that IHM and HM are essentially equivalent.

Theorem 4.3. *Let $H \in \mathbb{C}^{n\times n}$ be a Hermitian matrix with zero diagonal entries. Then there is a global minimizer $\boldsymbol{z}^*$ of $f(\boldsymbol{z}) = \boldsymbol{z}^H H \boldsymbol{z}$ in $\mathcal{U}^n$ with $|z_i^*| = 1$, $i = 1, \dots, n$.*

Proof. For a feasible solution $\boldsymbol{z} \in \mathcal{U}^n$ set $J(\boldsymbol{z}) = \{j : |\tilde{z}_j| < 1\}$. Let $\tilde{\boldsymbol{z}}$ be a global minimizer of f with minimal number of elements in $J(\tilde{\boldsymbol{z}})$. If $J(\tilde{\boldsymbol{z}}) = \emptyset$ we are done, as then $\boldsymbol{z}^* = \tilde{\boldsymbol{z}}$ satisfies the statement.

Otherwise let $k \in J(\tilde{\boldsymbol{z}})$, i.e. $|\tilde{z}_k| < 1$. Consider the other values of $\tilde{\boldsymbol{z}}$ being fixed. Then $f_k(\xi, \pi_k(\tilde{\boldsymbol{z}})) = \psi[0, u, b](\xi) = 2\Re(\xi\overline{u}) + b$ with

$$u = \sum_{j \neq k} h_{kj}\tilde{z}_j, \qquad b = \sum_{i,j \neq k} h_{ij}\overline{\tilde{z}_j}\tilde{z}_j.$$

Denote the minimum of f by f^* and the minimum of $f_k(\cdot, \pi_k(\boldsymbol{z}))$ by f_k^*. Then $f^* \leq f_k^*$ and $\tilde{z}_k$ is a minimizer of $f_k(\cdot, \pi_k(\tilde{\boldsymbol{z}}))$ as $f_k(\tilde{z}_k, \pi_k(\tilde{\boldsymbol{z}})) = f(\tilde{\boldsymbol{z}}) = f^*$. By Lemma 2.18 in the case $a = 0$ a minimizer of $f_k(\cdot, \pi_k(\tilde{\boldsymbol{z}}))$ is either arbitrary (if $u = 0$) or $-\mathrm{Normed}\,[u]$, especially there is a minimizer z'_k of $f_k(\cdot, \pi_k(\tilde{\boldsymbol{z}}))$ with $|z'_k| = 1$. Set $\tilde{\boldsymbol{z}}' = (\tilde{z}_1, \ldots, \tilde{z}_{k-1}, z'_k, \tilde{z}_{k+1}, \ldots, \tilde{z}_n)^T$. Then

$$f^* \leq f(\tilde{\boldsymbol{z}}') = f_k(z'_k, \pi_k(\tilde{\boldsymbol{z}})) \leq f_k(\tilde{z}_k, \pi_k(\tilde{\boldsymbol{z}})) = f(\tilde{\boldsymbol{z}}) = f^*.$$

Thus equality holds and $\tilde{\boldsymbol{z}}'$ is a global minimizer of f with $J(\tilde{\boldsymbol{z}}') < J(\tilde{\boldsymbol{z}})$. Contradiction. □

Theorem 4.4. *Let $H \in \mathbb{C}^{n\times n}$ be a Hermitian matrix and $D = \mathrm{diag}(d_1, \ldots, d_n) \in \mathbb{R}^{n\times n}$ a diagonal matrix. Then for $\boldsymbol{z} \in \partial\mathcal{U}^n$*

$$\boldsymbol{z}^H(H + D)\boldsymbol{z} = \boldsymbol{z}^H H\boldsymbol{z} + \mathrm{trace}(D)$$

Proof. Simple calculation yields

$$\begin{aligned}\boldsymbol{z}^H(H + D)\boldsymbol{z} &= \sum_{i,j=1}^{n} h_{ij}\overline{z_i}z_j + \sum_{i=1}^{n} d_i\overline{z_i}z_i \\ &= \sum_{i,j=1}^{n} h_{ij}\overline{z_i}z_j + \sum_{i=1}^{n} d_i = \boldsymbol{z}^H H\boldsymbol{z} + \mathrm{trace}(D).\end{aligned}$$

□

Corollary 4.5 (The problem HM can be formulated as IHM up to a constant)**.** *Let H be a positive semidefinite Hermitian matrix. Let $H' = H - \mathrm{diag}(h_{11}, \ldots, h_{nn})$ be the matrix H with erased diagonal. Then*

$$\min_{z\in\partial\mathcal{U}^n} \boldsymbol{z}^H H\boldsymbol{z} = \min_{z\in\mathcal{U}^n} \boldsymbol{z}^H H'\boldsymbol{z} + \mathrm{trace}(H)$$

and there is a common minimizer.

Proof. By Theorem 4.3 there is a global minimizer $\boldsymbol{z}^* \in \operatorname{argmin}_{z \in \mathcal{U}^n} \boldsymbol{z}^H H' \boldsymbol{z}$ with $\boldsymbol{z}^* \in \partial \mathcal{U}^n$. Especially

$$\boldsymbol{z}^* \in \operatorname*{argmin}_{z \in \partial \mathcal{U}^n} \boldsymbol{z}^H H' \boldsymbol{z} = \operatorname*{argmin}_{z \in \partial \mathcal{U}^n} \boldsymbol{z}^H H \boldsymbol{z} + \operatorname{trace}(H) = \operatorname*{argmin}_{z \in \partial \mathcal{U}^n} \boldsymbol{z}^H H \boldsymbol{z}.$$

So $\boldsymbol{z}^*$ is a common minimizer, and as $\boldsymbol{z}^* \in \partial \mathcal{U}^n$ by Theorem 4.4 we have

$$\min_{z \in \partial \mathcal{U}^n} \boldsymbol{z}^H H \boldsymbol{z} = (\boldsymbol{z}^*)^H H \boldsymbol{z}^* = (\boldsymbol{z}^*)^H H' \boldsymbol{z}^* + \operatorname{trace}(H) = \min_{z \in \mathcal{U}^n} \boldsymbol{z}^H H' \boldsymbol{z} + \operatorname{trace}(H).$$

□

4.1.3 Literature Survey

All quadratic programming problems are problems where the objective and all equality and inequality constraints are linear or quadratic. So both HM and IHM are quadratic problems. However, IHM has the special structure that only the objective is non-convex while all constraints are convex. Both types of problems have been studied in literature. For a survey of general quadratic programming problems see [Thoai 2005], for problems with convex constraints see e.g. [Nowak 2000].

Almost all solution approaches are based on the following concepts: outer approximation, branch and bound or a combination of both.

We first explain the outer approximation concept. Consider the problem to minimize a concave function f over the domain C. Denote the minimizer by x^*. The outer approximation technique constructs a sequence C_i of sets such that $C_1 \supset C_2 \supset \cdots \supset C$. The sets are constructed in such a way that f can be minimized over the domain C_i, denote the respective minimizer by x_i^*. Then $f(x_i^*) \leq f(x^*)$ for all i and the algorithm converges if $x_i^* \to x^*$ is guaranteed for $i \to \infty$. Furthermore, the algorithm stops when $x_i^* \in C$ for some i.

The branch and bound approach can be applied to minimize a function f over a compact set C. There in the i-th step the domain C is partitioned into sets $C_{i_1}, \ldots, C_{i_{k_i}}$. For each of these sets an upper and a lower bound for the best possible value is computed. Sets with a lower bound greater than the best found upper bound can be removed, as they can never contain the global minimum. This procedure is repeatedly applied. The algorithm converges if eventually the bounds converges to the optimal value.

However, these general algorithms are only practically usable if they can exploit some special structure of the problem. The requirement that all constraints are convex is an example of such a structure. In [Nowak 2000] a branch and cut algorithm for this type of problems is described. We present a more problem adapted branch and bound algorithm in the remaining part of this chapter.

4.2 Custom Branch and Bound

In this section we consider functions of the type $f(\boldsymbol{z}) = \boldsymbol{z}^H H \boldsymbol{z}$ for a positive definite matrix H. If H is only positive semidefinite, by Theorem 4.4 we can replace it by a positive definite matrix $H+D$ where D is a diagonal matrix with non-negative entries. This substitution does not affect the optimal solution point.

We now describe the branch and bound algorithm. Some details are postponed and explained later in this chapter.

Definition 4.6 (Partition Sets). *A partition is a set $\mathcal{Q} = \{Z^1, \dots, Z^m\}$ of sets such that $\operatorname{int}(Z^i) \cap \operatorname{int}(Z^j) = \emptyset$ for all $i \neq j$. It is called a partition of Ω, if $Z^1 \cup \cdots \cup Z^m = \Omega$. The sets Z^i are called partition sets.*

Definition 4.7 (Branching). *A branching on a partition $\mathcal{Q} = \{Z^1, \dots, Z^m\}$ is a subdivision of one partition set Z^k, i.e. it defines a new partition*

$$\mathcal{Q}' = \{Z^1, \dots, Z^{k-1}, Z^{k,1}, \dots, Z^{k,l}, Z^{k+1}, \dots, Z^m\}$$

where $Z^{k,1}, \dots, Z^{k,l}$ is a partition of Z^k.

Definition 4.8 (Optimal Value, Bounds). *The optimal value of f on a set Z is denoted by $f^*(Z) = \min\{f(\boldsymbol{z}) : \boldsymbol{z} \in Z\}$. In particular, we define $f^*(\emptyset) = \infty$. A lower bound function lb is a function such that $lb(Z) \leq f^*(Z)$ for all set Z. Similar an upper bound function ub is a function such that $ub(Z) \geq f^*(Z)$ for all sets Z.*

Note that the upper bound $ub(Z)$ is not an upper bound for the function f on Z, but for the minimal value $f^*(Z)$ of f on Z.

Definition 4.9 (Domain Reduction). *A domain reduction P constructs a domain set $P(Z)$ from a domain set Z such that $P(Z) \subset Z$ and a global minimizer $\boldsymbol{z}^* \in Z$ is contained in $P(Z)$.*

It might happen that $P(Z) = \emptyset$. Then Z does not contain any global optimal solution.

Definition 4.10 (Impact). *The impact of a variable z_k on the objective f is*

$$\gamma_k = \sup\left\{ \left\| \frac{\partial f(\boldsymbol{z})}{\partial \overline{\boldsymbol{z}_k}} \right\| : \boldsymbol{z} \in \partial \mathcal{U} \right\}.$$

Lemma 4.11. *The impact in the circle rotation problem is*

$$\gamma_k = \sum_{j=1}^{n} |h_{kj}| .$$

Proof. We compute by (4.1) for $\boldsymbol{z} \in \partial\mathcal{U}$

$$\left|\frac{\partial f(\boldsymbol{z})}{\partial \overline{\boldsymbol{z}_k}}\right| = \left|\sum_{j\neq k} h_{kj} z_j + |h_{kk}|\, z_k\right| \leq \sum_{j=1}^{n} |h_{kj}|\, |z_j| = \sum_{j=1}^{n} |h_{kj}|\,,$$

where equality holds for some $\boldsymbol{z} \in \partial\mathcal{U}$. □

Definition 4.12 (Angle Size)**.** *Let be $Z \subset \partial\mathcal{U}$. The angle size of Z is defined by*

$$\rho(Z) = \inf\{\beta - \alpha : Z \subset \exp(\imath[\alpha, \beta]), \alpha \leq \beta\}.$$

Intuitively, $\rho(Z)$ is the size of the smallest angle containing Z.

In the circle rotation problem, the initial domain is $\partial\mathcal{U}^n$, so all partition sets have the form $Z = Z_1 \times \cdots \times Z_n \subset \partial\mathcal{U}^n$. We call such partition sets *domain sets*. In the following algorithm all domain sets Z are connected, i.e. the domain Z_k always is a unit circle arc.

Algorithm 1: Circle rotation branch and bound.

```
Data: Target Approximation Quality $\varepsilon > 0$
1  $Z \leftarrow symmetry\_breaking(\partial\mathcal{U}^n)$;
2  $Z \leftarrow domainreduction(Z)$;
3  $lb^* \leftarrow lb(Z)$;
4  $ub^* \leftarrow ub(Z)$;
5  $\mathcal{Q} \leftarrow \{Z\}$;
6  while $ub^* > lb^* \cdot (1 + \varepsilon)$ do
7      take $\hat{Z} \in \mathcal{Q}$;
8      divide $\hat{Z}$ into $\{\hat{Z}^j, j \in J\}$;
9      $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{\hat{Z}\} \cup \{domainreduction(\hat{Z}^j) : j \in J\}$;
10     $lb^* \leftarrow \max\{lb(Z) : Z \in \mathcal{Q}\}$;
11     $ub^* \leftarrow \min\{ub(Z) : Z \in \mathcal{Q}\}$;
12 return $Z \in \mathcal{Q}$ with minimal $ub(Z)$;
```

The branch and bound procedure for the circle rotation problem is shown in Algorithm 1. As f is a positive definite Hermitian form, the optimal value is strictly positive. Below we explain the essential parts of the algorithm.

- The symmetry breaking of $\partial\mathcal{U}^n$ is done in line 1. As $f(\boldsymbol{z}) = f(\alpha\boldsymbol{z})$, for each α with $|\alpha| = 1$ there is a continuum of optimal solutions. This symmetry can be broken by fixing one or more variables to 1. It is obvious that one variable can be fixed, but in some cases more than one variable can be fixed to 1. This is described in Section 4.3.

- The lower bound evaluation function $lb(Z)$ in line 3 and line 10 is defined as $lb(Z) := \min\{f(\boldsymbol{z}) : \boldsymbol{z} \in \mathrm{conv}(Z)\}$, where $\mathrm{conv}(Z)$ denotes the convex hull of Z. As this is a convex optimization problem, it can be evaluated by any standard non-linear solver. However, in Section 4.6 we show how to compute the lower bound more efficiently by the block coordinate descent method.
- For the upper bound evaluation function $ub(Z)$ in line 4 and line 11, any feasible solution value $f(\boldsymbol{z})$ with $\boldsymbol{z} \in Z$ would suffice. However, the quality of the upper bound is crucial, so we solve the non-convex problem to minimize $f(\boldsymbol{z})$ for $z \in Z$ to a local minimum and set the solution to be $ub(Z)$. Depending on the starting point, several solutions can occur. To make this function well defined, we have to specify a starting point and the concrete optimization algorithm. Similar to the lower bound this might be solved by any non-linear solver but can be done more efficiently by the block coordinate descent method. Details are presented in Section 4.7.
- Line 7 comprises the choice of the element $\hat{Z} \in \mathcal{Q}$ to branch on. We observe that in the circle rotation problem usually the upper bound is close to optimal very early, and hence most of the time is spent by improving the lower bound. So we choose $\hat{Z} \in \{Z \in \mathcal{Q} : lb(Z) \text{ minimal}\}$.
- In line 8 the branching on $\hat{Z}$ is performed by dividing it to subdomains $\{\hat{Z}^j, j \in J\}$. There are many possible branching heuristics. We only consider branching on single rotations. This means, we always take the domain $\hat{Z}_k \subset \partial\mathcal{U}$ of a variable z_k and divide this one-dimensional domain into subdomains. The details are described in Section 4.4.
- In line 2 and line 9 the domain reduction algorithm P is applied. By Definition 4.9 this algorithm removes parts of the set Z that cannot contain a global minimizer. The domain reduction is one very important part in the branch and bound algorithm, as it can cut off regions of the feasible domain and, hence, reduces the number of branches. The algorithm is described in Section 4.5.

4.3 Symmetry Breaking

As $f(\boldsymbol{z}) = f(\alpha\boldsymbol{z})$ for each α with $|\alpha| = 1$ there is a global minimizer with $z_n = 1$. We could take a minimizer $\boldsymbol{z}^*$, then $\boldsymbol{z}' = \boldsymbol{z}^*/z_n^*$ is also a minimizer with $z_n' = 1$. So for symmetry breaking we can fix one variable to 1.

However, there might still be rotation symmetry. Consider a matrix implied by an electronic circuit that is not connected, with circle centers in the origin. This matrix can be reordered, such that it is block diagonal. In such cases, for each block a variable can be fixed. We now formalize this idea.

Definition 4.13. *A k-block diagonal matrix is a matrix A of the following form*

$$A = \begin{pmatrix} A_1 & 0 & \dots & 0 \\ 0 & A_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A_k \end{pmatrix} \quad \textit{with} \quad A_j \in \mathbb{C}^{n_j \times n_j}.$$

Definition 4.14. *A Hermitian matrix $A \in \mathbb{C}^{n\times n}$ is called k-reducible if there is a permutation matrix Q such that $Q^T AQ$ is k-block diagonal.*

All elements mapped into the same matrix block by the permutation Q are called a partition.

Note that every matrix is a 1-block diagonal matrix and thus is 1-reducible. Sometimes a matrix is called block diagonal if it is at least 2-block diagonal. Similar it is called reducible if it is at least 2-reducible.

We now show that for a k-reducible matrix H we can fix k variables to 1.

Lemma 4.15. *Let H be k-reducible. Let j_i be an element of partition i, $i = 1, \dots, k$. Then there is a minimum of $f(\boldsymbol{z}) = \boldsymbol{z}^H H\boldsymbol{z}$ over $\partial\mathcal{U}$ with $z_{j_i} = 1$ for $i = 1, \dots, k$.*

Proof. W.l.o.g. by permutation we can assume that H is block diagonal, i.e. $H = \operatorname{diag}(H_1, \dots, H_k)$ with $H_j \in \mathbb{C}^{n_j\times n_j}$. Set $\boldsymbol{z} = (\boldsymbol{z}^1, \dots, \boldsymbol{z}^k)$, where $\boldsymbol{z}^j$ is the vector of entries of $\boldsymbol{z}$ corresponding to block H_j. Then we have

$$f(\boldsymbol{z}) = \left((\boldsymbol{z}^1)^H, \dots, (\boldsymbol{z}^k)^H\right) \begin{pmatrix} H_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & H_k \end{pmatrix} \begin{pmatrix} \boldsymbol{z}^1 \\ \vdots \\ \boldsymbol{z}^n \end{pmatrix} = \sum_{j=1}^{k} (\boldsymbol{z}^j)^H H_j \boldsymbol{z}^j.$$

Let now $\boldsymbol{z} = (\boldsymbol{z}^1, \dots, \boldsymbol{z}^k) \in \partial\mathcal{U}^n$ be an optimal solution. Set $\boldsymbol{w} = (\boldsymbol{w}^1, \dots, \boldsymbol{w}^k)$ with $\boldsymbol{w}^j = \boldsymbol{z}^j/z_1^j$ where z_1^j denotes the first element of $\boldsymbol{z}^j$. Then

$$f(\boldsymbol{w}) = \sum_{j=1}^{k} (\boldsymbol{w}^j)^H H_j \boldsymbol{w}^j = \sum_{j=1}^{k} \frac{1}{\left|z_1^j\right|^2} (\boldsymbol{z}^j)^H H_j \boldsymbol{z}^j = \sum_{j=1}^{k} (\boldsymbol{z}^j)^H H_j \boldsymbol{z}^j = f(\boldsymbol{z}).$$

□

Especially for a k-reducible matrix one variable per block can be fixed to 1. As the influence of variable z_k on the objective is related to the impact γ_k, for each block we fix the variable z_k with maximal impact γ_k.

4.4 Branching Schemes

We only consider branching on single variables. In a first step we choose a variable to branch on, in the second step we split the domain of the variable into subdomains.

4.4.1 Variable Selection

The first step is the selection of a variable to branch on. Denote by $\hat{Z} = (\hat{Z}_1, \dots, \hat{Z}_n)$ the domain set to branch. A common heuristic is to make the most important decisions first. As the hardest part of the circle rotation problem is to increase the lower bound, the branching scheme should select a variable whose restriction increases the lower bound as much as possible.

According to these heuristics, we now state properties of rotations z_k that make them desirable to branch on. Recall that $lb(\hat{Z}) = \min \left\{ f(\boldsymbol{z}) : \boldsymbol{z} \in \operatorname{conv} \hat{Z} \right\}$ and denote by $\boldsymbol{z}^{lb}$ a solution where the minimum is attained. Note that $\boldsymbol{z}^{lb} \in \mathcal{U}^n$ but usually $\boldsymbol{z}^{lb} \notin \partial\mathcal{U}^n$.

- A variable z_k with great impact γ_k on the objective should be preferred (see Definition 4.10).
- A variable z_k should be preferred for branching if it has a great degree of freedom, i.e. the angle size $\rho(Z_k)$ is maximal. This is motivated by the observation that dividing a big domain into smaller ones is a bigger restriction than dividing an already small domain into the same number of even smaller ones.
- A variable z_k should be preferred for branching, if $\left|z_k^{lb}\right|$ is small. If $\left|z_k^{lb}\right|$ is close to 1, it is unlikely that a restriction of Z_k increases the lower bound significantly. However, if z_k^{lb} is close to zero, a restriction of Z_k restricts $\operatorname{conv}(Z_k)$ such that z_k^{lb} is infeasible for the lower bound problem subject to the restricted domain. So z_k^{lb} is pushed to the boundary and consequently the lower bound improves.

The heuristic chooses the variable to branch on according to the aspects described above. The variable z_k selected for branching is chosen by

$$k \in \operatorname*{argmax}_{j=1,\dots,n} \ \gamma_j \cdot \rho(\hat{Z}_j) \cdot \left(1 - \left|z_j^{lb}\right|\right).$$

Then $\hat{Z}_k$ is divided into subsets $\hat{Z}_k^j$, $j \in J$. Thus we partition $\hat{Z}$ into

$$\left\{ \hat{Z}_1 \times \dots \times \hat{Z}_{k-1} \times \hat{Z}_k^j \times \hat{Z}_{k+1} \times \dots \times \hat{Z}_n \subset \partial\mathcal{U}^n \ : \ j \in J \right\}$$

There is still freedom how to choose $\hat{Z}_k^j$, $j \in J$. This is described in the next section.

4.4.2 Branching on the Selected Element

We now assume that a variable z_k to branch on has been selected. The set before branching is $\hat{Z}_k$ and we want to subdivide it into sets $\hat{Z}_k^j, j \in J$. As in the previous section, denote by $z_k^{lb} \in \mathcal{U}$ the k-th variable of the solution, for which the lower bound value is attained. By the branching there are several things to be achieved:

- The lower bound should be increased. This means the region containing z_k^{lb} should be cut of.
- The number of remaining branches should be kept small. This can be done by either generating few sets $\hat{Z}_k^j, j \in J$ or by ensuring that these sets can be removed by domain reduction as their lower bound is too large.

However, these goals are contradictory. Dividing the set $\hat{Z}_k$ into many smaller arcs, cuts of z_k^{lb} and the new lower bound solutions are closer to the circle boundary. However, many branches have to be considered in the algorithm.

Even if the number of partition sets is fixed, there are different ways to generate these branches. For example, for three partition sets Figure 4.1 shows two alternatives with different focus. In Figure 4.1a the branching is done in a way such that two branches have a large distance to the current lower bound solution. So it is likely that they are dropped as their lower bound is too large. However, for one partition set the lower bound z_k^{lb} remains feasible, especially the lower bound is not improved. In contrast in Figure 4.1b the lower bound is guaranteed to be improved, therefore it is likely that two branches remain in the algorithm, as their lower bound is not large enough to be skipped.

There are the same considerations if we have to subdivide a proper circular arc $\hat{Z}_k \subsetneq \partial\mathcal{U}$. However, there the boundary of the outer arcs in the subdivision is fixed and we divide this arc into equidistant parts.

4.5 Domain Reduction Algorithm

The domain reduction algorithm is of particular importance within the branch and bound scheme. It drops parts of the domain that are guaranteed not to contain the global minimum. There is a trade-off for domain reduction. Of course, the domain reduction algorithm costs time in each branching node. However, it avoids the algorithm from branching in the skipped region and thus reduces the number of branches.

We propose a domain reduction algorithm based on computational geometry. The concepts and algorithms required for our domain reduction are introduced in Section 4.5.1.

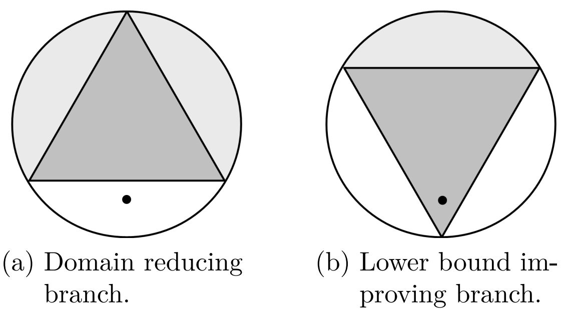

(a) Domain reducing branch.
(b) Lower bound improving branch.

Figure 4.1: Different possibilities to divide $\partial\mathcal{U}$ into three partition sets. The point marks z_k^{lb} and the unit circle is the initial domain of z_k^{lb}. By branching the domain is separated into three parts. The white circle segments are parts with probably small lower bound while the light gray parts are likely to have a larger lower bound. The dark gray triangle is not contained in the domain of z_k^{lb} for any of the branches. In (a) the lower bound is not improved, however, it is likely that two branches may be skipped by their lower bound. In (b) the lower bound is improved, but only one branch is likely to be skipped.

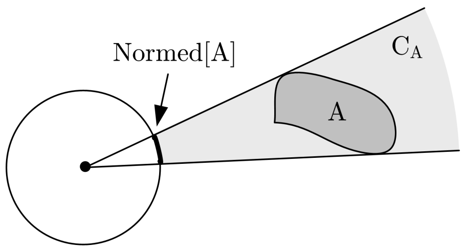


Figure 4.2: Normalization Normed $[A]$ of a set A.

4.5.1 Geometric Computations

In this section we introduce set computations. While most of the concepts can be considered in general spaces, we only consider the space $\mathbb{C}$.

Let $a, b \in \mathbb{C}$ denote elements, $A, B \subset \mathbb{C}$ sets and h a function on $\mathbb{C}$. Then $h(A) = \{h(a) : a \in A\}$. Especially the Minkowski sum is $A + B = \{a + b : a \in A, b \in B\}$, a set can be rotated by $aB = \{ab : b \in B\}$ and translated by $a + B = \{a + b : b \in B\}$.

Recalling Definition 2.7, the normalization of A is Normed $[A] = \{\text{Normed}\,[a] : a \in A\}$. This has also a geometric interpretation visualized in Figure 4.2. Assume A to be a connected set. Let C_A be the smallest cone in the origin such that $A \subset C_A$. Then Normed $[A]$ is the set of points of the cone C_A that lie on the unit circle, i.e. Normed $[A] = C_A \cap \partial\mathcal{U}$. In the literature the problem of computing C_A is known as aperture angle problem in [Ahn et al. 2007] or as apex angle problem in [Bose; Seara; Sethia 2004].

While these definitions are rather general, we only need sets that are the interior of a closed curve. Let A, B such sets and $C_1 = \delta(A)$ and $C_2 = \delta(B)$ be the curves enclosing A and B, then the convolution curve $C_1 * C_2$ is defined by summing the points having the same normal direction. Denote the normal on C_i as N_i it is

$$C_1 * C_2 = \{a + b : a \in C_1, b \in C_2, N_1(a) \parallel N_2(b), \langle N_1(a), N_2(b) \rangle > 0\}\,.$$

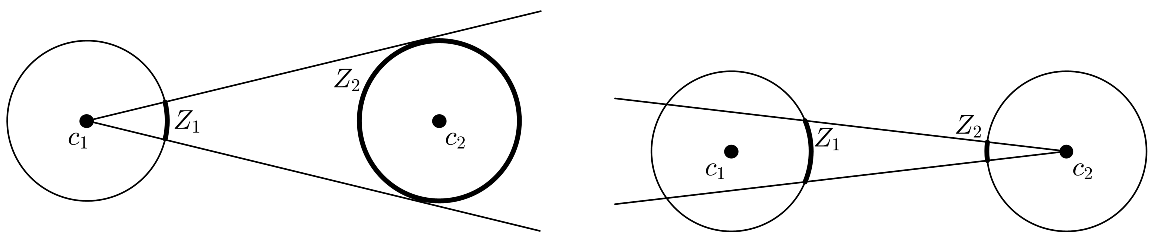
Note that the Minkowski sum of convex sets is convex again. Similar, if the curves C_1 and C_2 enclose convex sets, then the interior of C_1 and C_2 is convex.

The strong relation between convolution and Minkowski sum is analyzed in [Bajaj; Kim 1989]. In general, the boundary of the Minkowski sum is a subset of the convolution, i.e. $\delta(A + B) \subset \delta(A) * \delta(B)$. However, if A and B are convex, equality holds. So for convex sets $\delta(A + B) = \delta(A) * \delta(B)$. Despite in general the convolution can be more efficiently computed than the Minkowski sum, for convex sets both computations are the same.

One of the simplest cases to consider are polygons. A polygon B is represented by the counterclockwise enumeration $b_1, \dots, b_k$ of its vertices. Now $a + B$ can be computed by translating all vertices and aB by multiplying all vertices. The Minkowski sum $A + B$ of two polygons is a polygon and can be computed by standard algorithms of computational geometry. For two convex polygons, the sum $A+B$ is itself convex and can be computed in linear time, see e.g. [Berg et al. 2000, Theorem 13.10]. Furthermore, for a polygon A the set $\mathrm{Normed}\,[A]$ can be computed in linear time, as the cone of each polygon arc $b_i b_{i+1}$ can be computed in constant time and the cone containing A is the union of the cones of the arcs.

A natural extension of polygons and a concept widely used in the literature are splinegons. They are a generalization of polygons, where each arc is represented by an algebraic curve. So a splinegon can be represented by a sequence $b_1, \beta_1, b_2, \dots, b_k, \beta_k$ where b_j are the vertices and β_j algebraic curves with endpoints b_j and b_{j+1}. Splinegons have first been considered in [Souvaine 1986], for a survey see [Dobkin; Souvaine 1990]. In particular, a polygon is a special splinegon, where each arc is a line segment. Many geometric algorithms developed for polygons can be generalized to splinegons, see e.g. [Bajaj; Kim 1988]. Scaling and translation can be applied componentwise. For a splinegon B the set $\mathrm{Normed}\,[B]$ can be computed by computing the union of all sets $\mathrm{Normed}\,[\beta_j]$.

The first algorithm for computing the convolution of splinegons has been presented in [Bajaj; Kim 1989]. This algorithm works for general, not necessarily convex splinegons. Let C_1 and C_2 be splinegons with k_1 resp. k_2 curves with fixed maximal degree, this algorithm has complexity $O(k_1^3 k_2^3)$. If both splinegons are convex, the complexity reduces to linear time $O(k_1 + k_2)$. In [Lee; Kim; Elber 1998] an algorithm for computation of the Minkowski sum of general splinegons based on the convolution was presented. Note that the Minkowski sum of splinegons of maximal degree d is a splinegon with maximal degree d.



(a) Domain Reduction of Z_1. (b) Domain Reduction of Z_2.

Figure 4.3: Iterative geometric domain reduction.

For convex splinegons the Minkowski sum is equal to the convolution. For this case the algorithm of [Bajaj; Kim 1989] has been further improved in [Kohler; Spreng 1995]. We give a brief sketch of the algorithm.

Consider convex splinegons A and B with boundary curves C_1 and C_2. To each point of the boundary of a curve a set of normal vectors is assigned. A non-differentiable vertex is considered to have continuously changing normal vectors. Now both curves are segmented into compatible parts, see [Kohler; Spreng 1995, Definition 6]. Thereby, two curves are defined as compatible if they have the same set of normal vectors. Two vertices are compatible, if their set of normal vectors intersect and a vertex and a point are compatible, if the normal vectors of the curve are a subset of the set of normal vectors of the point. Then similarly to the algorithm for convex polygons, the algorithm proceeds and iterates simultaneously through both splinegons and componentwise computes the convolution of the compatible segments.

4.5.2 Geometric Domain Reduction Algorithm

We now use the concepts and algorithms of the previous section to construct a domain reduction algorithm. We first give a geometric intuition of the domain reduction and formalize it later in this section.

Consider two unit circles with centers c_1 and c_2 and connected pins with offset 1. The idea of the geometric domain reduction is visualized in Figure 4.3. Assume that at the beginning there is no restriction on the rotations z_1 and z_2, i.e. we have $Z_1 = Z_2 = \partial\mathcal{U}$. We know that for each possible rotation of z_2 the optimal position of the pin of circle 1 is on the straight line from c_1 to pin 2. Thus, as is shown in Figure 4.3a, we can reduce the possibly optimal rotations Z_1 of circle 1. In the next step displayed Figure 4.3b, the set Z_2 can be reduced. This procedure is repeatedly applied until no significant reduction of the sets is possible.

In Figure 4.3a the set Z_1 is the intersection of the circle around c_1 and the cone from c_1 containing Z_2. Hence, it is the normalization of the translated set Z_2. In Theorem 4.18 we generalize this idea to Hermitian minimization problems, where Z_2 is replaced by the Minkowski sum of circular arcs. As this Minkowski sum cannot be computed efficiently, in Theorem 4.19 the circular arcs are replaced by convex splinegons. For these splinegons, the Minkowski sum and their normalization can be computed efficiently by the methods of Section 4.5.1.

Lemma 4.16 (Composition of Domain Reductions)**.** *Let $P_1, \ldots, P_k$ be domain reductions as defined in Definition 4.9. Then $P := P_k \circ \cdots \circ P_1$ is a domain reduction.*

Proof. $P(Z) \subset Z$ and $\boldsymbol{z}^* \in Z \implies \boldsymbol{z}^* \in P(Z)$ can be seen by induction. □

Lemma 4.17. *Let P' be a domain reduction and P a mapping such that $P'(Z) \subset P(Z) \subset Z$ for all domain sets Z. Then P is a domain reduction.*

Proof. We have $P(Z) \subset Z$ by assumption for all Z. Let $\boldsymbol{z}^* \in Z$ be an optimum, then $\boldsymbol{z}^* \in P'(Z)$ implies $\boldsymbol{z}^* \in P(Z)$. □

Theorem 4.18. *Let $Z = Z_1 \times \cdots \times Z_n$ be a domain set. For all k set*

$$\Gamma'_k(Z) = -\mathit{Normed}\left[\sum_{j \neq k} q_{kj} Z_j\right] \cap Z_k, \tag{4.2}$$

$$P'_k(Z) = Z_1 \times \cdots \times Z_{k-1} \times \Gamma'_k(Z) \times Z_{k+1} \times \cdots \times Z_n. \tag{4.3}$$

Then P'_k is a domain reduction.

Proof. Obviously $\Gamma'_k(Z) \subset Z_k$ and thus $P'_k(Z) \subset Z$.

Now assume an optimal solution $\boldsymbol{z}^* \in Z$. By (4.1) we get

$$f_k(\xi, \pi_k(\boldsymbol{z})) = \psi\left[q_{kk}, \sum_{j \neq k} q_{kj} z_j, \sum_{i,j \neq k} q_{ij} \overline{z_i} z_j\right](\xi)$$

with $q_{kk} > 0$. With $z_j^* \in Z_j$ for all j Lemma 2.18 implies

$$z_k^* \in \operatorname*{argmin}_{\xi \in \partial \mathcal{U}} f_k(\cdot, \pi_k(\boldsymbol{z})) = -\mathrm{Normed}\left[\sum_{j \neq k} q_{kj} z_j\right] \subset -\mathrm{Normed}\left[\sum_{j \neq k} q_{kj} Z_j\right].$$

As we already assumed $z_k^* \in Z_k$, we know $z_k^* \in \Gamma'_k(Z)$. Especially it follows $\boldsymbol{z}^* \in P'_k(Z)$. Thus P'_k is a domain reduction. □

The disadvantage of P'_k in the above Theorem 4.18 is that it cannot be computed efficiently. Thus we now define another domain reduction. This is not as strong as P'_k but can be computed efficiently.

Theorem 4.19. *Let $Z = Z_1 \times \cdots \times Z_n$ be a domain set. For all k set*

$$\Gamma_k(Z) = -\mathit{Normed}\left[\sum_{j \neq k} q_{kj} \operatorname{conv}(Z_j)\right] \cap Z_k, \tag{4.4}$$

$$P_k(Z) = Z_1 \times \cdots \times Z_{k-1} \times \Gamma_k(Z) \times Z_{k+1} \times \cdots \times Z_n. \tag{4.5}$$

Then P_k is a domain reduction.

Proof. For all Z we have $\Gamma_k(Z) \subset Z_k$ and thus $P_k(Z) \subset Z$. Furthermore,

$$\begin{aligned}
& Z_j \subset \operatorname{conv}(Z_j) \\
\Longrightarrow\ & \sum_{j \neq k} q_{kj} Z_j \subset \sum_{j \neq k} q_{kj} \operatorname{conv}(Z_j) \\
\Longrightarrow\ & -\operatorname{Normed}\left[\sum_{j \neq k} q_{kj} Z_j\right] \subset -\operatorname{Normed}\left[\sum_{j \neq k} q_{kj} \operatorname{conv}(Z_j)\right] \\
\Longrightarrow\ & \Gamma'_k(Z) \subset \Gamma_k(Z) \\
\Longrightarrow\ & P'_k(Z) \subset P_k(Z)
\end{aligned}$$

and by Lemma 4.17 the claim follows. □

Corollary 4.20. *Denote by P'_k the domain reductions defined in Theorem 4.18 and by P_k the domain reductions defined in Theorem 4.19. Then the mappings $P' = P'_n \circ \cdots \circ P'_1$ and $P = P_n \circ \cdots \circ P_1$ are domain reductions.*

Proof. Immediate consequence from Lemma 4.16, Theorem 4.18 and Theorem 4.19. □

We explain the meaning of this theorem. Assume a domain set Z contains an optimal solution. Then with the domain reductions P' or P we can compute a new, potentially smaller angle set that also contains an optimal solution. So we are performing a domain reduction by dropping feasible regions that do not contain the optimal solution. Although P' yields the smaller set, in practice we use P which is easier to compute.

The Minkowski sum in each P_k can be computed efficiently by the algorithms given in Section 4.5.1. The set $\operatorname{conv}(Z_j)$ is a splinegon of maximal degree 2. So $q_{kj} \operatorname{conv}(Z_j)$ is a splinegon of degree 2. Therefore, $\sum_{j \neq k} q_{kj} \operatorname{conv}(Z_j)$ is a splinegon of maximal degree 2 and can be computed efficiently. Thus we can compute $\Gamma_k(Z)$, $P_k(Z)$ and subsequently $P(Z)$.

The initial domain reduction step starting from the domain set $Z^1 = Z$ computes $Z^{t+1} = P(Z^t)$. It stops, if either $Z^t = \emptyset$ or $Z^{t+1} \approx Z^t$, i.e. the domain reduction does not make significant restrictions to the domain set. We cannot require equality $Z^{t+1} = Z^t$, as the generated sequence of domain sets converges to a solution but might

Algorithm 2: Domain reduction algorithm.

```
Data: Domain Set $Z = Z_1 \times \cdots \times Z_n$
1 repeat
2     $Z' \leftarrow Z$;
3     for $k = 1, \ldots, n$ do
4         $Z_k \leftarrow \Gamma'_k(Z)$;
5 until $Z = \emptyset$ or $Z \approx Z'$;
6 return $\Gamma$;
```

never reach it, see Example 4.21. For some predefined $\delta > 0$ the formulation $Z^{t+1} \approx Z^t$ can be stated rigorously as

$$\sum_{j=1}^{n} \left|\rho(Z_k^{t+1}) - \rho(Z_k^t)\right| \leq \delta,$$

where ρ is the angle size as defined in Definition 4.12.

Example 4.21. *Consider the circle rotation problem with the following settings. Circle 1 is centered in $c_1 = -a$ and has a pin with offset $p_1 = 1$. Circle 2 is centered in $c_2 = +a$ and has a pin with offset $p_2 = -1$. There is one net connecting two pins with weight $\mu_1 = 1$. This circle rotation problem visualized in Figure 4.4 is equivalent to the Hermitian minimization problem with*

$$H = \begin{pmatrix} 1 & 1 & -2a \\ 1 & 1 & -2a \\ -2a & -2a & a^2 \end{pmatrix}.$$

For $a \geq 1$ the optimal solution is $\boldsymbol{z}^* = (1, 1, 1)^T$.

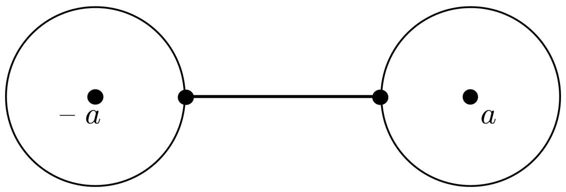


Figure 4.4: As explained in Example 4.21, the domain reduction converges to the optimum but never reaches it.

Assume that z_3 is fixed, i.e. $Z_3 = \{1\}$. Then we get the mappings

$$\Gamma'_1(Z) = -\mathit{Normed}[Z_2 - 2a] \cap Z_1,$$
$$\Gamma'_2(Z) = -\mathit{Normed}[Z_1 - 2a] \cap Z_2.$$

Denote by $S(x) = \partial\mathcal{U} \cap \{z \in \mathbb{C} : \Re(z) \geq x\}$ the part of the unit circle boundary with real part greater or equal to x. Starting the domain reduction with $Z_1^0 = S(x_0)$ for

$0 \leq x_0 \leq a$, $Z_2^0 = \partial\mathcal{U}$ and $Z_3 = \{1\}$ for Z_1, we iteratively obtain the domains $Z_1^t = S(x_t)$ with

$$r_{t+1} = 2a + \frac{x - 2a}{\sqrt{1 + 4a(a - x_t)}}, \qquad x_{t+1} = \frac{r_{t+1}}{\sqrt{r_{t+1}^2 + \frac{1-x^2}{1+4a(a-x)}}}.$$

For $a \geq 1$ and $x_0 < 1$, the sequence x_k converges to 1 but never attains it.

4.6 Lower Bound Evaluation

In Algorithm 1, for each branch a lower bound is computed. The lower bound to the solution is the worst lower bound of any branch. Furthermore, branches whose lower bound value is larger than the best found solution, can be removed from the set $\mathcal{Q}$ of domain sets. Therefore, it is important to compute good lower bounds efficiently.

Recall that the domain set $Z = Z_1 \times \cdots \times Z_n \subset \partial\mathcal{U}^n$ is connected and thus each Z_j is a unit circle arc. Furthermore, recall that $\operatorname{conv}(Z) = \operatorname{conv}(Z_1) \times \cdots \times \operatorname{conv}(Z_n)$. The lower bound is defined as $lb(Z) := \min\{f(\boldsymbol{z}) : \boldsymbol{z} \in \operatorname{conv}(Z)\}$. We first state the problem as non-linear optimization problem. This problem is a convex optimization problem satisfying Slater's condition and thus can be solved by standard techniques. However, in this section we present a block coordinate descent method to solve the problem more efficiently.

We first describe the domain sets of the single variables by non-linear equations. If a variable domain Z_j consists of a single element, the same is true for $\operatorname{conv}(Z_j)$. So there is only one feasible solution which is optimal. In practice, these variables are fixed and not be part of the optimization problem. Hence, we can w.l.o.g. assume that all variable domains Z_j are a unit circle arc $\exp(\imath[\alpha_j, \beta_j])$ with $\alpha_j < \beta_j$.

Lemma 4.22. *Let $W \subset \partial\mathcal{U}$. Then there are $d \in \mathbb{R}$, $s \in \mathbb{C}$ such that*

$$\operatorname{conv}(W) = \{z \in \mathbb{C} : \quad |z|^2 - 1 \leq 0, \quad d - \Re(z\overline{s}) \leq 0\}.$$

Proof. Assume that W is a proper unit circle arc, i.e. $W = \exp(\imath[\alpha, \beta])$ with $\alpha < \beta < \alpha + 2\pi$. Define

$$d = \cos\left(\frac{\alpha - \beta}{2}\right), \qquad s = \exp\left(\frac{\imath}{2}(\alpha + \beta)\right).$$

Then by Definition 2.8 the positive closed half space $H_{\geq} = H_{\geq}(s, d)$ to the right of the line from $\exp(\imath\alpha)$ to $\exp(\imath\beta)$ is $H_{\geq}(s, d) = \{w \in \mathbb{C} : \Re(w\overline{s}) \geq d)\}$ and in particular $W = \partial\mathcal{U} \cap H_{\geq}$ and $\operatorname{conv}(W) = \mathcal{U} \cap H_{\geq}$.

If $W = \partial\mathcal{U}$, then $\operatorname{conv}(W) = \mathcal{U} = \{W : |w|^2 \leq 1\}$. In this case, we may set $d = -2$ and $s = 1$ in order to make the constraint $d - \Re(z\overline{s}) \leq 0$ redundant. □

A lower bound $lb(Z) = \min\{f(\boldsymbol{z}) : \boldsymbol{z} \in \operatorname{conv}(Z)\}$ can be computed by solving the problem

$$\begin{array}{lll} \min & \boldsymbol{z}^H H \boldsymbol{z} & \\ \text{s. t.} & |z_i|^2 - 1 \leq 0 & \\ & d_i - \Re(z_i \overline{s_i}) \leq 0, & i = 1, \dots n. \end{array} \tag{4.6}$$

This is a convex optimization problem as f and $\operatorname{conv}(Z)$ are convex. We now consider the dual problem. As the problem is convex, there is no duality gap. The Lagrangian is with $\boldsymbol{\eta} \bullet \boldsymbol{s}$ denoting the component wise multiplication

$$\begin{aligned} L(\boldsymbol{z}, \boldsymbol{\mu}, \boldsymbol{\eta}) &= \boldsymbol{z}^H H \boldsymbol{z} + \sum \mu_i (|z_i|^2 - 1) + \sum \eta_i (d_i - \Re(z_i \overline{s_i})) \\ &= \boldsymbol{z}^H (H + \sum \mu_i \boldsymbol{e}_i \boldsymbol{e}_i^T) \boldsymbol{z} - \Re((\boldsymbol{\eta} \bullet \boldsymbol{s})^H \boldsymbol{z}) - \sum \mu_i + \sum \eta_i d_i. \end{aligned}$$

The Lagrangian dual now is for $(\boldsymbol{\mu}, \boldsymbol{\eta}) \geq 0$

$$\Theta(\boldsymbol{\mu}, \boldsymbol{\eta}) = \min\{L(\boldsymbol{z}, \boldsymbol{\mu}, \boldsymbol{\eta}) : \boldsymbol{z} \in \mathbb{C}^n\}.$$

As $H \succ 0$ and $\boldsymbol{\mu} \geq 0$ it follows $H + \sum \mu_i \boldsymbol{e}_i \boldsymbol{e}_i^T \succ 0$. So for fixed $(\boldsymbol{\mu}, \boldsymbol{\eta}) \geq 0$ the Lagrangian is a quadratic positive definite form in $\boldsymbol{z}$. The evaluation of the dual is done by unconstrained minimization of a positive definite quadratic function.

Lemma 4.23. *For $(\boldsymbol{\mu}, \boldsymbol{\eta}) \geq 0$ the Lagrangian dual is*

$$\Theta(\boldsymbol{\mu}, \boldsymbol{\eta}) = -\frac{1}{4} (\boldsymbol{\eta} \bullet \boldsymbol{s})^H (H + \sum \mu_i \boldsymbol{e}_i \boldsymbol{e}_i^T)^{-1} (\boldsymbol{\eta} \bullet \boldsymbol{s}) - \sum \mu_i + \sum \eta_i d_i.$$

Proof. For fixed dual variables $(\boldsymbol{\mu}, \boldsymbol{\eta}) \geq 0$, we compute the unique minimizer $\boldsymbol{z}_0$ by

$$\begin{aligned} & \nabla_{\overline{z}} L(\boldsymbol{z}_0, \boldsymbol{\mu}, \boldsymbol{\eta}) = 0 \\ \iff & (H + \sum \mu_i \boldsymbol{e}_i \boldsymbol{e}_i^T) \boldsymbol{z}_0 - \frac{1}{2} (\boldsymbol{\eta} \bullet \boldsymbol{s}) = 0 \\ \iff & \boldsymbol{z}_0 = \frac{1}{2} (H + \sum \mu_i \boldsymbol{e}_i \boldsymbol{e}_i^T)^{-1} (\boldsymbol{\eta} \bullet \boldsymbol{s}). \end{aligned}$$

Then $\Theta(\boldsymbol{\mu}, \boldsymbol{\eta}) = L(\boldsymbol{z}_0, \boldsymbol{\mu}, \boldsymbol{\eta})$ yields the result. □

By duality theory with $(\boldsymbol{\mu}, \boldsymbol{\eta}) \geq 0$ and $\boldsymbol{z} \in \operatorname{conv}(Z)$ we get

$$\Theta(\boldsymbol{\mu}, \boldsymbol{\eta}) \leq lb(Z) \leq f(\boldsymbol{z}).$$

As the problem is strictly convex and Slater's condition holds, there is a unique primal solution $\boldsymbol{z}^* \in \operatorname{conv}(Z)$ and unique Lagrange multipliers $(\boldsymbol{\mu}^*, \boldsymbol{\eta}^*) \geq 0$ such that equality $\Theta(\boldsymbol{\mu}^*, \boldsymbol{\eta}^*) = f(\boldsymbol{z}^*)$ holds.

Now we apply the block coordinate descent method described in Section 2.6. Geometrically this method iteratively keeps all but one circle rotation fixed and then computes the optimal rotation of this circle. If this optimization of a single circle can be done efficiently, the method converges fast in practice.

4.6.1 Coordinate Descent Method Application to Rotation Problem

We now apply the block coordinate descent method to the lower bound problem $\min\{f(\boldsymbol{z}) : \boldsymbol{z} \in \operatorname{conv}(Z)\}$ as stated in (4.6). Block coordinate descent methods have been applied to quadratic problems over convex sets, see e.g. [Han 1988] and [Tseng 1993] and these methods can be immediately applied for lower bound evaluation. However, the lower bound problem has a special structure which can be exploited to simplify the algorithm and its analysis.

We already know by Lemma 2.18 and (4.1) that we can compute the optimal relaxed rotation of one circle if all other circles are fixed. In this section we show that we can also efficiently compute the optimal rotation if the rotation is restricted to $\operatorname{conv}(W)$ for a unit circle arc W and fixed other rotations. This observation makes the block coordinate descent method fast for the lower bound computation.

We emphasize that all points in the sequence are feasible, i.e. we approximate the optimal solution value from above. As the algorithm usually never reaches the minimum $lb(Z)$, by the primal algorithm we only get an upper bound for $lb(Z)$. To get a lower bound for $lb(Z)$, we have to consider the dual problem. It is easy to compute the dual variables for the subproblem of rotating a single circle.

By applying the results shown in Section 2.6, we can show that the dual variables of the subproblems converge to the dual variables of the global optimization problem. As the problem is convex, there is no duality gap. So while solving the optimization problem $lb(Z)$ with the block coordinate descent method, we additionally get a sequence of dual solutions converging to $lb(Z)$ from below.

The domain is $\operatorname{conv}(Z) = \operatorname{conv}(Z_1) \times \cdots \times \operatorname{conv}(Z_n)$. Thus the subproblem is to minimize the objective with respect to $z_k \in \operatorname{conv}(Z_k)$ and all other rotations being fixed. From (4.1) we know $f_k(\cdot, \pi_k(\boldsymbol{z})) = \psi[a, u, b]$ with $a > 0$. Using the formulation of Lemma 4.22 to represent $\operatorname{conv}(Z_k)$, the subproblem $\operatorname{argmin}\{f_k(z, \pi_k(\boldsymbol{z})) : z \in \operatorname{conv}(Z_k)\}$ can be formulated as

$$\begin{aligned} \min_{z} \quad & a\,|z|^2 + 2\Re(z\overline{u}) + b \\ \text{s. t.} \quad & |z|^2 - 1 \leq 0 \\ & d - \Re(z\overline{s}) \leq 0. \end{aligned} \tag{4.7}$$

Theorem 4.24. *Problem* (4.7) *has a unique primary solution and unique Lagrange multipliers. Both the optimal primal solution and the Lagrange multipliers can be efficiently computed.*

Proof. As $a > 0$ the function $\psi = \psi[a, u, b]$ is strictly convex. The domain is strictly convex and bounded. Slater's condition is satisfied. So there is a unique solution with unique Lagrange multipliers. We now show a way to compute them efficiently.

Stating the KKT conditions there are Lagrange multipliers $\mu, \eta \geq 0$ with

$$(a+\mu)z + u - \eta s = 0 \tag{4.8}$$
$$|z|^2 - 1 \leq 0 \tag{4.9}$$
$$d - \Re(z\overline{s}) \leq 0 \tag{4.10}$$
$$\mu(|z|^2 - 1) = 0 \tag{4.11}$$
$$\eta(d - \Re(z\overline{s})) = 0 \tag{4.12}$$

We solve the KKT-equations and have to distinguish several cases.

Case 1: $\mu = \eta = 0$. Then none of the constraints is active, so the minimum is in the interior. From (4.8) we get that $z = -\frac{u}{a}$ is the unique KKT-point.

Case 2: $\mu = 0$, $\eta > 0$. From (4.12) we get $\Re(z\overline{s}) = d$. Furthermore, (4.8) implies $z = \frac{1}{a}(\eta s - u)$. Thus,

$$\begin{aligned} ad = a\Re(z\overline{s}) &= \Re\left((\eta s - u)\overline{s}\right) = \eta - \Re(u\overline{s}) \\ \iff \eta &= ad + \Re(u\overline{s}) \\ \implies z &= \left(d + \frac{1}{a}\Re(u\overline{s})\right) s - \frac{u}{a}, \end{aligned}$$

which is the unique KKT-point.

Case 3: $\mu > 0$, $\eta = 0$. Then (4.11) implies $|z| = 1$ and from (4.8) we get

$$(a+\mu)z = -u \implies |a+\mu|\,|z| = |u| \implies a + \mu = |u| \implies \mu = |u| - a,$$

and $z = \frac{-u}{|u|}$ is the unique KKT-point.

Case 4: $\mu > 0$, $\eta > 0$. Then from (4.12) we get $\Re(z\overline{s}) = d$, and (4.11) implies $|z| = 1$. Consequently,

$$|z\overline{s}| = 1 \wedge \Re(z\overline{s}) = d \iff z\overline{s} = d \pm i\sqrt{1-d^2} \iff z = sd \pm \imath s\sqrt{1-d^2}.$$

In both cases the Lagrange multipliers μ and η can be computed by solving (4.8) as a real 2×2 equation system.

The set of possible KKT-points is given by

$$O_R[\psi, Z] = \left\{ -\frac{u}{a},\ -\frac{u}{|u|},\ sd + \imath s\sqrt{1-d^2},\ sd - \imath s\sqrt{1-d^2},\ \left(d + \frac{1}{a}\Re(u\overline{s})\right) s - \frac{u}{a} \right\}.$$

For each case, the Lagrange multipliers can be computed. □

We now apply the block coordinate descent method described in Section 2.6 to the lower bound problem. We start with an initial feasible solution $\boldsymbol{z}^0 \in \operatorname{conv}(Z)$. Then we iteratively compute $\boldsymbol{z}^{t+1}$ from $\boldsymbol{z}^t$. This is done as follows. For those z_k that are not fixed and $k = 1 \ldots, n$ set

$$\begin{aligned} w_j &:= \begin{cases} z_j^{t+1} & \text{if } j < k \\ z_j^t & \text{otherwise} \end{cases} \\ \psi_k^t &:= \psi \left[q_{kk}, \sum_{j \neq q} q_{kj} w_j, \sum_{i,j \neq k} q_{ij} \overline{w_i} w_j \right] \\ z_k^{t+1} &:= \operatorname*{argmin}_{\xi \in \operatorname{conv}(Z_k)} \psi_k^t(w_1, \ldots, w_{k-1}, \xi, w_{k+1}, \ldots, w_n). \end{aligned}$$

From Theorem 4.24 we know that both z_k^{t+1} and the Lagrange multiplier μ_k^{t+1} and η_k^{t+1} are unique and can be efficiently computed. As the domain is closed and convex, the algorithm converges to a stationary point according to Theorem 2.35. As the problem is strictly convex, this is the unique global minimum.

Corollary 4.25. *Denote the constructed sequence of primal variables by $\boldsymbol{z}^t$ and the sequence of dual variables by $(\boldsymbol{\mu}^t, \boldsymbol{\eta}^t)$. Then*

$$\begin{aligned} &\forall t: \quad \Theta(\boldsymbol{\mu}^t, \boldsymbol{\eta}^t) \leq lb(Z) \leq f(\boldsymbol{z}^t), \\ &\lim_{t \to \infty} \boldsymbol{z}^t = \boldsymbol{z}^*, \\ &\lim_{t \to \infty} (\boldsymbol{\mu}^t, \boldsymbol{\eta}^t) = (\boldsymbol{\mu}^*, \boldsymbol{\eta}^*), \\ &\lim_{t \to \infty} \Theta(\boldsymbol{\mu}^t, \boldsymbol{\eta}^t) = lb(Z) = \lim_{t \to \infty} f(\boldsymbol{z}^t). \end{aligned}$$

Proof. The first statement is the weak duality theorem. The other statements follow from Corollary 2.38 and Theorem 2.40 and the absence of a duality gap. □

In practice it is too expensive, to compute the dual value in every iteration. Evaluating the dual function from Lemma 4.23 requires the solution of an equation system. So for the lower bound evaluation we apply a block coordinate descent method until the primal variables numerically converged. Then we perform one more iteration in which we also compute the dual variables $(\boldsymbol{\mu}^{t+1}, \boldsymbol{\eta}^{t+1})$. As they also converge when the primal variables converge, they are close to the optimal duals $(\boldsymbol{\mu}^*, \boldsymbol{\eta}^*)$ and thus $\Theta(\boldsymbol{\mu}^{t+1}, \boldsymbol{\eta}^{t+1}) = lb(Z) - \epsilon$ for a very small $\epsilon > 0$. If the gap still is too large, we have to perform more primal iterations.

4.7 Upper Bound Evaluation

In Algorithm 1, for each branch an upper bound is computed. As the algorithm terminates, when the gap between the upper bound and the lower bound is small, it is important to compute good upper bound values efficiently.

In this section we apply the block coordinate descent method as shown in Section 2.6 to compute an upper bound. Hence, we have to efficiently compute the minimum $\operatorname{argmin}\{f_k(z, \pi_k(\boldsymbol{z})) : z \in Z_k\}$ of the subproblem if all variables except z_k are fixed. From (4.1) we know that $f_k(z, \pi_k(\boldsymbol{z})) = \psi[a, u, b]$ with $a > 0$ and Lemma 4.26 shows, how to solve the subproblem.

Lemma 4.26. *Let $\Gamma = [\alpha, \beta]$ be an angle set, $Z = \exp(\imath\Gamma)$ and $\psi = \psi[a, u, b]$ with $a > 0$. Then we can efficiently compute*

$$Z^* := \operatorname*{argmin}_{z \in Z} \psi(z).$$

In particular:

- *For $u = 0$: ψ is constant on $\partial\mathcal{U}$ and $Z^* = Z$.*
- *For $u \neq 0$ and $u_0 := -\text{Normed}[u] \in Z$ we get $Z^* = \{u_0\}$.*
- *For $u \neq 0$ and $u_0 := -\text{Normed}[u] \notin Z$ we get*

$$Z^* = \operatorname*{argmin}_{w \in \{\exp(\imath\alpha), \exp(\imath\beta)\}} |w - u_0|$$

Proof. In the case $u = 0$ by definition $\psi(z)$ is constant on $Z \subset \partial\mathcal{U}$ and the statement holds.

For $u \neq 0$ from Remark 2.17 we get

$$\psi(z) = a \left| z + \frac{u}{a} \right|^2 - \frac{|u|^2}{a} + b$$

which is minimal for minimal value of $\left| z + \frac{u}{a} \right|$, i.e. for the $z \in Z$ with minimal distance to $\frac{u}{a}$. Figure 4.5. visualizes this situation.

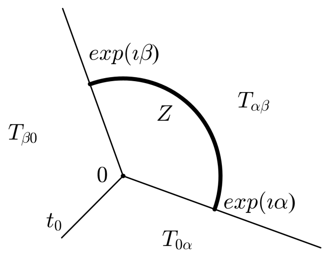


Figure 4.5: Possible cases to consider. t_α is the ray through $\exp(\imath\alpha)$, t_β is the ray through $\exp(\imath\beta)$ and t_0 is the angle bisector of the angle from β to α. The closed areas between the rays are denoted by $T_{\alpha\beta}$, $T_{\beta 0}$ and $T_{0\alpha}$.

Set $u_0 := -\text{Normed}[u] = -\frac{u}{|u|}$ and $u_1 := -\frac{u}{a}$. Then u_0 and u_1 are on the same ray from zero and thus u_0 and u_1 are in the same area of $T_{\alpha\beta}$, $T_{\beta 0}$ and $T_{0\alpha}$.

In particular, $u_1 \in T_{\alpha\beta}$ if and only if $u_0 \in Z$. In this case, u_0 is the closest point to u_1.

Otherwise if $u_1 \in T_{\beta 0}$ the closest point to u_1 in Z is $\exp(\imath\beta)$. Similar if $u_1 \in T_{0\alpha}$ the closest point to u_1 in Z is $\exp(\imath\alpha)$. Especially, if u_1 is on the ray t_0, it has the same distance to $\exp(\imath\alpha)$ and $\exp(\imath\beta)$. $\square$

Therefore, we can apply the block coordinate descent method stated in Section 2.6 to the upper bound problem. The assumptions of Theorem 2.35 are not satisfied as the solution is not unique. Hence, the argument is not guaranteed to converge. However, as ψ is bounded from below, by Remark 2.36 the function values converge and the algorithm terminates. As the method is feasible in every iteration, the computed value is a valid upper bound for the optimal value.

4.8 Numerical Results

In this section we present the computational results of the branch and bound algorithm for the circle rotation problem.

4.8.1 Evaluation Settings

4.8.1.1 Problem Instances

For the numerical evaluation we chose the instances described in Section 3.6.1.2 in detail. We briefly summarize how they were generated. For each of the circuits in Section 2.8 denote by A the sum of all circle areas. Then a problem instance for the rotation problem is created as follows. For a circuit instance with area A and an area factor $a_f \geq 0$ the centers of the circles are independently uniformly distributed in a square with area $a_f A$. For small area factors the circles are likely to overlap, while for large area factors they are less dense. We chose the area factors

$$a_f \in \{0,\ 0.001,\ 0.002,\ 0.005,\ 0.01,\ 0.02,\ 0.05,\ 0.1,\ 0.2,\ 0.5,\ 1,\ 2,\ 5,\ 10\}.$$

4.8.1.2 Algorithms

We evaluate the branch and bound algorithm with different settings. Refer to the notation of Algorithm 1 on page 66.

- Initial symmetry breaking is done as described in Section 4.3.

- We choose the element for branching as described in Section 4.4.1. For the branching scheme on one element, there are the two options described in Section 4.4.2 and shown in Figure 4.1. The domain reducing branching scheme is shown in Figure 4.1a, the lower bound improvement branching scheme is shown in Figure 4.1b.
- We can either do no domain reduction or the geometric domain reduction described in Section 4.5. Due to the lack of robust computational geometry libraries, we did not exactly implement the domain reduction as described in Section 4.5 but used outer convex polygonal approximations instead of splinegons. Details of our implementation are described in Section 4.8.1.3.
- Lower bound and upper bound evaluation are done by the block coordinate descent methods described in Section 4.6 and Section 4.7. The algorithm terminates, if during the last n single step optimizations the total improvement is less than 10^{-8}. This high precision is necessary, as we solve the global problem up to a very high accuracy. Numerical issues are not considered.

  For each branch, the lower and upper bound and the corresponding solution vectors are stored. Additionally, the values do not have to be reevaluated, if the previous solution to the lower bound resp. feasible solution is still feasible for the new branch.
- We set the target approximation quality to $\varepsilon_T = 10^{-8}$. Furthermore, we used a time limit of $t_{max} = 1h$.

4.8.1.3 Implemented Domain Reduction Algorithm

Implementing the algorithms in Section 4.5.1 robustly with respect to numerical inaccuracies is difficult. Due to the lack of existing libraries, we used a different implementation.

Recall that for some k we have compute

$$\Gamma_k(Z) = -\text{Normed}\,[A] \cap Z_k \qquad \text{with} \qquad A = \sum_{j \neq k} q_{kj} \operatorname{conv}(Z_j).$$

The hardest task is to compute the Minkowski sum efficiently. Obviously summands with $q_{kj} = 0$ can be ignored.

The first idea is that we can use outer polygonal approximations of the splinegons $q_{kj} \operatorname{conv}(Z_j)$. Especially for small domain sets, these polygons can be good a approximation with even a small number of nodes. The Minkowski sum of convex polygons is a standard algorithm in computational geometry, see e.g. [Berg et al. 2000, Chapter 13]. However, for the full domain set $Z_j = \partial\mathcal{U}$, the set $q_{kj} \operatorname{conv}(Z_j)$ is a disc. But the polygonal approximation of a disc with high precision requires polygons with a large number of nodes.

The second idea is that the Minkowski sum of discs and fixed points is very easy to compute. Denote by J_{full} the indices of the variables with $q_{kj} \neq 0$ and $Z_j = \partial\mathcal{U}$ and by J_{fixed} the indices of the fixed variables with $q_{kj} \neq 0$. Furthermore, let $w_j \in \partial\mathcal{U}$, $j \in J_{fixed}$ be their fixed value. Denote by J_{proper} the indices of the variables with $q_{kj} \neq 0$ and Z_j neither a single element nor a full circle.

In our implementation we first compute the Minkowski sum of all fixed angles and all full circles.

$$A' := \sum_{j \in J_{full}} q_{kj} D(0,1) + \sum_{j \in J_{fixed}} q_{kj} w_j = D\left(\sum_{i \in J_{fixed}} q_{kj} w_j, \sum_{j \in J_{full}} |q_{kj}| \right).$$

Now there are two cases. If $J_{proper} = \emptyset$, we are done and can set $A = A'$.

Otherwise we create outer polygonal approximations W_j to $q_{kj} \operatorname{conv}(Z_j)$, $j \in J_{proper}$ and W to A'. Then we compute the Minkowski sum

$$A := \sum_{j \in J_{proper}} q_{kj} W_j + W.$$

For both cases the aperture angle of A can be easily computed. Thus, as $\Gamma_k(Z)$ is the intersection of the aperture angle of A and Z_k, it can be computed as described above.

4.8.2 Measured Values

For each of the problem instances we ran each of the algorithms once. As the algorithm is deterministic and for most problem instances the running time is long enough, there is no need for multiple runs.

The following values were measured:

Objective value ub_A: The best feasible objective value ub^* at the end of Algorithm 1.

Lower bound value lb_A: The lowest lower bound lb^* in the set of domain sets at the end of Algorithm 1.

Approximation quality ε_A: The approximation quality is $\varepsilon_A = ub_A/lb_A - 1$.

CPU time t_A: The total CPU time consumed by the algorithm.

Number of iterations: This is the number of processed nodes in the search tree, which is the number of iterations of the loop in Algorithm 1, plus one for the root node.

CPU time for domain reduction t_{dr}: The total CPU time used for domain reduction.

CPU for upper bound evaluations t_{ub}: The total CPU time used for upper bound evaluation. Note that not always an upper bound evaluation has to be performed, as sometimes the solution of the previous branch is feasible for the new branch and such a reevaluation was not required.

CPU time for lower bound evaluations t_{lb}: The total CPU time used for lower bound evaluation. Again, sometimes the lower bound of the previous branch is feasible to the new branch.

4.8.3 Comparison of the Branching Schemes

We evaluated all instances with and without domain reduction for both branching schemes. We compared CPU time, number of iterations and approximation quality. The results were essentially independent of the branching scheme for all instances and all measured values. So in the remaining part of this evaluation, we only consider the branching scheme for lower bound improvement as shown in Figure 4.1b.

4.8.4 Domain Reducing and Non Domain Reducing Algorithm

Figure 4.6 shows the CPU time, the number of iterations and the final approximation quality for the algorithm with and without domain reduction. The target approximation quality was set to $\varepsilon = 10^{-8}$, and the running time was limited by $t_{max} = 1h$. We observe that the algorithm with domain reduction yields an approximation quality of $\varepsilon_A = 0$ for many instances, such that global optimality of the solution is ensured. However, due to finite precision arithmetic there might still be numerical errors. In order to use a logarithmic scale in the Figures 4.6e and 4.6f, we considered all approximation qualities $\varepsilon_A \leq \varepsilon$ as ε. A violet coloring indicates an instance for which $\varepsilon_A = 0$ is achieved.

Recall the problem structure analyzed in Section 3.6.2. We know that the problem is obviously harder for a larger number of circles. Furthermore, it is harder for smaller area factors. The smaller the area factor and the larger the number of circles, the larger is the number if local optima. However, for area factors around 0.005, the ratio between the worst and the best local optimal value is maximal.

In Figure 4.6 we see that the behavior of the branch and bound algorithm is related to this problem structure. We now explain these graphs. Note that the algorithm terminates with running time $t_A \leq t_{max}$ if and only if the approximation quality is $\varepsilon_A \leq \varepsilon$.[1] For simplification we say the algorithm converges if and only if it reaches the target approximation quality, i.e. $\varepsilon_A \leq \varepsilon$.

[1]There is the possibility that the algorithm reaches the approximation quality ε in the step, where its running time exceeds t_{max}, so there is not strictly an equivalence. However, this occurrence is very unlikely in practice.

Domain Reducing Algorithm

First we look at the algorithm with domain reduction. The CPU time t_A is shown in Figure 4.6a, the iterations in Figure 4.6c and the final approximation quality ε_A in Figure 4.6e. Figures 4.7 and 4.8 show the approximation quality and CPU time in more detail.

In Figure 4.6a a plateau can be recognized for CPU time $= t_A = 3.6{\cdot}10^{-6} ms = 1h$. This plateau is approximately a rectangle starting at 380 circles and area factor $a_f = 0.002$ and reaching to smaller area factors and larger numbers of circles. For these problem instances, the algorithm did not converge. This can also be seen in Figure 4.6e, as in this area $\varepsilon_A > \varepsilon$. Of course, for these instances the algorithm takes many iterations (only restricted by the time limit). So in Figure 4.6c there also seems to be a plateau. However, as the running time is bounded and an iteration takes longer for larger instances, the value of the plateau is decreasing for increasing number of circles.

At the boundary of the area of non-convergence, the approximation quality improves. So Figure 4.7f shows that the median approximation quality for $a_f = 0.002$ is ten times better than for $a_f = 0.001$, while for $a_f = 0.005$ the algorithm always converges.

Next to this region of non-converging instances the number of iterations and the running time decreases for both for decreasing number of circles and even more significantly for increasing area factors (see Figures 4.8a, 4.8b and 4.8f). While for small area factors, the number of iterations is 10000, it gets constantly 1 for larger area factors. For these instances, the algorithm reaches the desired approximation quality without branching. For almost all of these instances the algorithm guarantees the approximation quality $\varepsilon_A = 0$. In this region the running time only increases for increasing number of circles.

Non Domain Reducing Algorithm

We now look at the algorithm without domain reduction. The CPU time is shown in Figure 4.6b, the iterations in Figure 4.6d and the final approximation quality in Figure 4.6f. Once again we refer to the details shown in Figures 4.7 and 4.8.

Similarly to the previous case, for small area factors there is a region of instances where the algorithm does not converge. This region approximately is a rectangle starting at 19 circles and area factor $a_f = 0.2$ and reaching to smaller area factors and larger numbers of circles. It is obvious that this area is much larger than in the case with domain reduction. In fact, as can be seen in Figure 4.8f the algorithm without domain reduction only converges for large area factors greater or equal to 0.5. Of special interest is the final approximation quality shown in Figures 4.6f and 4.7f. For increasing area factors the approximation quality decreases, even if the algorithm does not converge.

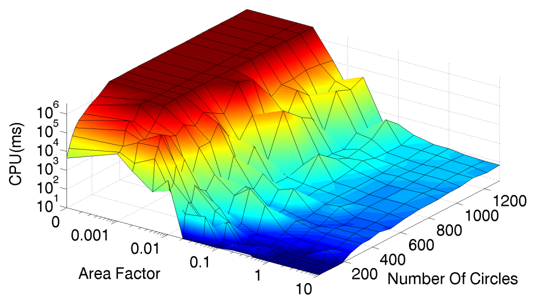
Algorithm Comparison

We now compare both algorithms. Usually introducing an additional domain reduction algorithm reduces the search space and thus the number of iterations. However, this search space reduction comes along with the price of additional computation time per iteration. Therefore, the running time might even increase by an additional domain reduction. However, we show that in our case the time improvement by search space reduction significantly dominates the extra computational effort of the domain reduction.

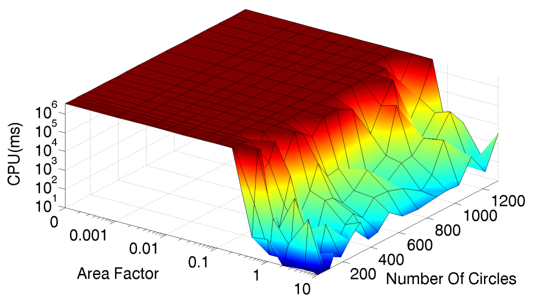
The algorithm with domain reduction always takes less or equal iterations than the algorithm without domain reduction. This is obvious for instances where both algorithm converges, as by the domain reduction some parts of the domain space are reduced and do not have to be explored by branching. For instances where the algorithms do not converge this is due to the fact that the running time is bounded and an iteration with domain reduction takes more time than an iteration without domain reduction.

Next we consider the running times shown in Figures 4.6a, 4.6b and 4.8. In all cases the domain reducing algorithm takes no longer than the non domain reducing algorithm. Equality occurs almost only in the case that neither of the algorithms converges and thus both are stopped by the time limit. For medium area factors (e.g. $a_f = 0.1$, see Figure 4.8d) the domain reducing algorithm takes only between few milliseconds and two seconds, while the non domain reducing algorithm does not even converge in one hour. For larger area factors, where both algorithms converge (e.g. $a_f = 2$, see Figure 4.8e) the running time of the non domain reducing algorithm is larger by orders of magnitude.

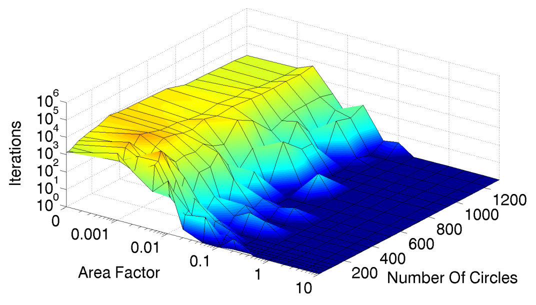
Finally, we compare the approximation quality. For small area factors $a_f \leq 0.002$ and larger problem instances neither of the algorithm converges, see Figures 4.7a and 4.7b. However, for these instances the final approximation quality ε_A of the domain reducing algorithm is significantly better than the approximation quality of the non domain reducing algorithm. For medium area factors the domain reducing algorithm converges and often even guarantees $\varepsilon_A = 0$, while the non domain reducing algorithm does not converge. Finally, for large area factors $a_f \geq 1$ both algorithms converge. However, as can be seen in Figure 4.7e, the domain reducing algorithm guarantees optimality $\varepsilon_A = 0$ while the non domain reducing algorithm does not.



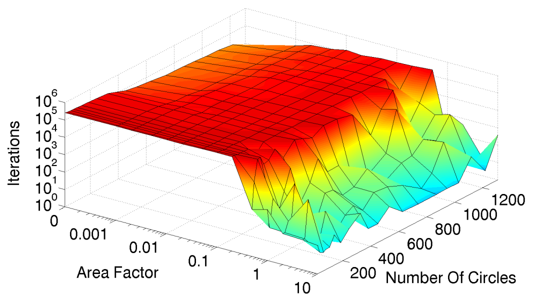
(a) Domain reduction, CPU time t_A in ms.



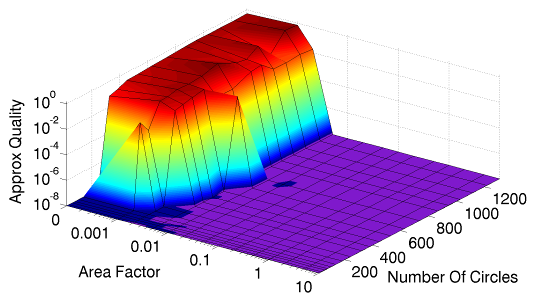
(b) No domain reduction, CPU time t_A in ms.



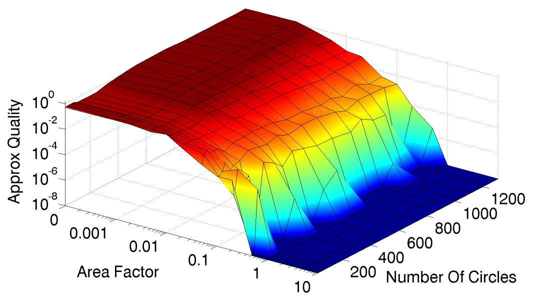
(c) Domain reduction, iterations.



(d) No domain reduction, iterations.

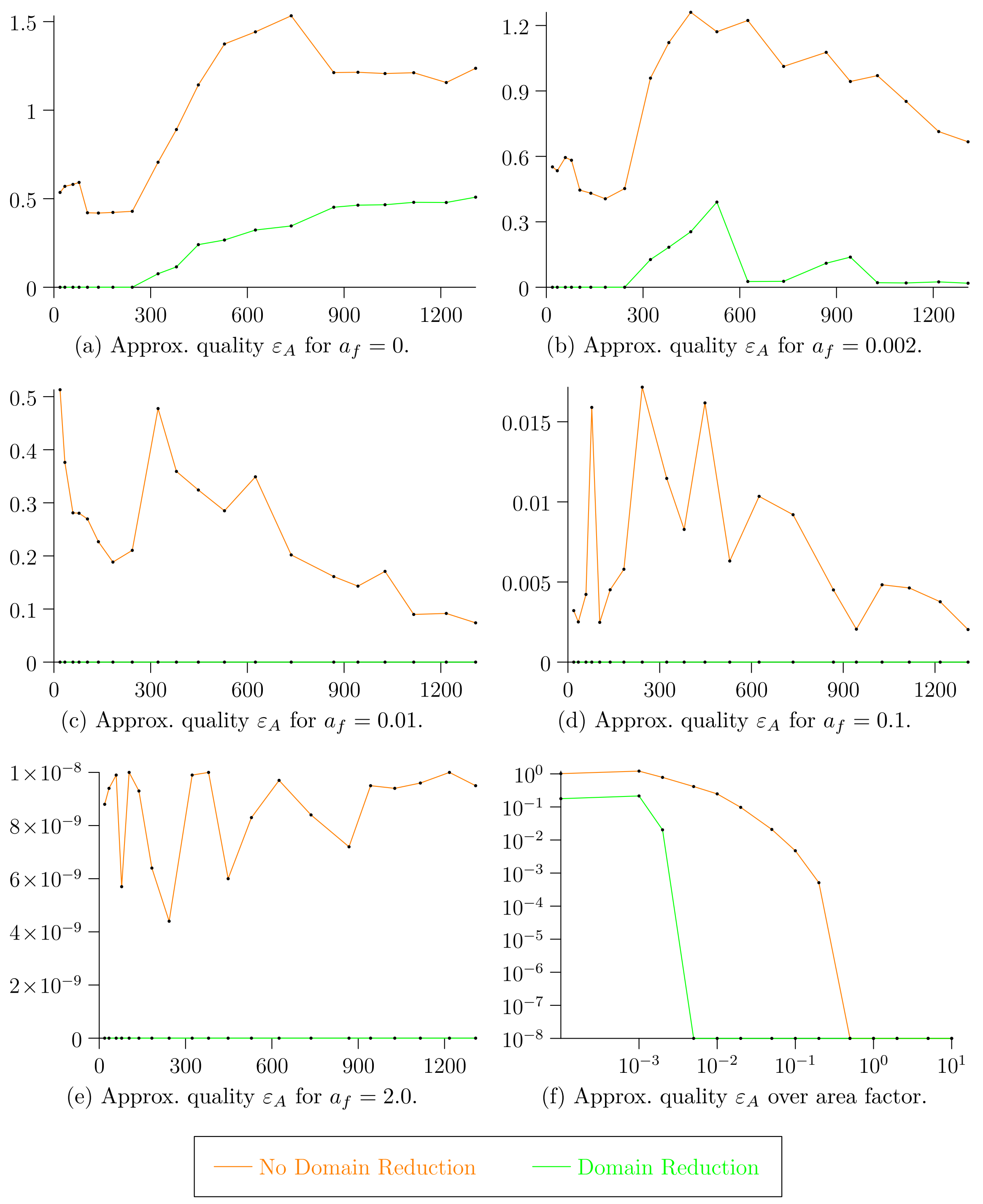


(e) Domain reduction, approximation quality ε_A.



(f) No domain reduction, approximation quality ε_A.

Figure 4.6: CPU time, iterations and approximation quality ε_A of the algorithm with and without domain reduction for target approximation quality $\varepsilon = 10^{-8}$ and run time limit $t_{max} = 1h = 3.6 \cdot 10^{-6} ms$. Note the logarithmic scale on the z-axis. In (e) the solutions with guaranteed optimality $\varepsilon_A = 0$ are colored violet.



(a) Approx. quality ε_A for $a_f = 0$.

(b) Approx. quality ε_A for $a_f = 0.002$.

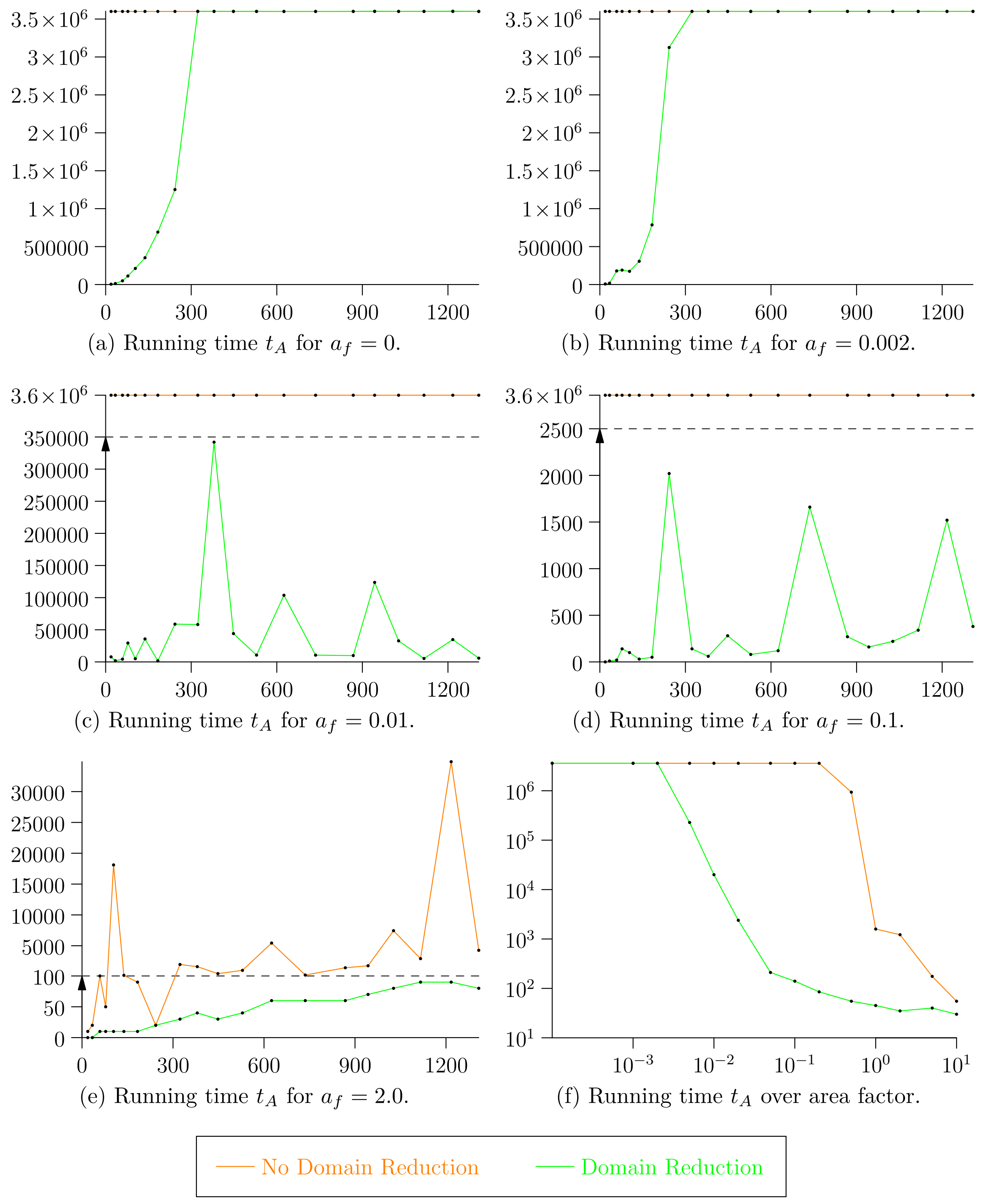
(c) Approx. quality ε_A for $a_f = 0.01$.

(d) Approx. quality ε_A for $a_f = 0.1$.

(e) Approx. quality ε_A for $a_f = 2.0$.

(f) Approx. quality ε_A over area factor.

Figure 4.7: In (a) – (e): For different area factors a_f the approximation quality ε_A (on the y-axis) is plotted over the number of circles (on the x-axis). In (f): Median of approximation qualities ε_A on the y-axis is plotted over the area factors (on the x-axis). The median is taken over all approximation qualities of runs for this area.



(a) Running time t_A for $a_f = 0$.

(b) Running time t_A for $a_f = 0.002$.

(c) Running time t_A for $a_f = 0.01$.

(d) Running time t_A for $a_f = 0.1$.

(e) Running time t_A for $a_f = 2.0$.

(f) Running time t_A over area factor.

Figure 4.8: In (a) – (e): For different area factors a_f the running time t_A (on the y-axis) is plotted over the number of circles (on the x-axis). In (f): Median of running times t_A on the y-axis is plotted over the area factors (on the x-axis). The median is taken over all running times of runs for this area. Note the scale break in (c) to (e).

4.8.5 Detailed Analysis of the Domain Reducing Algorithm

As the domain reducing algorithm outperforms the non domain reducing algorithm for all measured values and all instances, we analyze it in more detail.

Performance Analysis

In Figure 4.9 the median of the relative amount of time used in different parts of the algorithm is shown. Note that due to technical limitations, measured times are always a multiple of 10ms and thus very imprecise for the short running times of the algorithm for area factors $a_f \geq 0.1$. However, one recognizes that most of the time is spent in the domain reduction algorithm, approximately 90 to 100 percent. Lower bound and upper bound evaluation approximately require the same amount of time.

The small increase of the time used for lower bound and upper bound evaluation for $a_f = 0$ is plausible. These bound evaluations are done by the block coordinate descent method. Especially the upper bound evaluation is very similar to the method described in Section 3.4.3. Therefore, recall the running time behavior of the block coordinate descent method shown in Figure 3.9, where a significant increase in running time can be observed for the area factor $a_f = 0$. Thus it is plausible that the upper bound evaluation takes more time for $a_f = 0$.

The domain reduction algorithm exerts a dominating influence on the running time. Due to the lack of a robust implementation, we used outer polygonal approximations of the domain sets instead of splinegons. However, compared to an exact splinegon representation, these polygons have many vertices. We expect a significant decrease of the running time, if a fast and robust splinegon implementation for the geometric domain reduction is used.

Lower and Upper Bound Progression

Figure 4.10 shows the progression of the upper and lower bound over the number of iterations for the circuit instance D1217. We observe that the upper bound almost never decreases visibly (with an exception in Figure 4.10d). The algorithm tends to find a good solution very early and spends most of the time by increasing the lower bound. This is plausible from the analysis of the problem structure in Section 3.6.2, as the ratio between the best and the worst local minimum is small. Furthermore, as can be seen in Figure 3.5, the block coordinate descent method used for upper bound evaluation tends to find very good solutions very often.

For the area factors $a_f = 0$ and $a_f = 0.001$ the initial lower bound is very low and improves very slowly. The larger the area factor is, the better is the lower bound in the first iteration. This can be reasoned similarly to the analysis in Section 3.6.2 and can be deviated from Theorem 3.8. The objective function is a Hermitian form created

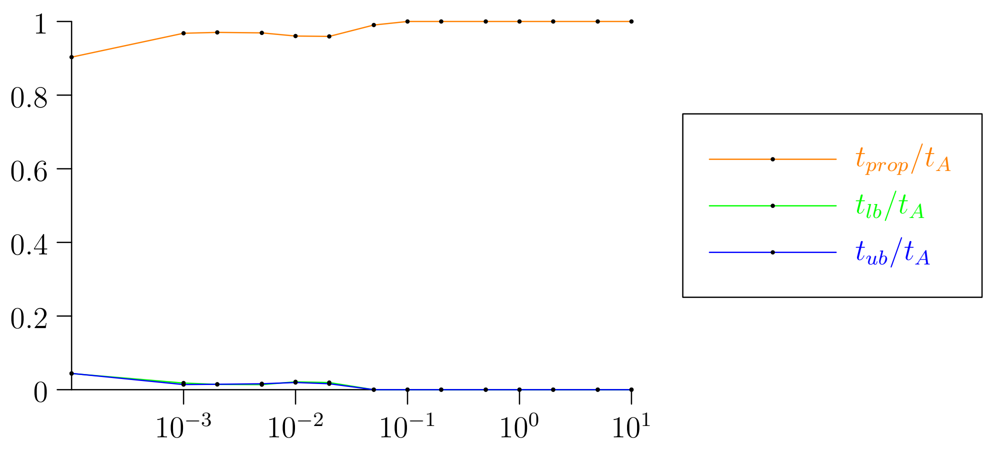


Figure 4.9: Comparison of the time used for domain reduction, lower bound evaluation and upper bound evaluation. The median of the relative domain reduction time, lower bound and upper bound evaluation time (t_{dr}/t_A, t_{lb}/t_A, t_{ub}/t_A) are plotted versus the area factors. Due to the limited precision of measurement for times below $10ms$, values for $a_f \geq 0.1$ are not significant.

as shown in Theorem 3.6. For increasing area factor the last variable related to the column $(\boldsymbol{u}, b)^H$ mostly has maximal impact and thus is fixed. After fixing one variable, the function has the form

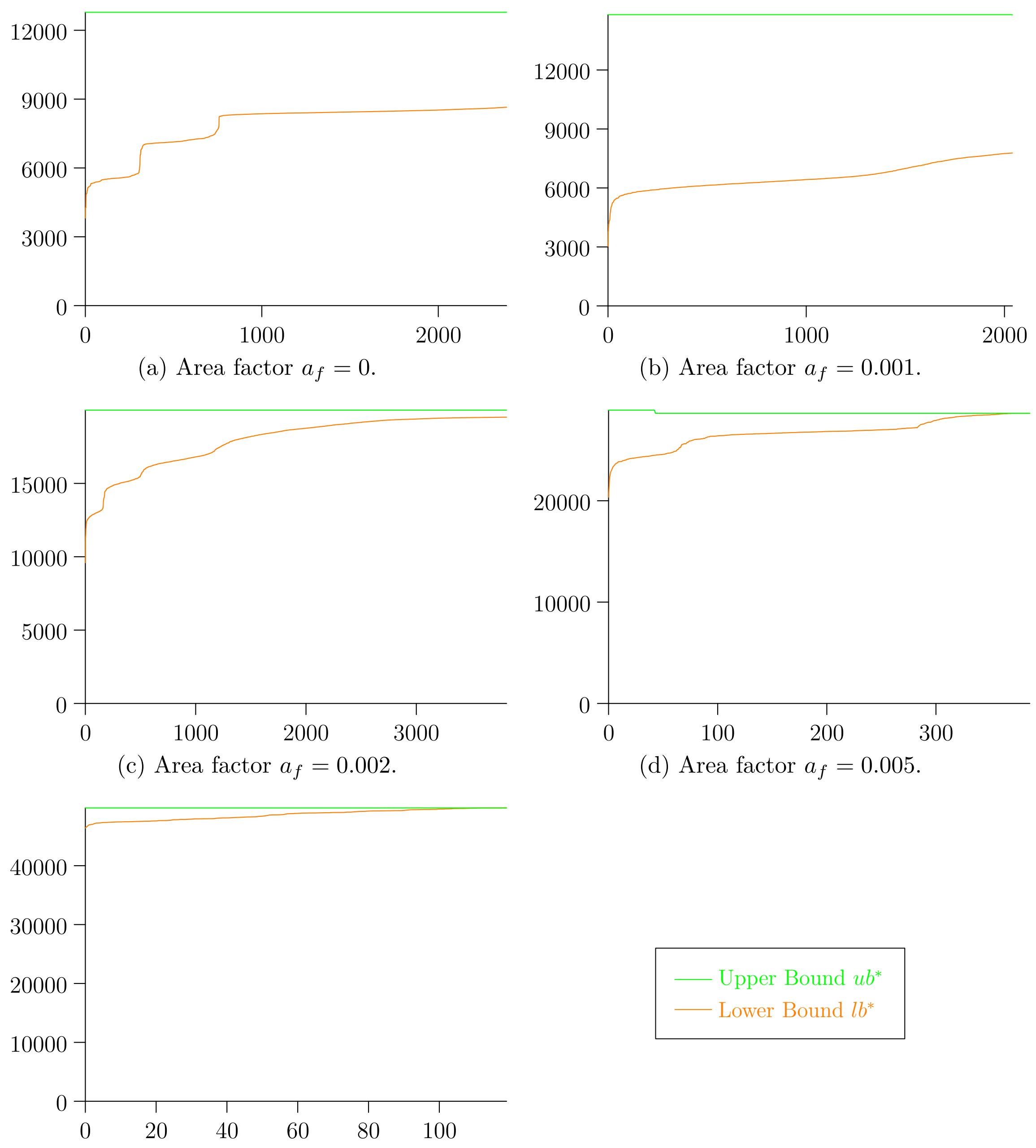
$$f(\boldsymbol{z}) = b + \boldsymbol{u}^H \boldsymbol{z} + \boldsymbol{z}^H \boldsymbol{u} + \boldsymbol{z}^H A \boldsymbol{z}.$$

Geometrically, for larger area factors the circles are less likely to overlap and thus it is more likely that the optimal solution of the relaxed problem is at the boundary. Furthermore, for increasing area factor the constant b (and also $\boldsymbol{u}$) becomes large. As the relaxation takes lower values only for the last term $\boldsymbol{z}^H A \boldsymbol{z}$, for increasing b and $\boldsymbol{u}$ the approximation quality improves.

Eventually for area factors $a_f \geq 0.005$ the initial lower bound starts close to the upper bound and converges very quickly.

4.8.6 Conclusion

In this section we evaluated the branch and bound algorithm for the circle rotation problem numerically. We showed that the algorithm with domain reduction is significantly superior to the algorithm without domain reduction in terms of number of iterations, CPU time and approximation quality. For larger area factors, the algorithm with domain reduction is even able to guarantee global optimality without branching. We also identified the domain reduction step to be the most time consuming part of the algorithm. However, we used polygonal approximations of the domain sets and the performance of the domain reduction might be improved by using more efficient implementations with splinegons or circular arc polygons.



(a) Area factor $a_f = 0$.

(b) Area factor $a_f = 0.001$.

(c) Area factor $a_f = 0.002$.

(d) Area factor $a_f = 0.005$.

(e) Area factor $a_f = 0.01$.

Figure 4.10: Behavior of the domain reducing algorithm for the circuit instance D1217. For different area factors a_f the upper bound ub^* and lower bound lb^* (on the y-axis) are plotted versus the number of iterations (on the x-axis). Note the different scales on both axes.

5 Single Net Circle Rotation Problem

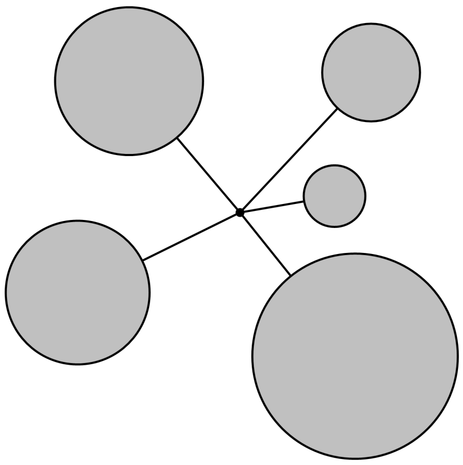


Figure 5.1: The single net circle rotation problem.

In this chapter we consider the circle rotation problem defined in Chapter 3, where circles are only connected by one net. For this problem a set of fixed centered, connected circles is given. There is one connection point (pin) per circle and all pins are connected by the same net. The single net circle rotation problem is to find a rotation of the circles such that the connection length is minimized.

The single net circle rotation problem in the star wire length model is equivalent to the following facility location problem: For a set of circles in the plane, find a point with minimal squared Euclidean distances to the boundaries of the circles. This facility location problem has recently been studied in literature and a big square small square algorithm has been presented in [Nezakati; Zaferanieh; Fathali 2009].

In this chapter we identify problem instances, where the problem can be solved to optimality efficiently. In Section 5.1 we formalize the single net circle rotation problem considered in this chapter. We show that the single net circle rotation problem and the described facility location problem are equivalent. In Section 5.2 we summarize the rare results for this problem that are known in the literature. In Section 5.3 we show that the problem can asymptotically be solved to optimality for instances, where the circles are small compared to their distances. In Section 5.4 we state a modification of the big square small square algorithm to solve the single net circle rotation problem to global optimality.

5.1 Problem Statement

We consider the circle rotation problem for circuits containing only a single net. Then w.l.o.g. we can assume the net to have weight $\mu' = 1$ in the star model. We further assume that each circle has exactly one connection pin to the net. Then, possibly by rotation, we can assume the offset of this pin to be real and non-negative. For some reason explained later, we denote this pin offset by $r_j \in \mathbb{R}^{\geq 0}$.

More formally, given is:

- A set $\mathcal{C}$ of n circles with fixed center positions $c_j \in \mathbb{C}$. The vector of the circle centers is $\boldsymbol{c} = (c_1, \dots, c_n)^T$.
- A set $\mathcal{P}$ of n pins with offset $r_j \in \mathbb{R}^{\geq 0}$ from the center of circle j. The vector of pin offsets is $\boldsymbol{r} = (r_1, \dots, r_n)^T$.

All pins are in the same net with weight $\mu' = 1$ in the star model. The single net circle rotation problem then is to rotate the circles such that the wire length is minimized.

Intuitively and by the results of Section 3.5 it can be seen that for fixed net centers it is possible, to rotate every circle in an optimal way. So for fixed net center s, the wire length in the strict model should be the sum of the squared distances from s to the circle boundaries. The wire length in the relaxed model should be the sum of the squared distances from s to the circle discs.

We define the problem based on this geometric intuition and then show that this problem is indeed equivalent to the single net circle rotation problem for the strict and the relaxed case.

Definition 5.1. *Let there be a circle with radius $r \in \mathbb{R}^{\geq 0}$ around the center $c \in \mathbb{C}$. Let $s \in \mathbb{C}$.*

- *The outer distance $f^{out}_{c,r}$ of s is the squared distance from s to the circle boundary if s is outside the circle and 0 otherwise. This is the squared distance from s to the circle disc.*
- *The inner distance $f^{in}_{c,r}$ of s is the squared distance from s to the circle boundary if s is inside the circle and 0 otherwise.*
- *The distance $f^{sum}_{c,r}$ of s is the squared distance from s to the circle boundary.*

Then it is

$$\begin{aligned} f^{out}_{c,r}(s) &= \max\left(|s-c| - r, 0\right)^2, \\ f^{in}_{c,r}(s) &= \min\left(|s-c| - r, 0\right)^2, \\ f^{sum}_{c,r}(s) &= (|s-c| - r)^2 = f^{out}_{c,r}(s) + f^{in}_{c,r}(s). \end{aligned}$$

Furthermore, we denote $f_j^{out} := f_{c_j,r_j}^{out}$ and similar $f_j^{in} := f_{c_j,r_j}^{in}$ and $f_j^{sum} := f_{c_j,r_j}^{sum}$. Finally, we set $f^{out}(s)$, $f^{in}(s)$ and $f^{sum}(s)$ as the sum over all circles.

Definition 5.2 (Point Location Problem)**.** *Given is a set of circles with centers c_j and radius r_j, $j = 1, \dots, n$.*

The relaxed point location problem *is to minimize f^{out}. Geometrically this means to find a point s such that the sum of squared Euclidean distances to the circle discs is minimized.*

The strict point location problem *is to minimize f^{sum}. Geometrically this means to find a point s such that the sum of squared Euclidean distances to the circle boundaries is minimized.*

Before we show the equivalence of the point location problem and the single net circle rotation problem we repeat some definitions of Section 3.5 in this context of a single net. It is $\mathrm{wl}_{star}^{j}(z_j, s)$ the wire length of the connection of circle j to the net center depending on the circle rotation z_j and the net center s. Then $\mathrm{wl}_{star}^{s,j}(s)$ and $\mathrm{wl}_{star}^{r,j}(s)$ is the minimum of these wire lengths depending on the net centers in the strict model (with $z_j \in \partial\mathcal{U}$) resp. the relaxed model (with $z_j \in \mathcal{U}$). Finally, $\mathrm{wl}_{star}^{s}(s)$ and $\mathrm{wl}_{star}^{r}(s)$ are the sum of all wire lengths.

Lemma 5.3. *It is* $\mathrm{wl}_{star}^{s,j} = f_j^{sum}$ *and* $\mathrm{wl}_{star}^{r,j} = f_j^{out}$.

Proof. By Theorem 3.14 we know

$$\mathrm{wl}_{star}^{j}(z_j, s) = \psi[a_j, u_j(s), b_j(s)](z_j) =: \psi(z_j)$$

with $a_j = r_j^2$, $u_j(s) = r_j(s - c_j)$ and $b_j(s) = |s - c_j|^2$.

As in Lemma 2.19 denote the minimum of ψ on $\partial\mathcal{U}$ by ψ_s^* and the minimum of ψ on $\mathcal{U}$ by ψ_r^*.

We first consider the strict problem. By Lemma 2.19 it is

$$\begin{aligned}\mathrm{wl}_{star}^{s,j}(s) &= \min_{z_j \in \partial\mathcal{U}} \mathrm{wl}_{star}^{j}(z_j, s) = \psi_s^* = a_j + b_j(s) - 2\,|u_j(s)| \\ &= r_j^2 + |s - c_j|^2 - 2r_j\,|s - c_j| = (|s - c_j| - r_j)^2 = f_j^{sum}(s).\end{aligned}$$

We now consider the relaxed problem. Then $|s - c_j| \geq r_j \iff u_j(s) \geq a_j$.

Case 1: $|s - c_j| \geq r_j$. Then $u_j(s) \geq a_j$ and by Lemma 2.19

$$\begin{aligned}\mathrm{wl}_{star}^{r,j}(s) &= \min_{z_j \in \mathcal{U}} \mathrm{wl}_{star}^{j}(z_j, s) = \psi_r^* = \psi_s^* \\ &= (|s - c_j| - r_j)^2 = \max\left(|s - c_j| - r_j, 0\right)^2 = f_j^{out}(s).\end{aligned}$$

Case 2: $|s - c_j| < r_j$. Then $u_j(s) < a_j$ and by Lemma 2.19

$$\begin{aligned}
\mathrm{wl}^{r,j}_{star}(s) &= \min_{z_j \in \mathcal{U}} \mathrm{wl}^{j}_{star}(z_j, s) = \psi^*_r \\
&= -\left(\frac{|u_j(s)|^2}{a_j}\right)^2 + b_j(s) = -\left(\frac{r_j^2 \, |s - c_j|^2}{r_j^2}\right)^2 + |s - c_j|^2 \\
&= 0 = \max\left(|s - c_j| - r_j, 0\right)^2 = f^{out}_j(s).
\end{aligned}$$

□

Corollary 5.4. *As an immediate consequence we get* $\mathrm{wl}^{s}_{star} = f^{sum}$ *and* $\mathrm{wl}^{r}_{star} = f^{out}$.

So, indeed the point location problem in Definition 5.2 is equivalent to the single net circle rotation problem.

5.2 Literature Survey

The point location problem stated in Definition 5.2 is a facility location minisum problem. Such problems have been analyzed in the literature.

In [Coope 1993] the problem to place a circle with variable radius and variable center that minimizes the distance to a set of points is analyzed. They analytically compute the optimal radius and show that the problem is equivalent to a variance problem. In [Drezner; Steiner; Wesolowsky 2002] this problem is considered for several distance measures. Furthermore, they consider a fast heuristic for the squared Euclidean distance problem.

In [Brimberg; Juel; Schöbel 2009] they consider the problem to place a circle with variable or fixed radius and variable center such that the distance to a set of given points is minimized. However, they analyze the Euclidean metric. Also in [Körner et al. 2009] they extend the results to general norms, but they do not consider the squared Euclidean metric.

In [Nezakati; Zaferanieh; Fathali 2009] they consider the general problem of placing a point such that the summed weighted Euclidean distance to the boundary of given circles is minimized. This is exactly the point location problem stated in Definition 5.2. As the problem is not convex, they apply the well known big square small square (BSSS) algorithm to this problem.

5.3 Asymptotically Optimal Algorithm

In this section we identify special instances where there is an easy algorithm to find the global minimum. We first sketch the algorithmic idea.

We know that the relaxed problem to minimize f^{out} is a convex optimization problem and, thus, can be solved to optimality. So the strict objective f^{sum} is the sum of a convex part f^{out} and a non-convex part f^{in}. Additional f^{in} is the sum of non-negative functions with compact support. The idea is that we first optimize f^{out} and take this as an initial solution to the optimization of $f^{sum} = f^{out} + f^{in}$. However, we only have to consider the summands of f^{in} which are non-zero at the minimum of f^{out}.

In Section 5.3.1 we generalize the circle rotation problem. In Section 5.3.2 we apply the obtained results to the rotation problem. To make the section more intuitive, we first give an outlook to the application of each statement.

5.3.1 Generalization of the Rotation Problem

In Lemma 5.5 we analyze, how the minimum of a convex function changes, if we add a non-negative function with bounded support. This allows us to estimate the region containing a minimum of $f^{out} + f^{in}$ based on the minimum of f^{out}. The main result of this lemma is that when f^{in} has bounded support which contains the minimum of f^{out}, the minimum of $f^{out} + f^{in}$ remains within the support.

In Lemma 5.6 we exploit the fact that f^{in}_j is concentric to the circle center and equal to zero at the circle boundary. We show that if circle j contains the minimum s_{out} of f^{out}, the minimum of $f^{out} + f^{in}_j$ is closer to the circle boundary than s_{out}.

In Lemma 5.7 for a special structure of f^{out} we show, how to find the global minimum of $f^{out} + f^{in}$. This allows us to achieve an asymptotically optimal algorithm for special instances of the single net circle rotation problem.

Lemma 5.5. *Let $G \subset \mathbb{R}^n$ and $f : G \to \mathbb{R}$ be a convex function and $g : G \to \mathbb{R}^{\geq 0}$ be a non-negative function. Let s_0 be a minimum of f. Then*

(i) If $g(s_0) = 0$, s_0 is also a minimum of $f + g$.
If f is strictly convex, s_0 is the unique minimum of $f + g$.

(ii) Let $U \subset G$. If $g(s) = 0$ for $s \in \partial U$ and $s_0 \in \overline{U}$, then a minimum of $f + g$ is in $\overline{U}$. If f is strictly convex, each minimum of $f + g$ is in $\overline{U}$.

(iii) Let $U \subset G$ and $g(s) = 0$ for $s \in \partial U$ and $s_0 \in \overline{U}$. Then

$$\min_{s\in G}(f + g)(s) = \min_{s\in G}(f + g \cdot 1_U)(s),$$
$$\operatorname*{argmin}_{s\in G}(f + g)(s) \subset \operatorname*{argmin}_{s\in G}(f + g \cdot 1_U)(s),$$

and if f is strictly convex

$$\operatorname*{argmin}_{s\in G}(f+g)(s) = \operatorname*{argmin}_{s\in G}(f+g\cdot 1_U)(s).$$

Proof. Proof of (i): For all $s \in G$ it is

$$(f+g)(s) \geq f(s) \overset{(*)}{\geq} f(s_0) = (f+g)(s_0).$$

If f is strictly convex, for $s \neq s_0$ in $(*)$ strict inequality holds, so s_0 is the unique minimum.

Proof of (ii): Let s_1 be a minimum of $f+g$. Assume $s_1 \notin \overline{U}$. Let s_2 be an intersection of ∂U and the line from s_0 to s_1, i.e. it is $s_2 = \lambda s_0 + (1-\lambda)s_1$ for some $\lambda \in (0,1)$. With $g(s_2) = 0$ and by convexity of f

$$\begin{aligned}(f+g)(s_2) = f(s_2) = f(\lambda s_0 + (1-\lambda)s_1) &\overset{(*)}{\leq} \lambda f(s_0) + (1-\lambda)f(s_1)\\ &\overset{(**)}{\leq} \lambda f(s_1) + (1-\lambda)f(s_1) = f(s_1) \leq (f+g)(s_1).\end{aligned}$$

So $s_2 \in \overline{U}$ is a minimum of $f+g$.

If f is strictly convex, in $(*)$ and $(**)$ strict inequality holds, so $(f+g)(s_2) < (f+g)(s_1)$ which contradicts the assumption of s_1 being a minimum.

Proof of (iii): Set $g_U = g\cdot 1_U$. There exists a minimum s_1 of $f+g_U$ in $\overline{U}$, i.e. $(f+g_U)(s_1) = \min_{s\in G}(f+g_U)(s)$. As $f+g_U \leq f+g$ it is

$$\min_{s\in G}(f+g)(s) \leq (f+g)(s_1) = (f+g_U)(s_1) = \min_{s\in G}(f+g_U)(s) \leq \min_{s\in G}(f+g)(s),$$

and so equality holds in each step. Let now $s' \in \operatorname{argmin}_{s\in G}(f+g)(s)$,

$$\min_{s\in G}(f+g_U)(s) \leq (f+g_U)(s') \leq (f+g)(s') \leq \min_{s\in G}(f+g)(s),$$

i.e. equality holds and especially

$$(f+g_U)(s') = \min_{s\in G}(f+g_U)(s) \implies s' \in \operatorname*{argmin}_{s\in G}(f+g_U)(s).$$

If now f is strictly convex, by applying (ii) to both $f+g$ and $f+g_U$, each minimum of $f+g$ and $f+g_U$ is in $\overline{U}$. As both functions are equal there, their minima are equal. □

Lemma 5.6. *Let $U \subset G \subset \mathbb{R}^n$ with U being the closed ball with radius r around the origin. Let $f: G \to \mathbb{R}$ be a strictly convex function and $g: G \to \mathbb{R}^{\geq 0}$ be a non-negative function only depending on $||s||$, i.e. $g(s) = h(||s||)$. Assume h is monotonic decreasing on $[0,r]$ and $h(r) = 0$.*

Let $s_0 \in \overline{U}$ be a minimum of f and s_1 be a minimum of $f+g$. Then it is

$$\begin{aligned}&||s_0|| \leq ||s_1|| \leq r,\\ &(f+g)(s_0) - (f+g)(s_1) \leq g(s_0).\end{aligned}$$

Proof. By Lemma 5.5 we have $||s_1|| \leq r$. Furthermore, it is

$$h(||s_1||) = g(s_1) = (f+g)(s_1) - f(s_1) \leq (f+g)(s_0) - f(s_0) = g(s_0) = h(||s_0||)$$

and by monotony of h it is $||s_1|| \geq ||s_0||$. It follows

$$(f+g)(s_0) - (f+g)(s_1) = f(s_0) - f(s_1) + g(s_0) - g(s_1) \leq g(s_0).$$

$\square$

Lemma 5.7. *Let U be the unit circle and $U \subset G \subset \mathbb{R}^2$. Let be $(x_0, y_0) \in U$ and $x_0 \geq 0$, $y_0 \geq 0$. Let f be an elliptic paraboloid*

$$f : G \to \mathbb{R}, \qquad f(x,y) = \frac{(x-x_0)^2}{a^2} + \frac{(y-y_0)^2}{b^2},$$

$g(x,y) = (1 - ||(x,y)||)^2$ and $h = f + g$. Let $V = \{(x,y) : x \geq x_0, y \geq y_0, x^2 + y^2 \leq 1\}$. Then:

(i) There is a global minimum of h in V.

(ii) If $x_0 = 0$ or $y_0 = 0$ the minimum can analytically be computed.

(iii) If $x_0 > 0$ and $y_0 > 0$: There is only one stationary point of h in V, which is the global minimum. For each point at the boundary of V there is a descent direction into the interior of V.

Proof. Note that there is neither a stationary point nor a local minimum in $x = y = 0$. So we assume $(x,y) \neq 0$.

We first prove (i). As $(x_0, y_0) \in U$ the global minimum (x_0, y_0) of f is in U. Thus by Lemma 5.5 we know that a global minimum of h is in U. Assume $(x_1, y_1) \in U \setminus V$ to be the global minimum. Then $x_1 < x_0$ or $y_1 < y_0$. Set $x_2 = x_0 + |x_1 - x_0|$ and $y_2 = y_0 + |y_1 - y_0|$. Then it is

$$|x_2| = x_0 + |x_1 - x_0| \geq x_0 + |x_1| - x_0 = |x_1|,$$

where equality holds if and only if $x_1 \geq x_0$. Similar $|y_2| \geq |y_1|$ with equality if and only if $y_1 \geq y_0$. By assumption of $x_1 < x_0$ or $y_1 < y_0$ it follows $\sqrt{x_2^2 + y_2^2} > \sqrt{x_1^2 + y_1^2}$

$$\begin{aligned} \Longrightarrow h(x_2, y_2) &= \frac{(x_2 - x_0)^2}{a^2} + \frac{(y_2 - y_0)^2}{b^2} + \left(1 - \sqrt{(x_2)^2 + (y_2)^2}\right)^2 \\ &< \frac{(x_1 - x_0)^2}{a^2} + \frac{(y_1 - y_0)^2}{b^2} + \left(1 - \sqrt{(x_1)^2 + (y_1)^2}\right)^2 = h(x_1, y_1) \end{aligned}$$

which contradicts the assumption of (x_1, y_1) being a minimum. This proves (i).

We now prove (ii). By (i) we can assume that $x \geq 0$ and $y \geq 0$. As we know that h is differentiable outside the origin and has no local minimum in the origin, its minimum must be a stationary point. Furthermore, we know that a global minimum is in V. Thus if we can show that there is at most one stationary point in V, this must be a global minimum.

In polar coordinates $x = r\cos(\varphi)$ and $y = r\sin(\varphi)$ we have

$$h(r,\varphi) = \frac{(r\cos(\varphi) - x_0)^2}{a^2} + \frac{(r\sin(\varphi) - y_0)^2}{b^2} + r^2 - 2r + 1.$$

With

$$A = \frac{r\cos(\varphi) - x_0}{a^2}, \qquad B = \frac{r\sin(\varphi) - y_0}{b^2}$$

for the partial derivatives we get

$$\begin{aligned} \frac{1}{2}\frac{\partial h}{\partial r} &= A\cos(\varphi) + B\sin(\varphi) + r - 1, \\ \frac{1}{2}\frac{\partial h}{\partial \varphi} &= -A\sin(\varphi) + B\cos(\varphi). \end{aligned}$$

To compute the stationary points the partial derivative have to be zero and we get

$$\begin{aligned} &\frac{\partial h}{\partial r} = 0 \quad \wedge \quad \frac{\partial h}{\partial \varphi} = 0 \\ \Longrightarrow\ & A = (1-r)\cos(\varphi) \quad \wedge \quad B = (1-r)\sin(\varphi) \\ \Longrightarrow\ & \cos(\varphi)[(1+a^2)r - a^2] = x_0 \quad \wedge \quad \sin(\varphi)[(1+b^2)r - b^2] = y_0. \end{aligned} \tag{$*$}$$

There we have to consider different cases:

Case 1: $x_0 = 0$, $y_0 = 0$, $a = b$. For this case we directly work with the objective and do not consider the derivatives. It is

$$h(r,\varphi) = \frac{a^2+1}{a^2}r^2 - 2r + 1 = \frac{a^2+1}{a^2}\left(r - \frac{a^2}{a^2+1}\right)^2 + \frac{1}{a^2+1}$$

which is minimal exactly for all (r,φ) with $r = \frac{a^2}{2(a^2+1)}$. Thus all local and global minima of h are a circle with radius $\frac{a^2}{2(a^2+1)}$ around the origin.

Case 2: $x_0 = 0$, $y_0 = 0$, $a \neq b$. Then by $(*)$ we have

$$\left[\cos(\varphi) = 0 \wedge r = \frac{y_0 + b^2}{1+b^2}\right] \quad \vee \quad \left[\sin(\varphi) = 0 \wedge r = \frac{x_0 + a^2}{1+a^2}\right]$$

Denote the first solution by z_1 and the second by z_2. For the Hessian it is

$$\begin{aligned} H(z_1) &= 2 \cdot \operatorname{diag}\left(1 + \frac{1}{b^2}, \frac{b^2(a+b)}{a^2(1+b^2)^2}(b-a)\right) \succ 0 \iff a < b, \\ H(z_2) &= 2 \cdot \operatorname{diag}\left(1 + \frac{1}{a^2}, \frac{a^2(a+b)}{b^2(1+a^2)^2}(a-b)\right) \succ 0 \iff a > b. \end{aligned}$$

The Hessian is semidefinite for $a = b$ which is not considered in this case. Hence, for $a > b$ the point z_1 is the only minimizer h in V. Similar for $a < b$ the point z_2 is the only minimizer of h in V.

Case 3: $x_0 = 0$, $y_0 > 0$. Then we can explicitly compute a set of 3 stationary points. The minimum of these points is the global minimizer. From $(*)$ we get the two cases $\cos\varphi = 0$ or $(1 + a^2)r - a^2 = 0$.

For $\cos\varphi = 0$ we have $\sin(\varphi) = 1$ and get as first stationary point

$$(r, \varphi) = \left(\frac{y_0 + b^2}{1 + b^2}, \frac{\pi}{2}\right).$$

Otherwise we have $(1 + a^2)r - a^2 = 0$ and conclude from $(*)$

$$\begin{aligned} r &= \frac{a^2}{1 + a^2} \\ \implies &\quad \sin(\varphi)\left(\frac{a^2 - b^2}{1 + a^2}\right) = y_0 \\ \implies &\quad a \neq b \quad \wedge \quad \sin(\varphi) = y_0\frac{1 + a^2}{a^2 - b^2}. \end{aligned}$$

This stationary point exists if and only if

$$a \neq b \quad \text{and} \quad y_0\frac{1 + a^2}{a^2 - b^2} \in [y_0, 1].$$

By comparison of these stationary points (if both exists), the minimizer can be computed.

Case 4: $x_0 > 0$, $y_0 = 0$. This case is symmetric to $x_0 > 0$, $y_0 = 0$.

We now proof (iii), so assume $x_0 > 0$ and $y_0 > 0$. For $x_0^2 + y_0^2 = 1$ we are done, so assume $x_0^2 + y_0^2 < 1$. Let $\varphi_0 = \arcsin(y_0)$ $\varphi_1 = \arccos(x_0)$, then for $(r, \varphi) \in V$ we have $\varphi \in [\varphi_0, \varphi_1]$. In particular, it is $0 < \varphi_0 < \varphi_1 < \frac{\pi}{2}$ and we have $\sin(\varphi) > 0$ and $\cos(\varphi) > 0$.

We get from $(*)$

$$r = \frac{x_0}{(1 + a^2)\cos(\varphi)} + \frac{a^2}{(1 + a^2)} \quad \wedge \quad r = \frac{y_0}{(1 + b^2)\sin(\varphi)} + \frac{b^2}{(1 + b^2)}.$$

Denote the left expression as $r_1(\varphi)$, the right expression by $r_2(\varphi)$. As $\cos(\varphi)$ is strictly decreasing in $(0, \pi/2)$, r_1 is strictly increasing. Similar, r_2 is strictly decreasing, and there is at most one intersection point $\varphi' \in [\varphi_0, \varphi_1]$. Furthermore, we have $r_1(\varphi_1) = 1$, $r_2(\varphi_0) = 1$,

$$x_0^2 + y_0^2 < 1 \implies \frac{x_0^2}{1-y_0^2} < 1 \implies \sqrt{\frac{x_0^2}{1-y_0^2}} < 1 \implies r_1(\varphi_0) = \frac{a^2 + \sqrt{\frac{x_0^2}{1-y_0^2}}}{1+a^2} < 1$$

and similar $r_2(\varphi_1) < 1$. By the intermediate value theorem we conclude that r_1 and r_2 have exactly one intersection on $[\varphi_1, \varphi_2]$. Thus, there is exactly one stationary point of h in V. As V contains the global minimum by (i), this stationary point is the global minimum.

It remains to show that for each point at the boundary of V there is a descent direction into V. Take a point (r_0, φ_0) at the boundary of V. We have to consider different cases.

Case 1: $r_0 \cos(\varphi_0) = x_0$ and $r_0 \sin(\varphi_0) = y_0$. Then $\boldsymbol{d} = (1,0)^T$ is a direction into V and

$$\boldsymbol{d}^T \nabla_{r,\varphi} h(r_0, \varphi_0) = \frac{\partial h}{\partial r}(r_0, \varphi_0) = r_0 - 1 = \sqrt{x_0^2 + y_0^2} - 1 < 0.$$

Case 2: $r_0 = 1$, $\cos(\varphi_0) = x_0$ and $\sin(\varphi_0) = 1$. Then $\boldsymbol{d} = (-\varepsilon, -1)^T$ is a direction into V for some small ε and

$$\boldsymbol{d}^T \nabla_{r,\varphi} h(r_0, \varphi_0) = \frac{2(\varepsilon + x_0)(y_0 - 1)}{b^2} < 0.$$

Case 3: $r_0 = 1$, $\cos(\varphi_0) = 1$ and $\sin(\varphi_0) = y_0$. This case is similar to case 2.

Case 4: $r_0 \cos(\varphi_0) = x_0$ and $y_0 < r_0 \sin(\varphi_0) < 1$. Then $\boldsymbol{d} = (0, -1)^T$ is a direction into V and

$$\boldsymbol{d}^T \nabla_{r,\varphi} h(r_0, \varphi_0) = -\frac{\partial h}{\partial \varphi}(r_0, \varphi_0) = -2x_0 \frac{r_0 \sin(\varphi_0) - y_0}{b^2} < 0.$$

Case 5: $r_0 \sin(\varphi_0) = y_0$ and $x_0 < r_0 \cos(\varphi_0) < 1$. This case is similar to case 4.

Case 6: $r_0 = 1$ and $\sin(\varphi_0) > y_0$ and $\cos(\varphi_0) > x_0$. Then $\boldsymbol{d} = (-1, 0)^T$ is a direction into V and

$$\boldsymbol{d}^T \nabla_{r,\varphi} h(r_0, \varphi_0) = -\frac{\partial h}{\partial \varphi}(r_0, \varphi_0) = -2x_0 \frac{r_0 \sin(\varphi_0) - y_0}{b^2} < 0.$$

So for each boundary point of V there is a descent direction into V. $\square$

By Lemma 5.7 either it is $x_0 = 0$ or $y_0 = 0$ and the optimum can be analytically computed, or we obtain the optimum by solving the non-linear program to minimize h in V. As there is only one stationary point in V and for each boundary point there is a descent direction into the interior of V, standard local non-linear solver converge to this stationary point. We say a minimum can *efficiently* be computed, if it can be computed analytically or by this non-linear program to minimize h on V.

Corollary 5.8. *Let U be the circle with radius r around (x_0, y_0) and $U \subset G \subset \mathbb{R}^2$. Denote $z = (x, y)$. Let*

$$f : G \to \mathbb{R}, \qquad f(z) = z^T A z + b^T z + c$$

be an elliptic paraboloid, i.e. $A \succ 0$. Let $g(x, y) = (r - ||(x - x_0, y - y_0)||)^2$. Then the minimum of $f + g$ can be efficiently computed.

Proof. By principal component analysis we can transform f to have its main axes parallel to the coordinate axes, i.e. to have the form of Lemma 5.7. This transformation does not change the structure of g. We additionally translate g to the origin. Then by Lemma 5.7 the result follows. □

Note in the settings of Corollary 5.8 that V includes the global minimum of f and the global minimum of $f + g$. One might think that a descending algorithm starting in the minimum of f would reach the optimum of $f + g$. This is obviously true, if we restrict the solutions to the set V. However, there exist examples where there is a strictly descending path from a minimum of f to a local minimum of $f + g$ that is not global. For an example see Figure 5.2.

5.3.2 Application to the Circle Rotation Problem

We now transfer the results of the previous section to the rotation problem.

Assumption 5.9. *In this section we assume that the circles do not overlap.*

Lemma 5.10. *If there are two or more circles, f^{out} is strictly convex.*

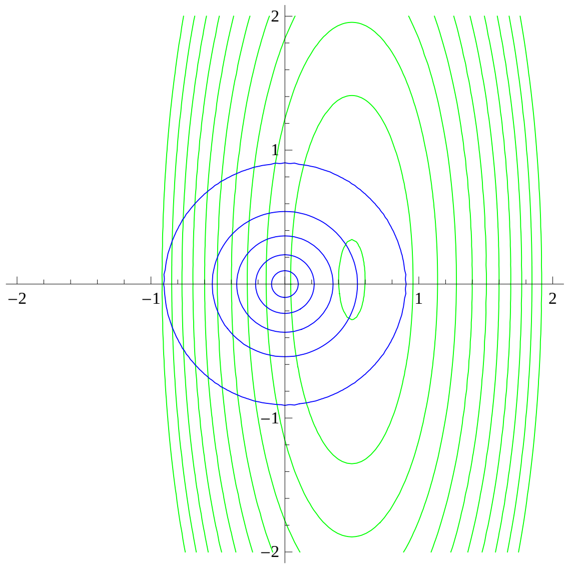
Proof. Each $f_i^{out}(s) = f_{c_i, r_i}^{out}(s) = \max(|s - c_i| - r_i, 0)^2$ is convex, and strictly convex outside the circle i. By assumption the circles do not overlap. Then f^{out} is the sum of convex functions, where the sum contains a strictly convex functions everywhere. Hence, f^{out} is strictly convex. □

Corollary 5.11. *Let s_{out} be the minimum of f^{out} and s_{sum} a minimum of f^{sum}. If s_{out} is within a circle j, then it is*

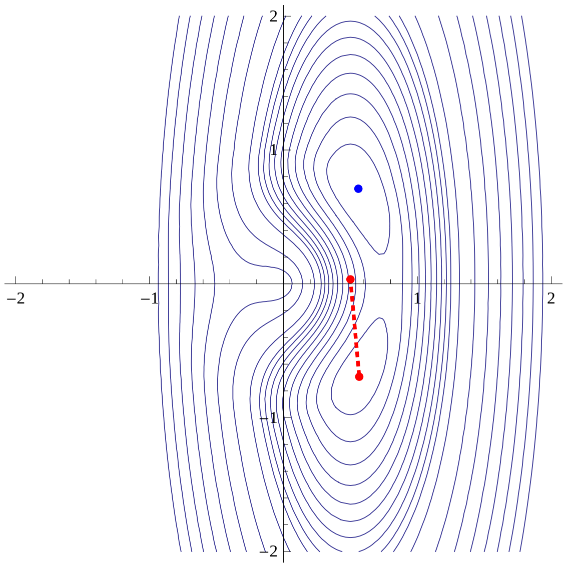
$$\begin{aligned} &|s_{out} - c_j| \le |s_{sum} - c_j| \le r_j, \\ &f^{sum}(s_{out}) - f^{sum}(s_{sum}) \le f^{in}(s_{out}) = f_j^{in}(s_{out}). \end{aligned}$$

If s_{out} is not in a circle, then $s_{out} = s_{sum}$.

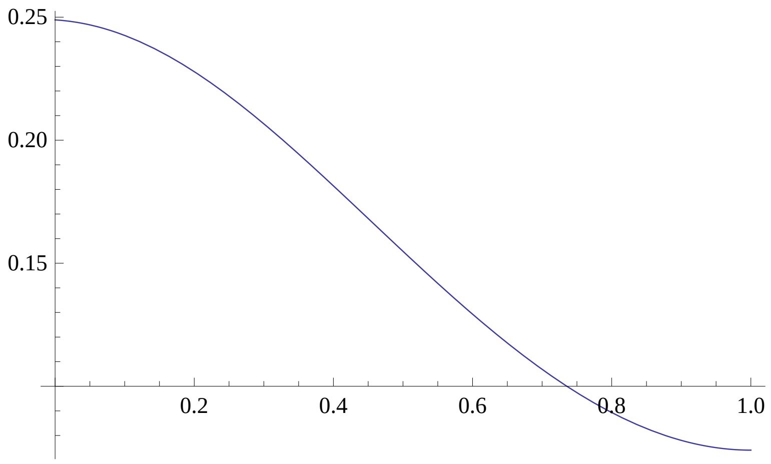
Proof. This is an immediate consequence of Lemma 5.6. □



(a) Contour lines of f (green, descending to the center) and g (blue, ascending to the center).



(b) Contour lines of $f + g$ and the path γ (red) from the global minimum of f to a non-global minimum of $f + g$. The blue point is the global minimum.



(c) The graph of $(f + g)(\gamma(t))$ shows that γ is strictly descending.

Figure 5.2: Consider the settings of Lemma 5.7 with $a = 1$, $b = 3$, $x_0 = \frac{1}{2}$ and $y_0 = 1/30$. Then by Lemma 5.7 the unique global minimum of $f + g$ is in the first quadrant. Numerical computations give the global minimum at $(x_g, y_g) = (0.5592, 0.7106)$ and a local but non-global minimum at $(x_l, y_l) = (0.5659, 0.7106)$. Consider the path $\gamma(t) = (1 - t)(x_0, y_0) + t(x_l, y_l)$. Then $(f + g)(\gamma(t))$ is a strictly decreasing path from the global minimum of f to a non-global minimum of $f + g$.

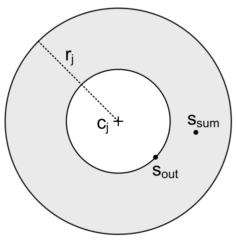


Figure 5.3: If the optimum s_{out} of f^{out} lies within the circle j with radius r_j around c_j, then the optimum s_{sum} of f^{sum} is in the annulus with radii $|s_{out} - c_j|$ and r_j around c_j.

Figure 5.3 visualizes the meaning of Corollary 5.11.

An approach is now, to optimize f^{out} first and if the optimum s_{out} is within a circle, restart the optimization of f^{sum} from s_{out}. For most cases the problem is very well behaved. This means, the algorithm usually finds the global optimum and, if not, the local optimum is nearly as good as the global optimum. However, optimality is not guaranteed. An example where the algorithm might fail is shown in Figure 5.4.

We now want to identify problems, where we can provably find a solution close to the optimum efficiently.

Lemma 5.12. *If all but one circle have radius* 0*, the global minimum can efficiently be computed.*

As an immediate consequence of the results of Section 3.5 we can compute the minimum analytically. However, we prove it by the results of this section.

Proof. W.l.o.g. circle n is the only circle with positive radius. If the optimum s_{out} of f^{out} is outside circle n, we are done. Otherwise f^{out} is an elliptic paraboloid. So by Corollary 5.8 the minimum can efficiently be computed. □

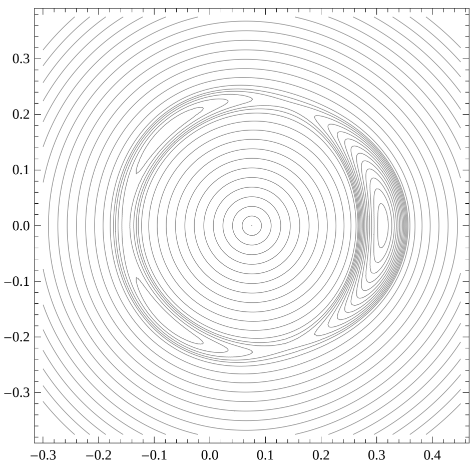
The question now is, whether we can extend this property to other problem instances. A natural extension would be to consider problems where the circles are not necessarily points but still small. Before we show such a result, we have to introduce the multivariate Taylor polynomial in real coordinates first.

Definition 5.13 (Multi-Index-Notation)**.** *Let* $\boldsymbol{\alpha} \in \mathbb{N}_0^n$ *and* $\boldsymbol{x} \in \mathbb{R}^n$*, then*

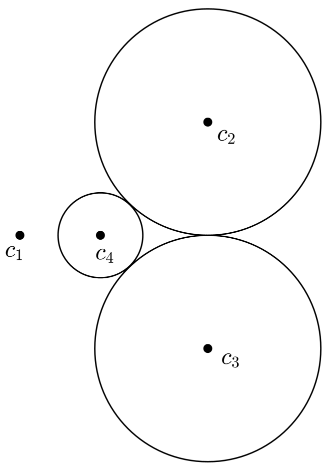
$$|\boldsymbol{\alpha}| = \sum_{i=1}^{n} \alpha_i \qquad \boldsymbol{\alpha}! = \prod_{i=1}^{n} \alpha_i! \qquad \boldsymbol{x}^{\boldsymbol{\alpha}} = \prod_{i=1}^{n} x_i^{\alpha_i}.$$

Then for the partial differential we define

$$\frac{\partial^{\boldsymbol{\alpha}} g}{\partial \boldsymbol{x}^{\boldsymbol{\alpha}}} = \frac{\partial^{\alpha_1}}{\partial x_1^{\alpha_1}} \cdots \frac{\partial^{\alpha_n}}{\partial x_n^{\alpha_n}}.$$



(a) Contour lines of f^{sum}.



(b) Circle positions.

Figure 5.4: There are four circles with $\boldsymbol{c} = (-1; 2+2i; 2-2i; 0.0755)^T$ and $\boldsymbol{r} = (0; 2; 2; 0.75)^T$. The optimum of f^{out} is $s_{out} = 0.755$. The optimal value of h is at $s_{sum} = 0.3112$ with $f^{sum}(s_{sum}) = 2.7467$. However, there are local optima at $s_{1,2} = -0.0771 \pm 0.1704$ with worse objective value $f^{sum}(s_{1,2}) = 2.7506$. The steepest descend directions show away from s_{out}. The problem can be modified by choosing $c_3 = s_{out} + \kappa$ with small κ. This does not change the optimum s_{out} of f^{out} and only changes the local optima of f^{sum} continuously. However, a local optimization algorithm starting in s_{out} might converge to any of these local optima. Especially for small $\kappa \in \mathbb{R}^{\leq 0}$ the algorithm does not find the global optimum.

Theorem 5.14 (Multivariate Taylor Polynomial)**.** *Let G be a domain and $g : G \to \mathbb{R}$ be a $k+1$ times continuously differentiable function. Let $\boldsymbol{x}_0 \in G$. Then the Taylor polynomial of order n around x_0 is*

$$T_{g,\boldsymbol{x}_0,n}(\boldsymbol{x}) := \sum_{|\boldsymbol{\alpha}|\leq k} \frac{1}{\boldsymbol{\alpha}!} \frac{\partial^{\boldsymbol{\alpha}} g(\boldsymbol{x}_0)}{\partial \boldsymbol{x}^{\boldsymbol{\alpha}}} (\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\alpha}}.$$

For the error term we get

$$R_{g,\boldsymbol{x}_0,n}(\boldsymbol{x}) = f(\boldsymbol{x}) - T_{g,\boldsymbol{x}_0,n}(\boldsymbol{x}) = \sum_{|\boldsymbol{\alpha}|=n+1} R_{g,\boldsymbol{x}_0,n,\boldsymbol{\alpha}}(\boldsymbol{x}) \cdot (\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\alpha}}$$

with a remainder satisfying

$$|R_{g,\boldsymbol{x}_0,n,\boldsymbol{\alpha}}| \leq \sup_{\boldsymbol{y}\in G} \left| \frac{1}{\boldsymbol{\alpha}!} \frac{\partial^{\boldsymbol{\alpha}} g(\boldsymbol{y})}{\partial \boldsymbol{x}^{\boldsymbol{\alpha}}} \right|.$$

Theorem 5.15. *Denote by s_{out} the minimum of f^{out}. If s_{out} is outside all circles, it is the global optimum of f^{sum}.*

Otherwise the minimum s_{out} lies within circle k. Suppose that there is a $\varepsilon > 0$ such that for all circles $j \neq k$ it is $\frac{r_j}{(|c_j - c_k| - r_k)^2}$. Denote the optimum of f^{sum} by s_{sum}. Then we can efficiently compute a point s_0 with

$$f^{sum}(s_0) \leq f^{sum}(s_{sum}) + 24(n-1)\varepsilon r^3.$$

Especially $f^{sum}(s_0) \to f^{sum}(s_{sum})$ for $\varepsilon \to 0$.

Proof. First consider the case that s_{out} is outside all circles. There is $f^{out} = f^{sum}$ and as $f^{sum} \geq f^{out}$ everywhere, then s_{out} is the global optimum of f^{sum}.

So now consider s_{out} to be in circle k. For this proof we view complex numbers as real vectors and especially gradients are real gradients. By translation we can assume $\boldsymbol{c}_k = (0,0)^T$. Furthermore, s_{out} is outside all other circles. Denote by U the disc with radius r_k around the origin and $s = (x,y)^T$.

We first do a detailed analysis of the functions f_j^{out} for a fixed j. Therefore, we set $r = r_j$, $(a,b) = c = c_j$ and $h = f_j^{out}$.

As all our Taylor polynomials are of order 2 around the origin, we omit these information in the index. Applying Theorem 5.14, for the Taylor expansion of order two around 0 in U we get

$$T_h(s) = h(0) + (\nabla h(0))^T s + \frac{1}{2} s^T (\nabla^2 h(0)) s,$$
$$R_h(s) = \sum_{|\alpha|=3} R_{h,\alpha}(s) \cdot s^\alpha,$$
$$|R_{h,\alpha}(s)| \leq \sup_{z\in U} \left| \frac{1}{\alpha!} \frac{\partial^\alpha h(z)}{\partial x^\alpha} \right|.$$

Computing the derivatives for $s \in U$

$$\begin{aligned} h(s) &= (|s-c|-r)^2, \\ \nabla h(s) &= 2(s-c) - \frac{2r}{|s-c|}(s-c), \\ \nabla^2 h(s) &= 2I - \frac{2r}{|s-c|^3}\begin{pmatrix} (y-b)^2 & -(x-a)(y-b) \\ -(x-a)(y-b) & (x-a)^2 \end{pmatrix}. \end{aligned}$$

Furthermore, it is for the third partial derivatives

$$\begin{aligned} \left|\frac{\partial^3 h}{\partial x^2 y}\right| &= \frac{2r}{|s-c|^5}\left|2(y-b)(x-a)^2 - (y-b)^3\right| \leq \frac{6r}{|s-c|^2}, \\ \left|\frac{\partial^3 h}{\partial x^3}\right| &= \frac{6r}{|s-c|^5}\left|(x-a)(y-b)^2\right| \leq \frac{6r}{|s-c|^2}. \end{aligned}$$

By symmetry of h and Schwarz theorem with $s \in U$ for $|R_h(s)|$ it is

$$\begin{aligned} |R_h(s)| &\leq 6 \sup_{z\in U}\left|\frac{1}{6}\frac{\partial^3 h(z)}{\partial x^3}\right| r_k^3 + 2\sup_{z\in U}\left|\frac{1}{2}\frac{\partial^3 h(z)}{\partial x^2 y}\right| r_k^3 \\ &\leq \frac{12r}{|s-c|^2}\cdot r_k^3 \leq \frac{12r}{(|c|-r_n)^2}\cdot r_k^3 \leq 12\varepsilon r_k^3. \end{aligned}$$

Now we consider the function f^{out} again. Note that $f_k^{out} = 0$ in U. It is

$$\begin{aligned} T_{f^{out}}(s) &= \sum_{j=1}^{n-1} T_{f_j^{out}}(s) \\ R_{f^{out}}(s) &= \sum_{j=1}^{n-1} R_{f_j^{out}}(s) \\ |R_{f^{out}}(s)| &\leq \sum_{j=1}^{n-1}\left|R_{f_j^{out}}(s)\right| \leq \sum_{j=1}^{n-1} 12\varepsilon r_k^3 = 12(n-1)\varepsilon r_k^3. \end{aligned}$$

Set $\rho(s) = f_k^{in}(s) + T_{f^{out}}(s)$. Then we have for $s \in U$

$$|f^{sum}(s) - \rho(s)| = \left|f^{out}(s) + f_k^{in}(s) - T_{f^{out}}(s) - f_k^{in}(s)\right| = |R(s)| \leq 12(n-1)\varepsilon r_k^3.$$

Now $T_{f^{out}}$ is an elliptic paraboloid. So by Corollary 5.8 the minimum s_0 of $\rho(s)$ can be efficiently computed. For this minimum we get with s_{sum} denoting the minimum of f^{sum}

$$\begin{aligned} & f^{sum}(s_0) - f^{sum}(s_{sum}) \\ &= f^{sum}(s_0) - \rho(s_0) + \rho(s_0) - \rho(s_{sum}) + \rho(s_{sum}) - f^{sum}(s_{sum}) \\ &\leq |f^{sum}(s_0) - \rho(s_0)| + |\rho(s_{sum}) - f^{sum}(s_{sum})| \\ &\leq 24(n-1)\varepsilon r_k^3 \end{aligned}$$

which converges to zero for $\varepsilon \to 0$. □

We now state a more intuitive formulation for the condition of Theorem 5.15. We already know that if all circles except one have radius 0, then the global minimum can be efficiently computed. So we expect the error to be small, if the circles are almost points. The next statement says that, if the radius of the circles are small compared to their distance to the circle containing s_{out}, then the error of the computed minimum is small.

Corollary 5.16. *Let the minimizer s_{out} of f^{out} be in circle k. Denote by s_{sum} the minimizer of f^{sum}. Assume that all circles have a positive distance to circle k, i.e. there is some $d > 0$ such that $|c_j - c_k| \geq r_j + r_k + d$. Furthermore, assume that the radii of the circles are small compared to their distances to the center of k, i.e. $\frac{r_j}{|c_j - c_k|} \leq \varepsilon$. Then we can efficiently compute an s_0 such that $f^{sum}(s_0) \to f^{sum}(s_{sum})$ for $\varepsilon \to 0$.*

Proof. Denote $c = |c_j - c_k|$. Then $c - r_k \geq d$ and $r_j \leq \varepsilon \cdot c$ and it follows

$$\begin{aligned}\frac{r_j}{(|c_j - c_k| - r_k)^2} &= \frac{r_j}{(c - r_k)^2} \leq \varepsilon \cdot \frac{c}{(c - r_k)^2} = \varepsilon \cdot \frac{1}{c - r_k}\left(1 + \frac{r_k}{c - r_k}\right) \\ &\leq \varepsilon \cdot \frac{1}{d}\left(1 + \frac{r_k}{d}\right) = \varepsilon \cdot \frac{d + r_k}{d^2}.\end{aligned}$$

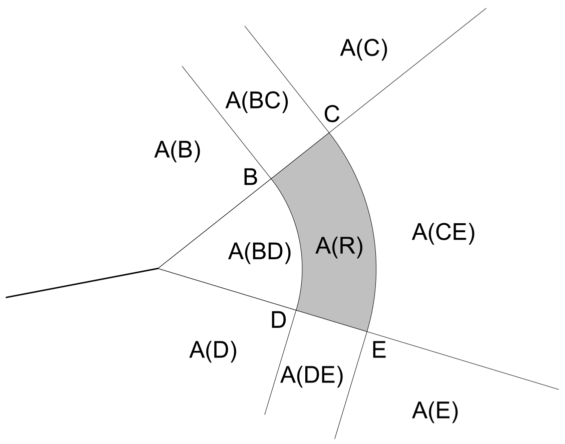
Theorem 5.15 yields the result. □

5.4 Global Algorithm BASSAS

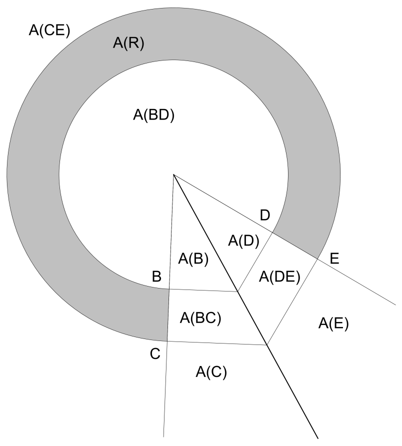
For location problems in the plane the BSSS algorithm is a well known branch and bound algorithm. In [Nezakati; Zaferanieh; Fathali 2009] it was applied to the location problem for points with minimal squared Euclidean distance to circles.

The BSSS algorithms was introduced in [Hansen; Peeters; Thisse 1981] and generalized in [Plastria 1992]. The plane is divided into rectangular regions, for which a lower bound can efficiently be computed. Upper bounds for the best point in a rectangle can easily be computed by evaluating the function at a point inside the rectangle. If now the lower bound of any rectangle is greater than the best upper bound, the rectangle can be discarded. Remaining rectangles are successively divided into smaller ones and the process is repeated. The algorithm stops, when a predefined tolerance is reached, i.e. the difference between lower bound and upper bound is small. The best solution found so far is taken as solution.

We modify this algorithm to work on annulus segments and call it big annulus segment small annulus segment (BASSAS) algorithm. Essentially, this is a BSSS algorithm in polar coordinates.



(a) Acute and obtuse angles.



(b) Reflex angle.

Figure 5.5: Distance computation to annulus ring segments. For both cases the plane is divided in nine areas. For points in $A(B), A(C), A(D), A(E)$ the distance to the ring segment is the distance to the point B, C, D, E. For points in $A(BC)$ and $A(DE)$ the distance to the ring segment is the distance to the line BC or DE. Similar for points in $A(BD)$ and $A(CE)$ the distance to the ring segment is the distance to the circular arc BD or CE. And finally for points in $A(R)$ the distance to the ring segment is zero.

Definition 5.17 (Annulus Segment)**.** *An annulus segment with radii R_{min}, R_{max}, angles φ_{min}, φ_{max} and center $c_0 \in \mathbb{C}$ is the set*

$$\begin{aligned}&AR(c_0, R_{min}, R_{max}, \varphi_{min}, \varphi_{max})\\ &= \{c_0 + R\exp(\imath\varphi) : R \in [R_{min}, \leq R_{max}], \varphi \in [\varphi_{min}, \varphi_{max}]\}\,.\end{aligned}$$

Lemma 5.18. *The distance $d(AR, c_1)$ of a point c_1 to an annulus ring segment AR can efficiently be computed.*

Proof. Although this can be proved analytically be, we refer to Figure 5.5. There the plane is separated to nine parts. For each point, the containing part can be determined. Furthermore, for each part the distance to the annulus ring segment is the distance of the point to a simple geometric object (point, circular arc, line) which can efficiently be computed. □

Lemma 5.19. *Let $D \subset \mathbb{C}$ be the disc with radius r around c and $U \subset \mathbb{C}$. Then for the distance $d(D, U)$ it is*

$$d(D, U) = \max(d(c, U) - r, 0).$$

Proof. Let $\mathrm{cl}(U)$ be the closure of U and denote by $u \in \mathrm{cl}(U)$ the element such that $d(c, u) = d(c, U)$.

Case 1: $d(c, U) \leq r$. Then it is

$$d(c,u) \leq r \implies u \in D \implies d(D,u) = 0 \implies d(D,U) = 0.$$

Case 2: $d(c, U) > r$. Then $u \notin D$. So let y be the intersection of the line from c to u and ∂D. Obviously $d(D,U) \leq d(y,u) = d(c,u) - r$. Now assume $d(D,U) < d(c,u) - r$. Then there is a $z \in D$ and a $v \in \text{cl}(U)$ such that $d(z,v) = d(D,U) < d(c,u) - r$. However, then

$$d(c,v) \leq d(c,z) + d(z,v) < r + d(c,u) - r = d(c,u) = d(c,U)$$

which is impossible as $v \in U$. So it follows $d(D,U) = d(c,u) - r$.

□

Algorithm 3: BASSAS algorithm to globally minimize f^{sum}.

```
Data: Target Approximation Quality $\varepsilon > 0$
1  $s_0 \leftarrow \text{argmin}_s\,(f^{out}(s))$;
2  if $s_0$ is not in a circle then return $s_0$;
3  $k \leftarrow$ circle containing $s_0$;
4  $\mathcal{Q} \leftarrow \{D_k\}$;
5  $lb^* \leftarrow \max\{lb(R) : R \in \mathcal{Q}\}$;
6  $ub^* \leftarrow \min\{ub(R) : R \in \mathcal{Q}\}$;
7  while $ub^* > lb^* \cdot (1 + \varepsilon)$ do
8      take $\hat{R} \in \mathcal{Q}$;
9      divide $\hat{R}$ into $\{\hat{R}^j,\, j \in J\}$;
10     $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{\hat{R}\} \cup \{\hat{R}^j\} : j \in J\}$;
11     $lb^* \leftarrow \max\{lb(R) : R \in \mathcal{Q}\}$;
12     $ub^* \leftarrow \min\{ub(R) : R \in \mathcal{Q}\}$;
```

In Algorithm 3 we state the algorithm BASSAS to globally minimize f^{sum}. There D_j denotes the disc with radius r_j around c_j. We explain the main steps:

Domain Sets All domain sets in $\mathcal{Q}$ are annulus rings during the algorithm (at the beginning degenerated as a circular disc).

Upper Bound Computation We can apply any local non-linear solver to compute an upper bound of $f^{sum}(s)$ on a domain R, as each feasible solution suffices.

Lower Bound Computation Note that all annulus ring segments $R \in \mathcal{Q}$ are subsets of D_k. Thus for the annulus ring R with maximal radius r it is

$$\begin{aligned} lb(R) &= \min_{s \in R} f^{sum}(s) = \min_{s \in R} \left(f_k^{in}(s) + \sum_{j \neq k} f_j^{out}(s) \right) \\ &\leq \min_{s \in R} f_k^{in}(s) + \min_{s \in R} \sum_{j \neq k} f_j^{out}(s) \leq r_k - r + \min_{s \in R} \underbrace{\sum_{j \neq k} f_j^{out}(s)}_{g(s)} . \end{aligned}$$

The last inequality holds, as the annulus ring segment R and D_k are concentric. Then it is $\min_{s \in R} f_k^{in}(s) = d(\partial D_k, R) = r_k - r$.

Hence, it remains to compute a lower bound of g. As $g(s)$ is a convex function, we can minimize it to global optimality over $\operatorname{conv}(R)$.

Furthermore, we can compute a lower bound as follows:

$$g(s) \geq \sum_{j \neq k} \min_{s \in R} f_j^{out}(s) = \sum_{j \neq k} d(D_j, R).$$

As $D_j, j \neq k$ and $R \subset D_k$ do not overlap by assumption, the distance $d(D_j, R)$ can be efficiently computed by Lemma 5.18 and Lemma 5.19.

Branching We choose the annulus ring segment $\hat{R} \in \mathcal{Q}$ with maximal lower bound to branch on. Let $\hat{R} = AR(c_0, R_{min}, R_{max}, \varphi_{min}, \varphi_{max})$. Then it is split into four parts by bisection of both the interval $[R_{min}, R_{max}]$ and the angle interval $[\varphi_{min}, \varphi_{max}]$. Thus it is with $R_m = (R_{min} + R_{max})/2$ and $\varphi_m = (\varphi_{min} + \varphi_{max})/2$

$$\begin{aligned} \{\hat{R}^j, j \in J\} = \{&AR(c_k, R_{min}, R_m, \varphi_{min}, \varphi_m),\ AR(c_k, R_{min}, R_m, \varphi_m, \varphi_{max}), \\ &AR(c_k, R_m, R_{max}, \varphi_{min}, \varphi_m),\ AR(c_k, R_m, R_{max}, \varphi_m, \varphi_{max})\}. \end{aligned}$$

6 Unit Circle Placements

In this chapter we consider placements of unit circles belonging to the same net. We consider placements with optimal area as well as placements with optimal wire length and how these placements are related.

This relation is of interest, as in electronics design the placement must have a short wire length and the packing must be dense.

The problem to find the densest packing of unit circles in the plane has extensively been studied in the literature. Also, the problem to pack unit circles such that their connection length is minimal has been analyzed. However, to our best knowledge the relation of these optimal packings has not been studied.

In this chapter we asymptotically prove for unit circle lattice packings that wire length optimal packings imply area optimal packings. We also show that the converse implication is not true. This fact supports to focus on wire length optimal placements in this thesis.

In Section 6.1 we summarize results of finite and infinite area optimal circle packings in the plane. We focus on free packings and characterize the packings with optimal packing density. In Section 6.2 we survey results from the literature known for packing circles such that their connection length is minimized. Applying the results of the first two sections of this chapter, in we analyze the relation of area optimal and wire length optimal packings in Section 6.3. We show that packings with optimal wire length are also asymptotically area optimal. However, the converse is not true, i.e. the packings with optimal area do not have optimal wire length.

6.1 Area Optimal Packings

6.1.1 Literature Survey

The densest packing of identical circles in the plane is the hexagonal lattice packing. Gauss proved that the hexagonal packing is the densest packing of all plane lattice packings. The first proof that this is indeed the densest of all (possibly non-lattice) packings was claimed by Thue in 1892 and 1910. However, the first flawless proof was done by Tóth around 1940 ([Böröczky 2004, Chapter 4]). The packing density of the hexagonal packing is $\pi/\sqrt{12}$.

For finite packings the problem is more complicated. For fixed boundaries there are only heuristics and some theoretical results for very few circles, usually based on computer aided optimality proves. For packing congruent circles in a square, see e.g. [Markót; Csendes 2005]. For packing congruent circles in a circle, optimal results are known for up to 13 and for 19 circles, see e.g. [Fodor 1999] and [Fodor 2003].

Therefore, in literature often so called free packings are considered. For free packings, there is no fixed boundary, instead the convex hull of the circles is considered. Based on [Groemer 1960] the problem of packing identical circles such that the convex hull is minimal is solved for most natural numbers in [Wegner 1986] by transformation to a number theoretic problem. Wegner shows that the first natural number for which his solution does not prove optimality is 121. In [Böröczky; Ruzsa 2007] it is shown that Wegner's solution proves optimality for 23/24 of all natural numbers.

6.1.2 Definitions

We state some definitions and results from literature first. They are based on [Böröczky 2004] and [Tóth 1972].

Definition 6.1 (Lattice)**.** *Let $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n \in \mathbb{R}^n$ be a set of linear independent vectors. A lattice is the subgroup*

$$\Lambda := \left\{ \sum_{i=1}^{n} \lambda_i \boldsymbol{v}_i : \lambda_i \in \mathbb{Z}, i = 1, \dots, n \right\}.$$

The vectors $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n$ are called a basis of Λ.

For the same lattice there exist different bases. However, the absolute value of the determinant of the basis vectors is uniquely determined by the lattice Λ and denoted by $\det(\Lambda)$.

The most common Lattices are $\mathbb{Z}^n$. For circle packing the two-dimensional hexagonal lattice A_2 with basis $(2,0)^T, (1,\sqrt{3})^T$ and $\det(A_2) = \sqrt{12}$ is important. A circle packing on this lattice is shown in Figure 6.1b.

Definition 6.2 (Packing)**.** *Let $\Pi \subset \mathbb{R}^2$ be a convex region. Let $B_i \subset \Pi$, $i \in I$, be a family of circles with centers c_i.*

- *A packing is a placement of the circles such that no circles overlap.*
- *A lattice packing is a packing such that the circle centers c_i lay on a lattice.*

For a region $G \subset \mathbb{R}^2$ we label the area as $A(G)$ and the perimeter as $P(G)$.

Definition 6.3 (Packing Density)**.** *Let* $\Pi \subset \mathbb{R}^2$ *be a convex region and* $\mathcal{B} = \{B_i, i \in I\}$ *be a circle packing in* Π*. With* D_R *we denote the circle with radius* R *around the origin. Then the packing density of* $\mathcal{B}$ *with respect to* Π *is*

$$\delta := \lim_{R\to\infty} \frac{\sum_{\{i:B_i\subset D_R\}} A(B_i)}{A(D_R \cap \Pi)}.$$

It can be shown that the definition for infinite packings does not depend on the choice of the origin.

Usually we are interested in either infinite packings with $\Pi = \mathbb{R}^2$ or finite packings with Π being the convex hull of the circles. Thus for a packing $\mathcal{B} = \{B_i, i \in I\}$, we define the packing density of $\mathcal{B}$ to be the packing density of $\mathcal{B}$ with respect to the convex hull $\operatorname{conv}\left(\bigcup_{i\in I} B_i\right)$ of all circles.

Lemma 6.4. *For a finite packing* $\mathcal{B} = \{B_i, i \in I\}$ *the packing density is*

$$\delta := \frac{\sum_{i=1}^{n} A(B_i)}{A(\Pi)}.$$

For a dense infinite lattice packing of congruent circles B *on the lattice* Λ *the packing density is*

$$\delta = \frac{A(B)}{\det(\Lambda)}.$$

Definition 6.5 (Voronoi Cell)**.** *Let* $\Pi \subset \mathbb{R}^2$ *be a convex region and* $B_i \subset \Pi$ *be a packing of congruent circles. The Voronoi cell* V_i *of circle* i *is the set of points in* Π *whose distance to* c_i *is not greater than to any other* c_j*,* $j \neq i$*.*

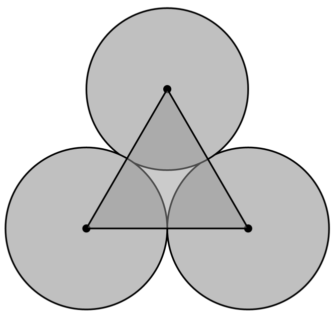
Definition 6.6 (Local Packing Density)**.** *Let* $\Pi \subset \mathbb{R}^2$ *be a convex region. Let* $B_i \subset \Pi$ *be a packing of congruent circles and* V_i *be the Voronoi cell of* B_i*. The local density of cell* i *is*

$$\delta_i := \frac{A(B_i)}{A(V_i)}.$$

6.1.3 Infinite Area-Optimal Packings

Intuitively, the densest packing of three circles of radius r is, to pack them at the corner points of a triangle of edge length $2r$. The ratio of the area of the triangle covered by circles and the total triangle area is independent of r and given by $\sigma = \pi/\sqrt{12}$. This packing is shown in Figure 6.1.

Claimed in 1892 by Thue and proven by Tóth around 1940 is the following famous result of Theorem 6.7.


(a) Triangle bound: The triangle bound σ is the ratio of the area of the triangle covered by circles and the total area of the triangle.

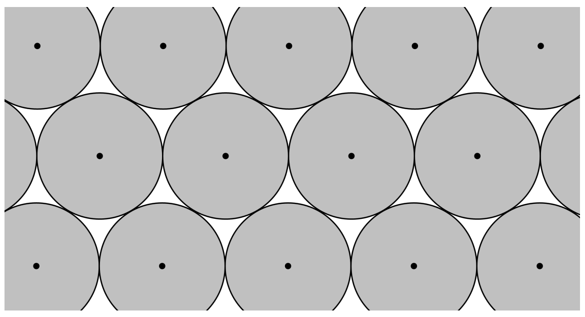

(b) The hexagonal lattice packing is the densest packing in the plane.

Figure 6.1: Triangle bound and the hexagonal lattice packing.

Theorem 6.7 (Densest Packing of Circles is the Hexagonal Lattice Packing)**.** *For each finite or infinite packing $\mathcal{B} = \{B_i, i \in I\}$ of unit circles (with more than two elements) it is $\delta \leq \sigma$. Furthermore, for all local densities it is $\delta_i \leq \sigma$. For the infinite packing, equality for all local densities only holds for the hexagonal lattice packing.*

If the Voronoi cell V_i is a polygon (always true if V_i is not at the boundary), it is $\delta_i = \sigma$ if and only if V_i is a regular hexagon.

6.1.4 Finite Area-Optimal Packings

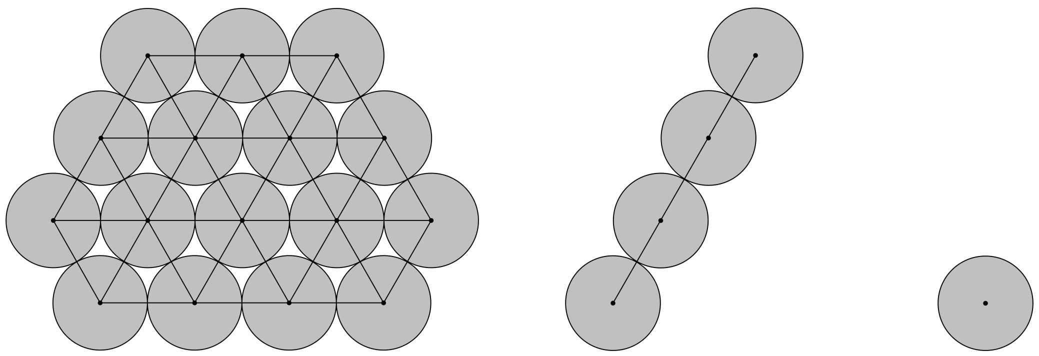
For finite packings the results are more complex. Based on [Groemer 1960], in [Wegner 1986] a conjecture of Tóth was proved that the area optimal packings are those hexagonal packings, which are as close to a regular hexagon as possible. We now state the main results.

Definition 6.8 (Groemer Packing)**.** *A Groemer packing is a hexagonal lattice packing where the convex hull of the circle centers is a polygon that can be triangulated in (possibly degenerated) triangles of edge length 2 in such a way that each triangle corner is the center of a circle.*

Theorem 6.9 (Thue-Groemer-Inequality)**.** *Let $\{B_1, \dots, B_n\}$ be a unit circle packing. Let Π be a convex region containing the circle centers. Then*

$$\frac{A(\Pi)}{\sqrt{12}} + \frac{P(\Pi)}{4} + 1 \geq n. \tag{6.1}$$

Equality holds if and only if the packing is a Groemer packing and Π is the convex hull of the circle centers.


(a) A Groemer packing is a dense lattice packing with hexagonal boundary.

(b) A line of circles is a degenerated Groemer packing.

(c) A single circle is a degenerated Groemer packing.

Figure 6.2: Different kinds of Groemer packings.

Now let $\mathcal{B} = \{B_1, \ldots, B_n\}$ be a packing. If Π' is a convex region containing the circles and Π_n is the convex hull of the circle centers, then $A(\Pi') \geq A(\Pi_n) + P(\Pi_n) + \pi$ and it follows

$$A(\Pi') \geq 2\sqrt{3}(n-1) + \frac{1}{2}(2-\sqrt{3})P(\Pi_n) + \pi. \tag{6.2}$$

Equality holds, if and only if $\mathcal{B}$ is a Groemer packing and Π' is its convex hull. So, the area $A(\Pi')$ of the convex hull Π' of a Groemer packing is minimal if the perimeter $P(\Pi')$ is minimal.

If $\mathcal{B}$ is a Groemer packing, Π_n is a convex polygon with at most six sides. So the isoperimetric inequality for hexagons yields $A(\Pi_n) \leq (\sqrt{3}/24)P(\Pi_n)^2$ and so by (6.1) we get $P(\Pi_n) \geq 2\left\lceil\sqrt{12n-3}-3\right\rceil$. So, it follows by (6.2) that

$$A(\Pi') \geq 2\sqrt{3}(n-1) + (2-\sqrt{3})\left\lceil\sqrt{12n-3}-3\right\rceil + \pi. \tag{6.3}$$

Definition 6.10 (Wegner Packing)**.** *A Groemer packing $B_1, \ldots, B_n$ is a Wegner packing if $P(C_n) = 2\left\lceil\sqrt{12n-3}-3\right\rceil$.*

The main result in [Wegner 1986] is the generalization of (6.3).

Theorem 6.11 (Wegner Theorem)**.** *Let $B_1, \ldots, B_n$ be a packing and Π' be a convex region containing all circles. Then* (6.3) *holds. Equality holds if and only if the packing is a Wegner packing.*

Hence, if there is a Wegner packing for a given number n of circles, all optimal packings are Wegner packings. However, there might exist multiple Wegner packings ($n = 18$) or no Wegner packing ($n = 121$). But even if there is no Wegner packing, there are near optimal Groemer packings.

Theorem 6.12 (Wegner Theorem)**.** *For all $n \geq 2$ there is a Groemer packing of n circles such that for the convex hull Π' we have*

$$A(\Pi') \leq 2\sqrt{3}(n-1) + (2-\sqrt{3})\left\lceil \sqrt{12n-3} - 2 \right\rceil + \pi. \tag{6.4}$$

The following theorem of [Böröczky; Ruzsa 2007] characterizes the number of circles for which Wegner packings exist.

Theorem 6.13. *For given $n \geq 2$, there is a Wegner packing of n unit circles if and only if*

$$\exists\, k, m \in \mathbb{N}_+ : \left\lceil \sqrt{12n-3} \right\rceil^2 + 3 - 12n \neq (3k-1)\cdot 9^m.$$

For $N \in \mathbb{N}$ let W_N be the set of Wegner numbers in $\{2, \ldots, N\}$. Then it is

$$\lim_{N\to\infty} \frac{|W_N|}{N} = \frac{23}{24}.$$

This yields the following conjecture [Böröczky; Ruzsa 2007].

Conjecture 6.14. *Any densest packing of n unit circles is some Groemer packing.*

6.2 Wire Length Optimal Packings

The problem to pack identical connected circles such that their wire length is minimized has been hardly studied in literature. In [Graham; Sloane 1990] the problem has first been considered and in [Chow 1995] it is solved for lattice packings. There it is proved that the optimal packings have a circular shape. In this section we state the main results of the publications [Graham; Sloane 1990] and [Chow 1995].

Definition 6.15. *For a finite set $C \subset \mathbb{R}^2$ of point with gravity center c_0, the wire length is*

$$\mathrm{wl}(C) := \sum_{c\in C} ||c - c_0||^2 .$$

Consider n unit circles which centers are pairwise connected. Let $c_1, \ldots, c_n$ be the centers of the circles. Ignoring the constant scaling, by Lemma 3.11 the wire length is equal to the wire length defined in Definition 6.15. The question for a given number n is to find a packing of n unit circles such that its wire length is minimal.

Although not proven, in [Graham; Sloane 1990] the following conjecture was stated.

Conjecture 6.16. *For $n \neq 4$ the packing with minimal wire length is a hexagonal lattice packing.*

The case $n = 4$ is an exception, as there are several placements with wire length 4.

We now state the main theoretical results of [Graham; Sloane 1990] and, if related, [Calderbank; Sloane 1987]. Note that in these papers a different scaling of the hexagonal lattice is used.

Definition 6.17 (Circular Cluster)**.** *Let $c_0 \in \mathbb{R}^2$ and $t \geq 0$. Then let $C = \{c \in A_2 : |c - c_0| \leq t\}$. If the centroid of C is c_0 again, C is called a circular cluster.*

Lemma 6.18 (Lower Bound on Wire Length)**.** *Let C be the centers of a circle packing with n circles. Then*

$$\mathrm{wl}(C) \geq \sum_{k=1}^{n} \left(\sqrt{\frac{2\sqrt{3}}{\pi}(k-1) + \frac{3}{4}} - \frac{\sqrt{3}}{2} \right)^2 = \frac{\sqrt{3}}{\pi} n^2 + O(n).$$

Lemma 6.19 (Wire Length of Circular Clusters)**.** *For circular clusters C with n circles in hexagonal lattices centered at lattice points it is*

$$\mathrm{wl}(C) = \frac{\sqrt{3}}{\pi} n^2 + O(n).$$

Corollary 6.20. *Denote C_n the centers of the wire length optimal packing with n circles. Then*

$$\lim_{n \to \infty} \frac{1}{n^2} \mathrm{wl}(C_n) = \frac{\sqrt{3}}{\pi}.$$

In [Chow 1995] based on Conjecture 6.16 mainly hexagonal lattice packings are considered. They call a package A_2-optimal if it is the optimal packing of all hexagonal lattice packings.

Theorem 6.21. *All A_2-optimal packings are circular clusters.*

They remarked that the converse is not true. There might be circular clusters that are not optimal. Furthermore, they proposed a polynomial time algorithm to find an optimal A_2-packing.

6.3 Asymptotic Analysis of Shaped Clusters

In this section we assume Λ to be a lattice such that each two lattice points have a distance at least two.

First we introduce some notations and state some results known in the literature. Later we prove the main results of this chapter. We consider packings of an increasing number of congruent circles in one net. Then asymptotically the packings with optimal wire length are area optimal, while the packings with optimal area are not wire length optimal.

Definition 6.22 (Center of Gravity). *Let $\Pi \subset \mathbb{R}^2$. The center of gravity $R(\Pi)$ is*

$$R(\Pi) := \frac{1}{A(\Pi)} \int_{\Pi} x \, \mathrm{d}x.$$

Definition 6.23 (Moment of Inertia). *Let $\Pi \subset \mathbb{R}^2$. Let the point x_0 be the center of rotation. Then the moment of inertia $J(\Pi)$ around x_0 is defined as*

$$J_{x_0}(\Pi) := \int ||x - x_0||^2 \, \mathrm{d}x.$$

With $J(\Pi)$ we denote the moment of inertia around the gravity center.

Theorem 6.24 (Parallel Axis Theorem). *Let be $\Pi \subset \mathbb{R}^2$. Let $x_0 \in \mathbb{R}^2$ be the gravity center of Π and $x_1 \in \mathbb{R}^2$ be arbitrary. Then*

$$J_{x_1}(\Pi) = J(\Pi) + A(\Pi) \cdot ||x_1 - x_0||^2 .$$

Definition 6.25 (Shaped Cluster). *Let $\Pi \subset \mathbb{R}^2$ be convex and compact. Then $\Lambda \cap \Pi$ is a shaped cluster. For $\lambda \geq 0$ we define the scaled shaped cluster by $\lambda\Pi := \{\lambda x : x \in \Pi\}$.*

By the definition of the Riemann integral and equality of Lebesgue and Riemann integral in the case of existence, the following statement holds.

Lemma 6.26. *Let $\Pi \subset \mathbb{R}^2$ be compact and $f : \Pi \to \mathbb{R}$ be continuous. Let $\Pi_\lambda \subset \Pi$ be a family of finite subsets. Let $V_\lambda(x)$ be the Voronoi cell of x with respect to Π_λ. If*

$$\lim_{\lambda\to\infty} \sup_{x\in\Pi_\lambda} A(V_\lambda(x)) = 0,$$

it is

$$\int_{\Pi} f(x) \, \mathrm{d}x = \lim_{\lambda\to\infty} \sum_{x\in\Pi_\lambda} f(x) A(V_\lambda(x)).$$

We now know that for $\Pi_\lambda = \frac{1}{\lambda}\Lambda \cap \Pi$ the area of the Voronoi cells not at the boundary is $\frac{\det(\Lambda)}{\lambda^2}$. For increasing λ the boundary cells become irrelevant. Then by Lemma 6.26 it follows

$$\int_{\Pi} f(x) \, \mathrm{d}x = \lim_{\lambda\to\infty} \sum_{x\in\Pi_\lambda} f(x) A(V_\lambda(x)) = \lim_{\lambda\to\infty} \frac{\det(\Lambda)}{\lambda^2} \sum_{x\in\frac{1}{\lambda}\Lambda\cap\Pi} f(x). \tag{6.5}$$

Especially for $f(x) = 1$ we get the identity

$$\lim_{\lambda\to\infty} \frac{|\lambda\Pi \cap \Lambda|}{\lambda^2} = \lim_{\lambda\to\infty} \frac{\left|\Pi \cap \frac{1}{\lambda}\Lambda\right|}{\lambda^2} = \lim_{\lambda\to\infty} \frac{1}{\lambda^2} \sum_{x\in\frac{1}{\lambda}\Lambda\cap\Pi} 1 = \frac{1}{\det(\Lambda)} \int_{\Pi} 1 \, \mathrm{d}x = \frac{A(\Pi)}{\det(\Lambda)}. \tag{6.6}$$

We now state the first main result of this section. Intuitively a hexagonal lattice packing in a shaped cluster can have suboptimal packing density because of poor local packing density at the boundary of the cluster. However, if we scale the shaped cluster, the boundary becomes less important. Thus for increasing scaling factor we expect the packing density of all shaped clusters to converge to the optimal packing density. This is stated in Theorem 6.27.

Theorem 6.27 (The asymptotic packing density of shaped clusters is optimal)**.** *Let* $\Pi \subset \mathbb{R}^2$ *be convex and compact. Denote by* $\mathcal{B}_\lambda$ *the packing of unit circles with center in* $\lambda\Pi \cap \Lambda$*. Denote the packing density of* $\mathcal{B}_\lambda$ *by* $\delta(\lambda\Pi \cap \Lambda)$*. Then*

$$\lim_{\lambda\to\infty} \delta(\lambda\Pi \cap \Lambda) = \frac{\pi}{\det(\Lambda)}.$$

Proof. Denote by $\operatorname{conv}(\mathcal{B}_\lambda)$ the convex hull of the circles in $\mathcal{B}_\lambda$. For the packing density it is by Lemma 6.4

$$\delta(\lambda\Pi \cap \Lambda) = \frac{|\mathcal{B}_\lambda| \cdot \pi}{A(\operatorname{conv}(\mathcal{B}_\lambda))}. \tag{6.7}$$

With D denoting the unit disc it is $\lambda\Pi \subset \operatorname{conv}(\mathcal{B}_\lambda) \subset \lambda\Pi + D$. Then it is

$$A(\lambda\Pi) \le A(\operatorname{conv}(\mathcal{B}_\lambda)) \le A(\lambda\Pi + D) = A(\lambda\Pi) + \pi$$

where the last equality holds by convexity of $\lambda\Pi$. It follows for $\lambda \to \infty$

$$\begin{aligned} &\frac{A(\lambda\Pi)}{\lambda^2} = A(\Pi) \quad \wedge \quad \lim_{\lambda\to\infty} \frac{A(\lambda\Pi) + \pi}{\lambda^2} = A(\Pi) \\ \Longrightarrow\ &\lim_{\lambda\to\infty} \frac{A(\operatorname{conv}(\mathcal{B}_\lambda))}{\lambda^2} = A(\Pi). \end{aligned}$$

Furthermore, it is by (6.6)

$$\lim_{\lambda\to\infty} \frac{|\mathcal{B}_\lambda|}{\lambda^2} = \lim_{\lambda\to\infty} \frac{|\lambda\Pi \cap \Lambda|}{\lambda^2} = \frac{A(\Pi)}{\det(\Lambda)}.$$

So we get by (6.7)

$$\lim_{\lambda\to\infty} \delta(\lambda\Pi \cap \Lambda) = \frac{\lim_{\lambda\to\infty} \frac{|\mathcal{B}_\lambda|}{\lambda^2} \cdot \pi}{\lim_{\lambda\to\infty} \frac{A(\operatorname{conv}(\mathcal{B}_\lambda))}{\lambda^2}} = \frac{\frac{A(\Pi)}{\det(\Lambda)}\pi}{A(\Pi)} = \frac{\pi}{\det(\Lambda)}.$$

□

We know that the wire optimal packings are shaped clusters $A_2 \cap D_r$. So, by Theorem 6.27 they have an asymptotic packing density of $\delta = \pi/\sqrt{12}$. We already know that optimal Groemer packings have an asymptotic packing density of $\delta = \pi/\sqrt{12}$. This means that if we optimize the wire length, asymptotically we also optimize the packing density.

Theorem 6.28. *Let* $\Pi \subset \mathbb{R}^2$ *be convex and compact. Then*

$$\lim_{\lambda\to\infty} \frac{\operatorname{wl}(\Lambda \cap \lambda\Pi)}{\lambda^4} = \frac{J(\Pi)}{\det(\Lambda)}.$$

Proof. We first compute with (6.5) and (6.6)

$$\lim_{\lambda\to\infty} R\left(\frac{1}{\lambda}\Lambda\cap\Pi\right) = \lim_{\lambda\to\infty} \frac{\lambda^2}{\left|\frac{1}{\lambda}\Lambda\cap\Pi\right|}\cdot\frac{1}{\lambda^2}\sum_{x\in\frac{1}{\lambda}\Lambda\cap\Pi} x = \frac{\det(\Lambda)}{A(\Pi)}\frac{1}{\det(\Lambda)}\int_\Pi x\,\mathrm{d}x = R(\Pi).$$

Then it is by applying (6.5)

$$\begin{aligned}\lim_{\lambda\to\infty}\frac{\mathrm{wl}(\Lambda\cap\lambda\Pi)}{\lambda^4} &= \lim_{\lambda\to\infty}\frac{1}{\lambda^2}\mathrm{wl}\left(\frac{1}{\lambda}\Lambda\cap\Pi\right) = \lim_{\lambda\to\infty}\frac{1}{\lambda^2}\sum_{x\in\frac{1}{\lambda}\Lambda\cap\Pi}\left(x - R\left(\frac{1}{\lambda}\Lambda\cap\Pi\right)\right)^2 \\ &= \frac{1}{\det(\Lambda)}\int_\Pi (x-R(\Pi))^2\,\mathrm{d}x = \frac{J(\Pi)}{\det(\Lambda)}.\end{aligned}$$

□

The following moments of inertia are known from literature. The moment of inertia of a disc D with radius r is $J(D) = \frac{1}{2}\pi r^4 = \frac{1}{2\pi}A(D)^2$. For a hexagon Π with given area the moment of inertia is $J(\Pi) \geq \frac{5}{18\sqrt{3}}A(\Pi)^2$, where equality holds for the regular hexagon.

Theorem 6.29. *Now denote by P_n the area optimal packing A_2-packing of n circles and by Q_n the wire length optimal packing for n circles. Then*

$$\begin{aligned}\lim_{n\to\infty}\frac{\delta(Q_n)}{\delta(P_n)} &= 1,\\ \limsup_{n\to\infty}\frac{\mathrm{wl}(P_n)}{\mathrm{wl}(Q_n)} &\geq \frac{5\pi}{9\sqrt{3}} \approx 1,00767.\end{aligned}$$

Proof. First statement: For all n it is $\delta(P_n) \geq \sigma = \pi/\sqrt{12}$. Furthermore, by Theorem 6.21 all Q_n are circular cluster, i.e. with D denoting the unit disc the centers of Q_n are $\lambda_n D\cap\Lambda$ for some λ_n. As $n\to\infty$ it is $\lambda_n\to\infty$ and thus by Theorem 6.27 it is

$$\lim_{n\to\infty}\delta(Q_n) = \lim_{\lambda\to\infty}\delta(\lambda D\cap\Lambda) = \frac{\pi}{\det(\Lambda)} = \frac{\pi}{\sqrt{12}}.$$

Hence, it is

$$\frac{\pi}{\sqrt{12}} \leq \delta(P_n) \leq \delta(Q_n) \to \frac{\pi}{\sqrt{12}}$$

and the first statement follows.

Second statement: We know by Theorem 6.12 and Theorem 6.13 that for $\frac{23}{24}$ of the natural numbers the area optimal packing P_n are hexagonal lattice packings. Let H be a regular hexagon and denote by n_k the infinite subsequence where the packing is the shaped cluster $\mu_{n_k}\Pi \cap H$. Then as $k \to \infty$, it is $\mu_{n_k} \to \infty$. Furthermore, recall that Q_n is the shaped cluster $\lambda_n D \cap \Lambda$. Thus, we get

$$\begin{aligned}\limsup_{n\to\infty} \frac{\mathrm{wl}(P_n)}{\mathrm{wl}(Q_n)} &\geq \lim_{k\to\infty} \frac{\mathrm{wl}(P_{n_k})}{\mathrm{wl}(Q_{n_k})} = \lim_{k\to\infty} \frac{\mathrm{wl}(\lambda_{n_k} H \cap \Lambda)}{\mathrm{wl}(\lambda_{n_k} D \cap \Lambda)} \\ &= \frac{\lim_{\lambda\to\infty} \frac{\mathrm{wl}(\lambda H \cap \Lambda)}{\lambda^4}}{\lim_{\lambda\to\infty} \frac{\mathrm{wl}(\lambda D \cap \Lambda)}{\lambda^4}} = \frac{J(H)}{J(D)} = \frac{5\pi}{9\sqrt{3}}.\end{aligned}$$

□

Hence, for A_2-packings the wire length optimal packings asymptotically are also optimal in packing density, while the converse is not true, i.e. the packing density optimal packings asymptotically are not wire length optimal.

Note that it is expected in Conjecture 6.14 that all densest packings are Groemer packings. Furthermore, in Conjecture 6.16 it is expected that the wire length optimal packings are indeed A_2 packings. Under these assumptions for all (not just A_2) packings, wire length optimal packings asymptotically are packing density optimal. Conversely, all packing density optimal packings have inferior wire length.

7 Circle Placement Problem

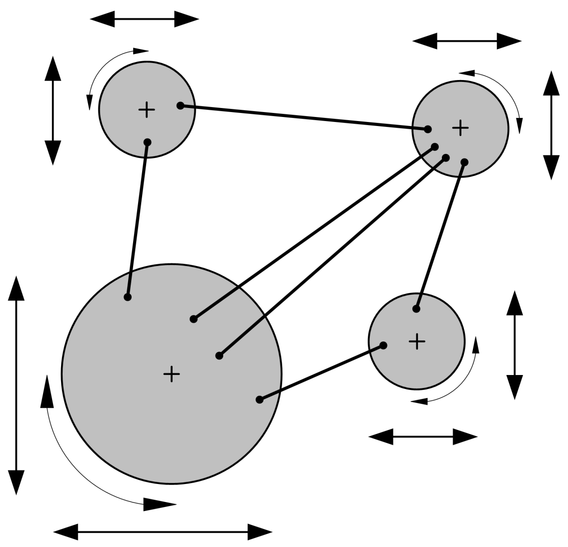


Figure 7.1: The circle placement problem is to rotate and translate connected circles such that they do not overlap and the wire length is minimized.

In this chapter we consider the *circle placement problem*. For this problem a set of fixed sized circles is given. Each circle has pins with fixed offset to the circle center. Hence, the positions of these pins depend on the location and the rotation of the circles. The pins of the circles are connected by nets. The circle placement problem is then to place and rotate the circles in the plane, such that they do not overlap and the wire length is minimized.

The circle placement problem is solved in an initial step of the rounded rectangle algorithm stated in Chapter 8. In this step, all rectangular components are represented as circles. The solution of this circle placement step is then used for transforming the circles to rectangles.

The circle placement problem is related to circle packing and facility layout problems, which already have been studied in the literature. Several heuristics such as monotonic basin hopping for circle packing problems or the attractor repeller model for facility layout problems have been proposed. However, in these approaches the circles centers are connected and the rotation is disregarded.

The circle placement problem is highly non-convex. Hence, heuristics are applied to reach good local optimal solutions. In this chapter we develop such heuristics. In particular, we extend and improve the attractor repeller model to generate good starting points for a constrained non-linear program. Furthermore, based on monotonic basin hopping we develop techniques to overcome local optima in a post-processing step.

In Section 7.1 we formulate the circle placement problem as non-linear program. In Section 7.2 we survey related results of the literature. To our best knowledge, the circle placement problem has not been considered in the literature. However, it is related to the circle packing problem and to facility layout problems. We summarize the known results of these areas that can be applied to the circle placement problem.

In Section 7.3 we extend the attractor repeller model known for the facility layout problem. This model yields good results in practice but has the theoretical drawback of being scaling variant. We propose several models that satisfy the important scaling invariance property. By comparing our models in Section 7.7.1, we show that one of our models outperforms the previously known model in terms of running time and solution quality.

In Section 7.4 we apply the attractor repeller model to the rectangle placement problem. In Section 7.5 the constrained non-linear program for the circle placement problem is analyzed. In particular, we show that it satisfies the important Mangasarian-Fromovitz constraint qualification (MFCQ) but not the linear independence constraint qualification (LICQ). In Section 7.6 we consider strategies to overcome local optimal solutions of the circle placement problem. Based on results of Chapter 3 we develop a monotonic basin hopping and a local search strategy. In Section 7.7.2 we show that our heuristics improve the solution quality, while the running time is only increased moderately.

In Section 7.7 we make a detailed numerical analysis of the algorithms proposed in this chapter. We show that they have moderate running times and lead to good solutions. Furthermore, we identify the best sequence of heuristics that is used as initial solution generator in Chapter 8.

7.1 Problem Statement

We now formalize the circle placement problem. Given is:

- A set $\mathcal{C}$ of n circles with fixed radii r_j, $j \in \mathcal{C}$.
- A set $\mathcal{P}$ of m pins. Pin l is connected to circle $j(l)$ and has fixed offset $p_l \in \mathbb{C}$ from its center.
- A set $\mathcal{N}$ of nets. Net k has m_k pins $\mathcal{P}_k$ and weight μ_k.

A placement is defined by the location and rotation of each circle. The optimization problem is to find a placement without overlap such that the wire length is minimized.

Placement Encoding

The location of a circle j is defined by the position $c_j \in \mathbb{C}$ of its center. The vector of circle centers is $\boldsymbol{c} = (c_1, \dots, c_n)^T \in \mathbb{C}^n$.

We studied different representations of the rotation in Chapter 3. The angle rotation encodes the rotation of circle i by its rotation angle φ_i. The Euclidean rotation represents the rotation of circle i by a $z_i \in \mathbb{C}$ and adds the constraint $|z_i| = 1$. This encoding yields an all quadratic program for the circle placement problem. However, it requires more variables and additional constraints than the angle rotation.

Based on our analysis in Chapter 3, the angle rotation is superior to the Euclidean rotation in practical implementations. Thus, we focus on the angle rotation in this chapter. However, sometimes the formulation in the Euclidean rotation is more convenient. For given angle rotations $\boldsymbol{\varphi} = (\varphi_1, \dots, \varphi_n) \in \mathbb{R}^n$ the Euclidean rotations $\boldsymbol{z}$ are given by

$$\boldsymbol{z} = (z_1, \dots, z_n)^T = (\imath\varphi_1, \dots, \imath\varphi_n)^T =: \exp(\imath\boldsymbol{\varphi}).$$

A solution to the circle placement problem is defined by the vector $\boldsymbol{c} \in \mathbb{C}^n$ of circle centers and $\boldsymbol{\varphi} \in \mathbb{R}^n$ of circle rotations.

Formulation as Non-Linear Program

The circles i and j do not overlap if $|c_i - c_j| \geq r_i + r_j$. To get a smooth formulation, in non-linear programs this is often stated as $|c_i - c_j|^2 \geq (r_i + r_j)^2$.

Furthermore, the computation of the wire length is similar to Section 3.1. In order to keep this chapter self contained, we briefly summarize the computation steps here that lead to the wire length formulation (3.1).

The absolute position q_l of pin l is given by $q_l = c_j + z_j p_l$ with $j = j(l)$. Let $\Phi = (\phi_{ij}) \in \{0,1\}^{m \times n}$ be the circle-to-pin matrix, i.e. $\phi_{lj} = 1$ if and only if pin l is on circle j. Furthermore, let $P = \operatorname{diag}(p_1, \dots, p_m)$ be the diagonal matrix of pin offsets. Then we get the absolute pin positions

$$\boldsymbol{q} = \Phi\boldsymbol{c} + P\Phi\boldsymbol{z}.$$

By Definition 2.43, the wire length is a positive semidefinite form $\boldsymbol{q}^H Q \boldsymbol{q}$ with $Q \in \mathbb{R}^{m \times m}$ of the pin positions. As $\boldsymbol{q}$ is linear in $(\boldsymbol{c}, \boldsymbol{z})$, the wire length $\operatorname{wl}(\boldsymbol{c}, \boldsymbol{z})$ is a positive semidefinite form of $(\boldsymbol{c}, \boldsymbol{z})$:

$$\begin{aligned} \mathrm{wl}(\boldsymbol{c},\boldsymbol{z}) &= \boldsymbol{c}^H(\Phi^T Q\Phi)\boldsymbol{c} + \boldsymbol{z}^H(\Phi^T P^H QP\Phi)\boldsymbol{z} + 2\Re(\boldsymbol{c}^H(\Phi^T QP\Phi)\boldsymbol{z}) \\ &= \boldsymbol{c}^H A\boldsymbol{c} + \boldsymbol{z}^H B\boldsymbol{z} + \Re(\boldsymbol{c}^H U\boldsymbol{z}) \end{aligned} \tag{7.1}$$

with

$$A = \Phi^T Q\Phi \in \mathbb{R}^{n\times n}, \qquad B = \Phi^T P^H QP\Phi \in \mathbb{C}^{n\times n}, \qquad U = 2\Phi^T QP\Phi \in \mathbb{C}^{n\times n}.$$

In previous chapters, we formulated the problems in complex variables and used the straight forward transformation into real variables in the numerical evaluations. This is not completely trivial in this case. With $\boldsymbol{c} = \boldsymbol{x} + \imath\boldsymbol{y}$ we explicitly formulate the wire length $\mathrm{wl}(\boldsymbol{c},\boldsymbol{\varphi})$ as real function depending on $\boldsymbol{x}, \boldsymbol{y}$ and $\boldsymbol{\varphi}$. Therefore, split $B = B_r + \imath B_i$ and $U = U_r + \imath U_i$ in real and imaginary part. As B is Hermitian, B_r is symmetric and B_i is skew symmetric. As U is not Hermitian, U_r and U_i are neither symmetric nor skew symmetric. Furthermore, set $\cos(\boldsymbol{\varphi}) = (\cos(\varphi_1), \dots, \cos(\varphi_n))$ and similar $\sin(\boldsymbol{\varphi}) = (\sin(\varphi_1), \dots, \sin(\varphi_n))$. Then

$$\begin{aligned} \mathrm{wl}(\boldsymbol{c},\boldsymbol{\varphi}) &= \mathrm{wl}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\varphi}) \\ &= \boldsymbol{x}^T A\boldsymbol{x} + \boldsymbol{y}^T A\boldsymbol{y} \\ &\quad + \cos(\boldsymbol{\varphi})^T B_r \cos(\boldsymbol{\varphi}) + \sin(\boldsymbol{\varphi})^T B_r \sin(\boldsymbol{\varphi}) + 2\sin(\boldsymbol{\varphi})^T B_i \cos(\boldsymbol{\varphi}) \\ &\quad + \boldsymbol{x}^T U_r \cos(\boldsymbol{\varphi}) + \boldsymbol{y}^T U_r \sin(\boldsymbol{\varphi}) + \boldsymbol{y}^T U_i \cos(\boldsymbol{\varphi}) - \boldsymbol{x}^T U_i \sin(\boldsymbol{\varphi}). \end{aligned} \tag{7.2}$$

We formulate the circle placement problem as non-linear program CPP

$$\begin{aligned} \min \quad & \mathrm{wl}(\boldsymbol{c},\boldsymbol{\varphi}) \\ \text{s. t.} \quad & (r_i + r_j)^2 - |c_i - c_j|^2 \le 0 \quad 1 \le i < j \le n. \end{aligned} \tag{7.3}$$

7.2 Literature Survey

The circle placement problem is non-convex and has many local minima. Any local optimization algorithm very likely converges to a rather poor local optimum.

There exist optimization techniques based on branch and bound or interval arithmetic that provably find the global optimal solution to non-convex optimization problems. However, these approaches have impractical running time for real world problems. More practical are multiple shooting algorithms. They do not guarantee global optimality but try to overcome poor local optima by doing several local optimization runs from different starting points. However, the selection of the starting points is crucial. To use such approaches, the problem structure must be exploited.

The circle placement problem is related to circle packing problems and even closer to facility layout problems. Therefore, we give a brief survey of the methods known in literature for these problems.

7.2.1 Circle Packing

The circle packing problem is to pack circles with given radius into a container (e.g rectangle, square, circle), such that the circles do not overlap and the container size is minimized. An often considered special case is the uniform circle packing, where all circles have the same radius.

Circle packing in different shapes differs only by the container constraints which enforce the circles to be in the container. For $a \in \mathbb{R}$ let $C(a)$ be a container increasing with a, e.g. a square with edge length a or a circle with radius a. Given is a set of n circles with radius r_i, $i = 1, \ldots, n$. The decision variables are the circle centers $c_i \in \mathbb{C}$ and the container size a. Then the circle packing problem to pack n circles with fixed radius into a the container $C(a)$ is

$$\begin{aligned} \min \quad & a \\ \text{s. t.} \quad & |c_i - c_j| \geq r_i + r_j, \quad 1 \leq i < j \leq n && (7.4) \\ & D(x_i, r_i) \subset C(a), \quad i = 1, \ldots, n. && (7.5) \end{aligned}$$

While (7.4) enforces the non-overlapping of the circles, (7.5) assure that the circles are within the container. As the non-overlap constraint (7.4) is not differentiable for $c_i = c_j$, it is often formulated as $|c_i - c_j|^2 \geq (r_i + r_j)^2$. This formulation is smooth.

Rare theoretical results exists for the circle packing problem. Even the simpler case of packing equal circles into the unbounded plane is a theoretically hard task, see Chapter 6. For equal circle packing of few circles there are some optimality proofs, usually computer aided by global solvers. See e.g. [Markót; Csendes 2005] for packing identical circles in a square or [Fodor 1999] and [Fodor 2003] for packing identical circles in a circle. However, these algorithms are impractical for real world problems.

Most literature focus on heuristic algorithms to find good solutions. Due to the non-overlapping constraints, circle packing problems have a large number of local optima which makes it computational intractable to find a global optimum. They have both a discrete and a continuous structure, in [Addis; Locatelli; Schoen 2008] described as funnel landscape. Any solution technology that solves circle packing problems efficiently has to exploit this special hybrid structure. For a recent survey see [Castillo; Kampas; Pintér 2008] and [M'Hallah; Hifi 2009]. We briefly summarize the main ideas here.

In [Maranas; Floudas; Pardalos 1995] the circle packing problem is solved as a non-linear global optimization problem. It can also be formulated as the difference of convex functions, see [Horst; Thoai 1999]. Furthermore, it falls into the class of all quadratic optimization problems analyzed in [Raber 1999]. However, to efficiently solve the problem, the geometric structure has to be exploited.

There exist some interesting non-linear reformulations. Several optimization methods stated in literature rely on the equivalence of the packing problem to the scattering problem. The *scattering problem* is to place n points in the unit square such that the smallest distance between two points is maximized. This problem is equivalent to minimize a square containing n unit circles.

In [Nurmela; Östergård 1997] the unit circle packing problem in a square is considered as scattering problem. They state it as the problem to minimize

$$\sum_{1 \le i < j \le n} \left(\frac{\lambda}{|c_i - c_j|^2} \right)^m$$

subject to c_i being in the unit square. Here λ is a scaling factor and m increased during the algorithm. As m tends to infinity, only the minimal distance of two circles has influence on the objective. By a trigonometric coordinate transformation, they transform this problem to an unconstrained non-convex problem and use multiple shoot approaches.

An idea in [Birgin; Sobral 2008] to improve performance of circle packing formulations is to replace the constraints (7.4) by the single constraint

$$\sum_{1 \le i < j \le n} \max(0, (r_i + r_j)^2 - |c_i - c_j|^2)^2 = 0. \tag{7.6}$$

In theory both equations are equivalent. However, in (7.4) $n(n-1)/2$ constraints have to be evaluated, while in (7.6) by sophisticated data structures for reasonable distributed circles only $O(n)$ constraints have to be evaluated.

A successful and for the general case most promising approach is monotonic basin hopping (MBH). This technique has been introduced in [Wales; Doye 1997], and since then is a well known technique in global optimization. It is is applied to circle packing problems in [Addis; Locatelli; Schoen 2008] and [Grosso et al. 2010]. It can be applied to circle packing problems in any kind of container and even be transferred to facility layout problems.

MBH is essentially a combination of local search and multiple shooting algorithms. Starting from a local optimum of the circle packing problem, the problem is resolved starting from a perturbation of the local optimal solution. If this run improves the objective, the new solution is chosen. Otherwise it is discarded. The algorithm stops after a fixed number of consecutive fails.

There are different perturbation schemes in the literature. In [Grosso et al. 2010] the circle center coordinates are randomly shifted in some interval $[-\Delta, \Delta]$. There are several extensions to this perturbation. Sometimes only a subset of the circles is shifted, sometimes the choice of the perturbation interval depends on the circle radius. A different rule is proposed in [Addis; Locatelli; Schoen 2008] for unequal circles. They randomly choose two circles with not too different radii and exchange the centers of these circles.

7.2.2 Facility Layout

Facility layout problems are closely related to the circle placement problem. In facility layout problems, shapes (e.g. rectangles or circles) of fixed size are to be placed in the plane, such that the shapes do not overlap and a weighted connection between the shapes is minimized. A recent survey of the literature on facility layout problems is given in [Castillo; Kampas; Pintér 2008].

Due to the non-overlapping constraints, circle facility layout problems have the same hybrid discrete and a continuous structure as circle packing problems. Accordingly, several discrete and continuous approaches exist. Recent algorithms use a hybrid strategy of discrete combinatorial and non-linear algorithms.

Facility layout problems for circular objects have been studied in literature. There the objective usually is the weighted sum of the distance of the circle centers c_i.

Definition 7.1 (Circle Facility Layout Problem)**.** *Let $w_{ij} \geq 0$ be the weight of the distance of the circles $i < j$. Let r_i be the radius of circle i. Denote $\boldsymbol{w} = (w_{ij})_{1\leq i<j\leq n}$. Then the facility layout problem $FLP(\boldsymbol{w}, \boldsymbol{r})$ is the non-linear program*

$$\min_{\boldsymbol{c}} \quad \sum_{1\leq i<j\leq n} w_{ij} \left|c_i - c_j\right|^2 \tag{7.7}$$

$$\text{s. t.} \quad \left|c_i - c_j\right|^2 \geq r_i + r_j^2, \quad 1 \leq i < j \leq n. \tag{7.8}$$

In the objective (7.7) and the constraints (7.8) the squared distance $\left|c_i - c_j\right|^2$ is used to make the model smooth.

In [Drezner 1980] the two phase DISCON (DISpersion-CONcentration) algorithm is considered. In the dispersion phase, starting from all circle centers in the origin, the circles are dispersed. Using this as an initial solution, in the concentration phase a dense packing is created.

In [Anjos; Vannelli 2006] the attractor-repeller model is proposed. In this model the non-overlapping constraints are included as penalty in the objective. Thus for each pair of circle i, j, beside the attractive force $w_{ij}\left|c_i - c_j\right|^2$ there is an additional repelling force $\alpha(r_i + r_j)^2 / \left|c_i - c_j\right|^2$. This repelling force pushes the circles away from each other and omits strong overlapping. The attractor repeller model yields good results in practice. We analyze and improve it in Section 7.3.

7.3 Generalized Attractor Repeller Models

In this section we identify an important theoretical and practical drawback of the attractor repeller model as stated in [Anjos 2001] and [Anjos; Vannelli 2006]. The structure of the solution of the attractor repeller model changes depending on scaling of circle radii and connection weights.

In Section 7.3.1 we explain the concept of scaling invariance for the facility layout problem. In Section 7.3.2 we summarize the original attractor repeller model stated in [Anjos; Vannelli 2006] and show that it is not scaling invariant. In the remaining sections we propose three scaling invariant modifications of the attractor repeller model. In Section 7.3.3 the global scaling invariant attractor repeller model is stated. This model is the smallest modification of the standard model to become scaling invariant. In Section 7.3.4 we propose the local scaling invariant attractor repeller model. This model attempts to control the overlapping of the circles by choosing an individual weight of the repeller force for each pair of circles. In Section 7.3.5 the limited range attractor repeller model is stated. By limiting the range of the repeller force, this model avoids that the circles are pushed too far away from each other. In Section 7.3.6 the we compare the different models.

7.3.1 Scaling Invariance

An important property of the facility layout problem $FLP(\boldsymbol{w}, \boldsymbol{r})$ defined in Definition 7.1 is its scaling invariance. Denote the objective (7.7) by $f[\boldsymbol{w}, \boldsymbol{r}]$ and the non-overlapping constraints (7.8) by $g[\boldsymbol{w}, \boldsymbol{r}]$.

Geometrically there are two intuitions.

- If we scale all connection weights by a common factor $\mu \geq 0$, the structure of the problem does not change. This means, the set of feasible solutions remains equal and the objective scales by a common factor. More formally
$$f[\mu\boldsymbol{w}, \boldsymbol{r}](\boldsymbol{c}) = \mu f[\boldsymbol{w}, \boldsymbol{r}](\boldsymbol{c}), \qquad g[\mu\boldsymbol{w}, \boldsymbol{r}](\boldsymbol{c}) = g[\boldsymbol{w}, \boldsymbol{r}](\boldsymbol{c}).$$
- If we scale all radii by a common factor $\gamma \geq 0$, the resulting solutions are only stretched. So $\gamma\boldsymbol{c}$ is feasibly to the stretched problem if $\boldsymbol{c}$ is feasible to the original problem. Furthermore, by this stretching the wire length scales by γ^2. More formally
$$f[\boldsymbol{w}, \gamma\boldsymbol{r}](\gamma\boldsymbol{c}) = \gamma^2 f[\boldsymbol{w}, \boldsymbol{r}](\boldsymbol{c}), \qquad g[\boldsymbol{w}, \gamma\boldsymbol{r}](\gamma\boldsymbol{c}) = \gamma^2 g[\boldsymbol{w}, \boldsymbol{r}](\boldsymbol{c}).$$

In particular, if $\boldsymbol{c}^*$ is the optimal solution to $FLP(\boldsymbol{w}, \boldsymbol{r})$ with objective value f^*, then $\gamma\boldsymbol{c}^*$ it the optimal solution to $FLP(\mu\boldsymbol{w}, \gamma\boldsymbol{r})$ with objective value $\mu\gamma^2 f^*$.

In the attractor repeller model the non-overlapping constraints are included as penalty, called repeller term, in the objective. The facility layout problem $FLP(\boldsymbol{w}, \boldsymbol{r})$ stated in Definition 7.1 is replaced by an unconstrained minimization of a function $\mathrm{ar}[\boldsymbol{w}, \boldsymbol{r}, \alpha] : \mathbb{C}^n \to \mathbb{R}$, where α is the weight of the repeller term. For larger α there is less overlap of the circles.

Definition 7.2 (Scaling Invariance). *A parametrized family of functions* $\operatorname{ar}[\boldsymbol{w}, \boldsymbol{r}]$ *is scaling invariant in connection weight if for* $\mu \geq 0$

$$\forall \boldsymbol{c} \in \mathbb{C}^n : \quad \operatorname{ar}[\mu \boldsymbol{w}, \boldsymbol{r}](\boldsymbol{c}) = \mu \operatorname{ar}[\boldsymbol{w}, \boldsymbol{r}](\boldsymbol{c}).$$

We say $\operatorname{ar}[\boldsymbol{w}, \boldsymbol{r}, \alpha]$ *is scaling invariant in size if for* $\gamma \geq 0$

$$\forall \boldsymbol{c} \in \mathbb{C}^n : \quad \operatorname{ar}[\boldsymbol{w}, \gamma \boldsymbol{r}](\gamma \boldsymbol{c}) = \gamma^2 \operatorname{ar}[\boldsymbol{w}, \boldsymbol{r}](\boldsymbol{c}).$$

Especially $\operatorname{ar}[\boldsymbol{w}, \boldsymbol{r}]$ *is scaling invariant in connection weight and size if and only if for* $\mu, \gamma \geq 0$

$$\forall \boldsymbol{c} \in \mathbb{C}^n : \quad \operatorname{ar}[\mu \boldsymbol{w}, \gamma \boldsymbol{r}](\gamma \boldsymbol{c}) = \mu \gamma^2 \operatorname{ar}[\boldsymbol{w}, \boldsymbol{r}](\boldsymbol{c}).$$

We say that an attractor repeller model is scaling invariant, if its objective function is scaling invariant.

7.3.2 Standard Attractor Repeller Model

In [Anjos 2001] and [Anjos; Vannelli 2006] the attractor repeller model with repelling force $(r_i + r_j)^2 / |c_i - c_j|^2$ is proposed. They consider the unconstrained optimization problem $AR0(\boldsymbol{w}, \boldsymbol{r}, \alpha)$

$$\min_{\boldsymbol{c}} \quad \operatorname{ar}_0[\boldsymbol{w}, \boldsymbol{r}, \alpha](\boldsymbol{c}) = \sum_{1 \leq i < j \leq n} w_{ij} |c_i - c_j|^2 + \frac{\alpha (r_i + r_j)^2}{|c_i - c_j|^2}. \tag{AR0}$$

The model (AR0) is neither scaling invariant in connection weight nor in size. For example, consider two circles connected with weight w_{12}. Assume $\alpha = 1$. Then ar_0 has its minimum for $|c_1 - c_2| = \sqrt[4]{w_{12}} \cdot \sqrt{r_1 + r_2}$. So, if we scale the radii by a common factor, the optimal center distance in (AR0) would not scale by the same factor. Similar, for greater weights $w_{12} > 1$ the minimum would be attained for smaller values $|c_1 - c_2|$. We propose advanced models fixing both problems.

7.3.3 Global Invariant Attractor Repeller Model

The attractor part in (AR0) is the connection length and should not be modified. Instead the lack of scaling invariance in (AR0) can be repaired by modifying the repeller term.

We define the global scaling invariant attractor repeller model $AR1(\boldsymbol{w}, \boldsymbol{r}, \alpha)$ as

$$\min_{\boldsymbol{c}} \quad \operatorname{ar}_1[\boldsymbol{w}, \boldsymbol{r}, \alpha](\boldsymbol{c}) = \sum_{1 \leq i < j \leq n} w_{ij} |c_i - c_j|^2 + \alpha \beta[\boldsymbol{w}] \frac{(r_i + r_j)^4}{|c_i - c_j|^2}. \tag{AR1}$$

The factor β only depends on $\boldsymbol{w}$. Suitable choices of β are discussed in Theorem 7.4.

Theorem 7.3. *The function family* ar_1 *is scaling invariant in size.*

Proof. By computation we get

$$\begin{aligned}\mathrm{ar}_1[\boldsymbol{w}, \gamma\boldsymbol{r}, \alpha](\gamma\boldsymbol{c}) &= \sum_{1\le i<j\le n} \gamma^2 w_{ij} \left|c_i - c_j\right|^2 + \alpha\beta[\boldsymbol{w}]\frac{\gamma^4(r_i+r_j)^4}{\gamma^2\left|c_i - c_j\right|^2} \\ &= \gamma^2\,\mathrm{ar}_1[\boldsymbol{w}, \boldsymbol{r}, \alpha](\boldsymbol{c}).\end{aligned}$$

□

Theorem 7.4. *The function family* ar_1 *is scaling invariant in connection weight, if and only if* $\beta[\gamma\boldsymbol{w}] = \gamma\cdot\beta[\boldsymbol{w}]$.

Proof. By computation we get

$$\begin{aligned}&\mathrm{ar}_1[\mu\boldsymbol{w}, \boldsymbol{r}, \alpha](\boldsymbol{c}) = \mu\,\mathrm{ar}_1[\boldsymbol{w}, \boldsymbol{r}, \alpha](\boldsymbol{c}) \\ \iff &\sum_{1\le i<j\le n} \mu w_{ij}\left|c_i - c_j\right|^2 + \alpha\beta[\mu\boldsymbol{w}]\frac{(r_i+r_j)^4}{\left|c_i - c_j\right|^2} \\ &= \mu\sum_{1\le i<j\le n} w_{ij}\left|c_i - c_j\right|^2 + \alpha\beta[\boldsymbol{w}]\frac{(r_i+r_j)^4}{\left|c_i - c_j\right|^2} \\ \iff &\beta[\mu\boldsymbol{w}]\sum_{1\le i<j\le n}\frac{(r_i+r_j)^4}{\left|c_i - c_j\right|^2} = \mu\beta[\boldsymbol{w}]\sum_{1\le i<j\le n}\frac{(r_i+r_j)^4}{\left|c_i - c_j\right|^2} \\ \iff &\beta[\mu\boldsymbol{w}] = \mu\beta[\boldsymbol{w}].\end{aligned}$$

□

Corollary 7.5. *If* $\beta[\gamma\boldsymbol{w}] = \gamma\cdot\beta[\boldsymbol{w}]$, *then the attractor repeller model* (AR1) *is scaling invariant in size and in connection weight.*

There is still some freedom in the choice of β. Another desired property is stated in Requirement 7.6.

Requirement 7.6. *Consider the facility layout problem for only two circles. Then for* $\alpha = 1$ *the minimum is attained for touching circles, i.e. for* $|c_1 - c_2| = r_1 + r_2$. *For this case, the attractor repeller model should have the same optimal solution as the facility layout problem, i.e. it should have a minimum if and only if* $|c_1 - c_2| = r_1 + r_2$.

Lemma 7.7. *Consider the modified attractor repeller model* (AR1) *for the facility layout problem instance with two circles. The the minimum is attained if and only if* $|c_1 - c_2| = \sqrt[4]{\frac{\alpha\beta[\boldsymbol{w}]}{w_{12}}}(r_1 + r_2)$.

Proof. For this case, the objective ar_1 in (AR1) only depends on the circle center distance $d = |c_1 - c_2|$. We consider ar_1 as a function d. Then

$$\begin{aligned} \mathrm{ar}_1(d) &= w_{12} d^2 + \alpha\beta[\boldsymbol{w}] \cdot \frac{(r_1 + r_2)^4}{d^2} \\ \mathrm{ar}_1'(d) &= 2 w_{12} d - 2\alpha\beta[\boldsymbol{w}] \cdot \frac{(r_1 + r_2)^4}{d^3} \end{aligned}$$

and thus the minimum is attained for d with

$$\begin{aligned} \mathrm{ar}_1'(d) = 0 &\iff w_{12} d^4 = \alpha\beta[\boldsymbol{w}](r_1 + r_2)^4 \iff d^4 = \frac{\alpha\beta[\boldsymbol{w}]}{w_{12}}(r_1 + r_2)^4 \\ &\iff d = \sqrt[4]{\frac{\alpha\beta[\boldsymbol{w}]}{w_{12}}}(r_1 + r_2). \end{aligned}$$

□

Hence, to achieve Requirement 7.6, for two connected circles it must be $\beta[\boldsymbol{w}] = w_{12}$.

Examples of normalization factors $\beta = \beta[\boldsymbol{w}]$ satisfying the above conditions are, with $|\{w_{ij} > 0\}|$ denoting the number of non-zero w_{ij},

$$\beta = \max_{i<j} w_{ij}, \qquad \beta = \min_{i<j} w_{ij}, \qquad \beta = \operatorname*{avg}_{w_{ij}>0} w_{ij} = \frac{1}{|\{w_{ij} > 0\}|} \cdot \sum_{i<j} w_{ij}. \tag{7.9}$$

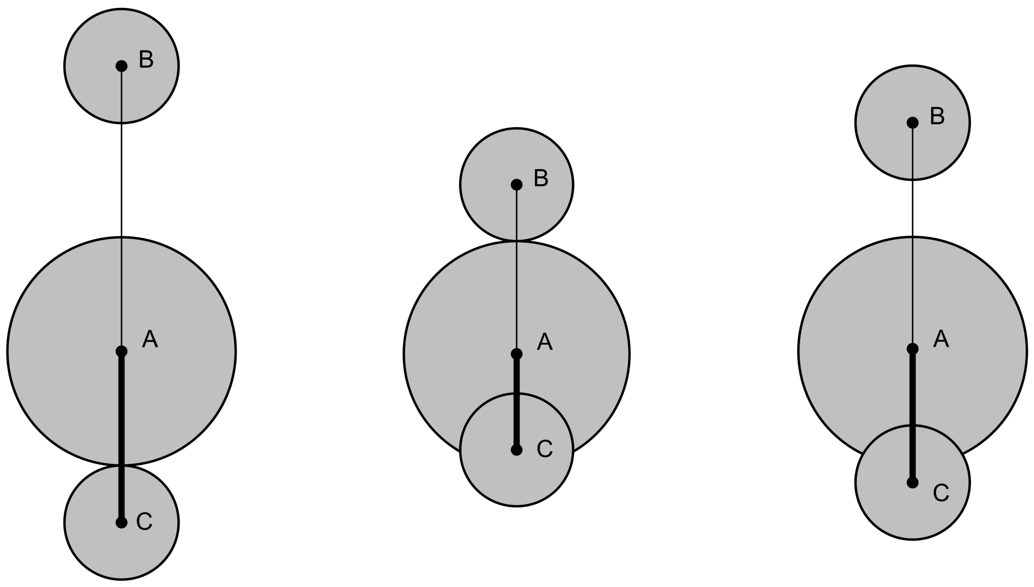
With these β, the attractor repeller model (AR1) is invariant of scaling in size and connection weight.

We now show the impact of different choices of β on the optimal solution. To get an intuition for the meaning of β, we only consider pairwise connections and ignore repelling forces of other circles. Hence, the following results in general do not hold exactly. This impact is visualized in Figure 7.2.

Figure 7.2a visualizes the choice $\beta[\boldsymbol{w}] = \max w_{ij}$ for $\alpha = 1$. The circles with the strongest connection do exactly not overlap. Circles with weaker connections have a large distance.

The choice $\beta[\boldsymbol{w}] = \min w_{ij}$ is shown in Figure 7.2b for $\alpha = 1$. The circles with the weakest connections do exactly not overlap, while circles with stronger connections strongly overlap. However, if not all circles are pairwise connected, it is $w_{ij} = 0$ for some i, j and thus $\beta = 0$.

Figure 7.2c shows $\beta[\boldsymbol{w}] = \operatorname{avg} w_{ij}$ for $\alpha = 1$. Circles with weak connections are pushed away from each other, while circles with strong connections overlap. However, the overlapping for strong connected circles is less than for choosing $\beta = \min w_{ij}$, and circles with weak connections have smaller distance than for $\beta = \max w_{ij}$. Additionally, this choice of β is more robust to changes of a single net weight.



(a) Choice $\beta = \max w_{ij}$. For $\alpha = 1$ the circles with the strongest connection do not overlap. Circles with weaker connections have a large distance.

(b) Choice $\beta = \min w_{ij}$. For $\alpha = 1$ the circles with the weakest connection do not overlap. Circles with strong connections overlap significantly.

(c) Choice $\beta = \operatorname{avg} w_{ij}$. For $\alpha = 1$ circles with stronger connections slightly overlap, while circles with weaker connections have a positive distance.

Figure 7.2: Optimal solutions of the circle repeller model (AR1) for different choices of β. The repelling force between A and C is omitted in this picture. The connection of the circles A and B is weak, the connection of A and C is strong.

It seems arbitrary to consider only positive weights w_{ij} in the last choice for β in (7.9). However, despite the global influence of the repelling function of two circles, repelling is a local property. One reason is that the repelling force is small for two circles with a large distance. Furthermore, assume i and j being two circles close together. Then the repelling forces on both circles i and j from all other circles are almost equal. Hence, by the repelling force of the other circles the circles i and j are neither pushed away nor together.

For electronic circuit design most weights w_{ij} are zero. In particular, the number of connections per circle does not increase for circuits with more circles. Instead the number of non-zero weights grows approximately linearly with the number of circles. Therefore, taking all weights into account would lead to small values of β for circuits with many circles, and therefore to strong overlapping for $\alpha = 1$.

7.3.4 Local Attractor Repeller Model

In the previous section and Figure 7.2, we have seen how to control the average overlap of the circles at the optimum by choosing β. However, it would be desirable to scale the net weights in Figure 7.2 such that A is touching both circles B and C. For this, we have to scale the repeller forces individually for each circle pair. We introduce scaling factors β_{ij}, $1 \leq i < j \leq n$ instead of β and get the attractor repeller model

$$\min_{\boldsymbol{c}} \quad \mathrm{ar}_2[\boldsymbol{w}, \boldsymbol{r}, \alpha](\boldsymbol{c}) = \sum_{1\leq i<j\leq n} w_{ij} \left|c_i - c_j\right|^2 + \alpha\beta_{ij}[\boldsymbol{w}] \cdot \frac{(r_i + r_j)^4}{\left|c_i - c_j\right|^2}. \tag{AR2}$$

To get optimality for both circles touching, in Figure 7.2 we have to choose $\beta_{ij} = w_{ij}$. However, for this choice unconnected circles do not repel each other. Therefore, a suitable choice is $\beta_{ij}[\boldsymbol{w}] = w_{ij} + \beta'[\boldsymbol{w}]$ with some small $\beta'[\boldsymbol{w}]$, e.g.

$$\beta_{ij}[\boldsymbol{w}] = w_{ij} + \frac{1}{n(n+1)} \sum_{1\leq k<l\leq n} w_{kl}.$$

Theorem 7.8. *The attractor repeller model* (AR2) *is scaling invariant in size and connection weight.*

Proof. It is $\beta_{ij}[\mu\boldsymbol{w}] = \mu\beta_{ij}$ and thus

$$\begin{aligned} \mathrm{ar}_2[\mu\boldsymbol{w}, \gamma\boldsymbol{r}, \alpha](\gamma\boldsymbol{c}) &= \sum_{1\leq i<j\leq n} \mu\gamma^2 w_{ij} \left|c_i - c_j\right|^2 + \alpha\beta_{ij}[\mu\boldsymbol{w}] \cdot \frac{\gamma^4(r_i + r_j)^4}{\gamma^2\left|c_i - c_j\right|^2} \\ &= \mu\gamma^2 \,\mathrm{ar}_2[\boldsymbol{w}, \boldsymbol{r}, \alpha](\boldsymbol{c}). \end{aligned}$$

□

7.3.5 Limited Range Attractor Repeller Model

The problem in all models considered so far is that we have to increase α to enforce the non-overlapping of all circles. In this process, some circles with weak connections are pushed far away from each other. One approach to avoid this was shown in Section 7.3.4 by introducing different weights β_{ij} depending on the connection strengths.

Another approach would be to stop the repelling force if two circles do not overlap. For this, we can multiply the repelling force by a step function

$$\Xi(x; x_0) = \begin{cases} 1 & \text{if } x \leq x_0, \\ 0 & \text{if } x > x_0. \end{cases}$$

However, the function $\Xi(x;x_0)$ is not continuous at $x=0$. A smooth approximation of $\Xi(x;x_0)$ is

$$\Sigma(x;\delta,x_0) = \frac{1}{1+\exp(\delta(x-x_0))}.$$

This function is approximately equal to 1 for values smaller than x_0 and approximately 0 for values greater than x_0. The steepness of the descent at x_0 is controlled by the parameter δ.

We assume some common $\beta[\boldsymbol{w}]$. Furthermore, the steepness $\delta[\boldsymbol{r}]$ depends on the circle radii. Then the limited range attractor repeller model is

$$\begin{aligned}\min_{\boldsymbol{c}} \quad \mathrm{ar}_3[\boldsymbol{w},\boldsymbol{r},\alpha] = &\sum_{1\le i<j\le n} w_{ij}\left|c_i-c_j\right|^2 \\ &+ \alpha\beta[\boldsymbol{w}]\cdot\left(\frac{(r_i+r_j)^4}{\left|c_i-c_j\right|^2}-1\right)\cdot\Sigma(\left|c_i-c_j\right|^2;\delta[\boldsymbol{r}];(r_i+r_j)^2).\end{aligned} \tag{AR3}$$

This model is also smooth.

Theorem 7.9. *If* $\beta[\mu\boldsymbol{w}]=\mu\beta[\boldsymbol{w}]$ *and* $\delta[\gamma\boldsymbol{r}]=\frac{1}{\gamma^2}\delta[\boldsymbol{r}]$, *then* (AR3) *is scaling invariant in connection weight and size.*

Proof. For such a $\delta[\boldsymbol{w}]$ it is

$$\begin{aligned}\Sigma(\left|\gamma c_i-\gamma c_j\right|^2;\delta[\gamma\boldsymbol{r}];(\gamma r_i+\gamma r_j)^2) &= \frac{1}{1+\exp(\delta[\gamma\boldsymbol{r}]\gamma^2(\left|c_i-c_j\right|^2-(r_i+r_j)^2))} \\ &= \Sigma(\left|c_i-c_j\right|^2;\delta[\boldsymbol{r}];(r_i+r_j)^2).\end{aligned}$$

Immediately we conclude

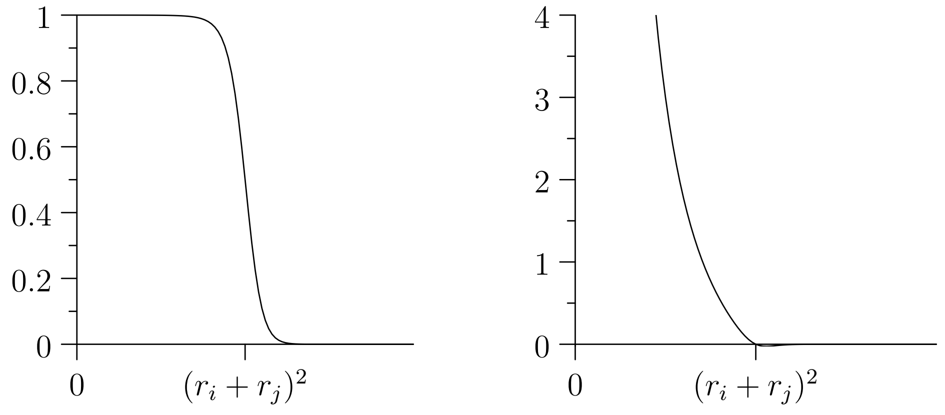
$$\mathrm{ar}_3[\mu\boldsymbol{w},\gamma\boldsymbol{r},\alpha](\gamma\boldsymbol{c}) = \mu\gamma^2\,\mathrm{ar}_3[\boldsymbol{w},\boldsymbol{r},\alpha](\boldsymbol{c}).$$

□

There still is some freedom in choosing $\delta[\boldsymbol{r}]$. An example is

$$\delta[\boldsymbol{r}] := \frac{1}{\mathrm{avg}_{i\in\mathcal{C}}\, r_i} = \frac{n}{\sum_{i=1}^n r_i}.$$

In contrast to the other attractor repeller models, the repelling force stops if two circles do not overlap. Therefore, circles with weak connections are not pushed to far away and we can choose larger values of $\beta[\boldsymbol{w}]$, i.e. $\beta[\boldsymbol{w}]=\max_{i<j} w_{ij}$.



(a) Step function for $\delta = 10$: $\Sigma(|c_i - c_j|^2 ; \delta, (r_i + r_j)^2)$.

(b) Limited range repeller function ar_3 of (AR3) for $\delta = 10$.

Figure 7.3: Step function and repeller function of (AR3) plotted over $|c_i - c_j|^2$.

7.3.6 Comparison of the Models

In this section we visualize the different tendencies of the models. A detailed numerical analysis is done in Section 7.7 for the circle placement problem.

Figure 7.4 visualizes placements for the circuit $V0093$ with all pin offsets set to zero. To generate a random starting point, we chose the circle centers uniformly of a square with area equal to the sum of all circle areas, see Section 3.6.1.2 for details. For this starting point we ran each attractor repeller model once with $\alpha = 10$. Even though not visualized in the picture, the connections between the large circles are stronger than most of the other connections.

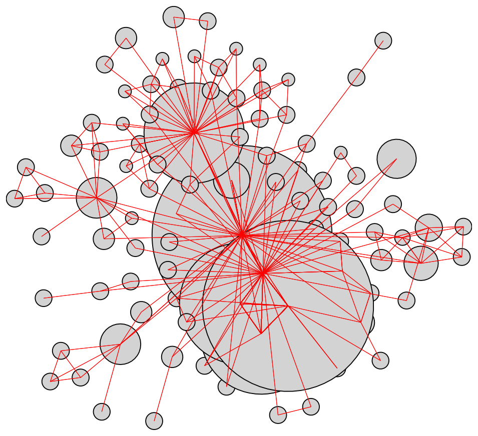
Figure 7.4a shows the solution for the standard attractor repeller model (AR0). The large circles strongly overlap while the small circles are widely spread. To get reasonable non-overlapping of the large circles, α must be significantly increased. However, this leads to very large gaps between the small circles.

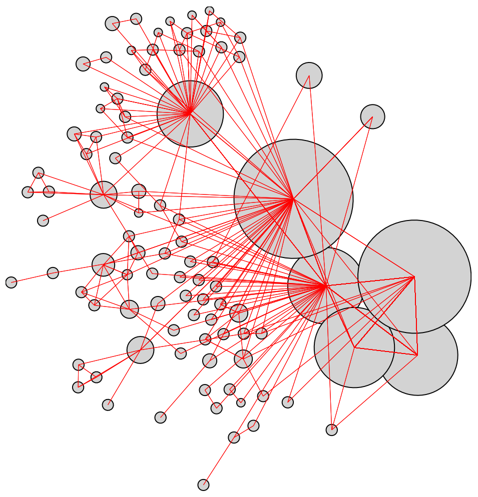
Figure 7.4b visualizes the result of the global invariant attractor repeller model (AR1) with $\beta = \operatorname{avg} w_{ij}$. There is little overlap of the large circles. The smaller circles are reasonable close together.

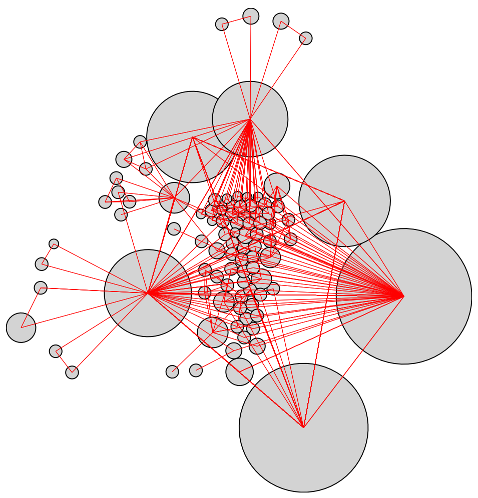
Figure 7.4c shows the local invariant model (AR2). Due to the strong connections of the large circles, their repelling force is also strong, hence they are pushed far away from each other. Similar, at the left side of the picture, there are strongly connected circles pushed away from each other by their strong repelling force. The weaker connected circles are reasonable distributed but still close together in the center of the placement. This model foils the concept of stronger connected circles being close to each other.

Figure 7.4d visualizes the limited range model (AR3) with $\beta = \max w_{ij}$. The circles are close together, as their repelling force is stopped for non-overlapping circles. However, this is also the disadvantage of the model. Due to the compact placement, the circles have no space to order themselves according to their connections, and the algorithm converges to a poor local minimum.

In the numerical analysis in Section 7.7 we show that the limited range model and the local invariant model are inferior to the standard model while the global invariant model is superior.


(a) Standard attractor repeller model (AR0).


(b) Global scaling invariant attractor repeller model (AR1) with $\beta = \operatorname{avg} w_{ij}$.


(c) Local attractor repeller model (AR2).

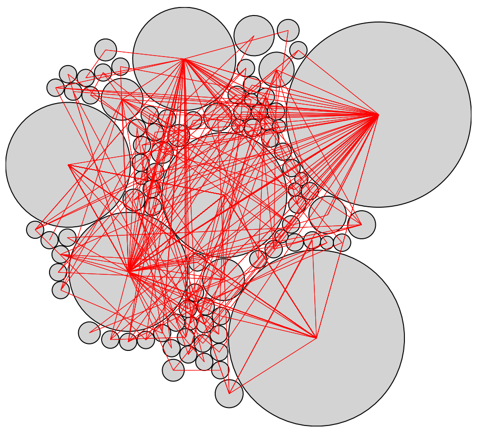

(d) Limited range attractor repeller model (AR3).

Figure 7.4: Comparison of the attractor repeller models. The circuit is $V0093$ with all pin offsets set to 0 and $\alpha = 10$.

7.4 Initial Solution Generation

In this section we consider the problem to generate an initial solution from where we start the non-linear program CPP stated in (7.3) on page 130. We briefly repeat this non-linear program here.

Assume the circle-to-pin matrix Φ being fixed. Then $CPP(Q, P, \boldsymbol{r})$ is the non-linear program

$$\begin{aligned} \min_{\boldsymbol{c},\boldsymbol{\varphi}} \quad & \mathrm{wl}[Q,P](\boldsymbol{c},\boldsymbol{\varphi}) = \boldsymbol{q}^H Q \boldsymbol{q} \qquad \text{with} \qquad \boldsymbol{q} = \Phi\boldsymbol{c} + P\Phi\boldsymbol{z} \\ \text{s. t.} \quad & (r_i + r_j)^2 - |c_i - c_j|^2 \leq 0, \end{aligned}$$

where $\boldsymbol{z} = \exp(\imath\boldsymbol{\varphi})$, P is the diagonal matrix of pin offsets, $\boldsymbol{r}$ the circle vector of radii and Q defines the wire length.

As this program is highly non-convex, the choice of an appropriate starting point is crucial for the final solution quality. The main ingredient of the initial solution generation is the application of the attractor repeller model analyzed and improved in Section 7.3.

7.4.1 Attractor Repeller Model for the Circle Placement Problem

In this section we replace the circle placement problem by an unconstrained problem based on the attractor repeller model. While the optimal solution of this new model might neither be feasible nor optimal to CPP, it distributes the circles according to their connections and gives a good starting point to solve CPP with a local non-linear solver.

Referring to the attractor repeller models for facility layout problems in Section 7.3, we leave the repeller part unchanged but replace the attractor part by the wire length.

Scaling the net weights by a factor $\mu \geq 0$ leads to the problem $CPP(\mu Q, P, \boldsymbol{r})$. Scaling the size by a factor γ gives the scaled problem $CPP(Q, \gamma P, \gamma r)$. Similar to the facility layout problem, CPP is scaling invariant with respect to size and to connection weight.

The attractor repeller models of Section 7.3 depend on circle center connections $\boldsymbol{w}$ that do not exist in the circle placement problem. However, a circle placement problem with all pins in the circle centers is a facility layout problem. We refer to Definition 7.1 of the facility layout problem as non-linear program $FLP(\boldsymbol{w}, \boldsymbol{r})$.

Lemma 7.10. *The circle placement problem $CPP(Q, 0, \boldsymbol{r})$ is equivalent to the facility layout problem $FLP(\boldsymbol{w}, \boldsymbol{r})$ with*

$$w_{ij} = -a_{ij} \qquad \textit{for} \qquad A = (a_{ij}) = \Phi^T Q \Phi. \tag{7.10}$$

Proof. As the constraints are the same, we only have to prove the equality of the objectives. For $CPP(Q, 0, \boldsymbol{r})$ it is

$$\operatorname{wl}[Q, 0](\boldsymbol{c}, \boldsymbol{\varphi}) = \boldsymbol{c}^H \Phi^T Q \Phi \boldsymbol{c} = \boldsymbol{c}^H A \boldsymbol{c} = \sum_{i<j} w_{ij} \left| c_i - c_j \right|^2 ,$$

where the last identity follows immediately from Lemma 2.45. □

In the following we use $\boldsymbol{w}$ as defined in (7.10). In particular, a scaling of Q is equivalent to a scaling of $\boldsymbol{w}$ by the same factor.

In Section 7.3 for $FLP(\boldsymbol{w}, \boldsymbol{r})$ we defined four attractor repeller models as unconstrained minimization of

$$\min_{\boldsymbol{c}} \quad \operatorname{ar}_t[\boldsymbol{w}, \boldsymbol{r}, \alpha](\boldsymbol{c}) = \sum_{i<j} w_{ij} \left| c_i - c_j \right|^2 + \alpha \operatorname{rep}_t[\boldsymbol{w}, \boldsymbol{r}](\boldsymbol{c}), \quad t = 0, \dots, 3.$$

The corresponding attractor repeller models for $CPP(Q, P, \boldsymbol{r})$ are created by replacing the attractor term by the wire length

$$\min_{\boldsymbol{c}, \boldsymbol{\varphi}} \quad \operatorname{ar}'_t[Q, P, \boldsymbol{r}, \alpha](\boldsymbol{c}, \boldsymbol{\varphi}) = \operatorname{wl}[Q, P](\boldsymbol{c}, \boldsymbol{\varphi}) + \alpha \operatorname{rep}_t[\boldsymbol{w}, \boldsymbol{r}](\boldsymbol{c}), \quad t = 0, \dots, 3.$$

Similar to the attractor repeller model for the facility layout problem, we define the invariance of this parametric family of functions.

Definition 7.11. *A parametrized family of functions* $\operatorname{ar}'[Q, P, \boldsymbol{r}, \alpha]$ *is scaling invariant in connection weight and size if and only if for* $\mu, \gamma \geq 0$ *and each* $\boldsymbol{c} \in \mathbb{C}^n$, $\boldsymbol{\varphi} \in \mathbb{R}^n$

$$\operatorname{ar}'[\mu Q, \gamma P, \gamma \boldsymbol{r}, \alpha](\gamma \boldsymbol{c}, \boldsymbol{\varphi}) = \mu \gamma^2 \operatorname{ar}'[Q, P, \boldsymbol{r}, \alpha](\boldsymbol{c}, \boldsymbol{\varphi}).$$

Theorem 7.12. *The attractor repeller model for the circle placement problem is scaling invariant if and only if the corresponding model for the facility layout problem is scaling invariant.*

Proof. With $\mu, \gamma \geq 0$ we have

$$\begin{aligned} \operatorname{wl}[\mu Q, \gamma P](\gamma \boldsymbol{c}, \boldsymbol{\varphi}) &= (\Phi \gamma \boldsymbol{c} + \gamma P \Phi \boldsymbol{z})^H \mu Q (\Phi \gamma \boldsymbol{c} + P \gamma \Phi \boldsymbol{z}) \\ &= \mu \gamma^2 (\Phi \boldsymbol{c} + P \Phi \boldsymbol{z})^H Q (\Phi \boldsymbol{c} + P \Phi \boldsymbol{z}) \\ &= \mu \gamma^2 \operatorname{wl}[\mu Q, \gamma P](\boldsymbol{c}, \boldsymbol{\varphi}). \end{aligned}$$

We conclude

$$\begin{aligned}
&\operatorname{ar}'_t[\mu Q, \gamma P, \gamma \boldsymbol{r}, \alpha](\gamma \boldsymbol{c}, \boldsymbol{\varphi}) = \mu\gamma^2 \operatorname{ar}'_t[Q, P, \boldsymbol{r}, \alpha](\boldsymbol{c}, \boldsymbol{\varphi}) \\
\iff &\operatorname{wl}[\mu Q, \gamma P](\gamma \boldsymbol{c}, \boldsymbol{\varphi}) + \alpha \operatorname{rep}_t[\mu \boldsymbol{w}, \gamma \boldsymbol{r}](\gamma \boldsymbol{c}) \\
&\quad = \mu\gamma^2(\operatorname{wl}[Q, P](\boldsymbol{c}, \boldsymbol{\varphi}) + \alpha \operatorname{rep}_t[\boldsymbol{w}, \boldsymbol{r}](\boldsymbol{c})) \\
\iff &\operatorname{rep}_t[\mu \boldsymbol{w}, \gamma \boldsymbol{r}](\gamma \boldsymbol{c}) = \mu\gamma^2 \operatorname{rep}_t[\boldsymbol{w}, \boldsymbol{r}](\boldsymbol{c}) \\
\iff &\sum_{i<j} \mu w_{ij} \left|\gamma c_i - \gamma c_j\right|^2 + \alpha \operatorname{rep}_t[\mu \boldsymbol{w}, \gamma \boldsymbol{r}](\gamma \boldsymbol{c}) \\
&\quad = \mu\gamma^2 \left(\sum_{i<j} w_{ij} \left|c_i - c_j\right|^2 + \alpha \operatorname{rep}_t[\boldsymbol{w}, \boldsymbol{r}](\boldsymbol{c}) \right) \\
\iff &\operatorname{ar}_t[\mu \boldsymbol{w}, \gamma \boldsymbol{r}, \alpha](\gamma \boldsymbol{c}) = \mu\gamma^2 \operatorname{ar}_t[\boldsymbol{w}, \boldsymbol{r}](\boldsymbol{c}).
\end{aligned}$$

□

Applying the models of Section 7.3 to the circle placement problem, we get four attractor repeller models for the circle placement problem:

- The standard attractor repeller model ($CPP - AR0$) it neither scaling invariant in net weight nor in size.
- The global attractor repeller model ($CPP - AR1$) it scaling invariant.
- The local attractor repeller model ($CPP - AR2$) it scaling invariant.
- The limited range attractor repeller model ($CPP - AR3$) it scaling invariant.

7.4.2 Feasibility Stretching

Some optimization algorithms, e.g. interior point methods, require a feasible starting point. In principle, we could increase the repeller factor α until the circles do not overlap anymore, i.e. the solution of the attractor repeller model is feasible to CPP. But increasing the repeller factor leads to numerical problems and eventually only blows up the placement without essentially changing the ordering of the circles.

In this section we show how this increasing of the repeller factor can be avoided. Instead we can stop the attractor repeller problem when the overlap is sufficiently small such that it is unlikely that the ordering of the circles changes in further steps. Then we stretch this solution to obtain feasibility for CPP.

Lemma 7.13. *Let $(\boldsymbol{c}, \boldsymbol{\varphi}) \in \mathbb{C}^n \times \mathbb{R}^n$ and $c_i \neq c_j$ for $i \neq j$. Let*

$$\lambda = \max_{1 \le i < j \le n} \frac{r_i + r_j}{|c_i - c_j|}.$$

Denote $(\boldsymbol{c}', \boldsymbol{\varphi}) = (\gamma \boldsymbol{c}, \boldsymbol{\varphi})$ for $\gamma > 0$. Then:

- *For $\gamma < \lambda$ the solution $(\boldsymbol{c}', \boldsymbol{\varphi})$ is infeasible to CPP.*
- *For $\gamma = \lambda$ the solution $(\boldsymbol{c}', \boldsymbol{\varphi})$ is exactly feasible to CPP, i.e. it is feasible and there are active constrains.*
- *For $\gamma > \lambda$ the solution $(\boldsymbol{c}', \boldsymbol{\varphi})$ is strictly feasible to CPP, i.e. it is feasible and all constraints are inactive.*

Proof. Let $\mathcal{M} = \left\{(i,j) : \lambda = \frac{r_i + r_j}{|c_i - c_j|}\right\}$.

Case 1: $\gamma < \lambda$. Then for $(i,j) \in \mathcal{M}$:

$$\left|c_i' - c_j'\right| = \gamma \left|c_i - c_j\right| < \frac{r_i + r_j}{|c_i - c_j|} \cdot |c_i - c_j| = r_i + r_j.$$

Hence, the circles i and j overlap, i.e. the solution is infeasible.

Case 2: $\gamma \geq \lambda$. Then for all i, j it is

$$\left|c_i' - c_j'\right| = \gamma \left|c_i - c_j\right| \geq \frac{r_i + r_j}{|c_i - c_j|} \cdot |c_i - c_j| = r_i + r_j.$$

where equality holds if and only if $(i,j) \in \mathcal{M}$ and $\gamma = \lambda$.

□

For a solution where no circles have identical centers, we can scale the solution such that the circles do not overlap and the scaled solution is feasible. Furthermore, if the solution is feasible but the distance of the circles is larger than desired, we can scale down the solution. Finally, by the choice of γ we can control how far the solution should be in the interior of the feasible region, i.e. how far the constraints should be away from being active.

7.5 Finding Local Minima

In this section we consider the problem to solve the non-linear program CPP to a local optimum by a standard non-linear solver. Heuristics to overcome poor local optima are addressed in Section 7.6. Some of the results are special cases of statements shown in Chapter 8 for a more general problem. However, to keep the chapters self contained, we proof them for the circle placement problem here.

In (7.3) we defined the non-linear program CPP for the circle placement problem as

$$\begin{array}{ll} \min & \mathrm{wl}(\boldsymbol{c}, \boldsymbol{\varphi}) \\ \text{s. t.} & g_{ij}(\boldsymbol{c}) = (r_i + r_j)^2 - |c_i - c_j|^2 \leq 0, \quad 1 \leq i < j \leq n. \end{array}$$

Despite this problem or related problems have often been considered in literature, we did not find any theorems about constraint qualifications for this problem.

Theorem 7.14 (MFCQ is satisfied)**.** *Let radii be positive, i.e. for all circles i it is $r_i > 0$. Then at each feasible point $(\boldsymbol{c}, \boldsymbol{\varphi})$ of CPP the Mangasarian-Fromovitz constraint qualification is satisfied.*

Proof. As there is no constraint on the rotations $\boldsymbol{\varphi}$, we ignore them in this proof and consider the g_{ij} as functions only depending on the circle centers $\boldsymbol{c}$. For the derivatives of the constraints it is

$$\frac{\partial g_{ij}}{\partial \overline{c_i}} = c_j - c_i, \qquad \frac{\partial g_{ij}}{\partial \overline{c_j}} = c_i - c_j, \qquad \frac{\partial g_{ij}}{\partial \overline{c_k}} = 0 \quad \text{for } k \notin \{i, j\}.$$

To show MFCQ for each feasible $\boldsymbol{c}$, we have to construct a vector $\boldsymbol{y} \in \mathbb{C}^n$ such that $\langle \boldsymbol{y}, \nabla_{\overline{c}} g_{ij}(\boldsymbol{c}) \rangle_{\mathbb{R}} < 0$ for all active constraints g_{ij} in $\boldsymbol{c}$.

Set $\boldsymbol{y} := \boldsymbol{c}$. Let g_{ij} be active in $\boldsymbol{c}$, then it is $|c_i - c_j|^2 = (r_i + r_j)^2$. So it follows

$$\begin{aligned} \langle \boldsymbol{y}, \nabla_{\overline{c}} g_{ij}(\boldsymbol{c}) \rangle_{\mathbb{R}} &= \Re \left(c_i (\overline{c_j - c_i}) + c_j (\overline{c_i - c_j}) \right) \\ &= -\Re \left((c_i - c_j)(\overline{c_i - c_j}) \right) \\ &= - |c_i - c_j|^2 = -(r_i + r_j)^2 < 0. \end{aligned}$$

Consequential MFCQ is satisfied in each feasible point. □

The Mangasarian-Fromovitz constraint qualification ensures that each minimizer is a KKT point. Furthermore, it is an important necessary and sufficient condition for many solvers to be convergent to KKT points, see e.g. [Haeser 2010].

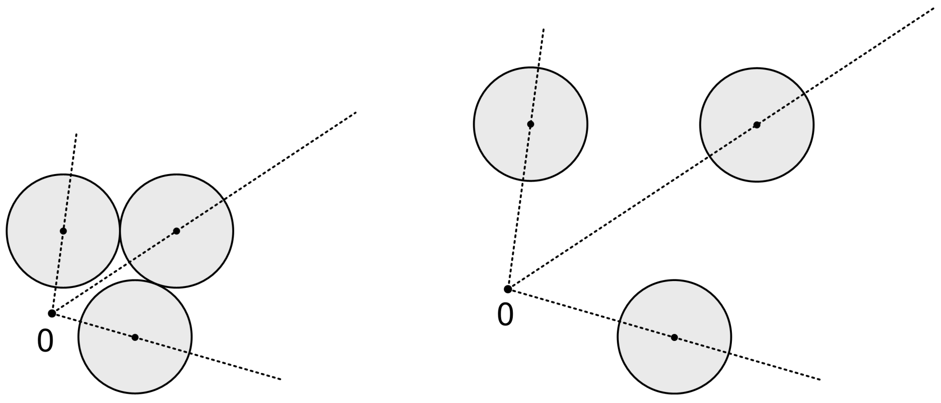
This direction $\boldsymbol{y}$ has the geometric interpretation shown in Figure 7.5. The rotations $\boldsymbol{\varphi}$ are kept fixed (as we ignored them within the proof). The circle centers are stretched away from the origin. As the radii are fixed, there arises a gap between previously touching circles. So no more circles are touching, thus all active constraints become inactive and MFCQ is satisfied.

We might also ask, if stronger constraint qualification such as the linear independence constraint qualification LICQ holds for this model. However, this is not the case as we are going to show in the remaining part of this section. For this, we introduce some theory of planar graphs, for a more detailed survey see [Jungnickel 2005]. Planar graphs are related to circle packings.

Definition 7.15 (Planar Graph)**.** *A graph is planar, if it can be drawn in the plane such that no two edges overlap.*

Definition 7.16 (Maximal Planar Graph)**.** *A planar graph is maximal planar if adding any edge to the graph destroys the planarity.*

Theorem 7.17. *A maximal planar graph with n nodes has $3n - 6$ edges.*



(a) Before stretching. (b) After stretching.

Figure 7.5: Geometric meaning of the $\boldsymbol{y}$ vector. The rotations remain unchanged, the circle centers are stretched away from each other.

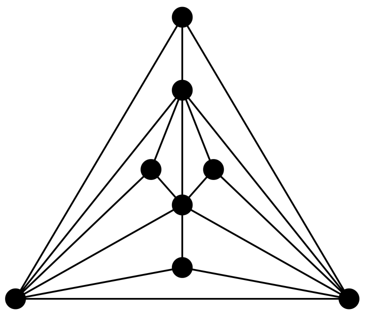


Figure 7.6: The maximal planar graph with 8 nodes.

Definition 7.18 (Circle Packing). *A circle packing is a placement of circles* $1, \dots, n$ *with radius* r_i *in the plane, such that the circles do not overlap. A circle packing is defined by the center* c_i *of each circle* i.

Definition 7.19 (Intersection Graph). *The intersection graph of a circle packing is a graph* $G(V, E)$*, where the set of nodes are the circles* $V = \{1, \dots, n\}$ *and for each pair of touching circles there is an edge, i.e.* $E = \{\{i, j\} : |c_i - c_j| = r_i + r_j\}$. *The intersection graph of a circle packing is simple and planar.*

Theorem 7.20 (Koebe-Andreev-Thurston Theorem, [Koebe 1936]). *Let* $G(V, E)$ *be a simple planar graph. Then there is a circle packing such that the intersection graph of the packing is isomorphic to* G.

Based on this we can now show the following theorem.

Theorem 7.21. *There are feasible solutions to* CPP *such that LICQ is violated.*

Proof. Let G be a maximal planar graph with $n \geq 7$ nodes and thus with $3n - 6$ edges (e.g. the graph shown in Figure 7.6).

By Theorem 7.20 there is a circle packing with n circles, which intersection graph is isomorphic to G. Let their radii be $\boldsymbol{r}$ and their centers be $\boldsymbol{c}$. With arbitrary rotations $\boldsymbol{\varphi}$ this packing is a feasible solution to CPP. Furthermore, for this solution there are $3n - 6$ active constraints.

However, the gradients of these constrains exist in the $2n$-dimensional subspace defined by the entries of the circle centers. As for $n \geq 7$ it is $3n - 6 > 2n$, the number of active constraint gradients exceeds the dimension, so these gradients cannot be linear independent. $\square$

7.6 Overcoming Local Optima

Due to the highly non-convex structure, the algorithm is likely to get trapped in a local, non-global minimum. In this section we consider heuristics to overcome local minima.

7.6.1 Equal Circle Swap Local Search (ECSLS)

Equal circle local search is a simple strategy to overcome local optima. It is mainly based on two observations:

1. For circle placement problems based on electronic circuit design instances, there are many circles with equal radii.

2. The circle rotation problem analyzed in Chapter 3 can be solved much more efficiently than the circle placement problem.

In Chapter 3 we have shown how to find a good local optimal solution of the rotation problem very quickly. In particular, feasible or almost feasible solutions to the circle placement problem are very different from the harder problem instances of the circle rotation problem with very small area factor. For these almost feasible instances, solving the rotation problem compared to solving the placement problem is faster by several orders of magnitudes. Thus, we can solve many circle rotation problems within a circle placement problem.

The proposed local search algorithm iterates over all touching pairs (i, j) of circles with equal radii. For each pair the circle centers of i and j are interchanged and the circle rotation problem is solved to a local minimum. If the wire length of the created solution is better than the wire length of the previous solution, the new solution is accepted.

This progress is repeatedly done for all circle pairs, unless there is no improvement in the last pass through all circles. It is shown in Algorithm 4.

Algorithm 4: Equal circle swap local search (ECSLS).

```
Data: starting point $X = (\boldsymbol{c}, \boldsymbol{\varphi})$
1  enumerate the pairs of touching circles by $(i_1, j_1), \dots, (i_N, j_N)$;
2  $k \leftarrow 1$;
3  $nrOfFails \leftarrow 0$;
4  while $nrOfFails < N$ do
5      $X' =$ interchange $i_k$ and $j_k$;
6      optimize rotations of $X'$;
7      if $\text{wl}(X') < \text{wl}(X)$ then
8          $X \leftarrow X'$;
9          $nrOfFails \leftarrow 0$;
10     else
11         $nrOfFails \leftarrow nrOfFails + 1$;
12     $k \leftarrow (k \mod N) + 1$;
13 return $X$;
```

Note that this algorithm is guaranteed to terminate, as there are only finitely many placements and thus the algorithm eventually reaches a placement from where it cannot improve by a circle swap. However, in practice we introduce a minimal improvement threshold ε and replace the condition in line 7 by $\text{wl}(X') \leq \text{wl}(X) - \varepsilon$.

7.6.2 Monotonic Basin Hopping (MBH)

Monotonic basin hopping was applied to circle packing problems in [Addis; Locatelli; Schoen 2008] and [Grosso et al. 2010]. It can be seen as a mixture of a multiple shoot approach and an iterative local search method. The simple algorithmic idea is shown in Algorithm 5.

Algorithm 5: Monotonic basin hopping (MBH).

```
Data: starting point $X$
Data: maximal number of tries $maxTries$
1  $X \leftarrow LO(X)$;
2  $k \leftarrow 0$;
3  while $k < maxTries$ do
4      $X' \leftarrow perturb(X)$;
5      $X'' \leftarrow LS(X')$;
6      if $f(X'') < f(X)$ then
7          $k \leftarrow 0$;
8          $X \leftarrow X''$;
9      else
10         $k \leftarrow k + 1$;
11 return $X$;
```

For the circle placement problem, monotonic basin hopping it is an extension of the local search method of Section 7.6.1 to circles with different radii. We will now describe the details of Algorithm 5 for the circle placement problem.

Local Optimization *LO* The local optimization *LO* to find a local optimum from a given starting point is used in the initial step in line 1 and repeatedly in line 5. In this step, we solve *CPP* as stated in (7.3) by any local non-linear solver.

Perturbation The perturbation used in line 4 is a crucial ingredient of the algorithm. It should be chosen in such a way that the algorithm can overcome local optima but the essential structure of the solution remains. The two strategies described in Section 7.2.1 for circle packing problems do not consider connections between circles and thus are not well suited for the circle placement problem.

We use the following idea: Let $\mathcal{T} = \{(i,j) : i < j, |c_i - c_j| = r_i + r_j\}$ be the pairs of touching circles. Then for a solution $(\boldsymbol{c}, \boldsymbol{\varphi})$ and two touching circles $(i,j) \in \mathcal{T}$ the solution with swapped circles i and j is $(\boldsymbol{c}', \boldsymbol{\varphi})$ with

$$c_i' = \frac{(r_i - r_j)c_i + 2r_jc_j}{r_i + r_j}, \qquad c_j' = \frac{(r_j - r_i)c_j + 2r_ic_i}{r_i + r_j}, \qquad c_k' = c_k \quad \text{for } k \notin \{i,j\}.$$

The geometric meaning of this swapping is shown in Figure 7.7. The rotations are kept. The circle centers are interchanged such that, seen along the line through the circle centers, the circles are swapped, i.e. their union occupies the same interval on the line but the circles have changed sides. This is equal to interchanging the circle centers (as in [Addis; Locatelli; Schoen 2008] or Section 7.1) if and only if the circles have the same radius.

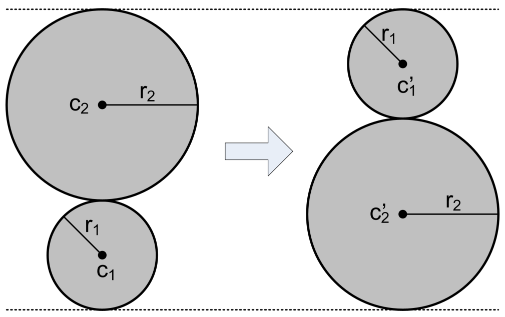


Figure 7.7: Geometric meaning of swapping two circles.

The perturbation of P_{ij} for $(i, j) \in \mathcal{T}$ now is defined by the following two steps:

1. Create $(\boldsymbol{c}', \boldsymbol{\varphi})$ by swapping circles i and j.
2. Solve the circle rotation problem locally, starting from $\boldsymbol{\varphi}$ for fixed centers $\boldsymbol{c}'$.

As described in Section 7.1, compared to the rotation problem the placement problem is much harder to solve and the running time of the algorithm is several orders of magnitudes slower. Thus, creating a perturbation P_{ij} and evaluating the objective for P_{ij} is very fast compared to the local optimization step including the feasibility restoration. Therefore, in the perturbation step in line 4 all perturbation $P_{ij}, (i, j) \in \mathcal{T}$ are performed and the best perturbation is chosen. If the following local optimization is not successful, this perturbation is marked as failed and in the next iteration the best remaining perturbation is chosen.

Algorithm 6 shows this monotonic basin hopping for the circle placement problem.

7.6.3 Combination of Monotonic Basin Hopping and Equal Circle Swap Local Search

In practice this monotonic basin hopping algorithm can be combined with the equal circle swap local search of Section 7.6.1. Then all perturbations P_{ij} of circles i, j with circles with equal radius are considered first and immediately accepted, if they improve the wire length. The monotonic basin hopping algorithm is only performed for circles with different radii. This combination of monotonic basin hopping and equal circle swap local search is shown in Algorithm 7.

Algorithm 6: Monotonic basin hopping for the circle placement problem.

```
Data: local optimal starting point $X = (\boldsymbol{c}, \boldsymbol{\varphi})$
Data: maximal number of tries $maxTries$
1  $k \leftarrow 0$;
2  compute $\mathcal{T}$;
3  order $P_{ij}$ for $(i,j) \in \mathcal{T}$ according to their wire length as $P^1, P^2, \ldots$;
4  while $k < maxTries$ do
5      $X' \leftarrow P^k(X)$;
6      $X'' \leftarrow LO(X')$;
7      if $\mathrm{wl}(X'') < \mathrm{wl}(X)$ then
8          $X \leftarrow X''$;
9          goto line 1;
10     else
11         $k \leftarrow k + 1$;
12 return $X$;
```

Algorithm 7: Combination of MBH and ECSLS for the circle placement problem.

```
Data: starting point $X = (\boldsymbol{c}, \boldsymbol{\varphi})$
Data: maximal number of MBH tries
1  $X \leftarrow$ local optimum starting from $X$;
2  loop
3      $X \leftarrow ECSLS(X)$ ;                    /* equal circle swap local search */
4      if no improvement in last $ECSLS$ and last $MBH$ then return $X$;
5      $X \leftarrow MBH(X)$ ;                          /* monotonic basin hopping */
6      if no improvement in last $ECSLS$ and last $MBH$ then return $X$;
```

7.7 Numerical Results

In this section we evaluate the algorithms stated in this chapter numerically. The complete algorithm to evaluate is shown in Figure 7.8. It consists of many individual analyzable and tunable parts. It does not make sense to consider all possible combinations. Therefore, we consider different aspects individually and then choose the best setting for the analysis of other parts of the algorithm.

In Section 7.7.1 we compare different attractor repeller variants stated in Section 7.4.1 for the circle placement problem. In Section 7.7.1.1 for each attractor repeller model we choose the best repeller weight α and decide if a feasibility stretch is applied. Based on this choice, in Section 7.7.1.2 we compare the different models. However, in Section 7.7.1 we do not consider any post processing steps as MBH or ECSLS, instead we use the result of CPP for comparison. The best attractor repeller model is analyzed in more detail in Section 7.7.1.3.

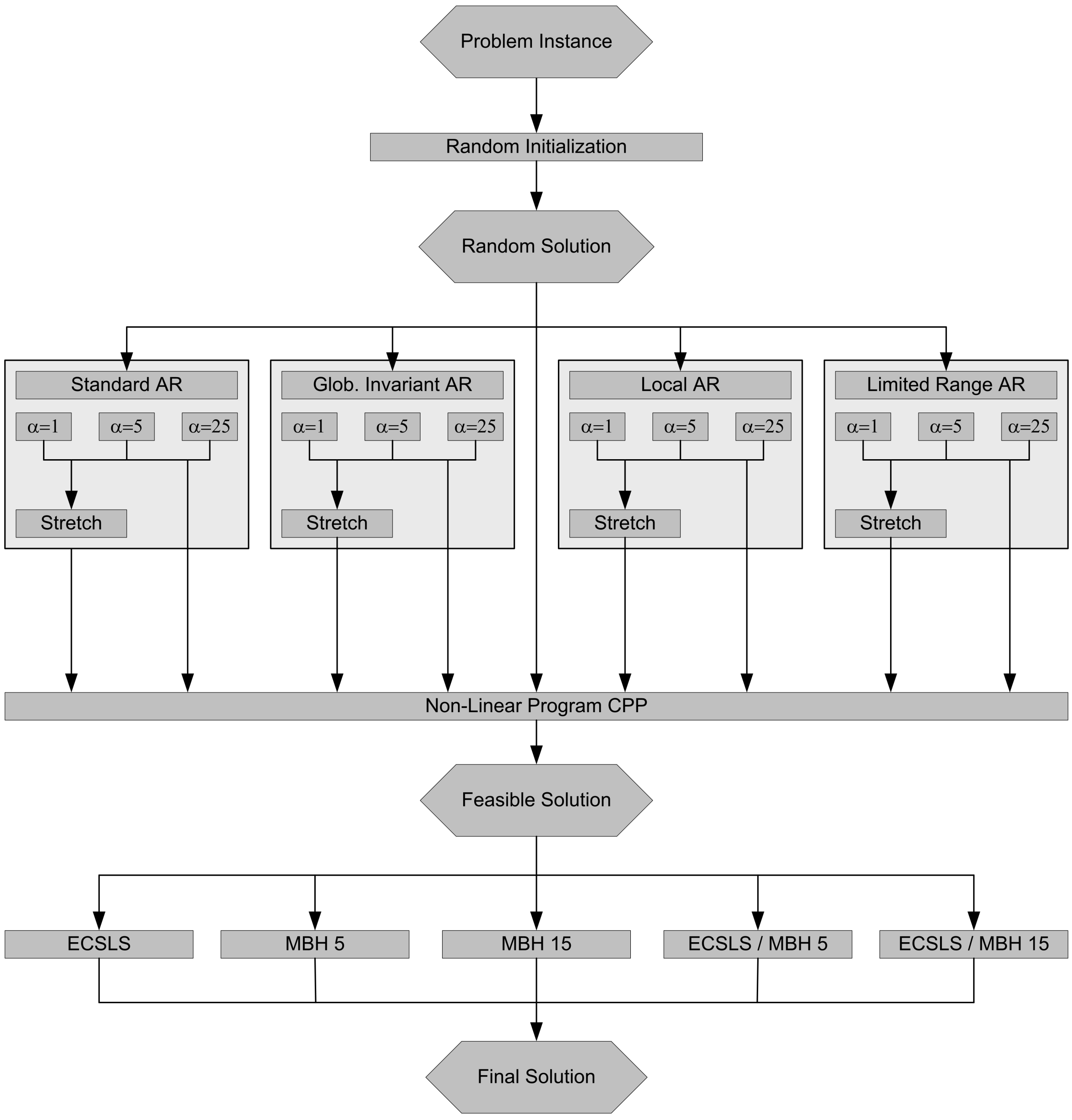


Figure 7.8: Algorithm flow of our circle placement algorithm.

In Section 7.7.2 we analyze different settings for MBH and ECSLS. We use the best attractor repeller model found in Section 7.7.1 and CPP to generate an initial solution for the post processing steps. To this solution we apply the local search stated in Section 7.6.1, monotonic basin hopping stated in Section 7.6.2 or their combination.

7.7.1 Attractor Repeller Models

In Section 7.3 we stated four attractor repeller models: the standard AR, the global invariant AR, the local AR and the limited range AR. Each of these models can be parametrized by the repeller weight α. Additionally, a feasibility stretching as described in Section 7.4.2 can be done after each of these models.

In the following, we present the numerical evaluation of the attractor repeller models for different settings. We applied them to the problem instances described in Section 2.8 where each component is represented as its circumcircle. To run a sufficient number of test instances, we only considered circuits with 323 or less circles.

For the global invariant attractor repeller model ($CPP - AR1$) we chose

$$\beta[\boldsymbol{w}] = \operatorname*{avg}_{w_{ij}>0} w_{ij} = \frac{1}{|\{w_{ij} > 0\}|} \cdot \sum_{i<j} w_{ij}.$$

For the local attractor repeller model ($CPP - AR2$) we chose

$$\beta_{ij}[\boldsymbol{w}] = w_{ij} + \frac{1}{n(n+1)} \sum_{k<l} w_{kl}.$$

For the limited range attractor repeller model ($CPP - AR3$) we used $\beta = \max w_{ij}$.

For each of the circuits, 100 random initial solutions were generated by placing the circle centers uniformly distributed in a square with area equal to the sum of the circle areas (i.e. we choose $a_f = 1$ in the procedure described in Section 3.6.1.2).

For these initial solutions, we ran all attractor repeller algorithms with $\alpha \in \{1, 5, 25\}$ and with and without feasibility stretching, denoted by S for stretching and N for no stretching. As the aim of the attractor repeller models is to generate a good initial solution for the non-linear program CPP, we applied CPP to the solutions of the attractor repeller models. Additionally we also immediately applied CPP to the random initial solutions.

In total we have six configurations for each of the four attractor repeller models. Considering also the immediate application of the non-linear program CPP to the random initial solution, there are twenty-five algorithm configurations. As it is confusing to immediately compare them, we first compare the different configurations for each attractor repeller model and eventually compare the best of these configurations with each other.

We ran Ipopt with $tol = 10^{-3}$ and $acceptable\_tol = 5 \cdot 10^{-3}$ and $max\_iter = 30000$ (which was never reached). Ipopt is an interior point solver with feasibility restoration techniques. However, it might have problems to converge if the starting point is very infeasible. Even for feasible starting points, convergence problems occurred. Therefore, we say a run converged, if both the attractor repeller run and the run the following run of CPP converged, this means they stop with the Ipopt result `Solve_Succeeded` or `Solved_To_Acceptable_Level`. If the run did not converge, usually the run of CPP terminated with `Infeasible_Problem_Detected` or rarely `Restoration_Failed`. In less than one percent of the runs the program crashed with insufficient memory when run from very poor starting points. As for these random initial solutions some attractor repeller models could not be evaluated, these runs were skipped.

Note that running time and convergence problems are likely to be solver dependent and might be different with an SQP-solver, while the objective is likely to be similar for different solvers.

7.7.1.1 Comparison of Different Settings for each AR-Model

For each attractor repeller model we present figures with the Dill instances. These figures show the number of circles on the x-axis. On the y-axis one of the following values is shown:

Median of the CPU-time in seconds: This time is the total running time of the attractor repeller run and the following CPP-run (random initialization and stretching are also taken into account but have no practical influence). If a run has not converged, the CPU time of this run is set to infinity, as we are not interested in fast but non-converged runs. As for some configurations more than half of the runs did not converge, the median CPU-time might be infinity. In these cases we take the maximum of the finite values. However, these points are marked with a circle while all other points are marked with a full disc.

Objective: For all problem instances we take the best solution found by any of the algorithms (note that, in contrast to the circle rotation problem, this is not likely to be the global optimum). In the figure the ratio of the median of the objectives divided by the best objective is displayed. Similar to the CPU-time, this value is set to infinity, if the run has not converged. If the median is infinite, the maximal finite value is taken and this point is marked by a circle.

Ratio of fails: As Ipopt has problems to converge for some configurations, this figure shows number of non-convergent runs divided by the total number of runs.

We do not include Egrain, Mekas, SapKit and Versiplektor into these figures, as each of these circuits has different characteristics. Instead the numerical results for these instances are presented in a tables. However, as these instances are small, all algorithms work reasonable well. So, in our analysis we focus in the Dill-instances.

Standard AR Figure 7.9 and Table 7.1 on page 158 show the standard attractor repeller model. For larger instances the attractor repeller runs without feasibility stretching rarely converge. This is due to the strong overlapping of large circles even for larger repeller factors α. Comparing the models with stretching, there is only a small difference between them. It looks like $\alpha = 1$ yields the best objective but has convergence problems for larger problem instances, while $\alpha = 5$ and $\alpha = 25$ do not show these difficulties. We decided to use $\alpha = 5$ with stretching as optimal setting.

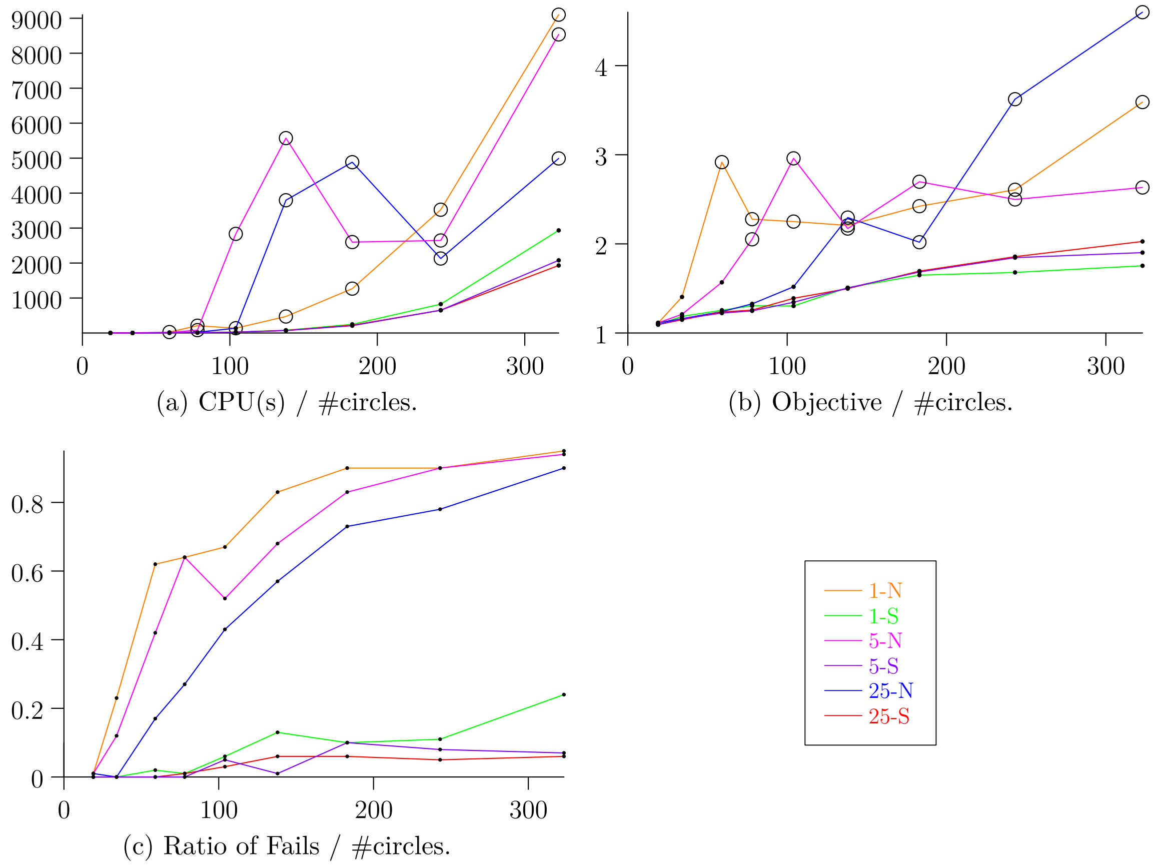
Global Invariant AR Figure 7.10 and Table 7.2 on page 159 display the global invariant attractor repeller model. For this model the setting with $\alpha = 1$ and without scaling has convergence problems for increasing number of circles. However, for larger values of α also the instances without stretching converge. This seems plausible, as for the global invariant model all circles are sufficiently non-overlapping even for moderate values of α. We have chosen $\alpha = 1$ with stretching to be the best setting for this algorithm.

Local AR The local attractor repeller model is shown in Figure 7.11 and Table 7.3 on page 160. As for the other algorithms, the setting $\alpha = 1$ without stretching leads to convergence problems for larger instances. However, for all other settings also the ratio of fails increases. The best convergence is reached for $\alpha = 25$ without stretching. In contrast to the other algorithms, scaling seems to be unprofitable, especially for large values of α. This can be explained as follows. This algorithm pushes the larger circles away from each other, as their weight of the repelling force is large. So it is very likely that the need for stretching is due to overlap of smaller circles in the center of the circuit. In a stretching step the larger circles are pushed even further, which might lead to numerical issues. However, without stretching, there is only overlap of smaller circles which might Ipopt be able to repair in a restoration phase. Furthermore, if α is small, in the final solution of the attractor repeller run the larger circles are still reasonable close and thus not pushed to far away by feasibility stretching.

As the ratio of fails increases for all settings, we have chosen $\alpha = 1$ with stretching, as this yields the significantly best objective values for almost all problem instance sizes.

Limited Range AR Figure 7.12 and Table 7.4 on page 161 visualize the limited range attractor repeller model. For this algorithm, the difference between the different values of α is smaller than for the other algorithms. This is due to the fact that by the large choice of $\beta = \max w_{ij}$ already small values α lead to significantly non-overlapping. However, larger values of α do not blow up the placement due to the limited range of the repelling force. Especially for the objective and the CPU time, almost all algorithm settings are equal. Due to the lower ratio of fails we have chosen $\alpha = 5$ without stretching to be the best setting for this algorithm.

Standard Attractor Repeller (AR0)



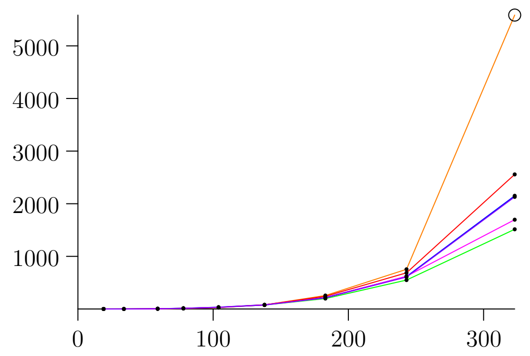
(a) CPU(s) / #circles. (b) Objective / #circles. (c) Ratio of Fails / #circles.

Figure 7.9: Standard AR – see Section 7.7.1.1.

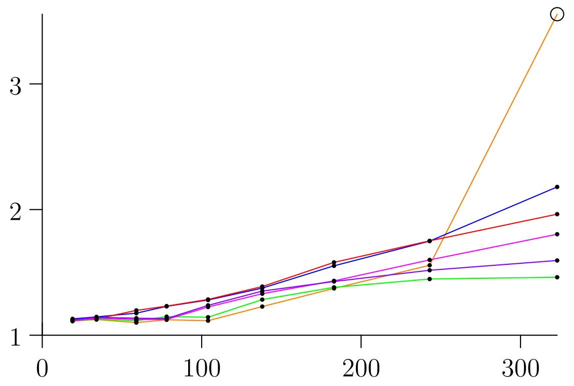
	Circuit	1-N	1-S	5-N	5-S	25-N	25-S
CPU(s)	E0031	0.265	0.210	0.180	0.180	0.170	0.190
	M0057	1.015	1.490	1.120	1.480	1.450	1.565
	S0041	3.615	1.550	2.395	1.455	1.475	1.205
	V0093	31.255	15.220	15.535	15.630	12.395	15.490
Objective	E0031	1.2181	1.1776	1.2127	1.1979	1.1865	1.1918
	M0057	1.1564	1.1840	1.2464	1.2654	1.3268	1.3199
	S0041	1.0966	1.0532	1.0739	1.0779	1.0709	1.0799
	V0093	1.0679	1.1081	1.0623	1.1080	1.0783	1.1040
Ratio of Fails	E0031	0.00	0.00	0.00	0.00	0.00	0.00
	M0057	0.00	0.00	0.00	0.00	0.00	0.00
	S0041	0.03	0.01	0.00	0.01	0.00	0.02
	V0093	0.12	0.00	0.00	0.00	0.00	0.00

Table 7.1: Standard AR – see Section 7.7.1.1.

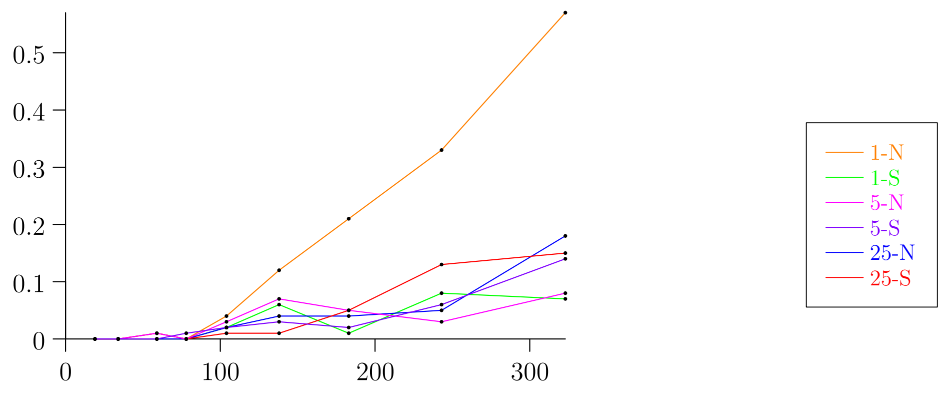
Global Invariant Attractor Repeller (AR1)



(a) CPU(s) / #circles.



(b) Objective / #circles.



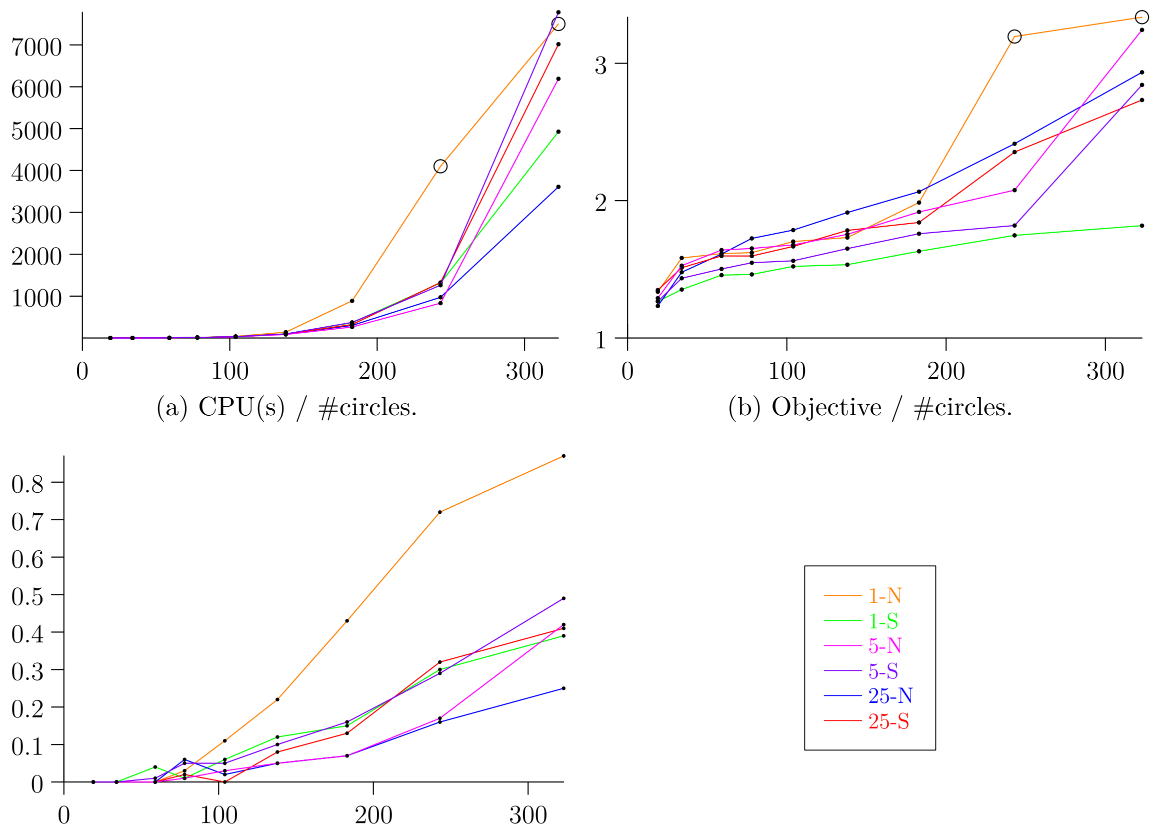
(c) Ratio of Fails / #circles.

Figure 7.10: Global invariant AR – see Section 7.7.1.1.

	Circuit	1-N	1-S	5-N	5-S	25-N	25-S
CPU(s)	E0031	0.160	0.170	0.190	0.180	0.280	0.205
	M0057	0.995	1.240	1.145	1.230	1.660	1.230
	S0041	0.925	1.160	0.920	1.095	1.135	1.240
	V0093	9.510	13.625	12.325	13.445	14.560	13.880
Objective	E0031	1.2134	1.2104	1.2468	1.2432	1.2541	1.2995
	M0057	1.1743	1.1885	1.2684	1.2812	1.3364	1.3273
	S0041	1.0551	1.0516	1.0498	1.0498	1.0569	1.0570
	V0093	1.0620	1.0974	1.1066	1.1217	1.1452	1.1476
Ratio of Fails	E0031	0.00	0.00	0.00	0.00	0.00	0.00
	M0057	0.00	0.00	0.00	0.00	0.00	0.00
	S0041	0.00	0.00	0.01	0.02	0.00	0.01
	V0093	0.00	0.02	0.02	0.00	0.01	0.00

Table 7.2: Global invariant AR – see Section 7.7.1.1.

Local Attractor Repeller (AR2)



(a) CPU(s) / #circles.

(b) Objective / #circles.

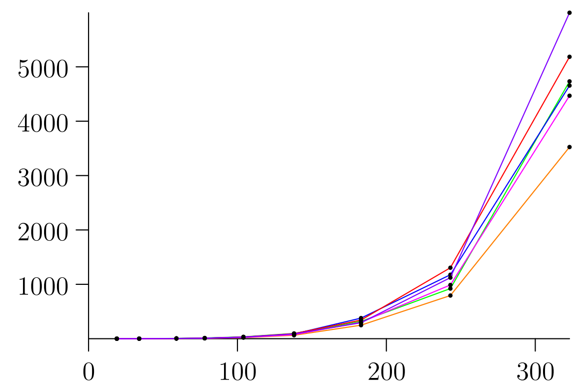
(c) Ratio of Fails / #circles.

Figure 7.11: Local AR – see Section 7.7.1.1.

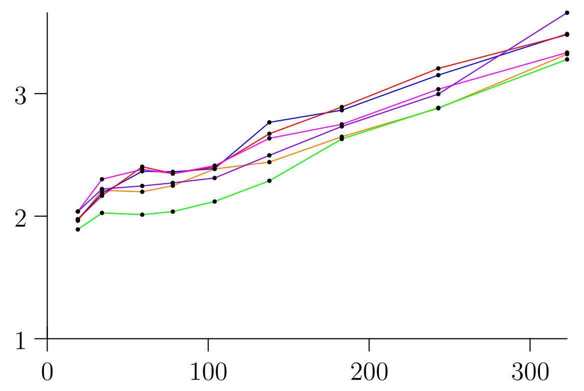
	Circuit	1-N	1-S	5-N	5-S	25-N	25-S
CPU(s)	E0031	0.180	0.220	0.230	0.235	0.290	0.245
	M0057	1.290	1.550	1.320	1.610	1.710	1.635
	S0041	0.995	1.225	1.020	1.250	1.305	1.290
	V0093	16.310	14.325	14.165	16.455	17.360	17.330
Objective	E0031	1.1421	1.1300	1.1700	1.1640	1.1708	1.2024
	M0057	1.4097	1.3113	1.4238	1.3466	1.4525	1.4807
	S0041	1.0756	1.0718	1.0807	1.0767	1.0868	1.0767
	V0093	1.1039	1.0896	1.2259	1.1837	1.8191	1.4909
Ratio of Fails	E0031	0.00	0.00	0.00	0.00	0.00	0.00
	M0057	0.00	0.00	0.00	0.00	0.01	0.00
	S0041	0.02	0.00	0.01	0.01	0.03	0.01
	V0093	0.02	0.01	0.00	0.03	0.00	0.05

Table 7.3: Local AR – see Section 7.7.1.1.

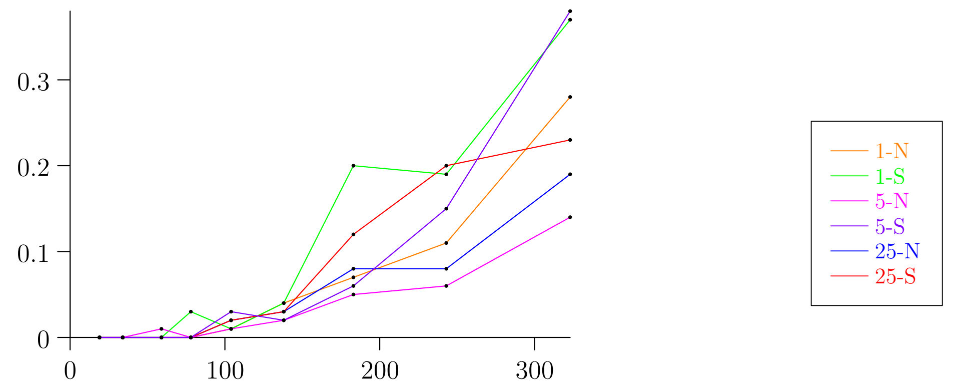
Limited Range Attractor Repeller (AR3)



(a) CPU(s) / #circles.



(b) Objective / #circles.



(c) Ratio of Fails / #circles.

Figure 7.12: Limited range AR – see Section 7.7.1.1.

	Circuit	1-N	1-S	5-N	5-S	25-N	25-S
CPU(s)	E0031	0.420	0.450	0.495	0.470	0.580	0.530
	M0057	2.545	2.630	1.625	2.010	1.670	1.560
	S0041	1.815	2.190	1.620	2.010	1.820	1.940
	V0093	11.820	17.100	11.515	13.630	17.300	15.500
Objective	E0031	2.4054	1.8509	2.5644	2.2014	2.6663	2.2806
	M0057	2.8649	2.1750	4.6878	3.6068	4.7866	4.9113
	S0041	1.1685	1.1272	1.2158	1.1880	1.2358	1.2205
	V0093	2.0174	1.7812	2.1494	2.0897	2.1431	2.1321
Ratio of Fails	E0031	0.01	0.00	0.00	0.00	0.00	0.00
	M0057	0.00	0.00	0.00	0.00	0.00	0.00
	S0041	0.00	0.01	0.00	0.00	0.00	0.00
	V0093	0.01	0.00	0.00	0.03	0.02	0.00

Table 7.4: Limited range AR – see Section 7.7.1.1.

7.7.1.2 Comparison of the Different AR-Models

In this section we compare the best tuned settings of all models. Furthermore, we also include the immediate application of the non-linear program CPP to the random initial solution into the comparison. Recall that we used the following settings:

- standard AR: $\alpha = 5$ with stretching
- global invariant AR: $\alpha = 1$ with stretching
- local AR: $\alpha = 1$ with stretching
- limited range AR: $\alpha = 5$ without stretching
- CPP without AR: CPP stated in (7.3) applied to the random initial solution

Figure 7.13 and Table 7.5 on page 163 visualize the comparison of these algorithms.

It can be seen that the final solutions of CPP without attractor repeller and the limited range attractor repeller model yield bad solutions. CPP gets stuck in a poor local optimum. The same is true for the limited range attractor repeller model. We designed the model to avoid the blow up of the placement. However, this also reduces the space to sort the circles according to their connections, so this model has the same problem as the immediate application of CPP. Even though not shown in the figures, in average on converging instances the limited range model and the CPP without AR lead to equally poor solutions. The worse solutions of the CPP without AR is due to the fact that many optimization runs did not converge, while for the limited range AR most of the runs converged. These convergence problems are also the reason for the long running time of CPP without AR, even if we only solve one non-linear program in contrast to the attractor repeller models, where two non-linear programs are solved.

The local attractor repeller model is better than limited range AR and CPP without AR in terms of the objective. However, it has convergence problems for larger instances and inferior running time to all other attractor repeller models.

The standard AR model and the global invariant AR model are almost similar in terms of running time and the ratio of fails, also the global invariant AR seems slightly better. However, the objective obtained by the invariant AR model is superior to all other models for almost all instances.

This result confirms the impression of Figure 7.4. The global invariant AR yields an equally degree of non-overlapping of all circles even of different size, while giving enough space for the circles to sort according to their wiring.

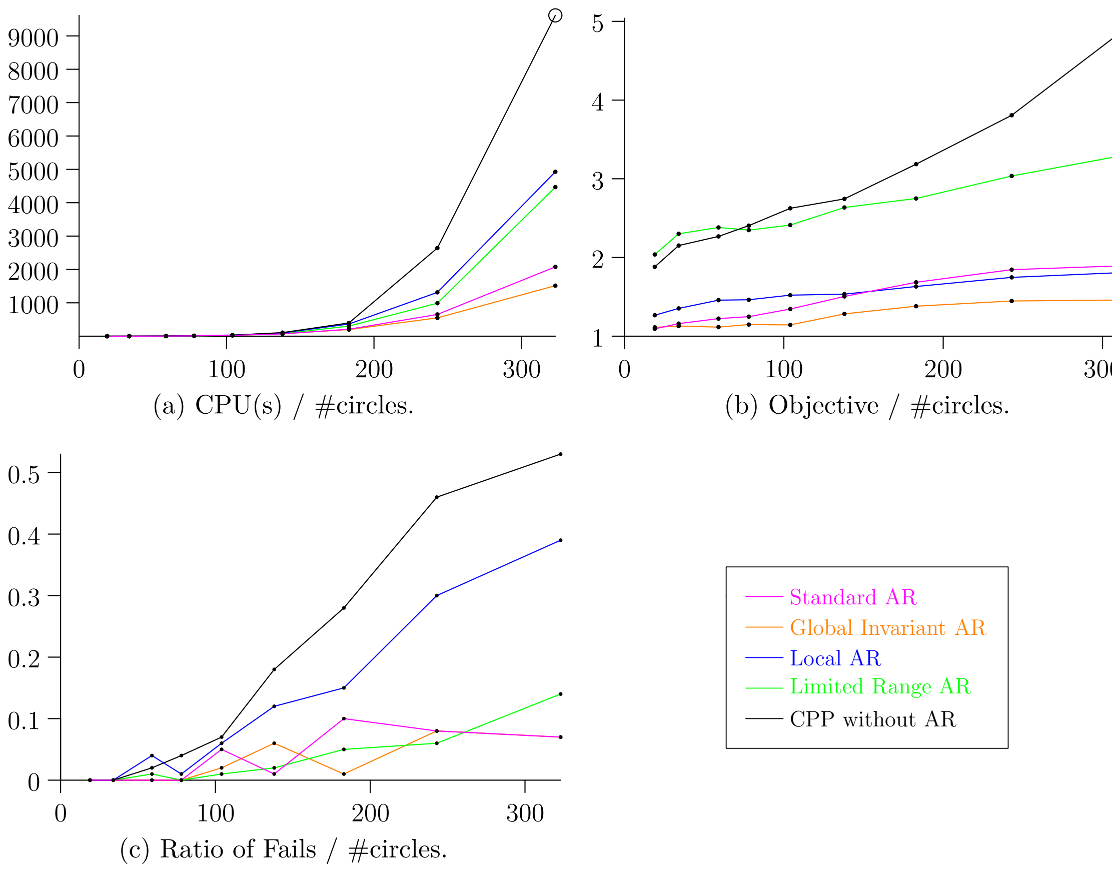


Figure 7.13: Comparison of AR – see Section 7.7.1.

	Circuit	Standard	Invariant	Local	Lim. Range	No AR
CPU(s)	E0031	0.180	0.170	0.220	0.495	0.570
	M0057	1.480	1.240	1.550	1.625	2.705
	S0041	1.455	1.160	1.225	1.620	2.580
	V0093	15.630	13.625	14.325	11.515	19.875
Objective	E0031	1.1979	1.2104	1.1300	2.5644	2.6430
	M0057	1.2654	1.1885	1.3113	4.6878	4.3518
	S0041	1.0779	1.0516	1.0718	1.2158	1.2692
	V0093	1.1080	1.0974	1.0896	2.1494	2.0508
Ratio of Fails	E0031	0.00	0.00	0.00	0.00	0.00
	M0057	0.00	0.00	0.00	0.00	0.00
	S0041	0.01	0.00	0.00	0.00	0.02
	V0093	0.00	0.02	0.01	0.00	0.00

Table 7.5: Comparison of AR – see Section 7.7.1.

7.7.1.3 Detailed Analysis of the Global Invariant AR

We have seen in the previous analysis that the global invariant attractor repeller model is significantly superior to the other models. Its best parameter choice is $\alpha = 1$ with stretching. As we use this algorithm in the remaining part of this thesis, we analyze it in more detail here. Figure 7.14 shows the quantiles of the objective ratio and the CPU time. While the worst case behavior can be bad, in more than 75% of the runs both the running time and the final objective are reasonable well, i.e. better than 50% away from the best found solution.

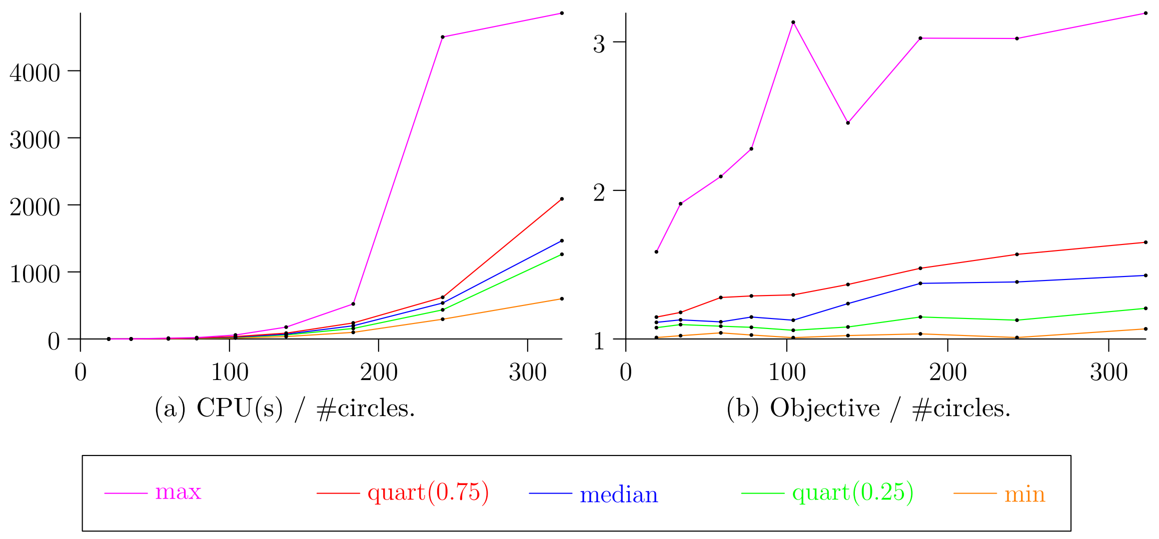


Figure 7.14: Quantiles of the global invariant attractor repeller model with $\alpha = 1$ and stretching. In (a) the running time is plotted over the number of circles. In (b) the ratio of the solution objectives divided by the best known solution objective is shown.

7.7.2 Local Search and Monotonic Basin Hopping

In this section we analyze the local search described Section 7.6.1 and the monotonic basin hopping described in Section 7.6.2.

In Section 7.7.1 we observed that the best attractor repeller model is the global invariant model with $\alpha = 1$ and feasibility stretching. Therefore, we use this model to generate a solution to the circle placement problem. After the attractor repeller run we apply CPP. This result is called the *initial solution*. Then we apply monotonic basin hopping, local search or both to this initial solution.

We compare the following algorithmic settings:

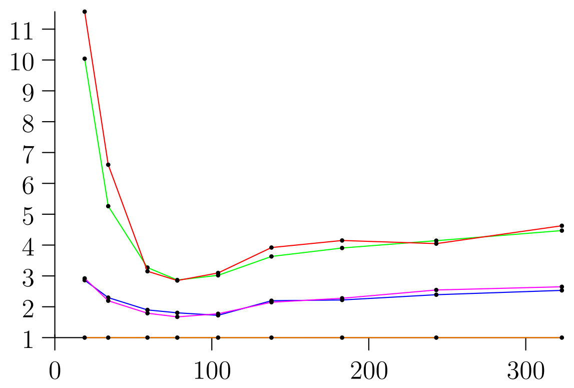
- ECSLS: Equal circle swap local search is performed.
- MBH 5: Monotonic basin hopping with maximum 5 fails is performed.
- MBH 15: Monotonic basin hopping with maximum 15 fails is performed.
- ECSLS / MBH 5: ECSLS and MBH5 is executed.
- ECSLS / MBH 15: ECSLS and MBH15 is executed.

The results are shown in Figure 7.15. In Figure 7.15a it can be seen that the running time mainly depends on the maximal number of monotonic basin hopping fails. Except for very small instances, the running time with 15 MBH fails is approximately five times the initial running time and the running time with 5 MBH fails is approximately 2.5 times the initial running time. If we only use equal circle swap local search, the running time is nearly not affected.

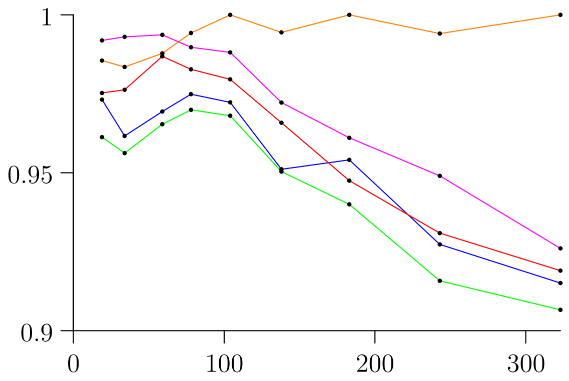
It is plausible that ECSLS has almost no influence on the running time, since in ECSLS only the circle rotation problem has to be solved, which is much simpler than the placement problem. In fact, we have seen in Chapter 3 that in can be done in milliseconds compared to up to more than thousand seconds for the initial solution generation.

Despite the very short running time, ECSLS has a large influence on the objective improvement. This can be seen in Figure 7.15b. The application of ECSLS without any monotonic basin hopping does not improve the objective much. But the combination of ECSLS and MBH is significantly superior to just applying MBH. For all but one instance, ECSLS / MBH 5 is even superior to MBH 15, even though its running time is only half as long. In summary, the key ingredient for the improvement is monotonic basin hopping, and an increase of the number of MBH fails improves the final solution quality. However, ECSLS in combination with MBH significantly improves the final solution for almost no additional running time.

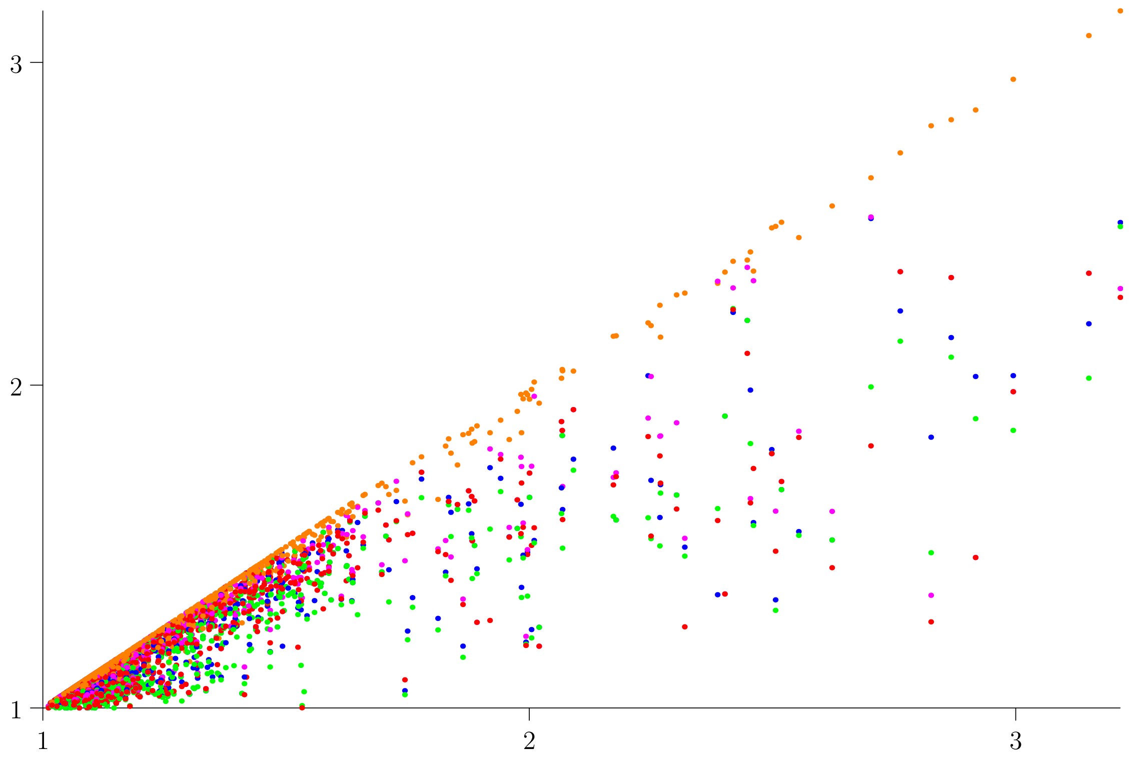
In Figure 7.15c the relative improvement achieved by the different algorithms for each run is shown. On the x-axis the ratio of the initial solution divided by the best known solution is shown, on the y-axis the ratio of the final solution divided by the best known solution. It can be seen that, as intuitively expected, the improvement is larger if the initial solution is worse. However, for poor initial solutions usually the final solution is still bad.



(a) CPU factor / #circles.



(b) Final objective quotient / #circles.



(c) Final objective quotient / initial objective quotient.

ECSLS	MBH 5	MBH 15	ECSLS / MBH 5	ECSLS / MBH 15

Figure 7.15: Comparison of different configurations of ECSLS and MBH. In (a) on the x-axis the number of circles is shown, on the y-axis the median running time of the total run divided by the running time of the initial solution generation. In (b) on the x-axis the number of circles is shown, on the y-axis the median quotient of the final objective and the initial solution objective. In (c) on the x-axis the normalized final solution objective is shown and on the y-axis the normalized initial solution objective. The normalized objective is the objective divided by the best known objective for this instance.

It is obvious that an increase of the number of MBH fails increases the running time and improves the solution quality. However, in contrast we could also perform multiple runs from different starting points. This is the classical contradiction between exploration and exploitation. Starting several runs without MBH from different starting points increases the exploration. In contrast, performing a single run with many MBH steps yields a good exploitation.

As a last result we compare approaches with approximately equal running time. Figure 7.15a motivates that the following algorithms are likely to have similar running times.

- Run the setting ECSLS five times and take the best solution.
- Run the setting ECSLS / MBH 5 twice and take the best solution
- Run the setting ECSLS / MBH 15 once.

The result of this comparison is shown in Figure 7.16. While the running times of these algorithms are approximately equal, for larger instances ECSLS / MBH 5 yields the best results while ECSLS / MBH 15 yields the worst result. So ECSLS / MBH 5 seems to be a good trade-off between exploration and exploitation. The algorithm overcomes local optima by monotonic basin hopping but is fast enough to be run multiple times. With this algorithm in the median we obtain solutions significantly better than 10% worse than the best known solution.

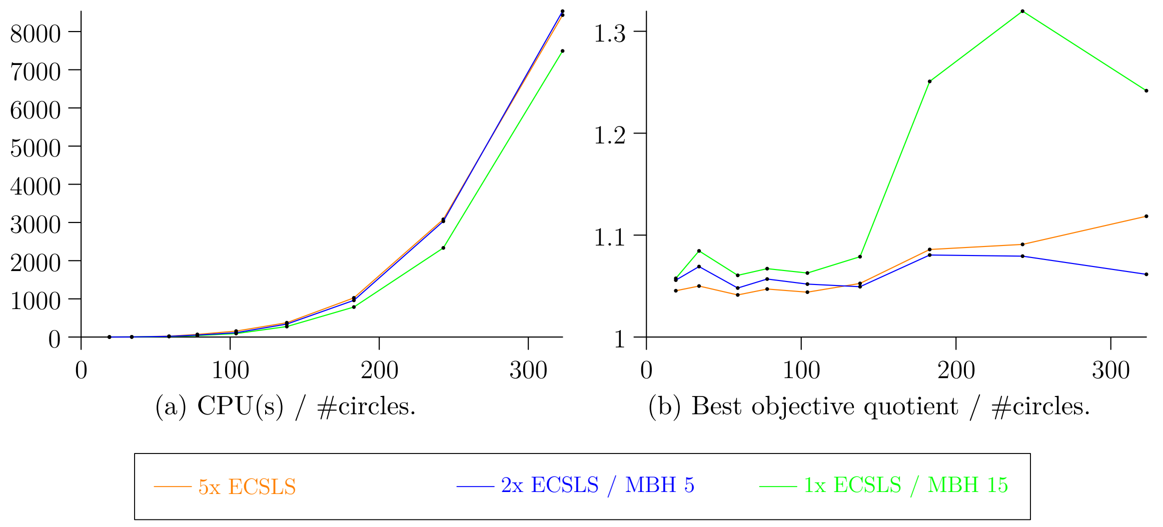


Figure 7.16: Comparison of post processing versus multiple shoot approaches. In (a) the median CPU time over the number of circles is shown, in (b) the median of the quotient of the best solution in this run and the best known objective is shown.

7.7.3 Conclusion

In this section we evaluated our algorithms for the circle placement problem numerically. We showed that our proposed global scaling invariant attractor repeller model is superior to previously existing models in terms of solution quality and running time. Furthermore, we demonstrated that our proposed local improvement algorithms equal circle swap local search (ECSLS) and monotonic basin hopping (MBH) are able to overcome local optima and to improve the solution quality by more than 5% for moderate increase in running time.

We evaluated the best configuration of our algorithm. The best attractor repeller model to generate the initial solution for the constrained non-linear optimization problem CPP is the global scaling invariant attractor repeller model with repeller weight $\alpha = 1$ and feasibility stretch. The best trade-off between running time and solution quality for the post processing step is the combination of equal circle swap local search and monotonic basin hopping with maximum 5 fails.

8 Rectangle Placement Problem

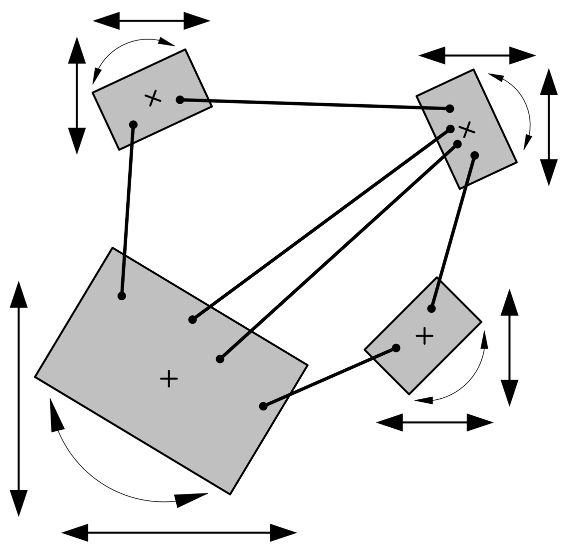


Figure 8.1: The rectangle placement problem is to rotate and translate connected rectangles such that they do not overlap and the wire length is minimized.

In this chapter we consider the rectangle placement problem. For this problem a set of fixed sized rectangular components is given. Each component has pins with fixed offset to the center of the component. The pins are connected by nets. The rectangle placement problem is to place and rotate the components in the plane, such that they do not overlap and the wire length is minimized.

The rectangle placement problem often appears in the context of electronic circuit design. The rectangles represent electronic components and the nets represent electric connections. In particular, the placement step in System-in-Package design is a rectangle placement problem. For System-in-Packages, we have to pack up to approximately 250 heterogeneous components. Rotation and non-overlapping of the components have to be considered within the algorithm.

Rectangle placement problems have been intensively studied in the literature. However, most approaches either focus on problem instances with up to 100 components or on instances with a large number of homogeneous components in VLSI design.

There is a lack of algorithms for placement instances with up to 250 heterogeneous components. Our proposed rounded rectangle algorithm is an analytic placement approach. In an initial step, we represent all components as circles and use the results of Chapter 7 to generate a good initial solution. Within a sequence of non-linear programs, the algorithm refines the components from circles to rectangles.

In Section 8.1 we formulate the rectangle placement problem. We survey the literature on this type of problems in Section 8.2. In Section 8.3 we state the rounded rectangle placement problem ($RRPP$) and show, that it satisfies the Mangasarian-Fromovitz constraint qualification (MFCQ) at each feasible point. In Section 8.4 we describe the rounded rectangle algorithm and prove its convergence to a feasible solution for the rectangle placement problem. Finally, in Section 8.5 we present a detailed numerical evaluation of the algorithm.

8.1 Problem Statement

We now formalize the rectangle placement problem. Therefore, we first define

$$\mathbb{C}' := \{z \in \mathbb{C} : \Re(z) \geq \Im(z) \geq 0\}.$$

In the rectangle placement problem given is:

- A set $\mathcal{C}$ of n rectangular components with fixed sizes $s_j \in \mathbb{C}'$, $j \in \mathcal{C}$.
- A set $\mathcal{P}$ of m pins. Pin l is connected to component $j(l)$ and has fixed offset $p_l \in \mathbb{C}$ from its center.
- A set $\mathcal{N}$ of nets. Net k has m_k pins $\mathcal{P}_k$ and weight μ_k.

Similar to Chapter 7, a placement is defined by the locations $\boldsymbol{c} = (c_1, \ldots, c_n) \in \mathbb{C}^n$ and the rotations $\boldsymbol{\varphi} = (\varphi_1, \ldots, \varphi_n) \in \mathbb{R}^n$ of the components. If convenient, we sometimes refer to the rotations by

$$\boldsymbol{z} = (z_1, \ldots, z_n) = \exp(\imath\boldsymbol{\varphi}) = (\exp(\imath\varphi_1), \ldots, \exp(\imath\varphi_n)).$$

The optimization problem is to find a placement with minimal wire length such that the components do not overlap. Additionally, the rotations might be restricted to multiples of a fixed angle, e.g. multiples of $\pi/2$. The wire length is defined as in (7.1) and (7.2). We briefly summarize it here.

The absolute position of pin l is $q_l := c_{j(l)} + z_{j(l)} p_l$. Denote by $\Phi = (\phi_{ij}) \in \{0,1\}^{m \times n}$ the component-to-pin matrix with $\phi_{lj} = 1$ if and only if pin l is on circle j. Furthermore, denote by $P = \operatorname{diag}(p_1, \ldots, p_m)$ the diagonal matrix of pin offsets. Then with a positive semidefinite matrix Q the wire length is

$$\operatorname{wl}(\boldsymbol{c}, \boldsymbol{\varphi}) = \boldsymbol{q}^H Q \boldsymbol{q} \qquad \text{with} \qquad \boldsymbol{q} = \Phi\boldsymbol{c} + P\Phi\boldsymbol{z}.$$

8.2 Literature Survey

In this section we give a survey of the literature on the rectangle placement problem. This problem naturally arises from electronic circuit design and, hence, much of the literature is related to this application. This survey is not exhaustive but covers the main approaches to the rectangle placement problem. We also include literature on the rectangle placement problem with modified wire length models and on related problems such as floorplanning. An extensive recent survey over a wide range of these algorithms in the context of VLSI design is given in [Nam; Cong 2007].

The approaches to the rectangle placement problem can be divided into discrete and analytical algorithms. Discrete algorithms are mostly based on metaheuristics, mixed integer programming and constraint programming. Analytical algorithms normally use formulations as non-linear optimization problem. Furthermore, these algorithms can be divided in flat and hierarchical placement algorithms. Flat algorithms consider the problem globally, while hierarchical algorithms separate the circuit via hypergraph partitioning and place each cluster individually.

There are several metaheuristic approaches to the rectangle placement problem. In [Murata et al. 1996] a simulated annealing solver based on the placement encoding by sequence pairs is presented. A survey of simulated annealing algorithms for the rectangle placement problem is given in [Wong; Leong; Liu 1988]. In [Areibi; Yang 2004] a combination of local search and genetic algorithms is applied to solve the rectangle problem with half perimeter wire length. These heuristics can handle large solution instances, but for acceptable running time they often yield substandard solution quality.

In [Faroe; Pisinger; Zachariasen 2003] the guided local search algorithm for the placement problem with half perimeter wire length is proposed. It starts from an initial solution generated by another optimization method and successively improves this solution by exploration of its neighborhood.

Also exact optimization methods have been applied to the rectangle placement problem. For example, in [Onodera; Taniguchi; Tamaru 1991] a branch and bound algorithm using the half perimeter wire length is proposed. This algorithm solves the problem to global optimality. However, only instances with up to six components can be solved directly, and larger instances must be decomposed. For practical problem instances exact optimization algorithms cannot be applied.

In [Berger 2010] the problems to place connected rectangles with respect to the area usage and the half perimeter wire length are analyzed. Using mixed integer formulations and constraint programming techniques, exact solution methods and heuristics are proposed. The proposed algorithms yield good solution quality with respect to packing density and wire length for small to medium sized problem instances, but they cannot handle instances with 100 or more components in acceptable running time.

Analytical algorithms for the placement problem have been studied in the literature. In [Alon; Ascher 1988] a non-linear penalty approach for the rectangle placement problem is proposed. This algorithm minimizes the wire length in the clique model and considers both arbitrary and discrete rotation of the rectangles. The non-overlapping is computed between the circumcircles of the rectangles and added as penalty term to the objective. However, transformation from this possibly overlapping placement of circumcircles to rectangles is not considered.

In [Sha; Dutton 1985] the rectangle placement problem is stated as non-linear problem including rotation and non-overlapping. They replace the rectangles by rectangles with half circles on two opposite sides and explicitly formulate the non-overlapping constraints by the concept of signed distances. However, they only consider connections between the rectangle centers.

In [Kahng; Reda; Wang 2005] a hierarchical analytic solver for large scale placement problems is proposed. After clustering the components based on the net structure, the clusters are placed separately by solving a non-linear program with a quadratic penalty formulation for the non-overlapping constraints. In [Luo; Anjos; Vannelli 2008] the floorplanning problem related to the rectangle placement problem is considered. In this algorithm an attractor repeller model for circular abstractions of the rectangular blocks is optimized. Keeping the order relations of this placement, the final floorplan is computed. In [Viswanathan; Chu 2004] an analytical solver for the cell design problem with a combination of the clique and the star wire length is proposed. However, this solver does not consider non-overlapping within the non-linear optimization phase, but generates the non-overlapping placement heuristically in post-processing steps.

8.3 Rounded Rectangle Placement Problem

The rounded rectangle algorithm consists of solving a sequence of non-linear programs where the components are abstracted as rounded rectangles. In this section we state and analyze such a non-linear program.

In Section 8.3.1 we state the discrete rotation constraint. In Section 8.3.2 we introduce the concept of rounded rectangles and formulate the non-overlapping of rounded rectangles as non-linear constraints. In Section 8.3.3 we formulate a non-linear program for the rounded rectangle placement problem and show that it satisfies the Mangasarian-Fromovitz constraint qualification in each feasible point.

8.3.1 Discrete Rotation

In practice, usually the components may not by rotated by an arbitrary angle. Instead they must only be rotated by multiples of $\pi/2$ or, less often, $\pi/4$. This constraint can easily be stated as non-linear equality constraint.

Lemma 8.1. *Let $\varphi_0 \neq 0$ be fixed. The angle φ is a multiple of φ_0 if and only if*

$$\sin\left(\pi \frac{\varphi}{\varphi_0}\right) = 0.$$

In particular, φ is a multiple of $\frac{\pi}{2}$ if and only if $\sin(2\varphi) = 0$.

Proof. It is $\sin(\alpha) = 0$ if and only if α is a multiple of π. Thus

$$\sin\left(\pi \frac{\varphi}{\varphi_0}\right) = 0 \iff \pi \frac{\varphi}{\varphi_0} \in \pi\mathbb{Z} \iff \varphi \in \varphi_0\mathbb{Z}.$$

With $\varphi_0 = \frac{\pi}{2}$ the second statement follows. □

Although easy to formulate, this constraint is problematic for non-linear solvers, as it leads to a discretized solution space. If this constraint is enforced to be satisfied in each iteration step, the components are not allowed to rotate at all. Therefore, in the algorithm this constraint is handled by a penalty approach: Its violation is added as weighted penalty to the objective. For zero or small penalty weights, the components can be arbitrarily rotated. By increasing the penalty parameter during the algorithm, the components are enforced to rotate orthogonally.

8.3.2 Non-Overlapping

The components are rectangles. However, the rotation of rectangles leads to many local optima, see e.g. Figure 8.2. For an initial placement we approximate the components by smooth shapes. The simplest and smoothest shape is a circle. If we represent each component by a circle, rotation is decoupled from non-overlapping. This reduces the number of local optima. For example, in Figure 8.2 the shown placement is not locally optimal if the small components are modeled as circles.

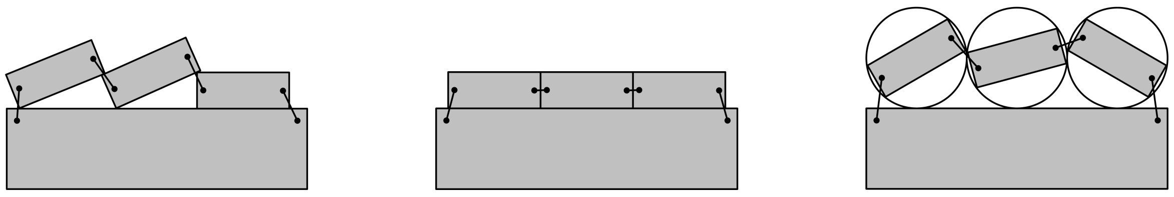


Figure 8.2: Local and global optimum of the rectangle placement problem. If the small components are circles, this placement is no local optimum.

If all components are represented as circles, the problem is equivalent to the circle placement problem which we have already discussed in Chapter 7 in detail. And in fact, we solve the circle placement problem to generate a good initial solution for the rounded rectangle algorithm.

However, a circle is a crude approximation of a rectangle. During the algorithm the approximation has to be refined. In our algorithm this refinement is done by modeling the components as rounded rectangles.

Definition 8.2 (Rounded Rectangle)**.** *A rounded rectangle $R(r, w)$ with $r \geq 0$ and $w \in \mathbb{C}'$ is the Minkowski sum of an inner rectangle with width $2\Re(w)$ and height $2\Im(w)$ and a disc with radius r. A rounded rectangle with $\Re(w) > 0$ and $\Im(w) = 0$ is called a stadium, a rounded rectangle with $w = 0$ is a circle.*

The set $c + \exp(\imath\varphi)R(r, w)$ is a placed rounded rectangle with center c and rotation φ.

Definition 8.3 (Degenerated Rounded Rectangle)**.** *A rounded rectangle $R(r, w)$ is called degenerated, if $r = 0$ and $\Im(w) = 0$.*

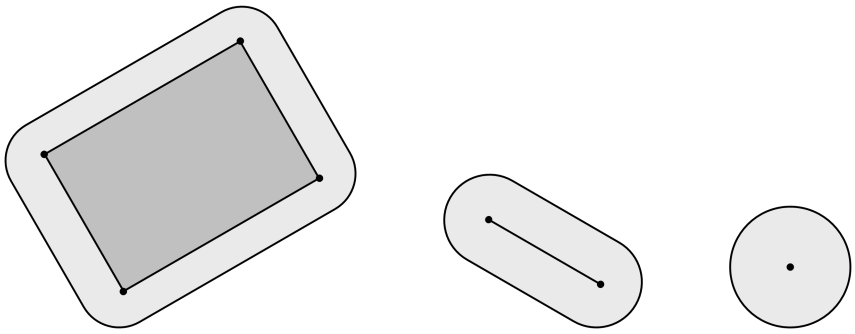


Figure 8.3: Rounded rectangle $R(r, w)$, stadium and circle.

The vertices of the inner rectangle of a rounded rectangle $R(r, w)$ are

$$C(w) = \{w,\ -w,\ \overline{w},\ -\overline{w}\}.$$

In particular, for a stadium there are only two vertices $C(w) = \{w, -w\}$ while for a circle the only vertex is $C(w) = \{0\}$. The vertices of the inner rectangle of the placed rounded rectangle $c + \exp(\imath\varphi)R(r, w)$ are

$$c + \exp(\imath\varphi)C(w) = \{c + \exp(\imath\varphi)\tilde{w} : \tilde{w} \in C(w)\}. \tag{8.1}$$

We have to assure that two components do not overlap. Two rounded rectangles do not overlap if the distance of their inner rectangles is greater than the sum of their radii.

We consider the case where both components i and j are circles $c_i + \exp(\imath\varphi_i)R(r_i, 0)$ and $c_j + \exp(\imath\varphi_j)R(r_j, 0)$ first. Then the two components i and j do not overlap if and only if the distance of their centers is greater than the sum of their radii, i.e.

$$|c_i - c_j|^2 \geq (r_i + r_j)^2$$

where the square is taken for smoothing.

For pairs of general rounded rectangles a similar constraint is possible. Therefore, we have to compute the distance of the inner rectangles. These functions are more complicated but explicitly computable. However, they are only once continuously differentiable. This approach has been considered in [Sha; Dutton 1985] for stadiums.

Another approach is to introduce separating hyperplanes. By the separating hyperplane theorem, two convex and compact sets can be separated by a hyperplane such that the sets are contained in the different half spaces. We apply this formulation for the separation of rounded rectangles.

As stated in Definition 2.8, a hyperplane with distance d and normal angle θ in the complex plane is

$$H(\theta, d) := \{x \in \mathbb{C} : \Re(x \cdot \overline{\exp(\imath\theta)}) = d\} = \{x \in \mathbb{C} : \langle x, \exp(\imath\theta)\rangle_{\mathbb{R}} = d\}.$$

Lemma 8.4. *Let $H := H(\theta, d)$ be a hyperplane and $H_{\leq}$ and $H_{\geq}$ the corresponding half spaces. Let $c + \exp(\imath\varphi)R(r, w)$ be a placed rounded rectangle. Define with $\chi \in \{-1, +1\}$*

$$h^{\chi}_{r,\tilde{w}}(c, \varphi, d, \theta) := r + \chi(d - \langle c + \tilde{w}\exp(\imath\varphi), \exp(\imath\theta)\rangle_{\mathbb{R}}).$$

Then it holds

$$\begin{aligned} R \subset H_{\leq} \quad &\Longleftrightarrow \quad h^{-1}_{r,\tilde{w}}(c, \varphi, d, \theta) \leq 0, \quad \forall \tilde{w} \in C(w), \\ R \subset H_{\geq} \quad &\Longleftrightarrow \quad h^{+1}_{r,\tilde{w}}(c, \varphi, d, \theta) \leq 0, \quad \forall \tilde{w} \in C(w). \end{aligned}$$

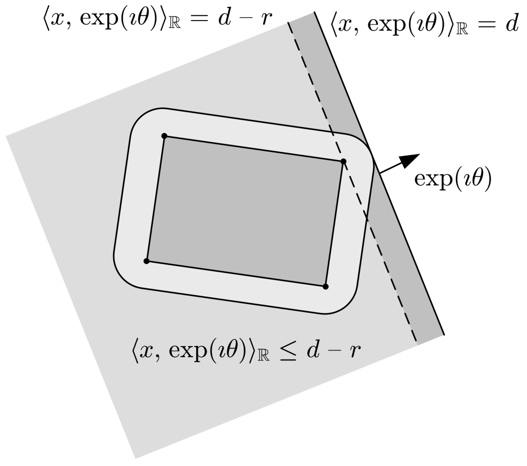

$\langle x, \exp(\imath\theta)\rangle_{\mathbb{R}} = d - r$
$\langle x, \exp(\imath\theta)\rangle_{\mathbb{R}} = d$
$\exp(\imath\theta)$
$\langle x, \exp(\imath\theta)\rangle_{\mathbb{R}} \leq d - r$

Figure 8.4: Half space containment of a placed rounded rectangle. The solid line is the hyperplane $H(\theta, d) = \{x : \langle x, \exp(\imath\theta)\rangle_{\mathbb{R}} = d\}$, the dashed line $H(\theta, d - r) = \{x : \langle x, \exp(\imath\theta)\rangle_{\mathbb{R}} = d - r\}$. The light gray area is the half space $H_{\leq}(\theta, d) = \{x : \langle x, \exp(\imath\theta)\rangle_{\mathbb{R}} \leq d - r\}$ and the union of the light and the dark gray area is the half space $H(\theta, d)_{\leq} = \{x : \langle x, \exp(\imath\theta)\rangle_{\mathbb{R}} \leq d\}$.

Proof. We only consider the case $\chi = -1$. This case is visualized in Figure 8.4.

The placed rounded rectangle R is in the half space $H_{\leq}(\theta, d)$ if and only if all of its vertices $c + \exp(\imath\varphi)C(w)$ are in the half space $H_{\leq}(\theta, d - r)$. From (8.1) we get for all $\tilde{w} \in C(w)$:

$$\langle c + \exp(\imath\varphi)\tilde{w}, \exp(\imath\theta)\rangle_{\mathbb{R}} \leq d - r \iff h^{-1}_{r,\tilde{w}}(c, \varphi, d, \theta) \leq 0.$$

□

To get a formulation of these constraints in real variables, we compute

$$\begin{aligned} &h^{\chi}_{r,\tilde{w}}(c, \varphi, d, \theta) \\ &\quad = r + \chi\Big(d - \Re(c)\cos(\theta) - \Im(c)\sin(\theta) - \Re(\tilde{w})\cos(\theta - \varphi) - \Im(\tilde{w})\sin(\theta - \varphi)\Big). \end{aligned}$$

For each pair $i < j$ where not i and j both are circles, we introduce a hyperplane $H_{ij} = H(\theta_{ij}, d_{ij})$ with normal angle θ_{ij} and distance d_{ij}. We want the component i to be in $H_{ij,\leq}$ and the component j to be in $H_{ij,\geq}$. For all pairs $i < j$ where not i and j both are circles, we obtain the non-overlapping constraints

$$\begin{aligned} h^{-1}_{r_i,\tilde{w}_i}(c_i, \varphi_i, d_{ij}, \theta_{ij}) &\leq 0, \qquad \forall \tilde{w}_i \in C(w_i), \\ h^{+1}_{r_j,\tilde{w}_j}(c_j, \varphi_j, d_{ij}, \theta_{ij}) &\leq 0, \qquad \forall \tilde{w}_j \in C(w_j). \end{aligned} \tag{8.2}$$

Remark 8.5. *To unify the notation, we make the convention $d_{ji} = d_{ij}$ and $\theta_{ji} = \theta_{ij}$. They are not considered as different variables but as different names for the same variable. Furthermore, define*

$$\chi_{ij} = \begin{cases} -1 & \text{if } i < j, \\ 0 & \text{if } i = j, \\ +1 & \text{if } i > j. \end{cases}$$

Then the constraints (8.2) *are equivalent to the constraint that for all pairs $i \neq j$ where not i and j both are circles it holds*

$$h^{\chi_{ij}}_{r_i,\tilde{w}_i}(c_i, \varphi_i, d_{ij}, \theta_{ij}) \leq 0, \qquad \forall \tilde{w}_i \in C(w_i).$$

For a rounded rectangle there are four constraints per half space containment, for a stadium two and for a circle one constraint. Additionally, we have to introduce two real variables for each hyperplane. However, all constraints are smooth.

8.3.3 Non-Linear Program Formulation

In this section we formulate the rounded rectangle placement problem as non-linear program. To use this formulation during an iterative algorithm in Section 8.4, we introduce some additional constraints. Each component i is modeled as a rounded rectangle $R(r_i, w_i)$ with fixed radius r_i and fixed half size w_i.

The placement is defined by the component centers $\boldsymbol{c} \in \mathbb{C}^n$ and the component rotations $\boldsymbol{\varphi} \in \mathbb{R}^n$. Furthermore, there are the separating hyperplanes for the component pairs $i < j$ given by the normal vector angle $\theta_{ij} \in \mathbb{R}$ and the distance $d_{ij} \in \mathbb{R}$. Hence, the vector of decision variables is $(\boldsymbol{c}, \boldsymbol{\varphi}, \boldsymbol{d}, \boldsymbol{\theta})$. Some variables, e.g. some angles φ_i, might be fixed. Other variable like the hyperplane variables for pairs of circles might be not occurring in the problem. These variables can be removed in the practical implementation. We keep them here for the ease of notation.

Before we state the non-linear program, we introduce the following sets:

- The set of components modeled as circles is denoted by $\mathcal{C}^\circ = \{i \in \mathcal{C} : w_i = 0\}$.
- The set of component pairs where both components are circles is $\mathcal{G}_0 := \{(i,j) : i,j \in \mathcal{C}^\circ, i < j\}$.
- The set of component pairs where not both components are circles is $\mathcal{G}_1 := \{(i,j) : i \notin \mathcal{C}^\circ \vee j \notin \mathcal{C}^\circ, i \neq j\}$.
- The set of components that must be orthogonal rotated is $\mathcal{C}^\perp \subset \mathcal{C}$.
- The set of components i with fixed rotation φ_i^{fix} is denoted by $\mathcal{C}^{fix} \subset \mathcal{C}$. The vector of fixed rotations is $\boldsymbol{\varphi}^{fix}$. Entries of $\boldsymbol{\varphi}^{fix}$ for non-fixed components are ignored.
- The set of components that should be penalized for non-orthogonal rotation is denoted by $\mathcal{C}^{rot} \subset \mathcal{C}$. The penalty factor is μ^{rot}.

W.l.o.g. we assume that $\mathcal{C}^{fix} \cap \mathcal{C}^\perp = \emptyset$. Otherwise, for $i \in \mathcal{C}^{fix} \cap \mathcal{C}^\perp$ either the problem is infeasible as $\sin(2\varphi_i^{fix}) \neq 0$ or the orthogonal rotation constraint is redundant.

The set of feasible placements is $D(\boldsymbol{w}, \boldsymbol{r}, \mathcal{C}^\perp, \mathcal{C}^{fix}, \boldsymbol{\varphi}^{fix})$. It is given by $(\boldsymbol{c}, \boldsymbol{\varphi}, \boldsymbol{d}, \boldsymbol{\theta}) \in D(\boldsymbol{w}, \boldsymbol{r}, \mathcal{C}^\perp, \mathcal{C}^{fix}, \boldsymbol{\varphi}^{fix})$ if the following constraints are satisfied:

$$(r_i + r_j)^2 - |c_i - c_j|^2 \leq 0, \qquad \forall (i,j) \in \mathcal{G}_0 \tag{8.3}$$
$$h^{\chi_{ij}}_{r_i,\tilde{w}}(c_i, \varphi_i, d_{ij}, \theta_{ij}) \leq 0, \qquad \forall \tilde{w} \in C(w_i), \qquad \forall (i,j) \in \mathcal{G}_1 \tag{8.4}$$
$$\sin(2\varphi_i) = 0, \qquad \forall i \in \mathcal{C}^\perp \tag{8.5}$$
$$\varphi_i - \varphi_i^{fix} = 0, \qquad \forall i \in \mathcal{C}^{fix} \tag{8.6}$$

Equation (8.3) enforces the non-overlapping of circle pairs, (8.4) separates all other pairs of components by hyperplanes. Equation (8.5) assures the orthogonal rotation of components in $\mathcal{C}^\perp$ and (8.6) enforces the fixed rotation of components in $\mathcal{C}^{fix}$.

Definition 8.6 (Rounded Rectangle Placement Problem)**.** *Denote by* $\operatorname{wl}(\boldsymbol{c},\boldsymbol{\varphi})$ *the wire length as defined in* (7.1)*. Then the rounded rectangle placement problem denoted by* $RRPP(\boldsymbol{w},\boldsymbol{r},\mathcal{C}^{\perp},\mathcal{C}^{fix},\boldsymbol{\varphi}^{fix},\mathcal{C}^{rot},\mu^{rot})$ *is*

$$\begin{aligned} \min \quad & \operatorname{wl}(\boldsymbol{c},\boldsymbol{\varphi}) + \mu^{rot} \sum_{i\in\mathcal{C}^{rot}} \sin^2(2\varphi_i) \\ \text{s. t.} \quad & (\boldsymbol{c},\boldsymbol{\varphi},\boldsymbol{d},\boldsymbol{\theta}) \in D(\boldsymbol{w},\boldsymbol{r},\mathcal{C}^{\perp},\mathcal{C}^{fix},\boldsymbol{\varphi}^{fix}). \end{aligned} \tag{$RRPP$}$$

In particular, we can state the rectangle placement problem as rounded rectangle placement problem.

Definition 8.7 (Rectangle Placement Problem (RPP))**.** *Set* $w_i = s_i$ *and* $r_i = 0$ *for* $i \in \mathcal{C}$ *and* $\mathcal{C}^{\perp} = \mathcal{C}$ *and* $\mathcal{C}^{fix} = \mathcal{C}^{rot} = \emptyset$*. Then the rectangle placement problem RPP is equivalent to* $RRPP(\boldsymbol{w},\boldsymbol{r},\mathcal{C}^{\perp},\mathcal{C}^{fix},\boldsymbol{\varphi}^{fix},\mathcal{C}^{rot},\mu^{rot})$*, i.e.*

$$\begin{aligned} \min \quad & \operatorname{wl}(\boldsymbol{c},\boldsymbol{\varphi}) \\ \text{s. t.} \quad & (\boldsymbol{c},\boldsymbol{\varphi},\boldsymbol{d},\boldsymbol{\theta}) \in D(\boldsymbol{w},\boldsymbol{r},\mathcal{C},\emptyset,0). \end{aligned}$$

We show that the problem satisfies the important Mangasarian-Fromovitz constraint qualification (MFCQ). This is a sufficient condition for many solvers to converge to Karush-Kuhn-Tucker points, see e.g. [Haeser 2010]. For this proof we need the following lemma.

Lemma 8.8. *Let* $(\boldsymbol{c},\boldsymbol{\varphi},\boldsymbol{d},\boldsymbol{\theta}) \in D(\boldsymbol{w},\boldsymbol{r},\mathcal{C}^{\perp},\mathcal{C}^{fix},\boldsymbol{\varphi}^{fix})$*. Let* $(i,j) \in \mathcal{G}_1$ *and* i *not be degenerated. Then*

$$\chi_{ij}(\Re(c_i \exp(-\imath\theta_{ij})) - d_{ij}) > 0.$$

Proof. Denote $v_{ij} = \exp(\imath\theta_{ij})$ and $z_i = \exp(\imath\varphi_i)$. By (8.4) we know for feasible points

$$\forall \tilde{w} \in C(w_i) : \qquad \chi(\Re((c_i + \tilde{w}z_i)\overline{v_{ij}}) - d_{ij}) \geq r_i$$

where the inequality is strict for at least one $\tilde{w} \in C(w_i)$ if component i is a proper rounded rectangle. With $\sum_{\tilde{w}\in C(w_i)} \tilde{w} = 0$ it follows

$$\begin{aligned} \chi_{ij}(\Re(c_i\overline{v_{ij}}) - d_{ij}) &= \chi_{ij}\left(\Re(c_i\overline{v_{ij}}) - d_{ij} + \frac{1}{|C(w_i)|}\Re\left(z_i\overline{v_{ij}} \cdot \sum_{\tilde{w}\in C(w_i)} \tilde{w}\right)\right) \\ &= \frac{1}{|C(w_i)|} \sum_{\tilde{w}\in C(w_i)} \chi_{ij}\Re(((c_i + z_i\tilde{w})\overline{v_{ij}}) - d_{ij}) \\ &\overset{(1)}{\geq} \frac{1}{|C(w_i)|} \sum_{\tilde{w}\in C(w_i)} r_i = r_i \overset{(2)}{\geq} 0. \end{aligned}$$

If i is a circle or a stadium and non-degenerated, it is $r_i > 0$ and so in (2) strict inequality holds. Otherwise i is a proper rounded rectangle and in (1) holds strict inequality. In both cases, in the chain strict inequality holds and so the assertion follows. □

As a generalization of Theorem 7.14, we now show that the Mangasarian-Fromovitz constraint qualification (MFCQ) is satisfied at each feasible point of $RRPP$.

Theorem 8.9 (MFCQ is satisfied)**.** *Let none of the components be degenerated. Assume $\mathcal{C}^{\perp} \subset \mathcal{C}$, $\mathcal{C}^{fix} \subset \mathcal{C}$ with $\mathcal{C}^{\perp} \cap \mathcal{C}^{fix} = \emptyset$ and arbitrary $\boldsymbol{\varphi}^{fix} \in \mathbb{R}^n$. Then MFCQ is satisfied for each feasible point $\boldsymbol{x} \in D(\boldsymbol{w}, \boldsymbol{r}, \mathcal{C}^{\perp}, \mathcal{C}^{fix}, \boldsymbol{\varphi}^{fix})$.*

Proof. Let $\boldsymbol{x} = (\boldsymbol{c}, \boldsymbol{\varphi}, \boldsymbol{d}, \boldsymbol{\theta})$ be the vector of decision variables. Set $\boldsymbol{y} = (\boldsymbol{c}, 0, \boldsymbol{d}, 0)$. We show that the equality constraint gradients are linearly independent and for all equality constraints g_{eq} it holds $\langle \nabla g_{eq}, \boldsymbol{y} \rangle_{\mathbb{R}} = 0$ and for all active inequality constraints g_{in} it is $\langle \nabla g_{in}, \boldsymbol{y} \rangle_{\mathbb{R}} < 0$.

Denote (8.3) by g^1_{ij}, (8.4) by $g^2_{ij,\tilde{w}}$, (8.5) by g^3_j and (8.6) by g^4_j. The set of active constraints $I^1 := \{(i,j) : g^1_{ij}(\boldsymbol{x}) = 0\}$ and $I^2 := \{(i,j,\tilde{w}) : g^2_{ij,\tilde{w}}(\boldsymbol{x}) = 0\}$.

Regarding the factor $\frac{1}{2}$ by Remark 2.15 for real variables, for the non-zero derivatives of g^1, g^2 with respect to $\boldsymbol{c}$ and $\boldsymbol{d}$ we get:

$$\frac{\partial g^1_{ij}}{\partial \overline{c_i}} = c_j - c_i \qquad \frac{\partial g^1_{ij}}{\partial \overline{c_j}} = c_i - c_j$$

$$\frac{\partial g^2_{ij,\tilde{w}}}{\partial \overline{c_i}} = -\frac{\chi_{ij}}{2} \exp(\imath \theta_{ij}) \qquad \frac{\partial g^2_{ij,\tilde{w}}}{\partial d_{ij}} = \frac{\chi_{ij}}{2}$$

Equality constraints for different rotations are not related. As $\mathcal{C}^{fix} \cap \mathcal{C}^{\perp} = \emptyset$, the equality constraint gradients are linear dependent if and only if there is a $j \in \mathcal{C}^{\perp}$ such that $\nabla g^3_j = 0$ or a $j \in \mathcal{C}^{fix}$ such that $\nabla g^4_j = 0$. However, it is $\nabla g^4_j \neq 0$ everywhere. Furthermore, for $j \in \mathcal{C}^{\perp}$ we have $\sin(2\varphi_j) = 0$ and conclude

$$\frac{\partial g^3_j}{\partial \varphi_j} = \cos(2\varphi_j) = \pm 1 \implies \nabla g^3_j \neq 0.$$

Thus, for all feasible solutions the equality constraint gradients are linear independent. Furthermore, for the equality constraints it follows immediately $\left\langle \nabla g^3_j, \boldsymbol{y} \right\rangle_{\mathbb{R}} = 0$ and $\left\langle \nabla g^4_j, \boldsymbol{y} \right\rangle_{\mathbb{R}} = 0$ for each j.

We now consider the inequality constraints. Let $(i,j) \in I^1$. Then

$$\begin{aligned}\left\langle \nabla g^1_{ij}, \boldsymbol{y} \right\rangle_{\mathbb{R}} &= \Re((c_j - c_i)\overline{c_i} + (c_i - c_j)\overline{c_j}) \\ &= -\Re((c_j - c_i)(\overline{c_j} - \overline{c_i})) = -\left|c_i - c_j\right|^2 = -(r_1 + r_2)^2 < 0.\end{aligned}$$

Let $(i,j,\tilde{w}) \in I^2$. By Lemma 8.8 we conclude

$$\left\langle \nabla g^2_{ij,\tilde{w}}, \boldsymbol{y} \right\rangle_{\mathbb{R}} = -\frac{\chi_{ij}}{2} \Re(\overline{c_i} \exp(\imath \theta_{ij}) - d_{ij}) < 0.$$

Hence, MFCQ holds at each $\boldsymbol{x} \in D(\boldsymbol{w}, \boldsymbol{r}, \mathcal{C}^{\perp}, \mathcal{C}^{fix}, \boldsymbol{\varphi}^{fix})$. $\square$

Similar to Theorem 7.14 for the circle placement problem, there is an intuitive meaning of the vector $\boldsymbol{y}$. While the rotations $\boldsymbol{\varphi}$ of the components and $\boldsymbol{\theta}$ of the hyperplanes are fixed, the centers $\boldsymbol{c}$ and the hyperplane distances $\boldsymbol{d}$ are stretched. So actually the plane is stretched and the fixed sized and fixed rotated components are placed on their new positions. By this procedure, there arises a gap between previously touching components and hyperplanes. All active inequality constraints become inactive, i.e. MFCQ is satisfied. This is visualized in Figure 8.5.

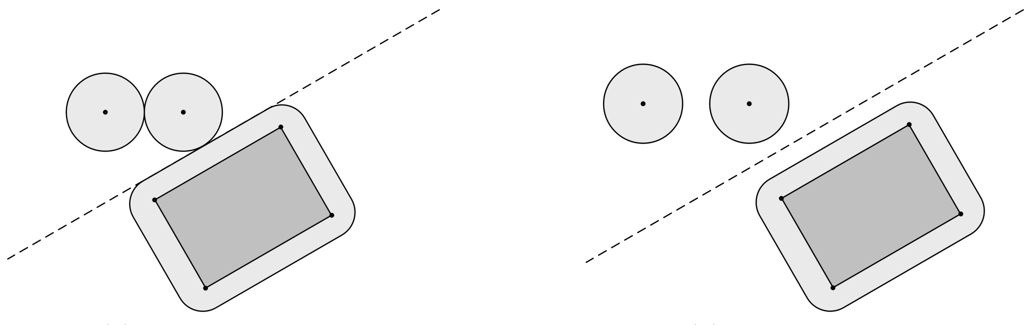

(a) Before stretching. (b) After stretching.

Figure 8.5: Geometric meaning of the $\boldsymbol{y}$ vector. The rotations remain unchanged and the locations are stretched away from each other.

Corollary 8.10. *There are feasible solutions to the rounded rectangle placement problem for which the linear independence constraint qualification (LICQ) is violated.*

Proof. If all components are modeled as circles, the rounded rectangle placement problem is equivalent to the circle placement problem CPP stated in (7.3). By Theorem 7.21 for CPP there are feasible solutions violating LICQ. Hence, there are feasible solutions to $RRPP$ violating LICQ. $\square$

In the vector $\boldsymbol{y}$ of Theorem 8.9 the rotations are kept fix. Later in this chapter the following question arises: If we rotate a component $i \notin \mathcal{C}^{\perp} \cup \mathcal{C}^{fix}$, can we stretch the components away from each other to stay feasible? Thus, we are now going to show that for each feasible point $(\boldsymbol{c}, \boldsymbol{\varphi}, \boldsymbol{d}, \boldsymbol{\theta})$ there is a feasible direction $(\boldsymbol{c}, \alpha \boldsymbol{e}_i, \boldsymbol{d}, 0)$ with some $\alpha > 0$ and $\boldsymbol{e}_i$ denoting the i-th unit vector. The factor α influences, how much the component i can be rotated for stretching the components away from each other.

We first proof the following extension of Lemma 8.8.

Lemma 8.11. *Let $(\boldsymbol{c}, \boldsymbol{\varphi}, \boldsymbol{d}, \boldsymbol{\theta}) \in D(\boldsymbol{w}, \boldsymbol{r}, \mathcal{C}^{\perp}, \mathcal{C}^{fix}, \boldsymbol{\varphi}^{fix})$. Let $(i,j) \in \mathcal{G}_1$.*

$$\chi_{ij} \left(\Re(\overline{c_i} \exp(\imath \theta_{ij})) - d_{ij} \right) \geq r_i + \Im(w_i).$$

Proof. We consider $\chi_{ij} = -1$, the other case is similar. The constraints enforce the placed rounded rectangle $c_i + \exp(\imath \varphi_i) R(r_i, w_i)$ to be contained in the negative half space of the hyperplane. As $\Re(w_i) \geq \Im(w_i)$, the hyperplane closest to c_i satisfying this condition has distance $r_i + \Im(w_i)$ from the center c_i. This implies the inequality. □

Lemma 8.12. *Let none of the components be degenerated. Assume $\mathcal{C}^\perp \subset \mathcal{C}$ and $\mathcal{C}^{fix} \subset \mathcal{C}$ with $\mathcal{C}^\perp \cap \mathcal{C}^{fix} = \emptyset$ and arbitrary $\boldsymbol{\varphi}^{fix} \in \mathbb{R}^n$. Let $\boldsymbol{x} := (\boldsymbol{c}, \boldsymbol{\varphi}, \boldsymbol{d}, \boldsymbol{\theta}) \in D(\boldsymbol{w}, \boldsymbol{r}, \mathcal{C}^\perp, \mathcal{C}^{fix}, \boldsymbol{\varphi}^{fix})$. Let $l \notin \mathcal{C}^\perp \cup \mathcal{C}^{fix}$.*

For each $\alpha \in \mathbb{R}$ with $|\alpha|\,|w_l| < r_l + \Im(w_l)$ the vector $\boldsymbol{y} = (\boldsymbol{c}, \alpha \boldsymbol{e}_l, \boldsymbol{d}, 0)$ is a feasible direction in $\boldsymbol{x}$. Furthermore, such an α exists and $\boldsymbol{x} + \delta \boldsymbol{y}$ is feasible for all $\delta \geq 0$.

Proof. We first show the existence of such an α. As component l is not degenerated, we have $r_l + \Im(w_l) > 0$ and, hence, an α with $|\alpha|\,|w_l| < r_l + \Im(w_l)$ exists.

Denote (8.3) by g^1_{ij}, (8.4) by $g^2_{ij,\tilde{w}}$, (8.5) by g^3_j and (8.6) by g^4_j. It can be seen that

$$\begin{aligned}
\forall (i,j) \in \mathcal{G}_1 :\quad & g^1_{ij}(\boldsymbol{x} + \delta \boldsymbol{y}) = (r_1 + r_2)^2 - (1+\delta)\,|c_i - c_j|^2 \leq -\delta\,|c_i - c_j|^2 \leq 0,\\
\forall j \in \mathcal{C}^\perp :\quad & g^j_3(\boldsymbol{x} + \delta \boldsymbol{y}) = g^j_3(\boldsymbol{x}) = 0,\\
\forall j \in \mathcal{C}^{fix} :\quad & g^j_4(\boldsymbol{x} + \delta \boldsymbol{y}) = g^j_4(\boldsymbol{x}) = 0.
\end{aligned}$$

It remains to show the statement for $g^2_{ij,\tilde{w}}$. For ease of notation set $r = r_i$, $c = c_i$, $d = d_{ij}$, $\theta = \theta_{ij}$ and $\chi = \chi_{ij}$. For $i \neq l$ we have by Lemma 8.11

$$\begin{aligned}
g^2_{ij,\tilde{w}}(\boldsymbol{x} + \delta \boldsymbol{y}) &= h^\chi_{r,\tilde{w}}((1+\delta)c, \varphi, (1+\delta)d, \theta)\\
&= r + \chi(d - \langle c + \tilde{w}\exp(\imath\varphi), \exp(\imath\theta)\rangle_{\mathbb{R}}) + \delta\chi(d - \langle c, \exp(\imath\theta)\rangle_{\mathbb{R}})\\
&\leq h^\chi_{r,\tilde{w}}(c, \varphi, d, \theta) - \delta(r_i + \Im(w_i)) \leq 0.
\end{aligned}$$

We now consider $i = l$. Then define

$$\begin{aligned}
h(\delta) :=& g^2_{ij,\tilde{w}}(\boldsymbol{x} + \delta \boldsymbol{y}) = h^\chi_{r,\tilde{w}}((1+\delta)c, \varphi + \alpha\delta, (1+\delta)d, \theta)\\
=& r + \chi(1+\delta)(d - \Re(\bar{c}\exp(\imath\theta))) - \chi\Re(\tilde{w}\exp(\imath(\varphi - \theta))\exp(\imath\alpha\delta))
\end{aligned}$$

We first show that $\frac{\partial h}{\partial \delta}(\delta) \leq 0$ for all $\delta > 0$. We compute the derivative with Wirtinger calculus and consider the factor $\frac{1}{2}$ for real derivatives by Remark 2.15. Then, applying Lemma 8.11 we get

$$\begin{aligned}
\frac{\partial h}{\partial \delta}(\delta) &= \frac{\chi}{2}(d - \Re(\bar{c}\exp(\imath\theta)) - \frac{\chi}{2}\alpha\Im(\tilde{w}\exp(\imath(\varphi - \theta))\exp(\imath\alpha\delta))\\
&\leq -\frac{1}{2}(r + \Im(\tilde{w})) + \frac{1}{2}\,|\alpha|\,|\tilde{w}| < 0.
\end{aligned}$$

Hence, the function h is decreasing for $\delta \geq 0$ and we conclude

$$g^2_{ij,\tilde{w}}(\boldsymbol{x} + \delta \boldsymbol{y}) = h(\delta) \leq h(0) = g^2_{ij,\tilde{w}}(\boldsymbol{x}) \leq 0.$$

□

8.4 Rounded Rectangle Algorithm

In this section we describe the rounded rectangle algorithm for the rectangle placement problem. This algorithm consists of solving a sequence of $RRPP$ instances. In the initial phase, all components are abstracted as circles, and we apply the methods described in Chapter 7 for the circle placement problem to generate a good placement. In the remaining part of the algorithm, the components are successively transformed from circles to rectangles.

8.4.1 Description of the Algorithm

For this transformation we exploit characteristics of electronic circuits. In electronic circuits there are components of very different size. In particular, there are some large components and many significantly smaller components. The placement of these larger components has strong impact on the final solution quality, so they should be placed good. The detailed placement of the smaller components is not required for the arrangement of the larger components. As the placement of the larger components is more important, the components are clustered by their size s_j into clusters $\mathcal{C}_t$, $t = 1, \dots, N$, such that components in the same cluster have similar size. Then the larger components are transformed from circles to rectangles first.

The algorithms runs through the refinement phases $t = 1, \dots, N$, where in the t-th phase:

- Components $i \in \mathcal{C}_{t'}$, $t' < t$, are encoded as rectangles with fixed rotation.
- Components $i \in \mathcal{C}_{t'}$, $t' > t$, are encoded as circles.
- Components $i \in \mathcal{C}_t$ are successively transformed from circles to rectangles. Simultaneously, the penalty for non-orthogonal rotation is successively increased.

In the remaining part of this section we formalize this algorithm.

Definition 8.13 (N-clustering)**.** *A N-clustering of the components is a partition of*

$$\mathcal{C} = \bigcup_{t=1}^{N} \mathcal{C}_t$$

into disjoint sets, such that for $i \in \mathcal{C}_t$ and $j \in \mathcal{C}_{t'}$ with $t < t'$ it is $|s_i| \leq |s_j|$.

We define the sets

$$\mathcal{C}_{<t} = \bigcup_{l<t} \mathcal{C}_l, \qquad \mathcal{C}_{>t} = \bigcup_{l>t} \mathcal{C}_l.$$

Definition 8.14. *Let $s \in \mathbb{C}'$. Then $R_\lambda(s)$ for $\lambda \in [0,1]$ is defined to be the rounded rectangle with half size λs and radius $(1-\lambda)\left|s\right|$, i.e.*

$$R_\lambda(s) := R((1-\lambda)\left|s\right|, \lambda s).$$

In particular, $R_1(s)$ is the rectangle with half size s and $R_0(s)$ is the enclosing circle of this rectangle.

We now state the problem that has to be solved within the steps of the rounded rectangle algorithm. In the t-th refinement phase, the components $i \in \mathcal{C}_{<t}$ are rectangles $R_1(s_i)$ and components $i \in \mathcal{C}_{>t}$ are circles $R_0(s_i)$, while components $i \in \mathcal{C}_t$ are rounded rectangles $R_\lambda(s_i)$ for some $\lambda \in [0,1]$. This yields $(\boldsymbol{r}, \boldsymbol{w})$ defined by

$$(r_i, w_i) = \begin{cases} (0, s_i) & i \in \mathcal{C}_{<t}, \\ ((1-\lambda)\left|s_i\right|, \lambda s_i) & i \in \mathcal{C}_t, \\ (\left|s_i\right|, 0) & i \in \mathcal{C}_{>t}. \end{cases}$$

The already fixed rotations are $\boldsymbol{\varphi}^{fix}$ and the penalty parameter is $\mu := \mu^{rot}$. Then the problem $P(t, \lambda, \mu, \boldsymbol{\varphi}^{fix})$ to solve is a rounded rectangle placement problem with $\mathcal{C}^{\perp} = \emptyset$, fixed components $\mathcal{C}^{fix} = \mathcal{C}_{<t}$. The components with rotation penalty are $\mathcal{C}^{rot} = \mathcal{C}_t$. Hence, it is

$$P(t, \lambda, \mu, \boldsymbol{\varphi}^{fix}) = RRPP(\boldsymbol{w}, \boldsymbol{r}, \emptyset, \mathcal{C}_{<t}, \boldsymbol{\varphi}^{fix}, \mathcal{C}_t, \mu). \tag{8.7}$$

This gives with $\mathcal{G}_0 = \{(i,j) : i,j \in \mathcal{C}_{>}t, i < j\}$ and $\mathcal{G}_1 := \{(i,j) : i \in \mathcal{C}_{\leq t} \vee j \in \mathcal{C}_{\leq t}, i \neq j\}$ for $P(t, \lambda, \mu, \boldsymbol{\varphi}^{fix})$ the formulation

$$\begin{aligned} \min \quad & \operatorname{wl}(\boldsymbol{c}, \boldsymbol{\varphi}) + \mu \sum_{i \in \mathcal{C}_t} \sin^2(2\varphi_i) && && (8.8) \\ \text{s. t.} \quad & (r_i + r_j)^2 - \left|c_i - c_j\right|^2 \leq 0, && \forall (i,j) \in \mathcal{G}_0 && (8.9) \\ & h^{\chi_{ij}}_{r_i, \tilde{w}}(c_i, \varphi_i, d_{ij}, \theta_{ij}) \leq 0, && \forall \tilde{w} \in C(w_i), \quad \forall (i,j) \in \mathcal{G}_1 && (8.10) \\ & \varphi_i - \varphi_i^{fix} = 0, && \forall i \in \mathcal{C}_{<t} && (8.11) \end{aligned}$$

The rounded rectangle algorithm stated in Algorithm 8 consists of solving a sequence of the rounded rectangle placement problems defined in (8.7).

We explain the steps of the algorithm here.

- In line 1 the components are clustered by their size.
- In line 2 the components are abstracted as circles and by the methods of Chapter 7 for the circle placement problem an initial placement is generated.
- In the initialization in line 3 no rotations are fixed and, as all components are circles, the hyperplanes are undefined. Thus all these values can be set arbitrary, e.g. to zero.

Algorithm 8: Rounded rectangle algorithm.

```
Data: Number of clusters $N$
1  $(\mathcal{C}_1, \dots \mathcal{C}_N) \leftarrow$ clustering of the components;
2  $(\boldsymbol{c}, \boldsymbol{\varphi}) \leftarrow$ solution of the circle placement problem;
3  $\boldsymbol{\varphi}^{fix} \leftarrow 0$; $\boldsymbol{d} \leftarrow 0$; $\boldsymbol{\theta} \leftarrow 0$;
4  for $t = 1$ to $N$ do
5      for $i \in \mathcal{C}_{t-1}$ do $\varphi_i^{fix} \leftarrow \varphi_i$;
6      for $(i, j)$ with $i < j$ do
7          if $i \in \mathcal{C}_t \vee j \in \mathcal{C}_t$ and not $i \in \mathcal{C}_{<t} \vee j \in \mathcal{C}_{<t}$ then
8              compute $\theta_{ij}$ and $d_{ij}$;
9      for $(\lambda, \mu)$ from $(0, 0)$ to $(1, \infty)$ do
10         $(\boldsymbol{c}, \boldsymbol{\varphi}, \boldsymbol{d}, \boldsymbol{\theta}) \leftarrow$ local optimum of $P(t, \lambda, \mu, \boldsymbol{\varphi}^{fix})$;
11 return $(\boldsymbol{c}, \boldsymbol{\varphi})$;
```

- In line 5 the rotations of the components that have been transformed to rectangles in the previous iteration are fixed. As fixed rotations are never changed in this algorithm, by this line during the algorithm the rotations of all components in $\mathcal{C}_{<t}$ are fixed.
- In line 6 to line 8 the new required hyperplanes are computed. The condition in line 7 is satisfied if and only if both components i and j have been circles in previous phases but at least one of them is no circle in this phase. For these pairs, the separating hyperplane has not been computed. To generate a feasible start point for the next solution, the hyperplane is computed in line 8 as shown in Lemma 8.15.
- In the loop in line 9 and line 10 the sequence of *RRPP* instances is solved. In this process, the rotation penalty μ is increased to ∞ and by increasing λ from 0 to 1 the components in $\mathcal{C}_t$ are transformed form circles to rectangles. It is not specified, how (λ, μ) increases to $(1, \infty)$. In our numerical evaluation, we use a fixed non-decreasing sequence of tuples. In Section 8.4.2 we show that each *RRPP* in line 10 is started from a feasible point.

The hyperplanes between two circles in line 8 are computed by the following lemma.

Lemma 8.15. *Let R_1 be the circle with radius r_1 and center c_1, and R_2 the circle with radius r_2 and center c_2. Set*

$$u = \frac{r_1 c_2 + r_2 c_1}{r_1 + r_2}, \qquad \theta = \arg(c_2 - c_1), \qquad d = \langle \exp(\imath\theta), u \rangle_{\mathbb{R}}, \qquad H = H(\theta, d).$$

Then the rectangles are contained in the different half spaces defined by H. More formally we have $R_1 \subset H_{\leq}$ and $R_2 \subset H_{\geq}$, or equivalently

$$h^{-1}_{r_1,0}(c_1, \varphi_1, d, \theta) \leq 0, \qquad h^{+1}_{r_2,0}(c_2, \varphi_2, d, \theta) \leq 0.$$

Proof. We only show the statement for h^{-1}, the proof for h^{+1} is similar. It is

$$\theta = \arg(c_2 - c_1) \implies \exp(-\imath\theta) = \frac{\overline{c_2 - c_1}}{|c_2 - c_1|}.$$

As the circles are non-overlapping, we know $|c_1 - c_2| \geq r_1 + r_2$. Then we get

$$\begin{aligned} h^{-1}_{r_1,0}(c_1, \varphi_1, d, \theta) &= r_1 - d + \Re(c_1 \exp(-\imath\theta)) \\ &= r_1 - \Re\left(\frac{r_1 c_2 + r_2 c_1}{r_1 + r_2} \cdot \frac{\overline{c_2 - c_1}}{|c_2 - c_1|}\right) + \Re\left(c_1 \cdot \frac{\overline{c_2 - c_1}}{|c_2 - c_1|}\right) \\ &= r_1 \left(1 - \frac{|c_2 - c_1|}{r_1 + r_2}\right) \leq 0. \end{aligned}$$

□

8.4.2 Each Starting Point is Feasible

We now show that in each optimization step in line 10 the initial solution is feasible to the problem. First in Lemma 8.16 we give a geometric intuition, as we show that for increasing λ the rounded rectangles R_λ are getting smaller. A rigorous proof is given in Theorem 8.18.

Lemma 8.16. *Let $s \in \mathbb{C}'$ and $0 \leq \lambda < \lambda' \leq 1$. Then $R_{\lambda'}(s) \subsetneq R_\lambda(s)$.*

Proof. By symmetry, w.l.o.g. it is sufficient to show the statement within the first quadrant of the Euclidean plane. This is visualized in Figure 8.6.

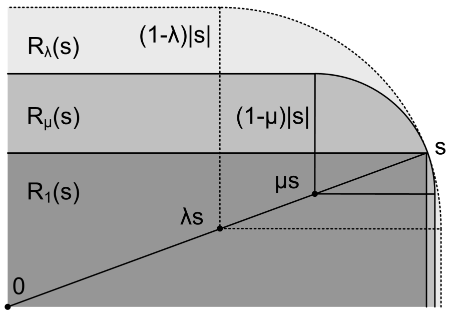


Figure 8.6: Containment of rounded rectangles $R_{\lambda'}(s)$ and $R_\lambda(s)$ for $0 \leq \lambda < \lambda' \leq 1$.

Except for the circular part, the strict containment is obvious. So, with $D(r, c)$ being the disc of radius r around c, it is sufficient to show that

$$D((1 - \lambda')\,|s|\,, \mu s) \subset D((1 - \lambda)\,|s|\,, \lambda s).$$

This statement follows by

$$\begin{aligned}
&x \in D((1-\lambda')\,|s|\,,\mu s)\\
&\implies |x-\lambda' s| \leq (1-\lambda')\,|s|\\
&\implies |x-\lambda s| \leq |x-\lambda' s| + |(\lambda'-\lambda)s| \leq (1-\lambda')\,|s| + (\lambda'-\lambda)\,|s| = (1-\lambda)\,|s|\\
&\implies x \in D((1-\lambda)\,|s|\,,\lambda s).
\end{aligned}$$

□

Theorem 8.18 shows that for fixed t within the loop in line 9 each solution is feasible to the next problem. Note that the rotation constraints remain unchanged. For the non-overlapping constraint this is intuitive by Lemma 8.16. Rectangles and circles remain equal. For $\lambda \leq \lambda'$ it holds $R_{\lambda'}(s_i) \subset R_\lambda(s_i)$ for each $i \in \mathcal{C}_t$, and if the larger components do not overlap, the smaller components do not overlap, too.

Lemma 8.17. *For $\chi \in \{-1,+1\}$ let h^χ be defined as in Lemma 8.4. Let $s \in \mathbb{C}$ being fixed,* $0 \leq \lambda \leq \lambda' \leq 1$ *and*

$$r = (1-\lambda)\,|s|\,, \quad \tilde{w} = \lambda s, \quad r' = (1-\lambda')\,|s|\,, \quad \tilde{w}' = \lambda' s.$$

Then for arbitrary $c \in \mathbb{C}$ and $\varphi, d, \theta \in \mathbb{R}$ it is

$$h^\chi_{r',\tilde{w}'}(c,\varphi,d,\theta) \leq h^\chi_{r,\tilde{w}}(c,\varphi,d,\theta).$$

Proof. We have

$$\begin{aligned}
&h^\chi_{r',\tilde{w}'}(c,\varphi,d,\theta)\\
&= \lambda'\Big[\chi\Re(s\exp(\imath(\varphi-\theta))) - |s|\Big] + |s| - d + \Re(c\exp(-\imath\theta))\\
&\leq \lambda\Big[\chi\Re(s\exp(\imath(\varphi-\theta))) - |s|\Big] + |s| - d + \Re(c\exp(-\imath\theta))\\
&h^\chi_{r,\tilde{w}}(c,\varphi,d,\theta).
\end{aligned}$$

□

Theorem 8.18. *Recall the notation of* (8.7)*. Let t and $\boldsymbol{\varphi}^{fix}$ be fixed, μ and μ' be arbitrary and $\lambda' \geq \lambda$. If $(\boldsymbol{c},\boldsymbol{\varphi},\boldsymbol{d},\boldsymbol{\theta})$ is feasible to $P(t,\lambda,\mu,\boldsymbol{\varphi}^{fix})$, then it is feasible to $P(t,\lambda',\mu',\boldsymbol{\varphi}^{fix})$.*

Proof. Let $(\boldsymbol{c},\boldsymbol{\varphi},\boldsymbol{d},\boldsymbol{\theta})$ be feasible to $P(t,\lambda,\mu,\boldsymbol{\varphi}^{fix})$. The rotation constraints (8.11) and the non-overlapping constraints (8.9) are the same for both problems. It remains to show the validity of (8.10) for $P(t,\lambda',\mu',\boldsymbol{\varphi}^{fix})$.

For circles $i \in \mathcal{C}_{>t}$ and rectangles $i \in \mathcal{C}_{<t}$ the constraint (8.10) is independent of λ and thus trivially satisfied. Therefore, assume $i \in \mathcal{C}_t$ being a rounded rectangle $R_{\lambda'}(s_i)$, i.e. $r'_i = (1-\lambda')\,|s_i|$ and $w'_i = \lambda s_i$. We have to show that for all $\tilde{w}' \in C(w'_i)$ it holds

$$h^{\chi_{ij}}_{r'_i,\tilde{w}'}(c_i,\varphi_i,d_{ij},\theta_{ij}) \leq 0.$$

We assume w.l.o.g. that $\tilde{w}' = w_i' = \lambda' s_i$, the other cases are similar. As the solution $(\boldsymbol{c}, \boldsymbol{\varphi}, \boldsymbol{d}, \boldsymbol{\theta})$ is feasible to $P(t, \lambda, \mu, \boldsymbol{\varphi}^{fix})$, we know for all $\tilde{w} \in C(w_i)$ and especially for $\tilde{w} = w_i = \lambda s_i$ that

$$h^{\chi_{ij}}_{r_i,\tilde{w}}(c_i, \varphi_i, d_{ij}, \theta_{ij}) \leq 0.$$

Applying Lemma 8.17 we conclude

$$h^{\chi_{ij}}_{r_i',\tilde{w}'}(c_i, \varphi_i, d_{ij}, \theta_{ij}) \leq h^{\chi_{ij}}_{r_i,\tilde{w}}(c_i, \varphi_i, d_{ij}, \theta_{ij}) \leq 0.$$

□

Now we know that for fixed t within the loop in line 9 each solution is feasible to the next problem. By Lemma 8.15 we can also compute the hyperplanes in line 8 such that the constraint (8.10) is satisfied for non-overlapping circles. Thus we know that in the first run for a cluster t the initial solution is feasible. Hence, in each iteration we start with a feasible solution.

8.4.3 Orthogonal Rotation is Achieved

It remains to show that the algorithm converges to a feasible solution of the rectangle placement problem. In particular, it remains to show that the components are orthogonally rotated, i.e. that for $\mu \to \infty$ we have $\sin(2\varphi_i) \to 0$.

Therefore, we assume t and $\boldsymbol{\varphi}^{fix}$ to be fixed and denote $\boldsymbol{x} := (\boldsymbol{c}, \boldsymbol{\varphi}, \boldsymbol{d}, \boldsymbol{\theta})$. Let X_λ be the set of feasible points of $P(t, \lambda, \mu, \boldsymbol{\varphi}^{fix})$. Then we define

$$\begin{array}{ll} \min & \mathrm{wl}(\boldsymbol{c}, \boldsymbol{\varphi}) \\ \text{s. t.} & \sin(2\varphi_i) = 0, \qquad \forall i \in \mathcal{C}_t \\ & \boldsymbol{x} \in X_\lambda \end{array} \tag{$P(\lambda)$}$$

and furthermore $P(\lambda, \mu) := P(t, \lambda, \mu, \boldsymbol{\varphi}^{fix})$, i.e.

$$\begin{array}{ll} \min & \mathrm{wl}(\boldsymbol{c}, \boldsymbol{\varphi}) + \mu \sum_{i \in \mathcal{C}_t} \sin^2(2\varphi_i) \\ \text{s. t.} & \boldsymbol{x} \in X_\lambda \end{array} \tag{$P(\lambda,\mu)$}$$

It can be seen that $(P(\lambda, \mu))$ is the quadratic penalty formulation of $(P(\lambda))$. We summarized the quadratic penalty approach for constrained non-linear programming in Section 2.5. In this section we transfer the results to $(P(\lambda, \mu))$.

For technical proof-related reasons we first have to show that the problem can be restricted to a compact subset.

Definition 8.19 (Touching Path, Touching Placement)**.** *Let i and j be two components. In a given placement, a touching path from i to j is a list $(l_1, \dots, l_m)$ of components such that $i = l_1$, $j = l_m$ and the components l_k and l_{k+1} are touching each other.*

A placement is touching, if for each pair of components there is a touching path between them.

Lemma 8.20. *There is a global optimal placement for $P(\lambda)$ that is touching. The same holds true for $P(\lambda, \mu)$.*

In particular, there is a compact set Y independent of λ and μ, which is guaranteed to contain a global minimizer of $P(\lambda)$ and of $P(\lambda, \mu)$.

Proof. If a set of components is connected by nets, they are touching at the optimum. Different sets of connected components can be moved together without changing the objective value. Hence, there is a touching global optimal placement.

Now let $\boldsymbol{x}^*$ be a touching placement and a global minimizer of $P(\lambda)$ (the case $P(\lambda, \mu)$ is similar). Then component i has circumcircle radius $|s_i|$. In particular, all components are in a square of edge length $s := 2\sum_{i \in \mathcal{C}} |s_i|$. Thus, the placement can be translated such that

$$\begin{aligned} \forall i \in \mathcal{C}: &\quad c_i \in [0, s] + \imath[0, s], \\ \forall i < j \in \mathcal{C}: &\quad d_{ij} \in [0, \sqrt{2}s]. \end{aligned}$$

Furthermore, all angles can be restricted to $[0, 2\pi]$. Hence, with $m = n(n-1)/2$ there is global minimizer

$$(\boldsymbol{c}^*, \boldsymbol{\varphi}^*, \boldsymbol{d}^*, \boldsymbol{\theta}^*) \in ([0, s] + \imath[0, s])^n \times [0, 2\pi]^n \times [0, \sqrt{2}s]^m \times [0, 2\pi]^m =: Y.$$

In particular, the set Y is compact. □

Theorem 8.21. *Let $\lambda_k \in [0, 1]$ and μ_k for $k \in \mathbb{N}$ be sequences with $\lim_{k\to\infty} \mu_k = \infty$ and $\lambda_k = 1$ for $k \geq k_0$ for some k_0. Furthermore, let $\boldsymbol{x}_k$ be the global minimizer of $P(\lambda_k, \mu_k)$. Then any convergent subsequence of $\boldsymbol{x}_k$ converges to a global minimizer of $P(1)$. In particular, such limit points of $\boldsymbol{x}_k$ exist.*

Proof. Let $k \geq k_0$. Then the set $X_\lambda = X_1$ remains fixed and the statement follows immediately from Theorem 2.33. Furthermore, by Lemma 8.20 there is a compact set Y such that we can choose a global minimizer $\boldsymbol{x}_k \in Y$ to $P(\lambda_k, \mu_k)$. By the Bolzano-Weierstrass theorem, the sequence $\boldsymbol{x}_k$ has a convergent subsequence. □

Unfortunately, this result is of little practical value, as finding the global minimizer of $P(\lambda_k, \mu_k)$ is almost as difficult as solving the initial rectangle placement problem to global optimality. For this reason, it would be appealing to transfer the local convergence result of Theorem 2.34, where the sequence points $\boldsymbol{x}_k$ only have to be a stationary points. Unfortunately, this result requires $X = \mathbb{R}^n$. However, we proof a similar result for the convergence of local minima of $P(\lambda_k, \mu_k)$ in Theorem 8.22.

Theorem 8.22. *Assume that* $\Im(s_i) > 0$ *for all* $i \in \mathcal{C}$. *Let* t *and* $\boldsymbol{\varphi}^{fix}$ *be fixed. Let* $\lambda_k \in [0, 1]$ *and* μ_k *for* $k \in \mathbb{N}$ *be a sequence with* $\lim_{k\to\infty} \lambda_k = 1$ *and* $\lim_{k\to\infty} \mu_k = \infty$.

Let Y *be the compact set defined in Lemma 8.20 and* $\boldsymbol{x}_k = (\boldsymbol{c}_k, \boldsymbol{\varphi}_k, \boldsymbol{d}_k, \boldsymbol{\theta}_k) \in Y$ *be a local minimizer of* $P(\lambda_k, \mu_k)$. *Let* $\boldsymbol{x}^* = (\boldsymbol{c}^*, \boldsymbol{\varphi}^*, \boldsymbol{d}^*, \boldsymbol{\theta}^*)$ *be a limit point of* $\boldsymbol{x}_k$.

Then for all $i \in \mathcal{C}_t$ *it holds* $\cos(\varphi_i^*) = 0$ *or* $\sin(\varphi_i^*) = 0$.

Proof. Let $\nu_i = 1$ if $i \in \mathcal{C}_t$ and $\nu_i = 0$ otherwise. The objective of $(P(\lambda, \mu))$ is

$$f_\mu(\boldsymbol{c}, \boldsymbol{\varphi}) := \mathrm{wl}(\boldsymbol{c}, \boldsymbol{\varphi}) + \mu \,\mathrm{pen}(\boldsymbol{\varphi}) \quad \text{with} \quad \mathrm{pen}(\boldsymbol{\varphi}) = \sum_{i\in\mathcal{C}} \nu_i \sin^2(2\varphi_i).$$

The gradient of wl is continuous and thus bounded on the compact set Y, i.e. there is a K_1 such that for $\boldsymbol{x} \in Y$ it is $||\nabla \mathrm{wl}(\boldsymbol{x})||_\infty \leq K_1$. Furthermore, there is a K_2 such that for all $\boldsymbol{x} \in Y$ we have $||\boldsymbol{x}||_\infty \leq K_2$.

For each λ the size of the rounded rectangle $i \in \mathcal{C}_t$ is defined by $r_i = (1 - \lambda) |s_i|$ and $w_i = \lambda s_i$. By Lemma 8.12 for $\boldsymbol{x} \in Y$ and $|\alpha| < \frac{\Im(s_i)}{|s_i|}$ the vector $\boldsymbol{y}_i(\boldsymbol{x}, \alpha) = (\boldsymbol{c}, \alpha \boldsymbol{e}_i, \boldsymbol{d}, 0)$ is a feasible direction at $\boldsymbol{x}$ in X_λ, as

$$\begin{aligned} |\alpha| \, |w_i| = |\alpha| \, \lambda \, |s_i| < |\alpha| \, |s_i| < \Im(s_i) &\leq (1 - \lambda)\Im(s_i) + \lambda\Im(s_i) \\ &\leq (1 - \lambda) \, |s_i| + \lambda\Im(s_i) = r_i + \Im(w_i). \end{aligned}$$

As $|\alpha| < 1/\sqrt{2} \leq 2\pi \leq K_2$, for such a $\boldsymbol{y} = \boldsymbol{y}_i(\boldsymbol{x}, \alpha)$ we have

$$||\boldsymbol{y}||_\infty \leq \max(||\boldsymbol{x}||_\infty, |\alpha|) \leq K_2 \implies \left|\boldsymbol{y}^T \nabla \mathrm{wl}(\boldsymbol{x})\right| \leq ||\nabla \mathrm{wl}(\boldsymbol{x})||_\infty \, ||\boldsymbol{y}||_\infty \leq K_1 K_2.$$

We take a subsequence of $\boldsymbol{x}_k$ converging to $\boldsymbol{x}^*$ and denote this by $\boldsymbol{x}_k$ again for ease of notation. Set $\alpha = \frac{\Im(s_l)}{2|s_l|}$ and $\boldsymbol{y}_{ik+} = \boldsymbol{y}_i(\boldsymbol{x}_k, +\alpha)$ and $\boldsymbol{y}_{ik-} = \boldsymbol{y}_i(\boldsymbol{x}_k, -\alpha)$. For all k the directions $\boldsymbol{y}_{ik+}$ and $\boldsymbol{y}_{ik-}$ are feasible. As $\boldsymbol{x}_k$ is a minimum, there does not exist a descent direction and we conclude

$$\begin{aligned} 0 \leq \boldsymbol{y}_{ik+}^T \nabla f_{\mu_k}(\boldsymbol{x}_k) \leq K_1 K_2 + \mu_k \alpha \boldsymbol{e}_i^T \nabla_\varphi \mathrm{pen}(\boldsymbol{x}_k), \\ 0 \leq \boldsymbol{y}_{ik-}^T \nabla f_{\mu_k}(\boldsymbol{x}_k) \leq K_1 K_2 - \mu_k \alpha \boldsymbol{e}_i^T \nabla_\varphi \mathrm{pen}(\boldsymbol{x}_k), \end{aligned}$$

and subsequently

$$\begin{aligned} K_1 K_2 \geq \left|\mu_k \alpha \boldsymbol{e}_i^T \nabla_\varphi \mathrm{pen}(\boldsymbol{x}_k)\right| &= 4\mu_k\alpha \left|\sin(2\varphi_i^k) \cos(2\varphi_i^k)\right| \\ \implies 0 \leq \left|\sin(2\varphi_i^k) \cos(2\varphi_i^k)\right| &\leq \frac{K_1 K_2}{4\mu_k \alpha}. \end{aligned}$$

As the right term converges to 0 for $\mu_k \to \infty$, we have $\sin(2\varphi_i^*) \cos(2\varphi_i^*) = 0$. $\square$

Note that the algorithm might converge to a point, where the components are rotated e.g. by $\pi/4$. This is similar to the case Theorem 2.34, where the penalty approach also can converge to a stationary point of the constraint violation. However, it seems unlikely that the algorithm converges to such a point, as this is a local maximum of the constraint violation. Thus, for increasing penalty parameter we expect the algorithm to overcome these local optima. And indeed, there is an extension of the algorithm that assures $\boldsymbol{x}^*$ to be feasible to $P(1)$.

Corollary 8.23. *In the settings of Theorem 8.22 denote by $\boldsymbol{y}_k^i$ the feasible direction defined in $\boldsymbol{x}_k$ for $i \in \mathcal{C}_t$. Take a fixed $0 < \delta \leq \frac{\pi}{4}$. If for each k*

$$f_{\mu_k}(\boldsymbol{x}_k) \leq f_{\mu_k}(\boldsymbol{x}_k + \delta \boldsymbol{y}_k^i), \quad \forall i \in \mathcal{C}_t,$$

then each limit point $\boldsymbol{x}^*$ of $\boldsymbol{x}_k$ is feasible to $P(1)$.

Proof. By Lemma 8.12 for the feasible direction $\boldsymbol{y}_k^i$ in $\boldsymbol{x}$ the point $\boldsymbol{x} + \delta \boldsymbol{y}_k^i$ is feasible for each $\delta \geq 0$. Take a subsequence of $\boldsymbol{x}_k$ converging to $\boldsymbol{x}^*$ and denote it by $\boldsymbol{x}_k$ again.

Then we have $\boldsymbol{y}_k^i \to \boldsymbol{y}^i := (\boldsymbol{c}^*, \alpha \boldsymbol{e}_i, \boldsymbol{d}^*, 0)$. Assume that $\cos(2\varphi_l^*) = 0$ for some $l \in \mathcal{C}_t$. Then $\sin^2(2\varphi_l^*) = 1$ and we conclude

$$\begin{aligned}
&\lim_{k\to\infty} \frac{1}{\mu_k} \Big(f_{\mu_k}(\boldsymbol{x}_k) - f_{\mu_k}(\boldsymbol{x}_k + \delta \boldsymbol{y}_k^l) \Big) \\
&= \lim_{k\to\infty} \frac{1}{\mu_k} \Big(\mathrm{wl}(\boldsymbol{x}_k) - \mathrm{wl}(\boldsymbol{x}_k + \delta \boldsymbol{y}_k^l) \Big) + \sin^2(2\varphi_l^k) - \sin^2(2(\varphi_l^k + \alpha\delta)) \\
&= 1 - \sin^2(2\varphi_l^* + 2\alpha\delta) > 0.
\end{aligned}$$

In particular, for large k we have

$$f_{\mu_k}(\boldsymbol{x}_k + \delta \boldsymbol{y}_k^l) < f_{\mu_k}(\boldsymbol{x}_k),$$

which contradicts the assumption.

Hence, $\cos(2\varphi_i^*) \neq 0$ for all $i \in \mathcal{C}$ and by Theorem 8.22 we conclude $\sin(2\varphi_i^*) = 0$ and the feasibility of $\boldsymbol{x}^*$. □

Corollary 8.23 suggests the following modification of an optimization step in the rounded rectangle algorithm. Take some small fixed $0 < \delta < \frac{\pi}{4}$. Assume a local optimum $\boldsymbol{x}_k$ of a problem $(P(\lambda, \mu))$ is found. If for some $i \in \mathcal{C}_t$ we have

$$f_{\mu_k}(\boldsymbol{x}_k + \delta \boldsymbol{y}_k^i) < f_{\mu_k}(\boldsymbol{x}_k),$$

restart a local optimization at $\boldsymbol{x}_k + \delta \boldsymbol{y}_k^i$. If the local optimization is a feasible descent method, this algorithm is guaranteed to converge to a feasible solution of the rectangle placement problem.

However, this problem is irrelevant in practice. The algorithm only might converge to some point with $\cos 2\varphi_l^* = 0$ if it stays in this stationary point in all iterations. Due to limited numerical accuracy of computers, this does not happen in practice.

8.5 Numerical Results

In this section we evaluate the rounded rectangle algorithm stated in Algorithm 8. In Section 8.5.1 we described details of the implemented algorithms and the measured values. In Section 8.5.2 we analyze the overall performance of the algorithm in terms of running time and wire length. In Section 8.5.3 the different phases of the algorithms are compared. In Section 8.5.4 for different circuit instances we present the placement after different phases of the algorithm.

8.5.1 Applied Methods and Measurements

For each of the instances described in Section 2.8 with up to 243 components we ran the algorithm for 100 random initial solutions.

The components were clustered into 2 clusters as follows. To each component $i \in \mathcal{C}$ the scalar value $\ln(|s_i|)$ is assigned. Then iteratively the clusters with closest distance were unified, until there were two remaining clusters. Table 8.1 on page 192 shows the number of components per cluster and their average size.

The initial placement generation was done by the configuration ECSLS / MBH 5 described in Section 7.7.2, we briefly summarize this setting here. Starting from a random initial solution we apply the global invariant attractor repeller model with repeller factor $\alpha = 1$ and feasibility stretch as described in Section 7.4. This solution is optimized by the constrained non-linear program CPP described in Section 7.5. Subsequently, we performed equal circle swap local search and monotonic basin hopping with maximum 5 fails as described in Section 7.6. The final solution of this circle placement algorithm is taken as the initial solution for the refinement phase of the rounded rectangle algorithm.

The sequence (μ, λ) in line 9 of the algorithm was chosen as

$$(\mu, \lambda) = \{(0; 0.0),\ (0; 0.1),\ (1; 0.3),\ (10; 0.95),\ (\infty; 1.0)\}, \tag{8.12}$$

where for $\mu = \infty$ the orthogonal rotation is removed from the objective and added as constraint to Ipopt. The run $(0; 0.0)$ is introduced for technical reasons: Geometrically this problem is equivalent to the last iteration of the previous phase, hence we start this run at a local optimum and Ipopt can compute the dual variables very quickly. In the remaining optimization runs for this cluster, we make a warm start with initialized duals. In the figures we include this run in the run $(\mu, \lambda) = (0; 0.1)$.

We configured Ipopt with the settings $tol = 10^{-3}$ and $acceptable\_tol = 5 \cdot 10^{-3}$ and $max\_iter = 30000$, which was never reached. As Ipopt is a filter method, it allows the solution to become infeasible to reduce the objective value. In the sequence of non-linear programs, we start close to the local optimum and reduce the amount of allowed infeasibility by setting $theta\_max\_fact = 10$. For details we refer to

Circuit	$\lvert\mathcal{C}_1\rvert$	$\lvert\mathcal{C}_2\rvert$	$\text{avg}_{i\in\mathcal{C}_1} \lvert s_i\rvert$	$\text{avg}_{i\in\mathcal{C}_2} \lvert s_i\rvert$
D0019	14	5	5.617	0.464
D0034	15	19	5.346	0.464
D0059	17	42	4.906	0.464
D0078	19	59	4.559	0.464
D0104	19	85	4.559	0.464
D0138	14	124	5.617	0.513
D0183	14	169	5.617	0.501
D0243	14	229	5.617	0.506
E0031	5	26	2.193	0.484
S0041	11	30	2.768	0.352
V0093	6	87	3.948	0.537
M0057	5	52	1.917	0.645

Table 8.1: Clustering of the circuits. In the second and third column, the number of components in each clusters is shown. In the fourth and fifth column the average size of the components per cluster is stated.

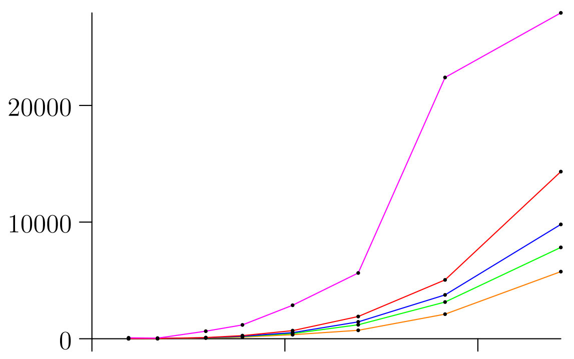
[Wächter; Biegler 2006]. The rare cases of convergence to infeasible solutions could be avoided by using a solver which stays feasible in every iteration, therefore these runs are removed from the evaluation. Additionally, we started with a small constraint penalty parameter *mu_init* = 0.001 and performed a warm start if the problem structure has not changed, i.e. the cluster t remained the same and rotation was considered as quadratic penalty term.

After each step we measured the CPU time and the wire length without the orthogonal rotation penalty. In particular, we use the terms objective and wire length synonymously, even for runs with rotation penalty. As the circle placement algorithm has been studied in Chapter 7, we consider it as a single step here and do not analyze its different phases.

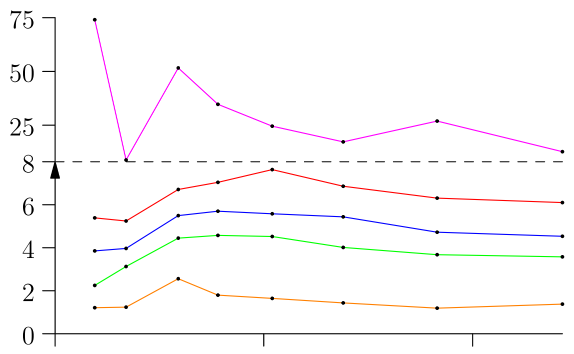
8.5.2 Overall Performance of the Algorithm

The overall performance of the algorithm is shown in Figure 8.7, the solution quality in Figure 8.7c and Table 8.2 on page 193. We do not have a lower bound and compare the runs against the best known result for this instance. It turns out that the algorithm is robust. While the worst case behavior might be poor, even for larger instances most of the times the objective is less than 40% away from the best known solution.

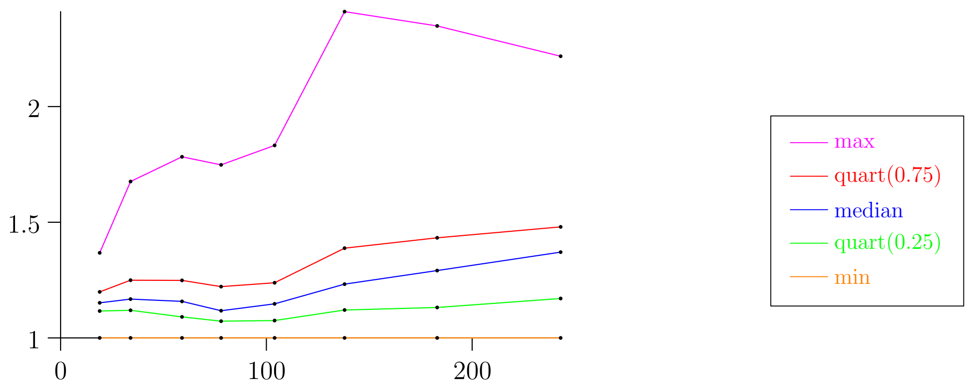
In Figure 8.7a the absolute CPU time in seconds is shown, in Figure 8.7b the CPU time divided by the CPU time of the initial solution generation. As expected, the absolute CPU time increases with the number of circles. In Figure 8.7b it can be seen that, except for some outliers, the algorithm is robust and usually takes less than 8 times as long as for the initial solution generation.



(a) CPU(s) / #circles.



(b) CPU ratio / #circles.



(c) Wire length / #circles.

Figure 8.7: In (a): quartiles of the CPU time in seconds versus the number of circles. In (b): quartiles of the ratio of the final CPU time and the CPU time of the initial solution. In (c): quartiles of the ratio of the final objective divided by the best known objective for this instance.

Instance	min	quart(0.25%)	median	quart(0.75%)	max
E0031	1.0000	1.0749	1.1375	1.2140	1.7987
S0041	1.0000	1.1550	1.2081	1.2597	1.5455
M0057	1.0000	1.1676	1.2899	1.3772	1.7612
V0093	1.0000	1.0340	1.0680	1.1207	2.5133

Table 8.2: Quartiles of the ratio of the final objective divided by the best known objective for this instance.

8.5.3 Analysis of the Different Phases

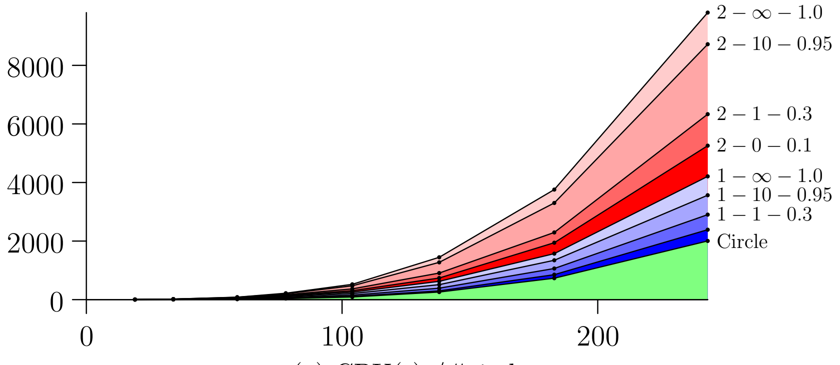
In Figure 8.8 the behavior of the algorithm for increasing number of circles is shown. The different graphs are labeled by a triple (t, μ, λ) where t is the cluster currently refined from circles to rectangles and μ and λ as in (8.12).

Figure 8.8a shows the median total CPU time in seconds versus the number of circles. The initial circle placement is colored green, the runs for the first cluster blue and for the second cluster red. In Figure 8.8b the CPU time is visualized, too. However, there we divided the CPU time of each phase by the CPU time required for the initial circle placement. In the graphics the median of these values is shown. Figure 8.8c visualizes the median wire length after each phase. Also not strictly necessary, the wire length reduces in each step, i.e. the improvement by refinement of the components to rectangles compensates for the more orthogonal rotation.

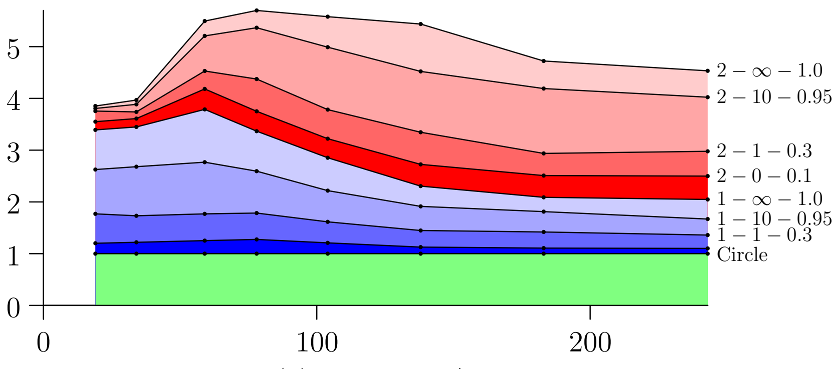
In Figure 8.8a we recognize the expected increase in running time for larger problem instances. This seams to be proportional to the increase for the initial circle placement solution. Indeed in Figure 8.8b we observe that, except for small instances, the ratio of the times spent in different phases of the algorithm remains approximately equal. In particular, in the refinement of the components of cluster C_2 approximately half of the total running time of the algorithm is spent.

Figure 8.8c shows that after run $1 - 10 - 0.95$ where the components of C_1 are almost refined to rectangles, the wire length is not significantly reduced. Additionally, in Figure 8.9 the relation of the solutions of different phases is displayed for each run. Figure 8.9a shows that the objectives for the initial circle placement and after the refinement of cluster 1 is related but still contains some diversification. In contrast, in Figure 8.9b is can be seen that the final objective and the objective after refinement of the components of the first cluster is strongly correlated.

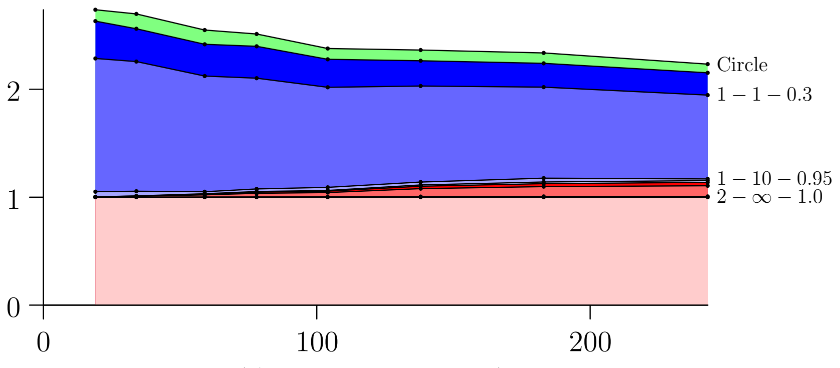
Hence, in the later phases the algorithm spends much time for little improvement in the objective and less important decisions. This is due to the large number of components in the second cluster and the significant increase of the number of variables and constraints to model them as rounded rectangles. In Table 8.3 on page 196 the size of the non-linear programs for the different phases is shown. In particular, the increase of the number of variables and constraints for the refinement of components of $\mathcal{C}_2$ can be seen.



(a) CPU(s) /#circles.

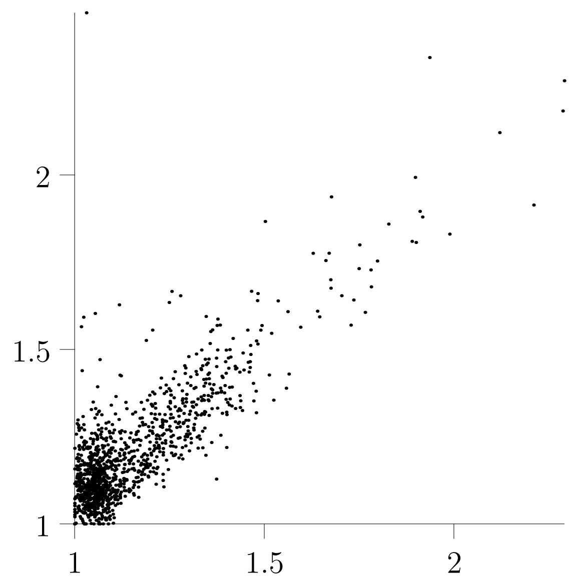


(b) CPU ratio /#circles.

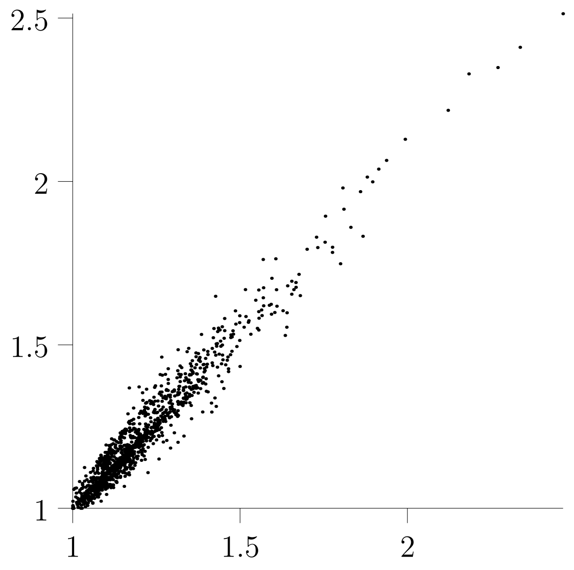


(c) Wire length ratio /#circles.

Figure 8.8: On the x-axis the number of circles is displayed. On the y-axis is shown: in (a) the median total CPU time in seconds; in (b) then median of the ratio of the total CPU time and the CPU time of the initial solution generation; in (c) the median ratio of the wire length after the run and the final wire length. The labels on the right are $t - \mu^{rot} - \lambda$.



(a) $1 - \infty - 1.0$ vs. Circle.



(b) $2 - \infty - 1.0$ vs. $1 - \infty - 1.0$.

Figure 8.9: Comparison of the objective ratios, i.e. the ratio of the solution objective divided by the best known objective for this instance and this run. In (a): objective ratio $1-\infty-1.0$ (on y-axis) versus objective ratio of initial circle placement. In (b): objective ratio $2 - \infty - 1.0$ (on y-axis) versus objective ratio of $1 - \infty - 1.0$ (on x-axis).

			Circles		$t = 1$		$t = 2$	
Circuit	$\|\mathcal{C}_1\|$	$\|\mathcal{C}_2\|$	#cons	#var	#cons	#var	#cons	#var
D0019	14	5	171	57	1088	379	1368	399
D0034	15	19	561	102	2436	882	4488	1224
D0059	17	42	1711	177	5519	1877	13688	3599
D0078	19	59	3003	234	8684	2818	24024	6240
D0104	19	85	5356	312	13013	3884	42848	11024
D0138	14	124	9453	414	17034	4068	75624	19320
D0183	14	169	16653	549	26754	5463	133224	33855
D0243	14	229	29403	729	42864	7323	235224	59535
E0031	5	26	465	93	1055	373	3720	1023
S0041	11	30	820	123	2525	893	6560	1763
V0093	6	87	4278	279	6471	1353	34224	8835
M0057	5	52	1596	171	2706	711	12768	3363

Table 8.3: Number of variables and number of constraints for the different phases of the algorithm.

8.5.4 Placement Examples after the Different Phases

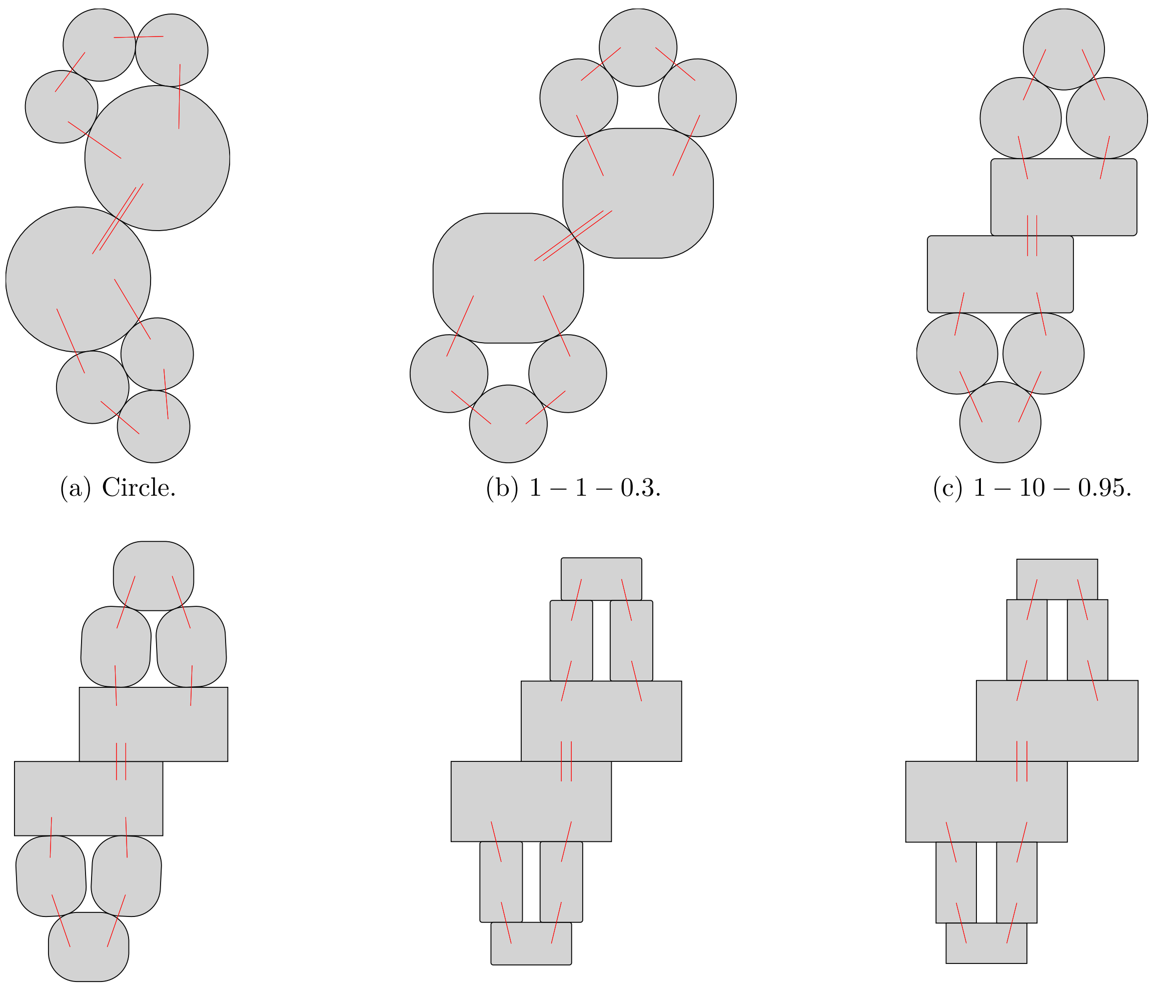
The Figures 8.10, 8.11 and 8.12 show the main steps of the algorithm after different phases. It can be seen, how the structure of the initial circle placement is preserved within the refinement steps. Furthermore, the refinement of the components $\mathcal{C}_2$ has little influence on the topology of the placement.

8.5.5 Conclusion

In this section we evaluated the rounded rectangle algorithm numerically. We showed that it can handle problem instances with up to 250 components and yield good solution quality in acceptable running time.

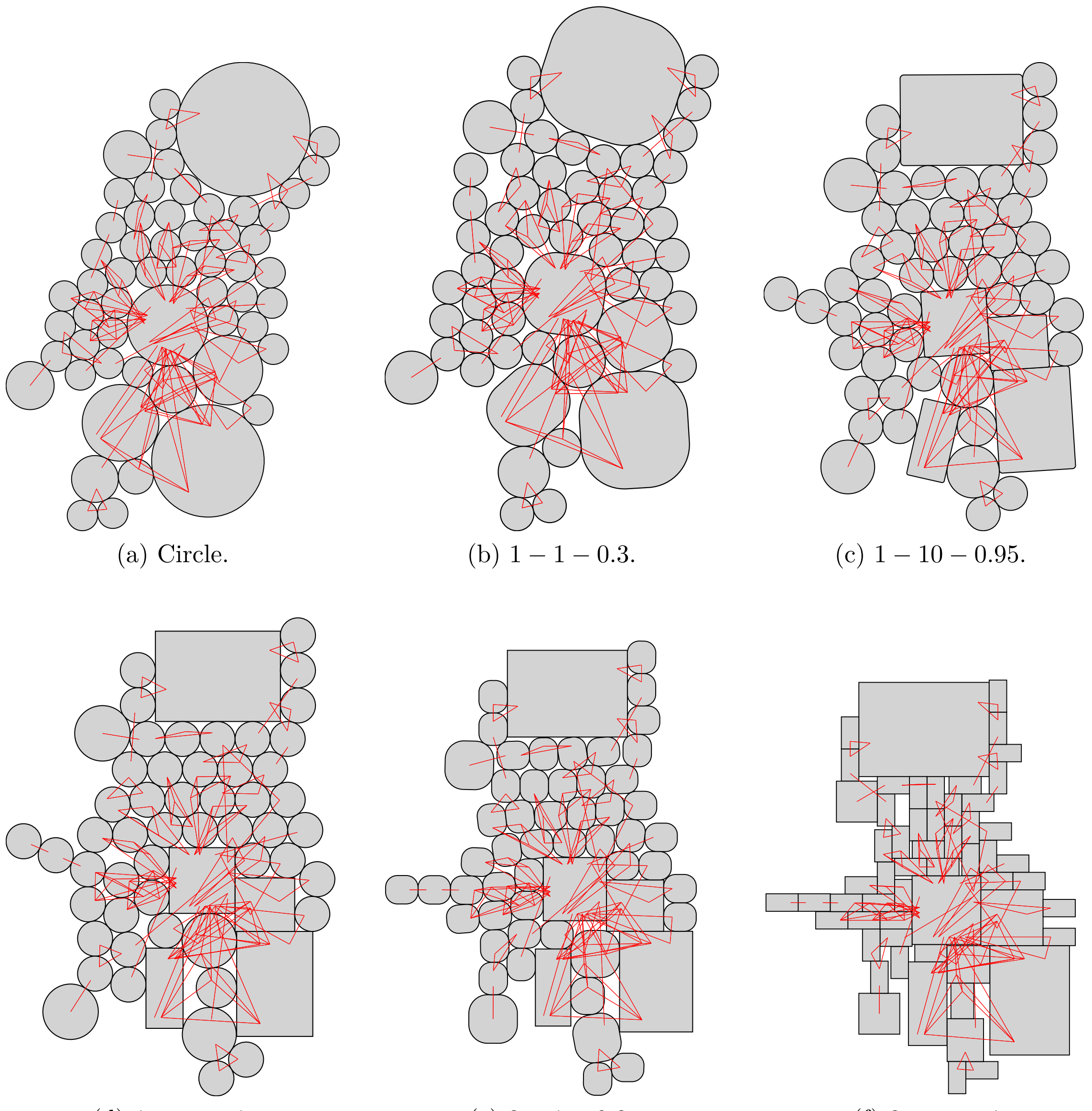
We showed, that there is a correlation between the quality of the initial circle placement and the quality of the final solution. In particular, we demonstrated that the algorithm spends a large amount of running time in the refinement of the smaller components, where there is little improvement in the wire length and the solution structure essentially remains unchanged.

We did not evaluate multiple shooting approaches, as we focused on the circle to rectangle refinement phase in this chapter. However, the running time of the initial circle placement is short compared to the overall running time. Hence, for practical algorithms it might be advantageously, to generate several circle placements and start the refinement from the best of these placements.


(a) Circle. (b) $1 - 1 - 0.3$. (c) $1 - 10 - 0.95$.

(d) $2 - 1 - 0.3$. (e) $2 - 10 - 0.95$. (f) $2 - \infty - 1$.

Figure 8.10: Steps of the rounded rectangle algorithm for an artificial instance.


(a) Circle.

(b) $1 - 1 - 0.3$.

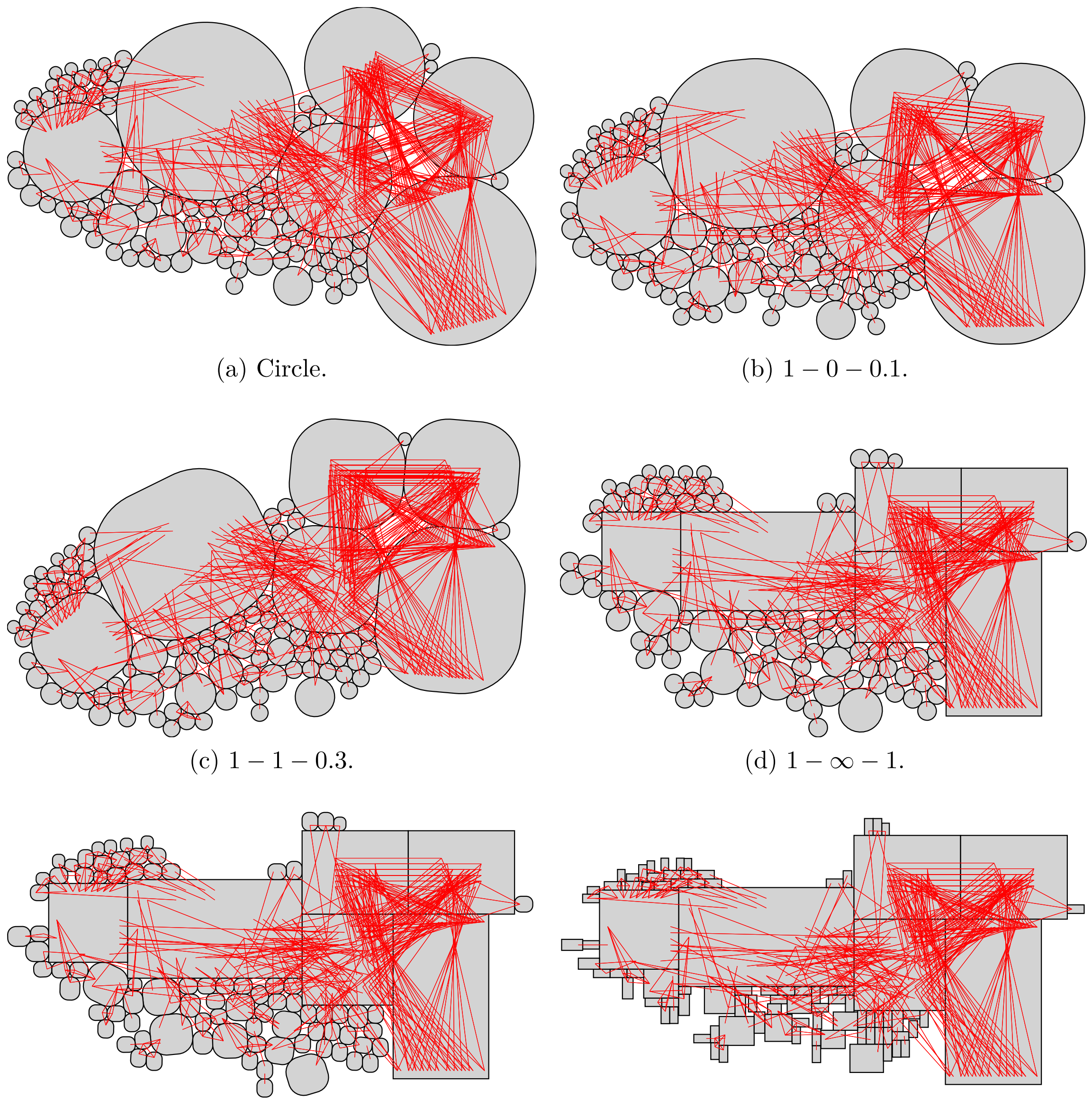
(c) $1 - 10 - 0.95$.

(d) $1 - \infty - 1$.

(e) $2 - 1 - 0.3$.

(f) $2 - \infty - 1$.

Figure 8.11: Important steps of the rounded rectangle algorithm for M0057.


(a) Circle.

(b) $1 - 0 - 0.1$.

(c) $1 - 1 - 0.3$.

(d) $1 - \infty - 1$.

(e) $2 - 1 - 0.3$.

(f) $2 - \infty - 1$.

Figure 8.12: Important steps of the rounded rectangle algorithm for V0093.

9 Conclusion and Outlook

This thesis has been motivated by the lack of algorithms for medium sized placement problems with 100 to 250 components per module side in 2.5D System-in-Package design. For these problems, we developed the rounded rectangle algorithm which solves a sequence of non-linear programs. In this thesis we considered different aspects of this algorithm, which are themselves of theoretical and practical interest. We conclude this thesis by summarizing the main results, and present opportunities for further research.

9.1 Conclusion

The main problems considered in this thesis are the circle rotation problem, the circle placement problem and the rectangle placement problem. The circle rotation problem is crucial for local improvement steps in the circle placement problem, which itself is an essential part of our algorithm for the rectangle placement problem. We evaluated all algorithms for real world problem instances.

We proved the NP-hardness of the circle rotation problem and its equivalence to a minimization problem known in the literature. We developed several local non-linear solution algorithms and showed that they find close-to-optimal solutions for real world problem instances efficiently. Furthermore, we proposed a branch and bound algorithm with efficient lower and upper bound estimations. In particular, we developed a novel domain reduction algorithm that enables the branch and bound algorithm to solve a wide range of problem instances to global optimality in very short running time.

For the circle placement problem we proved that our non-linear formulation satisfies the Mangasarian-Fromovitz constraint qualification (MFCQ). To face the non-convexity of the problem, we enhanced algorithms from the literature. In particular, we extended and improved the attractor repeller model and showed that our novel scaling invariant model yield significantly better solutions than previously existing models. Applying our circle rotation algorithms, we developed local search and monotonic basin hopping algorithms to overcome local optima, and we demonstrated that they notably improve the solution quality while only moderately increasing the running time.

We presented the rounded rectangle algorithms for the rectangle placement problem. Starting from a solution of the circle placement problem, we explained how to refine the circles to rectangles in a sequence of non-linear programs. We showed that each of these non-linear programs satisfies MFCQ and generates a feasible solution to the next problem. Furthermore, we proved that the algorithm converges to a solution for the rectangle placement problem. We numerically demonstrated that the algorithm yields solutions of good quality in moderate running time for instances with up to 250 components.

9.2 Outlook

For further research, there are several open questions regarding the rounded rectangle algorithm.

The algorithm spends most of the time in later iterations, where the wire length improvement and the change of the solution is small. In contrast, solvers based on metaheuristics such as simulated annealing or local search have their strength in improving existing solutions locally. A hybrid approach seems promising, where the exploration is done by the rounded rectangle algorithm, but after the refinement of larger components to rectangles, local search strategies for placing the smaller rectangles are used.

For the circle placement problem the number of non-overlapping constraints increases quadratically in the number circles. Additionally, for the rectangle placement problem also the number of variables grows quadratically. However, once an approximate placement is generated, most of the constraints never become active. In trust region approaches the constraints which can not be active within the trust region can be efficiently identified and, hence, ignored within the trust region. Another approach is to compute the Voronoi diagram for an approximate placement and preserving its structure in the remaining algorithm. This might be done by adding the vertices of the Voronoi diagram and possibly additional intermediate points as variables to the optimization problem and enforcing all components to stay in their Voronoi cells. For this approach the increase in constraints and variables is linear.

Bibliography

Addis, Bernardetta; Locatelli, Marco; Schoen, Fabio [2008]. *Efficiently packing unequal disks in a circle.* In: *Operations Research Letters* 36.1, pp. 37–42. ISSN: 0167-6377.

Ahn, Hee-Kap et al. [2007]. *Aperture-angle and Hausdorff-approximation of convex figures.* In: *Proceedings of the twenty-third annual symposium on Computational geometry.* SCG '07. ACM, pp. 37–45. ISBN: 978-1595937056.

Alizadeh, Farid [1995]. *Interior Point Methods in Semidefinite Programming with Applications to Combinatorial Optimization.* In: *SIAM Journal on Optimization* 5.1, pp. 13–51. ISSN: 1095-7189.

Alon, Amir; Ascher, Uri [1988]. *Model and solution strategy for placement of rectangular blocks in the Euclidean plane.* In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 7.3, pp. 378–386. ISSN: 0278-0070.

Anjos, Miguel F. [2001]. *New Convex Relaxations for the Maximum Cut and VLSI Layout Problems.* PhD thesis. University of Waterloo.

Anjos, Miguel F.; Vannelli, Anthony [2006]. *A New Mathematical-Programming Framework for Facility-Layout Design.* In: *INFORMS Journal on Computing* 18.1, pp. 111–118. ISSN: 1526-5528.

Areibi, Shawki; Yang, Zhen [2004]. *Effective memetic algorithms for VLSI design automation = genetic algorithms + local search + multi-level clustering.* In: *Evolutionary Computation* 12.3, pp. 327–353. ISSN: 1063-6560.

Bajaj, Chanderjit; Kim, Myung [1989]. *Generation of configuration space obstacles: The case of moving algebraic curves.* In: *Algorithmica* 4.1, pp. 157–172. ISSN: 0178-4617.

Bajaj, Chanderjit; Kim, Myung-Soo [1988]. *Algorithms for planar geometric models.* In: *Automata, Languages and Programming.* Ed. by Lepistö, Timo; Salomaa, Arto. Vol. 317. Lecture Notes in Computer Science. Springer, pp. 67–81. ISBN: 978-3540194880.

Bazaraa, Mokhtar S.; Sherali, Hanif D.; Shetty, CM [1993]. *Nonlinear Programming: Theory and Algorithms.* 2nd edition. John Wiley and Sons, New York. ISBN: 978-0471557937.

Berg, Mark de et al. [2000]. *Computational Geometry: Algorithms and Applications.* 2nd edition. Springer. ISBN: 978-3540779735.

Berger, Martin [2010]. *Bicriteria Optimization in Electronic Design Automation – Placement Algorithms for 2.5D System-in-Package Integration.* PhD thesis. Fachbereich Mathematik, TU Kaiserslautern.

Bertsekas, Dimitri P. [1999]. *Nonlinear Programming.* 2nd edition. Athena Scientific. ISBN: 978-1886529007.

Bezdek, James C.; Hathaway, Richard J. [2003]. *Convergence of alternating optimization.* In: *Neural, Parallel & Scientific Computations* 11.4, pp. 351–368. ISSN: 1061-5369.

Birgin, Ernesto G.; Sobral, F. N. C. [2008]. *Minimizing the object dimensions in circle and sphere packing problems.* In: *Computers & Operations Research* 35.7, pp. 2357–2375. ISSN: 0305-0548.

Böröczky, Károly J. [2004]. *Finite Packing And Covering.* Cambridge University Press. ISBN: 978-0521801577.

Böröczky, Károly J.; Ruzsa, Imre Z. [2007]. *Note on an inequality of Wegner.* In: *Discrete & Computational Geometry* 37.2, pp. 245–249. ISSN: 0179-5376.

Bose, Prosenjit; Seara, Carlos; Sethia, Saurabh [2004]. *On computing enclosing isosceles triangles and related problems.* In: *Proceedings of the 16th Canadian Conference on Computational Geometry (CCCG'04)*, pp. 120–123.

Brimberg, Jack; Juel, Henrik; Schöbel, Anita [2009]. *Locating a minisum circle in the plane.* In: *Discrete Applied Mathematics* 157.5, pp. 901–912. ISSN: 0166-218X.

Calderbank, A. Robert; Sloane, Neil J. A. [1987]. *New Trellis codes based on lattices and cosets.* In: *IEEE Transactions on Information Theory* 33.2, pp. 177–195. ISSN: 0018-9448.

Castillo, Ignacio; Kampas, Frank J.; Pintér, János D. [2008]. *Solving circle packing problems by global optimization: Numerical results and industrial applications.* In: *European Journal of Operational Research* 191.3, pp. 786–802. ISSN: 0377-2217.

Chow, Timothy Y. [1995]. *Penny-packings with minimal second moments.* In: *Combinatorica* 15.2, pp. 151–158. ISSN: 0209-9683.

Coope, Ian D. [1993]. *Circle fitting by linear and nonlinear least squares.* In: *Journal of Optimization Theory and Applications* 76.2, pp. 381–388. ISSN: 0022-3239.

Dobkin, David P.; Souvaine, Diane Loring [1990]. *Computational geometry in a curved world.* In: *Algorithmica* 5.1, pp. 421–457. ISSN: 0178-4617.

Drezner, Zvi [1980]. *DISCON: A New Method for the Layout Problem.* In: *Operations Research* 28.6, pp. 1375–1384. ISSN: 0030-364X.

Drezner, Zvi; Steiner, Stefan; Wesolowsky, George O. [2002]. *On the circle closest to a set of points.* In: *Computers & Operations Research* 29.6, pp. 637–650. ISSN: 0305-0548.

Faroe, Oluf; Pisinger, David; Zachariasen, Martin [2003]. *Guided Local Search for Final Placement in VLSI Design.* In: *Journal of Heuristics* 9.3, pp. 269–295. ISSN: 1381-1231.

Fodor, Ferenc [1999]. *The Densest Packing of 19 Congruent Circles in a Circle.* In: *Geometriae Dedicata* 74.2, pp. 139–145. ISSN: 0046-5755.

Fodor, Ferenc [2003]. *The densest packing of 13 congruent circles in a circle.* In: *Contributions to Algebra and Geometry* 44.2, pp. 431–440. ISSN: 0138-4821.

Garey, Michael R.; Johnson, David S. [1979]. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman. ISBN: 978-0716710455.

Goemans, Michel X.; Williamson, David P. [1995]. *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming.* In: *Journal of the ACM* 42.6, pp. 1115–1145. ISSN: 0004-5411.

Graham, R. L.; Sloane, N. J. A. [1990]. *Penny-packing and two-dimensional codes.* In: *Discrete & Computational Geometry* 5.1, pp. 1–11. ISSN: 0179-5376.

Groemer, Helmut [1960]. *Über die Einlagerung von Kreisen in einen konvexen Bereich.* In: *Mathematische Zeitschrift* 73.3, pp. 285–294. ISSN: 0025-5874.

Grosso, Andrea et al. [2010]. *Solving the problem of packing equal and unequal circles in a circular container.* In: *Journal of Global Optimization* 47.1, pp. 63–81. ISSN: 0925-5001.

Haeser, Gabriel [2010]. *On the global convergence of interior-point nonlinear programming algorithms.* In: *Computational & Applied Mathematics* 29, pp. 125–138. ISSN: 1807-0302.

Han, Shih-Ping [1988]. *A successive projection method.* In: *Mathematical Programming* 40.1, pp. 1–14. ISSN: 0025-5610.

Hansen, Pierre; Peeters, Dominique; Thisse, Jacques Francois [1981]. *On the location of an obnoxious facility.* In: *Sistemi Urbani* 3, pp. 299–317.

Håstad, Johan [2001]. *Some optimal inapproximability results.* In: *Journal of the ACM* 48.4, pp. 798–859. ISSN: 0004-5411.

Horst, R.; Thoai, N. V. [1999]. *DC Programming: Overview.* In: *Journal of Optimization Theory and Applications* 103.1, pp. 1–43. ISSN: 0022-3239.

Jungnickel, Dieter [2005]. *Graphs, Networks and Algorithms.* Springer. ISBN: 978-3540637605.

Kahng, Andrew B.; Reda, Sherief; Wang, Qinke [2005]. *Architecture and Details of a High Quality, Large-Scale Analytical Placer.* In: *Proceedings of the ACM/IEEE International Conference on Computer Aided Design.* IEEE Computer Society, pp. 891–898. ISBN: 978-0780392540.

Koebe, Paul [1936]. *Kontaktprobleme der konformen Abbildung.* In: *Berichte der Sächsischen Akademie der Wissenschaft Leipzig 88*, pp. 141–164.

Kohler, Markus; Spreng, Michael [1995]. *Fast computation of the C-space of convex 2D algebraic objects.* In: *International Journal of Robotics Research* 14.6, pp. 590–608. ISSN: 0278-3649.

Körner, Mark et al. [2009]. *General minisum circle location.* In: *Proceedings of the 21st Canadian Conference on Computational Geometry (CCCG2009)*, pp. 111–114.

Lee, In-kwon; Kim, Myung-Soo; Elber, Gershon [1998]. *The Minkowski Sum of 2D Curved Objects.* In: *Proceedings of Israel-Korea Bi-National Conference on New Themes in Computerized Geometrical Modeling*, pp. 155–164.

Lengauer, Thomas [1990]. *Combinatorial Algorithms for Integrated Circuit Layout.* John Wiley & Sons. ISBN: 978-0471928380.

Luo, Chaomin; Anjos, Miguel F.; Vannelli, Anthony [2008]. *Large-scale fixed-outline floorplanning design using convex optimization techniques.* In: *ASP-DAC '08: Proceedings of the 2008 conference on Asia and South Pacific design automation.* IEEE Computer Society Press, pp. 198–203. ISBN: 978-1424419227.

Luo, Zhi-Quan; Luo, Xiaodong; Kisialiou, M. [2003]. *An efficient quasi-maximum likelihood decoder for PSK signals.* In: *IEEE International Conference on Acoustics, Speech, and Signal Processing 2003 (ICASSP '03).* Vol. 6, pp. 561–564. ISBN: 978-0780376632.

Luo, Zhi-Quan et al. [2007]. *Approximation Bounds for Quadratic Optimization with Homogeneous Quadratic Constraints.* In: *SIAM Journal on Optimization* 18.1, pp. 1–28. ISSN: 1052-6234.

Maranas, Costas D.; Floudas, Christodoulos A.; Pardalos, Panos M [1995]. *New results in the packing of equal circles in a square.* In: *Discrete Mathematics* 142.1-3, pp. 287–293. ISSN: 0012-365X.

Markót, Mihály Csaba; Csendes, Tibor [2005]. *A New Verified Optimization Technique for the "Packing Circles in a Unit Square" Problems.* In: *SIAM Journal on Optimization* 16.1, pp. 193–219. ISSN: 1052-6234.

M'Hallah, Rym; Hifi, Mhand [2009]. *A Literature Review on Circle and Sphere Packing Problems: Models and Methodologies.* In: *Advances in Operations Research*, pp. 1–23.

Murata, Hiroshi et al. [1996]. *VLSI Module Placement based on Rectangle-Packing by the Sequence-Pair.* In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 15.12, pp. 1518–1524. ISSN: 0278-0070.

Nam, Gi-Joon; Cong, Jason [2007]. *Modern Circuit Placement – Best Practices and Results.* Springer. ISBN: 978-0387368375.

Nezakati, Ahmad; Zaferanieh, Mehdi; Fathali, Jafar [2009]. *BSSS Algorithm for the Location Problem with Minimum Square Error.* In: *Advances in Operations Research.*

Nocedal, Jorge; Wright, Stephen J. [2006]. *Numerical Optimization.* 2nd edition. Springer Series in Operations Research and Financial Engineering. Springer. ISBN: 978-0387303031.

Nowak, Ivo [2000]. *Dual Bounds and Optimality Cuts for All-Quadratic Programs with Convex Constraints.* In: *Journal of Global Optimization* 18.4, pp. 337–356. ISSN: 0925-5001.

Nurmela, Kari J.; Östergård, Patric R. J. [1997]. *Packing up to 50 Equal Circles in a Square.* In: *Discrete and Computational Geometry* 18.1, pp. 111–120. ISSN: 0179-5376.

Onodera, Hidetoshi; Taniguchi, Yo; Tamaru, Keikichi [1991]. *Branch-and-bound placement for building block layout.* In: *Proceedings of the 28th Conference on ACM/IEEE Design Automation.* ACM Press, pp. 433–439. ISBN: 978-0897913959.

Plastria, Frank [1992]. *GBSSS: The generalized big square small square method for planar single-facility location.* In: *European Journal of Operational Research* 62.2, pp. 163–174. ISSN: 0377-2217.

Polityko, David D. [2008]. *Physikalischer Entwurf für die vertikale SiP Integration.* PhD thesis. Electrical Engineering and Computer Sciences, Berlin Institute of Technology.

Raber, Ulrich [1999]. *Nonconvex All-Quadratic Global Optimization Problems: Solution Methods, Application and Related Topics.* PhD thesis. Universität Trier.

Richter, Christian et al. [2007]. *Technology Aware Modeling of 2.5D-SiP for Automation in Physical Design.* In: *Proceedings of the 9th Electronics Packaging Technology Conference*, pp. 623–630. ISBN: 978-1424413232.

Sait, Sadiq M.; Youssef, Habib [1999]. *VLSI Physical Design Automation, Theory and Practice.* World Scientific. ISBN: 978-9810238835.

Semple, J.; Zlobec, S. [1986]. *On the continuity of a Lagrangian multiplier function in input optimization.* In: *Mathematical Programming* 34.3, pp. 362–369. ISSN: 0025-5610.

Sha, Lu; Dutton, Robert W. [1985]. *An analytical algorithm for placement of arbitrarily sized rectangular blocks.* In: *DAC '85: Proceedings of the 22nd ACM/IEEE conference on Design automation.* ACM, pp. 602–608. ISBN: 978-0818606359.

So, Anthony Man-Cho; Zhang, Jiawei; Ye, Yinyu [2007]. *On approximating complex quadratic optimization problems via semidefinite programming relaxations.* In: *Mathematical Programming* 110.1, pp. 93–110. ISSN: 0025-5610.

Souvaine, Diane Loring [1986]. *Computational geometry in a curved world.* PhD thesis. Princeton University.

Thoai, Nguyen [2005]. *General Quadratic Programming.* In: *Essays and Surveys in Global Optimization.* Ed. by Audet, Charles; Hansen, Pierre; Savard, Gilles. Springer, pp. 107–129. ISBN: 978-0387255705.

Todd, Michael J. [2001]. *Semidefinite Optimization.* In: *Acta Numerica* 10, pp. 515–560.

Tóth, László Fejes [1972]. *Lagerungen in der Ebene auf der Kugel und im Raum.* Springer. ISBN: 978-3540054771.

Tseng, Paul [1993]. *Dual coordinate ascent methods for non-strictly convex minimization.* In: *Mathematical Programming* 59, pp. 231–247. ISSN: 0025-5610.

Vandenberghe, Lieven; Boyd, Stephen [1996]. *Semidefinite Programming.* In: *SIAM Review* 38.1, pp. 49–95. ISSN: 0036-1445.

Viswanathan, Natarajan; Chu, Chris Chong-Nuen [2004]. *FastPlace: efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model.* In: *ISPD '04: Proceedings of the 2004 international symposium on Physical design.* ACM, pp. 26–33. ISBN: 978-1581138177.

Wächter, Andreas; Biegler, Lorenz T. [2006]. *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming.* In: *Mathematical Programming* 106.1, pp. 25–57. ISSN: 0025-5610.

Wales, David J.; Doye, Jonathan P. K. [1997]. *Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms.* In: *The Journal of Physical Chemistry A* 101.28, pp. 5111–5116. ISSN: 1089-5639.

Wegner, Gerd [1986]. *Über endliche Kreispackungen in der Ebene.* In: *Studia Scientiarum Mathematicarum Hungarica* 21, pp. 1–28.

Wong, Danny F.; Leong, Hon Wai; Liu, Chang. L. [1988]. *Simulated annealing for VLSI design.* Kluwer Academic Publishers. ISBN: 978-0898382563.

Zhang, Shuzhong; Huang, Yongwei [2006]. *Complex Quadratic Optimization and Semidefinite Programming.* In: *SIAM Journal on Optimization* 16.3, pp. 871–890. ISSN: 1052-6234.

Curriculum Vitae

Scientific Career

Jul. 2011 – today	Member of the research staff in the Department of Optimization at the Fraunhofer ITWM, Kaiserslautern
Sep. 2008 – Jul. 2011	PhD study in mathematics at the Technical University of Kaiserslautern, funded by a scholarship of the Fraunhofer ITWM
Dec. 2006 – Aug. 2008	Member of the research staff in the Department of Optimization at the Fraunhofer ITWM, Kaiserslautern
Oct. 2004 – Nov. 2006	Main study period in Mathematics at the University of Siegen. Title of the diploma thesis: "Graphentheoretische Modellierung eines automatisierungstechnischen Echtzeitnetzwerks und Algorithmenentwurf zum Kommunikations-Scheduling"
Oct. 2003 – Sep. 2004	Exchange study in mathematics at the University of Edinburgh, Scotland
Oct. 2001 – Sep. 2003	Basic study period in mathematics at the University of Siegen

Publications

The following publication contains related work to this thesis.

Richter, Christian; Polityko, David D.; Hefer, Jan; Guttowski, Stephan; Reichl, Herbert; Berger, Martin; Nowak, Uwe; Schröder, Michael [2007]. *Technology Aware Modeling of 2.5D-SiP for Automation in Physical Design.* In: *Proceedings of the 9th Electronics Packaging Technology Conference*, pp. 623–630. ISBN: 978-1424413232.

Akademischer Lebenslauf

Akademische Karriere

Jul. 2011 – heute	Wissenschaftlicher Mitarbeiter in der Abteilung Optimierung am Fraunhofer ITWM, Kaiserslautern
Sep. 2008 – Jul. 2011	Promotion in Mathematik an der Technichen Universität Kaiserslautern, finanziert durch ein Stipendium des Fraunhofer ITWM
Dec. 2006 – Aug. 2008	Wissenschaftlicher Mitarbeiter in der Abteilung Optimierung am Fraunhofer ITWM, Kaiserslautern
Oct. 2004 – Nov. 2006	Hauptstudium Mathematik an der Universität Siegen. Diplomarbeitsthema: “Graphentheoretische Modellierung eines automatisierungstechnischen Echtzeitnetzwerks und Algorithmenentwurf zum Kommunikations-Scheduling”
Oct. 2003 – Sep. 2004	Auslandsstudium Mathematik an der Universität Edinburgh, Schottland
Oct. 2001 – Sep. 2003	Grundstudium Mathematik an der Universität Siegen

Publikationen

Die folgende Publikation stehen in Bezug zu dieser Arbeit.

Richter, Christian; Polityko, David D.; Hefer, Jan; Guttowski, Stephan; Reichl, Herbert; Berger, Martin; Nowak, Uwe; Schröder, Michael [2007]. *Technology Aware Modeling of 2.5D-SiP for Automation in Physical Design.* In: *Proceedings of the 9th Electronics Packaging Technology Conference*, pp. 623–630. ISBN: 978-1424413232.