TECHNISCHE UNIVERSITÄT MÜNCHEN
Zentrum Mathematik

Optimal Containment

Lucia Barbara Roth

Vollständiger Abdruck der von der Fakultät für Mathematik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Die Dissertation wurde am 20. Mai 2009 bei der Technischen Universität München eingereicht und durch die Fakultät für Mathematik am 15. Dezember 2009 angenommen.

Abstract

We study algorithmic aspects of optimal containment problems in this thesis. We consider 1- and k-containment problems under homothety, with the special case of the k-center problem, as well as rotational containment problems, especially some problems involving cylinders. We discuss the complexity of such problems alongside algorithms to compute bounds for the optimal value, relying amongst other on geometric inequalities, mathematical programming formulations, and the concept of core-sets to achieve these results. The thesis includes experimental studies of the practical performance of the described algorithms, too.

Zusammenfassung

Thema dieser Arbeit sind optimale Containment-Probleme unter algorithmischen Gesichtspunkten. Es werden 1- und k-Containment-Probleme unter Homothetie betrachtet, darunter der Spezialfall des k-Center-Problems, sowie Containment-Probleme unter Rotation, insbesondere solche mit Zylindern. Dabei sind sowohl die Komplexität der Probleme als auch Algorithmen zur Näherung des Optimalwerts von Interesse. Hierzu werden unter anderem geometrische Ungleichungen, Formulierungen als mathematische Programme, sowie Core-Sets eingesetzt. Die Arbeit enthält weiterhin Tests zur praktischen Überprüfung der beschriebenen Algorithmen.

Contents

1 Introduction

Containment problems are questions from applied geometry which aim at finding or approximating an extremal body in a given class of bodies such that it either contains or is contained in another given body [67]. We consider optimal containment problems and related questions from an algorithmic viewpoint.

Maybe the most famous containment problem is the question of the smallest enclosing ball of a point set, originally posed by J.J. Sylvester in 1857 [129]. This problem is still subject to research today [140] – alongside many other containment questions. In this thesis, we focus on some specific classes of containment problems (introduced in Chapter 2), and particularly on the following aspects of the topic:

- We ask for the complexity of a given problem in general dimension and for algorithms to solve or approximate the problem.
- On the practical side, we discuss implementations of algorithms and their performance on concrete examples.

We now proceed with some exemplary applications of containment problems, mention related topics, and finish the introduction with an outline of the thesis and the acknowledgments.

1.1 Applications of Containment Problems

A variety of real-world applications motivates studying containment problems. An example of an application of the smallest enclosing ball problem in 2D is the placement of an emergency facility for a set of "customers" modeled as demand points. We seek a center of the point set minimizing the maximum Euclidean distance to the demand points. When not only one but k facilities can be placed, we have the Euclidean k-center problem, where we look for k centers instead of one. Besides location planning, 2D and 3D containment problems frequently arise in robotics and computer graphics, whereas high-dimensional applications can be found in mathematics itself, for instance in optimization and statistics (see [66] for examples).

Many other shapes besides Euclidean balls can arise in containment problems. In some applications, other distance measures are simply more natural. For instance, in [82], polygons are covered by rectangles which is motivated by finding good layouts for maps. Protein structures can be encoded into binary strings [55], and so can occurrences of keywords in text documents. In order to analyze similarity of binary strings, the Hamming distance and its geometric counterpart, the l_1 distance (see also [59]), are suitable measures since they count the number of different entries (in contrast to, e.g., the l_∞ distance). In other cases, there is no particular reason to prefer a specific norm to measure distance, but some norms have desired computational properties such as the l_1 norm in the medical diagnosis problem described in [108]. We now present two examples of location problems where more “exotic” shapes arise.

1.1.1 Robot Example

Consider the situation where a robot arm should be able to reach a set of object locations in the plane. The arm may rotate freely in any direction, so it can reach objects within a circle whose radius depends on the arm length. Now assume the robot itself may also move along tracks above the plane, for instance, two orthogonal tracks as depicted in Figure 1.1. The robot can reach all the objects whenever

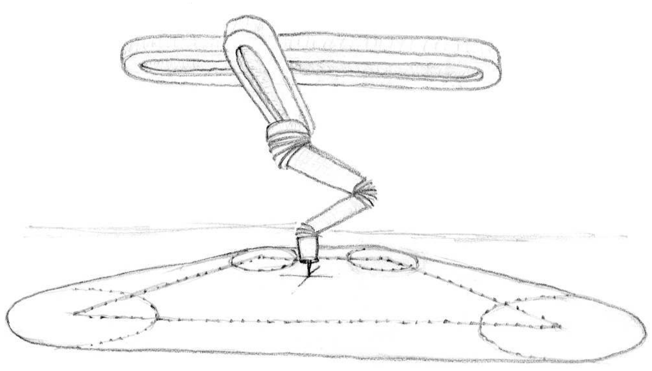


Figure 1.1: A robot arm, hinged to two axis parallel tracks.

the object locations in the plane can be covered by the resulting shape, i.e., the Minkowski sum of a circle and a rectangle (see Figure 1.2). Placing the robot can therefore be modeled as a containment problem in 2D. (A proper definition of the classes of containment problems we consider can be found in Chapter 2.)

In case we have a robot arm with joint limits in a way that it only reaches a section of the full circle (compare [72]), we get a conical section – just as in the following example.

1.1.2 Ray Source Example

Let a sender emit signals which should reach a set of given target objects in the plane or in 3-space. If the source emits equally to all sides, we get a disk or a ball. However, the source is often bounded on its sides, limiting the directions of the emitted rays or waves such that only a section of the ball remains (see Figure 1.2). We get the same type of shapes when the objects are required to be within the field of sight of a camera. Finding the minimal required reach and the corresponding optimal location of the sender is also a containment problem. In practice, one might like to place several senders, too, maybe with different transmission ranges.

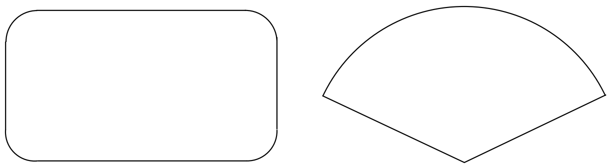


Figure 1.2: Possible shapes for containment problems from robot and ray source examples.

1.1.3 Extremity Correction Example

A number of applications of containment problems can be found in the life sciences. We consider another example, this time from human medicine, where deformities of the extremities (especially the legs) are to be corrected. Such deformities can be congenital or post-traumatic. Frequently, the affected bones are not only malaligned but also too short. Such deformities can be corrected by a method of G. Ilizarov [86], [87]. It involves cutting the bone, and, with help of an externally attached apparatus, slowly extending the cut making the bone tissue grow in between. Whereas this method is incriminating for the patient due to the external apparatus, a new technique [22] places the device ("intramedullary nail") directly into the bone marrow. Though the new method is more patient-friendly, it entails higher demands on the planning of the operation and the precision of the nail placement. Locating the cylindrical nail (see Figure 1.3) within the bone marrow can be modeled as a containment problem, where the bone data is extracted from CT scans. The approach is explained in detail in Section 5.3; see also [36].

1.2 Related Problems

We shortly mention some further topics which are strongly connected to the containment problems we consider in this thesis.

1.2.1 Clustering

Given a set of objects, most frequently data points, we seek to partition it into a number of clusters, optimizing a given objective function. The k-center problem mentioned before is a typical clustering problem. However, many other variants

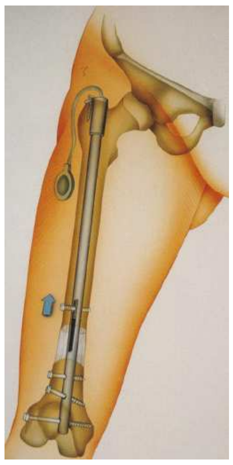


Figure 1.3: Schematic illustration of an implantable nail used in extremity correction. Picture credit: © 2004 Beinverlängerung.de

of clustering problems are studied. Instead of minimizing the maximum distance to the cluster center, one can also choose to minimize the sum of distances, which is known as the k-median problem. Discrete clustering problems require choosing the cluster centers from a discrete set. In other settings, it may be desirable not to consider the distance of the clustered objects to the cluster center, but their pairwise distances [14]. Moreover, the number of clusters need not be fixed but can also be subject to the objective function. The complexity of such problems is studied in [51]. For a survey on clustering, see [117].

1.2.2 Covering, Packing, and Piercing Problems

Covering and packing problems are closely related to containment questions. Covering asks for a set of objects such that their union contains a given body; packing requires that the intersection of the set of packed objects has empty interior and that its union is contained in a given body. Packing of polygons is studied for instance in [1] and [15]. A famous question in this context is sphere packing. Piercing problems, in contrary, seek for a hitting set of points such that each member of a set of given objects contains at least one point. Note that a number of Euclidean unit balls can be pierced by k points if and only if the set of centers of the balls can be covered by k unit balls [113], [123]. In this sense, k-piercing problems for Euclidean balls are polar to Euclidean k-center problems.

1.2.3 Facility Location and Motion Planning

Location problems seek for the optimal placement of a number of facilities, most frequently in 2D, 3D, or within a given discrete set. The placement of supply units for a given set of customers can be modeled as a k-median problem; for emergency facilities, k-center is an appropriate choice [47]. However, these problems often involve capacities, costs, and additional constraints. Typically, there are also existing facilities which should be considered in the problem formulation. Moreover, the facilities need not be modeled as points but may also be more complicated objects such as lines or polygonal chains. See [48] for a survey. An interesting variant is the optimal placement of obnoxious or hazardous facilities, e.g., waste disposal sites or pipelines. Here, it is desirable to maximize the distance instead of minimizing it [23], [46], [97]. The problem of finding a widest empty corridor [24] arises when an object has to be transported through a set of obstacles. Such motion planning problems also come up in robotics research, and some of them can be formulated as inner containment problems. See the survey [121] on motion planning.

1.3 Thesis Outline and Main Results

We now outline the structure of this thesis and provide a summary of the main results. To begin with, Chapter 2 contains basic definitions and formally introduces containment problems.

In Chapter 3, we consider the basic 1-containment problem under homothety, that is covering a point set by a homothetic copy of a given container. We review known methods for the problem, and extend some LP and SOCP formulations to more general container shapes. We also propose a cutting-plane algorithm for the problem. Though it may need an exponential number of iterations in general dimension and is therefore not polynomial in the worst case, we can provide examples where it is superior to known polynomial methods in moderate dimensions, for instance when the container is the cross polytope and the number of input points gets large.

Chapter 4 addresses containment under homothety, too, but with several containers. This is a generalization of the well-known k-center problem, also allowing for non-symmetric and different containers. We introduce a diameter partitioning scheme based on distance graphs which can be used to compute upper and lower bounds with provable guarantees for 2-containment problems. The analysis of the method is based on (partly new) geometric inequalities, and we provide results for general container shapes and also different containers. For some special cases of the k-containment problem, we are able to verify the existence of polynomial time methods.

Moreover, we consider a branch-and-bound procedure to approximate the general k-containment problem. It can be used to find fast approximations of the Euclidean k-center problem, and improves on previous implementations for this problem. We discuss how to further accelerate the method using a new MICP-formulation which is integrated into the branch-and-bound. It is also possible to use the bounds computed by the diameter partitioning scheme to speed up the procedure.

Finally, we consider the case where not a point set but its convex hull has to be covered by k Euclidean balls. We see that the core-set results for the Euclidean k-center problem can be extended and derive a method to construct these core-sets in general dimension. Thereby, we prove the existence of a polynomial time approximation scheme for this problem.

In Chapter 5, we allow rotations in addition to homotheties. We are especially interested in containment problems where the containers are rotational symmetric about an axis. We discuss their complexity and structural properties, and show that covering a point set on the unit sphere with an anchored circular double cone is $\mathbb{NP}$-complete. In particular, we consider problems involving cylinders and provide an overview of methods to compute a-priori bounds for smallest enclosing (anchored) cylinders.

It is then shown that core-sets for the anchored 1-ray problem exist whose size depends only on the desired accuracy and the shape of the point set, but not on the number of input points or the dimension. We can extend these results to anchored cylinder, anchored k-ray, and anchored k-line problems. We then consider approximation algorithms for these problems, based on core-sets and a convex programming formulation for the anchored 1-ray. We prove the existence of a polynomial time approximation scheme when k is fixed and the ratio of the maximal norm of a point from the input set and the optimal radius is bounded by a constant. We also discuss an implementation of the resulting algorithm for the anchored cylinder problem and computational examples.

Finally, we introduce a cylinder problem originating from extremity surgery. We provide an approximation algorithm based on semidefinite programming, and discuss practical questions in 3D. In addition, we show that the problem is $\mathbb{NP}$-complete in general dimension, and depict relations to geometric transversal theory.

An overview of some open questions related to these problems can be found in Chapter 6.

1.4 Acknowledgments

Co-authors. Some of the work presented in this thesis has been published previously. Chapters 3 and 4 are joint work with René Brandenberg, and major parts thereof also appear in [37] and [38]. The extremity correction part in Section 5.3 resulted from an interdisciplinary project[1], and is joint work with René Brandenberg, Tobias Gerken, and Peter Gritzmann, see [36]. I would like to thank my co-authors for their work, and for sharing their thoughts and ideas with me.

[1]see `http://www-m9.ma.tum.de/Allgemeines/Extremit%E4tenkorrektur`

Grants. Throughout most time of my PhD research, I was supported by the "Deutsche Forschungsgemeinschaft" through the graduate program "Applied Algorithmic Mathematics" which I gratefully acknowledge. I received financial support to take part in PreDoc Course "Optimization Methods in Discrete Geometry" (in Berlin, 2006), too, and would like to thank the organizers and everybody else involved for this valuable experience. I also want to thank René Brandenberg for organizing a stay at University College London in 2008 and supporting it with his Walther-von-Dyck grant, and David Larman for inviting us.

Thanks. I thank Peter Gritzmann for offering me the possibility to join his research group and work on these questions, for always supporting me and my work, and for his helpful comments on drafts. I thank René Brandenberg for his support, for introducing me to this area of mathematics and pointing out interesting problems, and for numerous helpful suggestions and inspiring discussions. I have enjoyed working at M9, and would also like to thank all other current and former group members, namely Klaudia Bachmeier, Oliver Bastert, Steffen Borgwardt, Julia Böttcher, David Bremner, Andreas Brieden, Markus Brill, Christoph Buchheim, Nico Düvelmeyer, Tobias Gerken, Soubhi Elias Janjal, Peter Heinig, Jan Hladky, Markus Jörg, Barbara Langfeld, Katja Lord, Diana Piguet, Tanja Stadler, Anusch Taraz, Barbara Wilhelm, and Andreas Würfl. I am grateful for many amusing conversations, for learning essential things about life in academia, and for help whenever I needed it.

This thesis contains some experimental studies, and it is pointed out when the implementations use code from student projects at M9. Therefore, I would like to thank Andreas Arnold, Kathrin Frankl, Brigitte Gölles, and Stefan König for their work.

Many thanks also to my fellow students and colleagues, especially Andrea, Irmi, Martin, and Vinzenz who often cheered me up.

Last but not least I want to thank my family and my husband for all their support and encouragement during the ups and downs while I was working on this thesis.

2 Preliminaries

In the following, we introduce the necessary concepts and notation in order to deal with problems involving the containment of objects in other objects.

The geometric objects we consider live in n-dimensional Euclidean space over $\mathbb{R}$, the real field. By $\|\cdot\|$, we denote the Euclidean norm and by $\langle\cdot,\cdot\rangle$ the standard scalar product. We use $e_1,\dots,e_n$ to denote the standard basis of $\mathbb{R}^n$. Any point $x \in \mathbb{R}^n$ is given by n coordinates, $x = (\xi_1,\dots,\xi_n)^T$, such that $x = \sum_{i=1}^n \xi_i e_i$.

2.1 Containment Problems

For optimal containment, among a fixed family of objects, we look for an element satisfying the containment conditions and optimizing a given measure function. In order to be able to handle this kind of problems algorithmically, we do of course need further conditions on both the objects involved and the measure function, as well as concrete containment conditions.

Containment problems can be defined in the following setting: Let $\mathcal{I}, \mathcal{O}$ be subsets of $\mathcal{P}(\mathbb{R}^n)$, where $\mathcal{P}(\cdot)$ denotes the power set. We call a problem a containment problem when the task can be classified as finding $I \in \mathcal{I}$, $O \in \mathcal{O}$ satisfying the containment condition $I \subset O$. For optimal containment, we distinguish between outer and inner containment problems. Outer containment problems are minimization problems. Let ω be a non-negative functional on $\mathcal{O}$ which is monotone with respect to set inclusion. For outer containment, the task is to find $I \in \mathcal{I}$, $O \in \mathcal{O}$ satisfying the containment condition such that $\omega(O)$ is minimal in $\mathcal{O}$. Analogously, for inner containment problems, ω is defined on $\mathcal{I}$, and we seek $I \in \mathcal{I}$, $O \in \mathcal{O}$, such that O contains I and $\omega(I)$ is maximal in $\mathcal{I}$. Of course, without further specification, such I, O need not even exist. For the problems we consider later on, we specify $\mathcal{I}$, $\mathcal{O}$, and ω.

We proceed with some necessary terminology, and then introduce the specific classes of containment problems we consider in Section 2.3. See [67] on general containment problems.

2.2 Definitions

Let $A, A' \subset \mathbb{R}^n$. By $\mathrm{conv}(A)$, $\mathrm{pos}(A)$, $\mathrm{aff}(A)$, and $\mathrm{lin}(A)$, we denote the *convex*, *positive*, *affine*, and *linear hull* of A, respectively. The *distance* $\mathrm{dist}(A, A')$ between

A and A' is defined as $\inf\{\|a - a'\| : a \in A, a' \in A'\}$. The *Minkowski sum* $A + A'$ is the set $\{a + a' : a \in A, a' \in A'\}$. For $\lambda \in \mathbb{R}$, λA denotes the set $\{\lambda a : a \in A\}$.

We consider translations, dilatations, and rotations in $\mathbb{R}^n$. A *translation* is a mapping $x \mapsto x + t$ where t is a vector in $\mathbb{R}^n$. A *dilatation* is a mapping $x \mapsto \lambda x$, where λ is a non-negative scalar. A *rotation* is a mapping $x \mapsto Ax$, where A is an orthogonal matrix with positive determinant. A *homothety* is a mapping composed by translations and dilatations. A *similarity* is a mapping composed by homotheties and rotations. By a translation, dilatation, rotation, homothety, or similarity of $A \subset \mathbb{R}^n$ we refer to its image under the specified map. The identity matrix is referred to as Id.

A *convex body* in $\mathbb{R}^n$ is a compact, convex set. Polytopes are convex bodies that are represented as the convex hull of a finite point set or as the intersection of a finite set of half-spaces. We refer to this as the vertex and hyperplane representation of the polytope. For short, we call polytopes in vertex representation *$\mathcal{V}$-polytopes* and polytopes in hyperplane representation *$\mathcal{H}$-polytopes*.

By $\mathbb{B} = \{x \in \mathbb{R}^n : \|x\| \leq 1\}$, we denote the *Euclidean unit ball*, and by $\mathbb{S} = \{x \in \mathbb{R}^n : \|x\| = 1\}$ the *Euclidean unit sphere*. The set $\{x \in \mathbb{R}^n : \langle e_i, x\rangle \leq 1,\ \langle -e_i, x\rangle \leq 1, 1 \leq i \leq n\} = [-1, 1]^n$ is the *unit cube*, and $\mathrm{conv}\{e_1, \ldots, e_n, -e_1, \ldots, -e_n\}$ denotes the *unit cross polytope* in $\mathbb{R}^n$.

In order to address the containment problems we consider in the following, we introduce the notion of a *container*. By a container, we denote an n-dimensional convex body C with $0 \in \mathrm{int}(C)$. We denote the set of all n-dimensional containers by $\mathcal{C}_n$. We can associate a distance (or gauge) functional F_C with a container C: $F_C(x) = \inf\{\lambda \geq 0 : x \in \lambda C\}$. We may write minimum instead of infimum here since C is closed, compact, and has the origin in its interior. For 0-symmetric containers, F_C is a norm on $\mathbb{R}^n$. For instance, the cube induces the ∞-norm and the cross-polytope induces the 1-norm. However, since we do not get a norm for containers C that are not 0-symmetric, we simply address the containers as objects in Euclidean space and do not consider general Minkowski spaces here.

Let us also define the Minkowski measure of symmetry. For any container $C \subset \mathbb{R}^n$, let s_C denote the *Minkowski symmetry* of C, that is the maximal dilatation factor ρ such that some translate of $-\rho C$ is contained in C. Obviously, $s_C \leq 1$, and we say that C is symmetric if and only if $s_C = 1$. In the latter case C can be translated such that the translate is 0-symmetric. Furthermore, $s_C \geq 1/n$ follows from John's theorem [93].

2.3 Basic Classes of Outer Containment Problems for Point Sets

In the following, we introduce the special containment problems considered in this thesis. In particular, we consider classes of outer containment problems with finite point sets as inner objects where the dimension n is part of the input.

The first class of containment problems we introduce is containment under homothety for point sets. Given a finite point set $P = \{p_1, \dots, p_m\}$ and a container, we are looking for the smallest homothetic copy of the container that contains the point set. Note that we minimize the dilation factor here, though other measures are possible, too.

Minimal 1-Containment Problem under Homothety (**MCP**$_{\text{Hom}}$)

Input:	$n \in \mathbb{N}$, $m \in \mathbb{N}$, $C \in \mathcal{C}_n$, $P = \{p_1, \dots, p_m\}$ with $p_i \in \mathbb{R}^n$ for $1 \le i \le m$
Task:	min ρ, such that $P \subset c + \rho C$, $c \in \mathbb{R}^n$

In the notion of Section 2.1, we can express the task using $\mathcal{I} = \{P\}$, $\mathcal{O} = \{c + \rho C :\ c \in \mathbb{R}^n,\ \rho \ge 0\}$, and $\omega(c + \rho C) = \rho$ for any instance of the MCP$_{\text{Hom}}$. With $R_{\text{Hom}}(P, C)$, we denote the optimal radius of the MCP$_{\text{Hom}}$ for P and C. A special case is the smallest enclosing ball problem, where the container is the Euclidean unit ball $\mathbb{B}$. Another example is the computation of the Minkowski symmetry s_C for a $\mathcal{V}$-polytope $V = \text{conv}(P)$, since $s_C = 1/R_{\text{Hom}}(-P, V)$.[1] We can also rotate the container instead of translating it:

Minimal 1-Containment Problem under Rotation and Dilatation (**MCP**$_{\text{Rot}}$)

Input:	$n \in \mathbb{N}$, $m \in \mathbb{N}$, $C \in \mathcal{C}_n$, $P = \{p_1, \dots, p_m\}$ with $p_i \in \mathbb{R}^n$ for $1 \le i \le m$
Task:	min ρ, such that $P \subset \rho\Phi(C)$, Φ a rotation

The optimal radius is denoted by $R_{\text{Rot}}(P, C)$. Allowing both rotations and translations, we get containment under similarity:

Minimal 1-Containment Problem under Similarity (**MCP**$_{\text{Sim}}$)

Input:	$n \in \mathbb{N}$, $m \in \mathbb{N}$, $C \in \mathcal{C}_n$, $P = \{p_1, \dots, p_m\}$ with $p_i \in \mathbb{R}^n$ for $1 \le i \le m$
Task:	min ρ, such that $P \subset c + \rho\Phi(C)$, $c \in \mathbb{R}^n$, Φ a rotation

[1] For other types of containers, computing the Minkowski symmetry is also a containment problem under homothety, but not with a point set as inner object. Note that s_C can be computed by linear programming both for $\mathcal{V}$- and $\mathcal{H}$-polytopes [67].

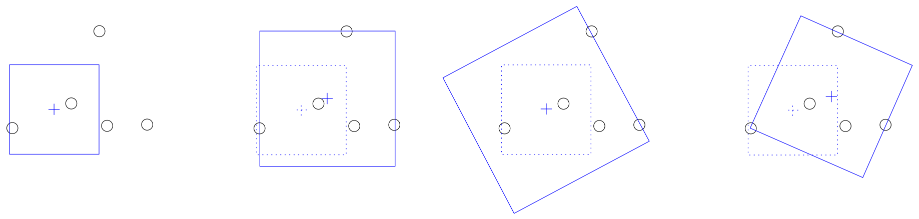


Figure 2.1: 2D Example of a point set P (black) and a container C (blue), and the corresponding solutions of $\mathrm{MCP_{Hom}}$, $\mathrm{MCP_{Rot}}$, and $\mathrm{MCP_{Sim}}$.

The optimal radius is denoted by $R_{\mathrm{Sim}}(P, C)$. A special case is the smallest enclosing finite cylinder, where $C = l + \mathbb{B} \cap l^{\perp}$, l a line segment in $\mathbb{R}^n$.[2] Figure 2.1 demonstrates the differences of the $\mathrm{MCP_{Hom}}$, the $\mathrm{MCP_{Rot}}$, and the $\mathrm{MCP_{Sim}}$ in an example where the container C is a cube in 2D. Naturally, it always holds that $R_{\mathrm{Hom}}(P, C) \leq R_{\mathrm{Sim}}(P, C)$ and $R_{\mathrm{Rot}}(P, C) \leq R_{\mathrm{Sim}}(P, C)$.

We are also interested in covering a point set with more than one container. We consider the problem where the number of containers k is fixed.

Minimal k-Containment Problem under Homothety ($\mathbf{MCP^k_{Hom}}$)

Input: $n \in \mathbb{N}$, $m \in \mathbb{N}$, $C_1, \ldots, C_k \in \mathcal{C}_n$, $P = \{p_1, \ldots, p_m\}$ with $p_i \in \mathbb{R}^n$ for $1 \leq i \leq m$

Task: min ρ, such that $P \subset \bigcup_{i=1}^k (c_i + \rho C_i)$, $c_i \in \mathbb{R}^n$ for $1 \leq i \leq k$

In the notion of Section 2.1, we now have $\mathcal{O} = \{\bigcup_i (c_i + \rho C_i) : \ c_i \in \mathbb{R}^n, \ \rho \geq 0\}$. When all k containers are identical, we talk about the *k-center problem*. A special case is the Euclidean k-center problem (see Figure 2.2), where $C_1 = \ldots = C_k = \mathbb{B}$. We use $R_{\mathrm{Hom}}(P, C_1, \ldots, C_k)$ for the optimal radius of the $\mathrm{MCP^k_{Hom}}$ for P and the containers $C_1, \ldots, C_k$. When all k containers are equal, we refer to $R^k_{\mathrm{Hom}}(P, C)$ for shortness. As before, we can also define k-containment under rotation and dilatation:

Minimal k-Containment Problem under Rotation and Dilatation ($\mathbf{MCP^k_{Rot}}$)

Input: $n \in \mathbb{N}$, $m \in \mathbb{N}$, $C_1, \ldots, C_k \in \mathcal{C}_n$, $P = \{p_1, \ldots, p_m\}$ with $p_i \in \mathbb{R}^n$ for $1 \leq i \leq m$

Task: min ρ, such that $P \subset \bigcup_{i=1}^k \rho \Phi_i(C_i)$, Φ_i rotations for $1 \leq i \leq k$

[2] Since we require containers to be compact, the usual smallest enclosing infinite cylinder does not meet the requirements. However, when l is long enough (depending on P, of course), the two problems are equivalent.

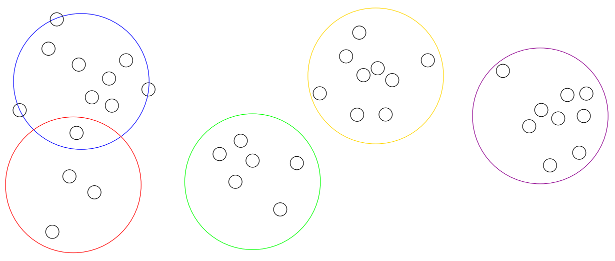


Figure 2.2: Illustration of an optimal solution of an instance of the Euclidean k-center problem in 2D, where $k = 5$.

Note that this definition requires all C_i to still intersect in the origin after rotation and scaling. This does of course not hold for k-containment under similarity:

Minimal k-Containment Problem under Similarity **($\mathbf{MCP}^k_{\mathrm{Sim}}$)**

Input:	$n \in \mathbb{N}$, $m \in \mathbb{N}$, $C_1, \ldots, C_k \in \mathcal{C}_n$, $P = \{p_1, \ldots, p_m\}$ with $p_i \in \mathbb{R}^n$ for $1 \leq i \leq m$
Task:	min ρ, such that $P \subset \bigcup_{i=1}^k (c_i + \rho\Phi_i(C_i))$, $c_i \in \mathbb{R}^n$, Φ_i rotations for $1 \leq i \leq k$

With these fundamental problems, we are able to classify a wide range of containment questions. In Chapters 3, 4, and 5 we consider both the general questions but also interesting special cases. For instance, in case of containment problems involving rotations, we restrain our considerations to highly symmetric objects since the general problem is suspected to be extremely hard. Moreover, problems arising in applications frequently involve additional constraints or provide further information. Therefore, we also consider containment problems with extra constraints, for instance in Section 4.2, where we address the $\mathrm{MCP}^k_{\mathrm{Hom}}$ with the centers c_i depending on each other.

Strictly speaking, we should always address the above problems as *outer* containment problems *for point sets*. For better readability and since we consider these cases in the major part of this thesis, we use the shorter notation. Nevertheless, other settings are possible. In Section 4.4, we address $\mathcal{V}$-polytopes as input, that is we try to cover the convex hull of a point set with the containers, and in Section 5.3, we consider an inner containment problem where the input is a set of ellipsoids. All those problems are covered by the general classification for containment problems introduced in Section 2.1.

Well studied containment problems include the minimum volume bounding box [21] and minimum volume enclosing ellipsoid [104], [127] of a point set. Extending the notation used above, these are containment problems under affinity. The 2-containment variants of these problems are also subject to research, including different objective functions [12], [26], [125]. Another example of outer containment under affinity is the computation of minimum volume enclosing simplices [67], [114].

2.4 Fundamental Terms

For the remainder of this chapter, we mention some basic terms from the areas of Computational Convexity, Algorithms, Complexity Theory, and Optimization we deal with in this thesis.

2.4.1 Computational Convexity

Convex containment problems in general dimension belong to the field of Computational Convexity (compare [67]).

Radii

A class of geometric measures strongly related to containment questions are geometric radii.

Definition 2.1 (outer j-radius)**.** *Let B be a convex body, and C a container. For any $1 \leq j \leq n$, the* outer j-radius of B with respect to C *is the infimum value $\rho \geq 0$, such that B is contained in $F + \rho C$ and F is a $(n-j)$-dimensional affine subspace of $\mathbb{R}^n$.*

Again, we may write minimum instead of infimum here since B and C are compact. When C is the Euclidean unit ball, we refer to Euclidean radii. Note that, similarly, one can define the inner j-radii of a convex body. See [65] and [66] on radii of convex bodies and their computation.

Solving an instance of containment under homothety is therefore equivalent to computing the optimal outer n-radius and center. When C is the Euclidean unit ball, the radii can be cast as containment under similarity except that the containers here are infinite for $j < n$ and therefore not compact (compare Section 2.3).

Core-Sets and Helly Numbers

The notion of *core-sets* in geometry inspired many recent publications, e.g., [18], [19], [43], [73], [75], [102], [141]. When considering containment problems or related questions for point sets, the idea is to extract a (desirably small) subset of points providing a good approximation of the complete set. Whether a given subset of P allows a good approximation naturally depends on the measure to be computed. Therefore, core-sets are defined with respect to a specific containment problem. In this thesis, we consider core-sets for 1-containment and k-containment problems under homothety [18], [19], [102] and for smallest enclosing cylinders [141]. See [4] for a survey on core-sets and the related concept of kernels.

Core-sets are strongly related to Helly-type theorems and Helly numbers. For illustration, an alternative formulation for Helly's theorem [79] is that for any point set P and any container C, $n+1$ points exist that suffice to find the optimal value of an instance of the $\mathrm{MCP}_{\mathrm{Hom}}$. This implies that for the $\mathrm{MCP}_{\mathrm{Hom}}$, a core-set of size $n+1$ exists yielding the exact solution (and not just an approximation). When the container is the Euclidean unit ball $\mathbb{B}$, we cannot do with less than $n+1$ points for the exact solution in general. We may also say that $\mathbb{B}$ has Helly number $n+1$. The cube, however, has Helly number 2 [30], implying that a point set P is covered by a cube if and only if each pair of points is covered. This immediately implies that any pair p_i, p_j of points in P maximizing $R_{\mathrm{Hom}}(\{p_i, p_j\}, [-1,1]^n)$ is a core-set providing us with the exact solution.

We are especially interested in the existence of core-sets whose size depends only on the desired approximation quality and is independent of both the dimension and the number of points in P. For some of the problems we consider, the running time of the best known exact algorithms depends exponentially on the input dimension. In these cases, polynomial time computation of dimension-independent core-sets enables approximation algorithms whose running times are only polynomial in the input dimensions. We address such questions in Sections 3.1, 4.3, 4.4, and 5.2, but before, we need to specify the terms from complexity theory used here.

2.4.2 Complexity and Algorithms

We assume the Turing machine or bit model of computation when making statements about complexity. See [56] on the Turing model and $\mathbb{NP}$-completeness. When talking about $\mathbb{NP}$-hardness and membership in $\mathbb{NP}$, we assume that we deal with the decision version of a containment problem, though we usually state them in the optimization version. Within this thesis, we use reductions from and to satisfiability problems. In particular, we show polynomiality by reducing problems to 2-SAT in Sections 4.1.2, 4.2.1, and 4.2.2, and prove $\mathbb{NP}$-hardness by reduction from 3-SAT in Sections 5.1.2 and 5.3.3.

When no fast method computing the optimal solution is at hand, one may settle for suboptimal solutions. We therefore consider approximation algorithms for optimal containment problems. For a given minimization problem and $\delta \geq 1$, we say that an algorithm computes a *factor δ approximation*, if the algorithm outputs a feasible solution that is at most by a factor of δ larger than the optimal value [134] (and vice versa for maximization problems). An algorithm is called a *polynomial time approximation scheme* (PTAS) if it determines a factor $(1+\varepsilon)$ approximation for any $\varepsilon > 0$ in polynomial time in the input size for any fixed $\varepsilon > 0$.

Of course, a theoretically efficient algorithm need not always be the best choice for an implementation. Worst-case super-polynomial methods can be extremely fast for "real-world" input (for instance the simplex method mentioned in the next section) and an algorithm with proven linear running time may be impractical due to huge constants. We therefore point out practical issues besides the theoretical complexity where we consider it appropriate in this thesis.

2.4.3 Optimization

Some containment problems as well as subproblems occurring in containment algorithms can be cast as special optimization problems.

LP and LP-type

A *linear program* (LP) in standard form is determined by a linear objective function which is to be minimized subject to a set of linear constraints. Widely used algorithms for LP are simplex and interior-point methods. See, for instance, [109] on LP and examples.

The notion of *LP-type problems* describes a class of (nonlinear) optimization problems satisfying a number of abstract conditions. The key observation is that this is sufficient to be able to apply simplex-like algorithms to these problems. A famous example is the Euclidean 1-center problem. See [57] on LP-type problems and Section 5.1.2 for another example.

QP, SOCP, and SDP

The arguably simplest generalization of LP is minimizing a convex quadratic objective function, resulting in a *convex quadratic program* with linear constraints (QP). We consider the more general *second-order cone programs* (SOCP), too, which may also involve quadratic constraints of the type that a vector of variables is in the cone of second order. A *semidefinite program* (SDP) may also contain constraints requiring that a matrix of variables is in the cone of symmetric positive semidefinite matrices. All SOCPs can be formulated as SDPs and both subsume linear and quadratic programming. On examples of SOCP see [106]; for SDP [80], [107], and [132]. LP, SOCP, and SDP are examples of optimizing over self-dual cones. The most common approach to such problems are interior-point methods [34], [112].

General Convex Programming

Besides the special cases of LP, SOCP, and SDP, we deal with optimization problems that cannot be formulated in such a scheme but are at least convex, that is, a convex objective function is minimized over a closed convex feasible region. By means of the ellipsoid method, convex programming problems in suitable representation and under mild boundedness assumptions are computationally tractable (see [69]). Essentially, an oracle to compute the objective function value and its subgradient at a given point as well as a separation oracle for the feasible region are required. However, in practice, almost exclusively other algorithms are used since the ellipsoid method is slow even for medium size problems [112]. Interior point methods for the general convex programming problem and especially practical algorithms are still subject to research (compare [34]).

Integer Programming

Moreover, we come across programs with integer constraints imposed on the variables. Even *integer linear programming* (ILP) is $\mathbb{NP}$-hard. See, for instance, [120] on ILP. *Mixed integer programming* (MIP) refers to programs where only some of the variables underlie the integer constraints. The most common methods for this kind of problems are branch-and-bound (B&B) and branch-and-cut algorithms.

3 1-Containment under Homothety

The first type of containment problems we consider in detail is the $\mathrm{MCP}_{\mathrm{Hom}}$, the 1-containment problem under homothety for point sets. We are interested in this problem not only for its occurring in applications itself, but also because it is the base case in solving harder containment problems. Therefore, particular attention is paid to approximation methods in practice. This chapter resumes joint work with René Brandenberg which is currently accepted for publication [37].

For suitable container representations, 1-containment under homothety for point sets is a convex programming problem (see Section 2.4.3) and is tractable by means of the ellipsoid method. In some cases, for instance with polytopal or ellipsoidal containers, specialized methods are available. The $\mathrm{MCP}_{\mathrm{Hom}}$ for balls and polyhedra is first considered in [49]. Here, we also present a cutting plane algorithm applying to the general case when the container is represented in terms of a separation oracle. Finally, the practical performance of the methods is discussed providing examples and experiments.

Since we require containers to be convex, the $\mathrm{MCP}_{\mathrm{Hom}}$ for a point set P is equivalent to the 1-containment problem for the convex hull of P. Of course, this does not hold for k-containment problems where covering $\mathcal{V}$-polytopes is a different problem, see Section 4.4.

3.1 Methods for Special Container Shapes

In this section, we review methods to solve or approximate the $\mathrm{MCP}_{\mathrm{Hom}}$ for some special container shapes, namely polytopes, Euclidean containers, and combinations thereof.

3.1.1 Polytopes

When C is restricted to facet or vertex presented polytopes, the $\mathrm{MCP}_{\mathrm{Hom}}$ can be formulated as an LP [66].

$\mathcal{H}$-Polytopes

In this case, C is presented as an intersection of half-spaces. We have assumed that $0 \in \mathrm{int}(C)$ for all containers C, so the offsets of the half-space defining inequalities can be normalized to 1. Let $C = \bigcap_{i=1}^{k} \{x : a_i^T x \leq 1\} \subset \mathbb{R}^n$. Then

$$P \subset c + \rho C \quad \Leftrightarrow \quad \langle a_i, p_j - c\rangle \leq \rho \text{ for all } 1 \leq i \leq k \text{ and } 1 \leq j \leq m.$$

Therefore, the smallest ρ with $P \subset c + \rho C$ for any $c \in \mathbb{R}^n$ is the optimal solution of the following LP (compare [66]):

$$\begin{aligned} \min\ &\rho \\ \rho + \langle a_i, c\rangle &\geq \max_{1 \leq j \leq m} \langle a_i, p_j\rangle \qquad 1 \leq i \leq k \\ \rho &\geq 0 \end{aligned} \tag{3.1}$$

Program (3.1) has $n + 1$ variables (ρ and c) and k inequalities (not counting non-negativity constraints). Note that the number of points in P affects only the right hand side computations of the LP, not the size of the program itself. For parallelotopal containers C, and especially when C is the cube, it is sufficient to check whether all pairs of points in P can be covered. That is, $R_{\text{Hom}}(P, C) = \max_{1 \leq j_1, j_2 \leq m} \langle a_i, p_{j_1} - p_{j_2}\rangle$ and it suffices to compute these values to determine the optimal radius.

$\mathcal{V}$-Polytopes

If the container is a polytope in vertex-representation, $C = \operatorname{conv}\{w_1, \ldots, w_k\}$ it holds that for any $\rho > 0$,

$$P \subset c + \rho C \quad \Leftrightarrow \quad \frac{1}{\rho}(p_j - c) \in \operatorname{conv}\{w_1, \ldots, w_k\} \text{ for all } 1 \leq j \leq m.$$

If $\dim(P) > 0$, the optimal radius is surely positive, and by setting $\rho' = 1/\rho$ and $c' = c/\rho$ we can again find an LP representation [66]:

$$\begin{aligned} \max\ &\rho' \\ \rho' p_j - c' - \sum_{i=1}^{k} \lambda_{ij} w_i &= 0 \qquad 1 \leq j \leq m \\ \sum_{i=1}^{k} \lambda_{ij} &= 1 \qquad 1 \leq j \leq m \\ \lambda_{ij} &\geq 0 \qquad 1 \leq i \leq k,\ 1 \leq j \leq m \end{aligned} \tag{3.2}$$

Program (3.2) has $km+n+1$ variables and $m(n+1)$ constraints (not counting non-negativity constraints). Consequently, the size of the LP depends quadratically on m, the number of points in P, since both the number of variables and the number of constraints are linear in m. Provided that we can choose between the vertex and the hyperplane representation for the container, and the sizes of the two representations are of the same order of magnitude (for instance when C is a simplex), we surely prefer LP (3.1) to LP (3.2). Yet, when the hyperplane representation of C is significantly larger than the vertex representation, we cannot apply the formulation in (3.1) directly. For instance when the container is the cross-polytope , it is impossible to use the $\mathcal{H}$-presentation even in moderate dimensions, but one is not forced to using LP (3.2) either. We present a cutting plane method based on $\mathcal{H}$-polytopal approximations of the container in Section 3.2.2. The experiments in Section 3.3.1 show that, even for the cross polytope, the cutting plane method is much faster than using LP (3.2).

3.1.2 Euclidean Containers

As mentioned before, the MCP_{Hom} is the well-known 1-center or smallest enclosing ball problem when the container is the Euclidean unit ball $\mathbb{B}$. It has attracted considerable attention and many different solution or approximation techniques are available. Their adequacy depends on the dimension, the number of points and the desired level of accuracy.

In [52], geometric properties of the Euclidean 1-center are used, especially the fact that at most $n + 1$ points define the ball uniquely and are situated on its surface. The problem is LP-type (see also [136]), and the resulting combinatorial algorithm computes the exact center and squared radius. It has exponential worst-case running time, yet it works well in practice and is widely accepted as the best solution method in the Euclidean setting. In [58], an exact solver for general convex quadratic programs is presented, and the experiments include a comparison of different solvers for the smallest enclosing ball.

The minimum enclosing ball problem can also be formulated as a second-order cone program

$$\min \rho, \text{ s.t. } \|p_j - c\| \le \rho, \qquad 1 \le j \le m,$$

and tests show that implementations can compete with the LP-type algorithms in terms of running time [102], [142].

We have already shortly addressed core-sets for geometric optimization problems in Section 2.4.1. In the case of the Euclidean 1-center problem, a very easy incremental core-set algorithm has been stated in [19] and improved in [18] and [102]. The principle is the following: Starting with an arbitrary point, compute a suitable approximation of the smallest enclosing ball for the core-set in each iteration and check whether the ball with its radius enlarged by a factor of $(1 + \varepsilon)$ already covers P, if not, add an uncovered point to the core-set. One can show that the radius increases in each iteration by a constant factor depending on ε, and therefore, the number of iterations and the number of points in the core-set also depend only on ε. This algorithm is important for Euclidean k-center computations, too, so we get back to it in Chapter 4. A similar analysis yields a basic subgradient method for the 1-center problem [18]. Two related algorithms are introduced in [140].

3.1.3 Combined Containers

We can now combine the linear and second-order cone constraints to formulate LPs and SOCPs for more complicated containers.

Intersections

When the container is given as the intersection of Euclidean balls and polytopes, we can formulate the $\mathrm{MCP_{Hom}}$ as follows. Suppose $C = \bigcap_{i=1}^{k}(c_i + r_i\mathbb{B}) \cap \bigcap_{i=1}^{l} Q_i$, where the Q_i are polytopes in $\mathcal{H}$- or $\mathcal{V}$-presentation.

$$\begin{aligned} &\min \rho \\ \|p_j - c - \rho c_i\| &\leq \rho r_i \qquad 1 \leq i \leq k,\ 1 \leq j \leq m \\ p_j - c &\in \rho Q_i \qquad 1 \leq i \leq k,\ 1 \leq j \leq m \end{aligned}$$

The $p_j - c \in \rho Q_i$ conditions can be expressed as linear constraints using the same manipulations as in Section 3.1.1.

Hulls and Minkowski Sums

Consider the kind of 2D shapes mentioned in Sections 1.1.2 and 1.1.1 for the robot placement and the sender location problems. They can be cast as the Minkowski sum of a box and a ball and as the intersection of a cone and a ball (compare Figure 1.2). We can also express the containment problems for these type of shapes in terms of SOCPs.

Let $C = \mathrm{pos}(t+Q) \cap \mathbb{B}$ be the intersection of the positive hull of either a Euclidean ball, a $\mathcal{V}$- or $\mathcal{H}$-polytope Q (where $t \in \mathbb{R}^n$, $t \notin \mathrm{int}(-Q)$) and a Euclidean ball. For $\mathcal{H}$-polytopes and balls, it can be cast as follows:

$$\begin{aligned} &\min \rho \\ p_j - c - \lambda_j t &\in \lambda_j Q \qquad 1 \leq j \leq m \\ \|c + p_j\| &\leq \rho \qquad 1 \leq j \leq m \\ \lambda_j &\geq 0 \qquad 1 \leq j \leq m \end{aligned}$$

For $\mathcal{V}$-polytopes, we can write the constraints as $p_j - c \in \lambda_j t + \lambda_j \,\mathrm{conv}(w_1, \ldots, w_k)$, $\lambda_j \geq 0$. Since this is equivalent to $p_j - c \in \mathrm{pos}(t + w_1, \ldots, t + w_k)$, we get

$$\begin{aligned} &\min \rho \\ p_j - c &= \sum_{i=1}^{k} \mu_{ij}(t + w_i) \qquad 1 \leq j \leq m \\ \|c + p_j\| &\leq \rho \qquad 1 \leq j \leq m \\ \mu_{ij} &\geq 0 \qquad 1 \leq i \leq k,\ 1 \leq j \leq m \end{aligned}$$

Suppose now that C is the Minkowski sum of a number of polytopes and a Euclidean ball, that is, $C = \sum_{i=1}^{k-1} Q_i + \mathbb{B}$, where $Q_1, \dots, Q_{k-1}$ are polytopes in $\mathcal{H}$- or $\mathcal{V}$-presentation. Since

$$p_j \in c + \rho C \quad \Leftrightarrow \quad p_j - c = \sum_{i=1}^{k} x_{ij} \text{ and } x_{ij} \in \rho Q_i, x_{kj} \in \rho\mathbb{B}$$

we can again express the problem as an SOCP. Of course, we can also skip the ball in this formulation. The following are two special cases of such Minkowski sums.

When C is a zonotope, that is the Minkowski sum of a number of line segments, $C = \sum_{i=1}^{k} [\alpha_i, \beta_i] z_i$ we obtain the following LP with ρ' and c' as in (3.2):

$$\begin{aligned} &\max\ \rho' \\ \rho' p_j - c' - \sum_{i=1}^{k} \mu_{ij} z_i &= 0 && 1 \le j \le m \\ \mu_{ij} &\in [\alpha_i, \beta_i] && 1 \le i \le k,\ 1 \le j \le m \end{aligned}$$

In case C is the outer parallel body $Q + \mathbb{B}$ of an $\mathcal{H}$-polytope $Q = \bigcap_{i=1}^{k} \{x \in \mathbb{R}^n : a_i^T x \le 1\}$, it suffices to solve the following SOCP:

$$\begin{aligned} &\min \rho \\ \|p_j - c - x_j\| &\le \rho && 1 \le j \le m \\ \langle a_i, x_j \rangle &\le \rho && 1 \le i \le k,\ 1 \le j \le m \end{aligned}$$

3.2 General Container Shapes

We now proceed to more general container shapes.

3.2.1 Previous Work

For general containers in suitable representation, the ellipsoid method can be used to show polynomiality of the 1-containment problem. In [66] it was shown that any instance of the $\mathrm{MCP}_{\mathrm{Hom}}$ can be approximated up to any given accuracy within polynomial time, if C is the unit ball of an l_p-space with $p \in \mathbb{N}$. The variant of the ellipsoid method stated there only requires a bounding body $B \supset C$ (e.g., a ball or box) and a separation oracle for C. The polynomiality proof needs a strong separation oracle. As stated in [112], the ellipsoid method is hardly a practical tool.

So far, practical implementations for general containers are rarely subject to research. Since the Euclidean case is well-studied both theoretically and practically, a natural question to ask is whether the fast algorithms for this problem generalize. Extensions of both the LP-type and the core-set method addressed in Section 3.1.2 to more general containers may be possible, however both methods essentially rely on a “half-space lemma” [31], [19, Lemma 2.2] which is not applicable if the container is not an ellipsoid. When the container is the Euclidean ball, the center of the smallest enclosing ball is unique. The “half-space lemma” states that any closed half-space containing the center of the smallest enclosing ball also contains a point from P lying on the boundary of the ball. Naturally, this holds for all ellipsoidal containers, since the statement is invariant under affine transforms. It does not hold for other than Euclidean containers, since, for $n \geq 3$ and any other symmetric container C, there exist point sets $P \subset \mathbb{R}^n$ and corresponding centers c such that $c \notin \operatorname{conv}(P)$ [98].

The existence of core-sets for more general containers whose size depends only on ε is an interesting question both from a theoretical and practical viewpoint. If P is a regular simplex with center 0 and $C = -P$, we need a subset of P of size depending on the dimension n to approximate the containment factor up to a given constant. So dimension independent core-set sizes for arbitrary combinations of P and C are impossible. For general, symmetric containers it is not known whether such core-sets exist. The incremental core-set algorithm does not provide the desired result, since it may generate core-sets whose size grows with the dimension if applied to other than Euclidean containers.

In the following, we introduce a cutting plane algorithm approximating the $\mathrm{MCP}_{\mathrm{Hom}}$. For previous work on cutting plane algorithms for this problem, see [115], where a general purpose cutting plane method is introduced and, among other problems, applied to the $\mathrm{MCP}_{\mathrm{Hom}}$.

3.2.2 Cutting Plane Algorithm

In order to be able to handle the container C in a cutting plane algorithm, we need to find suitable cutting planes. This can be done with help of a separation oracle, which also enables us to present the results as general as possible.

Separation Oracles

A strong separation oracle for C, given a point p, either identifies $p \in C$ or returns a vector a such that $\max_{x \in C} \langle a, x \rangle = 1$ and $\langle a, p \rangle > 1$ [69].

Efficient strong separation oracles can be provided for a huge class of presentations of convex bodies. For instance, whenever a subgradient of the objective $\varphi(x) = \min\{\rho : P \subset x + \rho C\}$ can be computed, we have such an oracle. Note that the ability to provide a separation oracle for C naturally relies on the representation of C. For instance, a separation oracle can be hard to deduce when C is represented as the integer hull of a polytope.

Lemma 3.1. *Let $a \in \mathbb{R}^n$ such that $-a$ is a subgradient of $\varphi(x) = \min\{\rho : P \subset x + \rho C\}$ at c, that is $\varphi(x) - \varphi(c) \geq \langle a, x - c\rangle$ for all $x \in \mathbb{R}^n$. Then a is an outer normal of a hyperplane supporting $c + \rho C \supset P$ at a point $p_j \in P$.*

Proof. The vector $-a$ is a subgradient of φ in c if and only if $(-a, -1)$ is an outer normal of a hyperplane supporting the epigraph of φ, $\operatorname{epi}(\varphi)$, in $(c, \varphi(c))$. The level sets of $\operatorname{epi}(\varphi)$ are $\bigcap_j (p_j - \rho C)$, so $-a$ is an outer normal of a hyperplane supporting $p_j - \rho C$ in c for some $p_j \in P$. Equivalently, a is an outer normal of a hyperplane supporting $c + \rho C$ at p_j. ☐

Like the ellipsoid algorithm, the following cutting plane method only requires an initial bounding body and a strong separation oracle for C. When only a weak separation oracle is available, we can perform binary search for the appropriate dilatation factor by calling the oracle successively, until we reach a sufficient approximation quality.

Algorithm

Let $P = \operatorname{conv}\{p_1, \ldots, p_m\}$ and let $\varepsilon > 0$ be the desired accuracy. During the run of the cutting plane algorithm, Algorithm 3.1, we maintain an $\mathcal{H}$-polytope H which is an outer approximation of the container C. At first, H is assigned an arbitrary $\mathcal{H}$-polytope containing C. In the following we call H the *bounding polytope*. As long as the approximation H of C is not sufficient, it will be refined adaptively by cutting hyperplanes. Let ρ, c be the solution of the $\mathrm{MCP}_{\mathrm{Hom}}$ with input P and H obtained via linear programming using the formulation from (3.1). As $C \subset H$, the value of ρ is a lower bound for the minimal radius $R_{\mathrm{Hom}}(P, C)$ of the $\mathrm{MCP}_{\mathrm{Hom}}$ with input P and C (see Figure 3.1).

Obviously, even if the objective φ is not given in an explicit form, $\varphi(c)$ for any fixed c can be computed using the oracle. Hence, we can easily check for every point p_j of P if $p_j \in c + \rho C$, and if not, compute $\bar{\rho} = \varphi(c)$, which is an upper bound for $R_{\mathrm{Hom}}(P, C)$.

If $\bar{\rho}/\rho \leq 1 + \varepsilon$ we are done. Otherwise, we refine the bounding polytope to $H \cap \{x : \langle a_{k+1}, x\rangle \leq 1\}$ where $\{x : \langle a_{k+1}, x\rangle \leq \bar{\rho}\}$ is a hyperplane supporting $\bar{\rho}C$ at a point $p_j - c \in P - c$, and continue iterating.

Finally, as each bounding polytope H is a subset of the preceding one, the sequence of lower bounds ρ is increasing. The sequence of upper bounds $\bar{\rho}$, though, need not decrease monotonically (see Figure 3.2). Hence, it makes sense to store the best obtained upper bound and the corresponding center.

Discussion

The presented cutting plane algorithm is intuitive as its approximation of the optimal value is based purely on polytopal approximations of the underlying container

Algorithm 3.1 Cutting plane algorithm

Input: $P = \operatorname{conv}\{p_1, \ldots, p_m\}$, C via separation oracle, $H = \bigcap_{i=1}^{k} \{x : \langle a_i, x \rangle \leq 1\}$ a bounding polytope of C, and $\varepsilon > 0$

Output: ε-approximation $\bar{\rho}$ of $R_{\mathrm{Hom}}(P, C)$ and corresponding center $\bar{c}$

```
$\bar{\rho} = \infty$
loop = **true**
**while** loop **do**
    solve the LP: $\rho_* := \min\{\rho : \ \rho + \langle a_i, c \rangle \geq \max_j \langle a_i, p_j \rangle \ \forall\, i\}$
    **if** $p_j - c \in \rho_* C \ \forall j$ **then**
        set $\bar{\rho} = \rho_*$
        $\bar{c} = c$
        loop = **false**
    **else**
        compute $\bar{\rho} = \min\{\bar{\rho}, \varphi(c)\}$
        set $\bar{c}$ to the corresponding $c$
    **end if**
    **if** $\bar{\rho}/\rho_* \leq 1 + \varepsilon$ **then**
        set loop = **false**
    **else**
        get $a_{k+1}$ from the strong separation oracle with input $(p_j - c)/\bar{\rho}$
        set $H = H \cap \{x : \langle a_{k+1}, x \rangle \leq 1\}$
    **end if**
**end while**
```

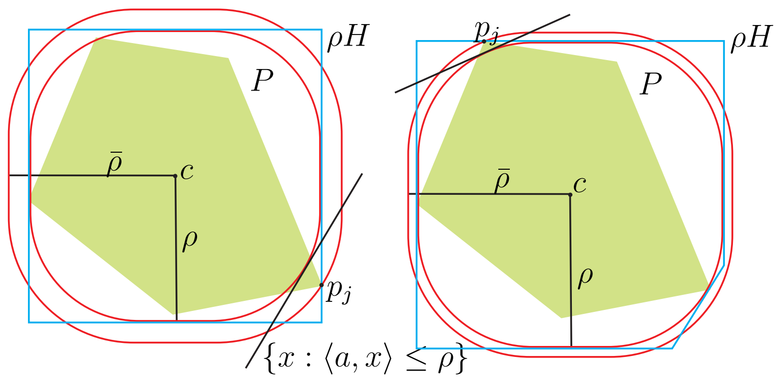


Figure 3.1: Two iterations of the cutting plane method: Here C is the Minkowski sum of a square and a disc and P contains the vertices of the green polygon. In the first step (left picture) H is the square circumscribed C. Obviously, c and ρ are an optimal solution for $\mathrm{MCP}_{\mathrm{Hom}}$ with P and H as input. Since p_j is a vertex with maximal distance to c (with respect to φ) the cut $\{x : \langle a, x \rangle \leq 1\}$ supporting C in $(p_j - c)/\bar{\rho}$ will be added to H. The right picture shows the same scene for the updated H. Surely, we get a better lower bound and here a better upper bound, too.

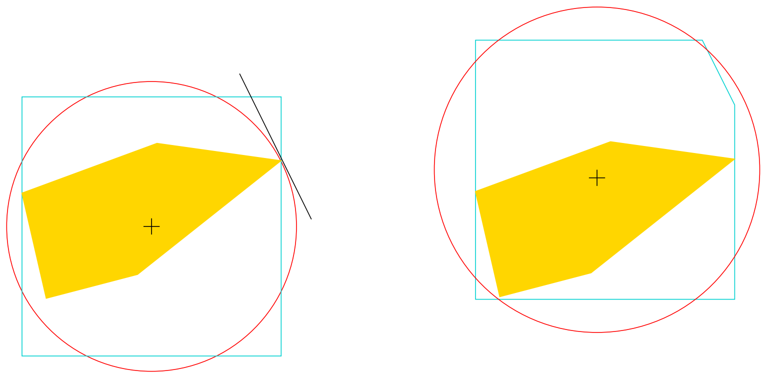


Figure 3.2: An example of the cutting plane method where C is the Euclidean unit ball and P is indicated by the vertices of the yellow polygon. The bounding polytope H is indicated in blue. The upper bound $\bar{\rho}$ (the radius of the red circle) increases after adding a cut.

C. In contrast, general purpose cutting plane methods try to generate better linear approximations of the involved convex constraints or the epigraph of the objective function. This is the case for the cutting plane approach described in [115], although this method can be reduced to operate directly on the container C in case of the $\mathrm{MCP}_{\mathrm{Hom}}$.

One may understand the cutting plane approach as a dual core-set method. Instead of finding a subset of the points in P with almost the same radius, the cutting plane method looks for a small subset of the hyperplanes describing C where it is essential for the containment.

It is possible to prove a theoretical upper bound on the running time of the cutting plane algorithm which is linear in m since the size of P is only involved in computing the two maxima for the right-hand side of the LP and the new upper bound. However, the method may not be polynomial in n. Essentially the number of iterations is bounded by $O(1/(q\varepsilon)^{n-1})$, where $q > \varepsilon$ is the ratio between the smaller and the bigger radius of the smallest annulus containing C. This can be done by estimating the number of possible hyperplane directions on the unit sphere for a given accuracy ε. Therefore, the same bound is valid for an a-priori approximation of C by a polytope Q in hyperplane representation, such that $Q \subset C \subset (1+\varepsilon)Q$. This shows that the algorithm converges and has a polynomial number of iterations in fixed dimensions.

However, since the analysis above does not pay credit to the adaptive principle of the cutting plane algorithm, the a-priori upper bound on the number of iterations is much too pessimistic compared to the computational results. For example, approximating the 3-dimensional Euclidean ball ($q = 1$) via equilateral triangulation by an $\mathcal{H}$-polytope consisting of 512 hyperplanes yields $\varepsilon = 0.03$. In 4-space, using barycentric subdivision [139], 9216 hyperplanes only suffice for $\varepsilon = 0.45$. The following tests for different data sets P and C show a substantially better general performance of the cutting plane algorithm.

We emphasize that fast methods for the $\mathrm{MCP}_{\mathrm{Hom}}$ are crucial especially when many instances occur within another approximation algorithm. Hence, this usually demands quick results in small dimensions, whereas high dimensions are out of reach anyway. Examples for this can be found in Chapters 4 and 5, where instances of the $\mathrm{MCP}^k_{\mathrm{Hom}}$ and the $\mathrm{MCP}_{\mathrm{Sim}}$ are approximated, creating a huge number of $\mathrm{MCP}_{\mathrm{Hom}}$ instances. Nevertheless, note that in some of the examples there, the problem size of the $\mathrm{MCP}_{\mathrm{Hom}}$ is even too small to benefit from the cutting plane approach. This happens when both the dimension is small and the point sets contain only a few points due to the use of core-set algorithms.

3.3 Experiments and Examples

We have already addressed several possibilities to tackle containment problems apart from the cutting plane algorithm, though some of those approaches only apply to special instances. Still, a natural question to ask is why we do not use a tool for general convex programming for the $\mathrm{MCP_{Hom}}$. In [63] such a framework for unifying convex programming (cvx) is proposed, providing a Matlab©[1] implementation, too. Some instances of the $\mathrm{MCP_{Hom}}$ can easily be formulated in cvx, others would require separate "graph implementations" (see [63] for details). One should keep in mind that the main purpose of the cvx-framework is to simplify the specification. In doing so, the performance is limited by the environment [64, p.6].

The experiments in the following are performed on a SUNW SPARC Sun Fire 440 Workstation (1.3 GHz) with 1.6 GB RAM. A C++ implementation of the cutting plane algorithm and Xpress-MP©[2] as LP-Solver are used. Further details of the implementation are discussed in Section 3.3.4.

A small test on different data sets (see Table 3.1) shows that cvx is not the appropriate choice for this kind of problems as even the smallest example takes more than five seconds.

input	n	3			10			30		
	m	100	1000	10000	100	1000	10000	100	1000	10000
cutting plane	iterations	10	20	11	93	71	49	641	702	751
	time (s)	0.03	0.05	0.11	0.10	0.15	0.68	5.77	8.68	30.46
cvx	time (s)	5.31	75.31	*	6.51	211.18	*	28.05	856.20	*

Table 3.1: The cutting plane algorithm and cvx in exemplary tests. The input polytopes P are samples of $(0,1)$-normally distributed points, the accuracy is 10^{-5} and C is formed by the intersection of five n-dimensional balls.

Secondly, the running time increases noticeably with the number of points in P. Considering the intersection of k balls as container (with notation as in Section 3.1.3), the SOCP formulation yields $O(kmn)$ constraints and variables

$$x_{ij} = c - p_j - \rho c_i \text{ and } \xi_{ij} = \rho r_i \quad \text{for } 1 \leq i \leq k,\ 1 \leq j \leq m$$

where the variables (x_{ij}, ξ_{ij}) underlie the second-order cone conditions. That is why data sets of 10000 points and a C defined by five balls already cause an "out of memory" error (indicated by a '*' in Table 3.1) in cvx. Directly using SeDuMi (the SDP solver inside cvx, see [116], [126]) does not cause the "out of memory" error but needs 1h 45min for the instance with 10000 points in $\mathbb{R}^3$. One can see that

[1]The MathWorks, see `http://www.mathworks.com`

[2]Dash Optimization, see `http://www.dashoptimization.com/`

the performance of the cutting plane algorithm deteriorates noticeably in higher dimensions in this example, but it has the important advantage that the number of constraints in the programs depends only on the number of iterations.

Surely, our implementation is specialized on a specific convex problem and the algorithm itself makes use of the given problem structure. For these reasons and because of the environments being different (Matlab© versus C++), an extensive comparison of the cutting plane approach to cvx (or similar solvers) would be of minor value. Instead, we present some examples to illustrate and analyze the course of the cutting plane algorithm.

3.3.1 Polytopal Containers

When C is a $\mathcal{H}$-polytope, the LP formulation in (3.1) is a convenient way to approach the problem. For a reasonable number of hyperplanes, we do not need the cutting plane algorithm here. In comparison, $\mathcal{V}$-polytopes as containers are more interesting.

In a first experiment (see Table 3.2), we compared the cutting plane algorithm with the direct approach via linear programming given in Section 3.1.1 for $\mathcal{V}$-polytopal containers. As mentioned before, the sizes of those LPs depend quadratically on the number of input points. The first $\mathcal{V}$-polytope that comes to mind is surely the regular cross polytope .

Considering for instance the 30-dimensional cross-polytope, it seems obvious that solving the containment problem via the hyperplane representation is simply impossible and one is forced to use the vertex representation as shown in Section 3.1.1. However, when $m = 10000$, the vertex representation of the cross polytope yields an LP (3.2) with a constraint matrix of about 300000 rows, 600000 columns, and 1500000 nonzero entries, whereas, applying the cutting plane algorithm, one can see that only about 200 of the 2^{30} hyperplanes in the hyperplane representation are enough to solve the containment problem for P. An accuracy of 10^{-14} (about the tolerance of the LP solver) is reached after less than 200 iterations. The algorithm computes a suitable approximation (depending on P) of the cross-polytope in facet representation, and as we can see from the formulation in (3.1), the corresponding LP can be solved much faster. Moreover, upper bounds and cutting planes can easily be generated, so the running times stay small (see Table 3.2). Clearly, we suggest to use the direct approach when the number of points m is very small.

3.3.2 Euclidean Containers

In a second experiment (see Figure 3.3) we considered six differently distributed types of data sets P of different sizes in several dimensions $n \leq 30$ with the Eu-

input	n	10			30		
	m	100	1000	10000	100	1000	10000
cutting plane	iterations	24	24	27	194	196	182
	time (s)	0.04	0.04	0.16	0.67	0.85	4.12
LP (3.2)	time (s)	0.37	15.20	2143.60	2.11	107.42	5594.63

Table 3.2: Running times of the cutting plane method compared to directly solving the linear program in 3.2. The container is the cross polytope in dimension n. The point sets P are samples of $(0, 1)$-normally distributed data points. In both cases, the problem is approximated up to an accuracy of $\epsilon < 10^{-14}$.

clidean unit ball as container C and $\varepsilon = 0.0001$. [3]

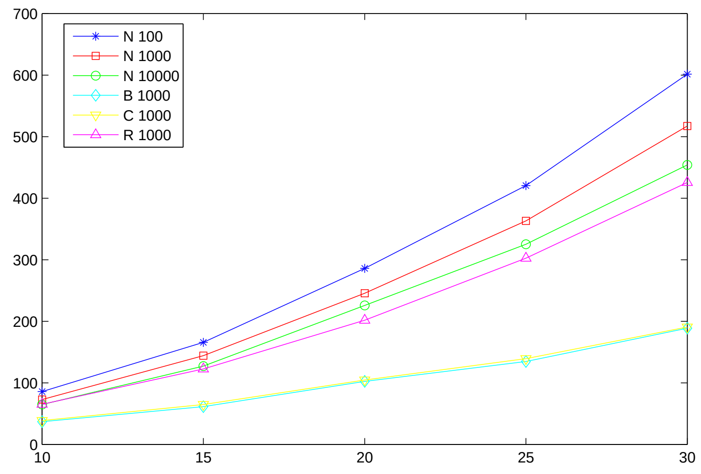


Figure 3.3: Averaged number of iterations for different data sets P in dimensions $n = 10, 15, 20, 25$, and 30. Here, C is the Euclidean unit ball and $\varepsilon = 0.0001$. N – $(0, 1)$ normally distributed data, B – points on the surface of the unit ball, C – vertices of the unit cube chosen at random, R – equally distributed data in $[0, 1]^n$. The numbers 100, 1000, and 10000 refer to the number of points in the input polytope P.

In high dimensions, the cutting plane method with container $\mathbb{B}$ cannot compete with specialized algorithms for the Euclidean 1-center problem. In [52], the implementation described there is compared to those from [102] and [142]. Their tests go up to dimension 2000 which is out of reach for our algorithm. Yet, our program

[3]A previous implementation of a cutting plane algorithm for the Euclidean case is described in [119].

seems to be competitive for moderate dimensions which is sufficient for many applications. For instance, all calculations are finished successfully within 1 second in dimension 10 and calculations with at most 1000 points within 6 seconds in dimension 30. The computations for the largest input sets (10000 points, dimension 30) take about 15 seconds on average. We sample 100 instances for each data point in Figure 3.3.

The results illustrated in Figure 3.3 corroborate that the number of vertices m of P has no influence on the number of iterations performed and therefore it has only secondary influence on the running time. This seems reasonable since we know from Helly's theorem that $n + 1$ points suffice to determine the $\mathrm{MCP}_{\mathrm{Hom}}$ solution. The shape of P on the other hand affects the number of iterations. Points distributed on the surface of a ball or randomly chosen vertices of a cube behave much better with growing dimension than test cases with normally or equally distributed data points. For the normally distributed data sets, bigger input data sets even reduce the number of iterations performed. This observation may be natural, as the shapes where the algorithm performs less iterations are more symmetric or ball-like.

In our tests, we also observe the accuracy (the difference between upper and lower bound) during the iteration process. The tests depicted in Figure 3.4 suggest that it is exponential with a rate depending on the dimension for the examples considered here.

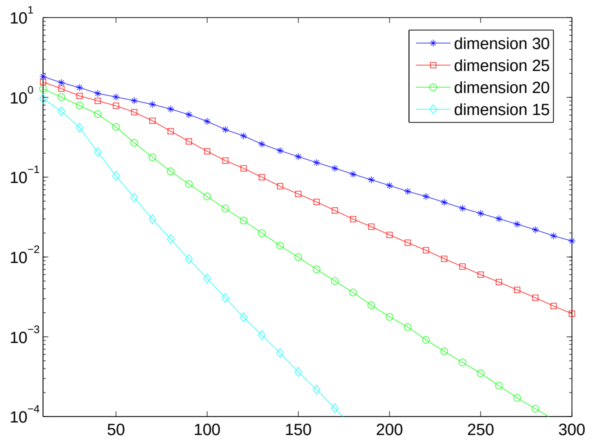


Figure 3.4: Exemplary analysis of the accuracy achieved by the cutting plane algorithm in different dimensions, where C is the Euclidean unit ball and P is again a normally distributed data set of 1000 points. The x-axis shows the number of iterations performed so far. For each dimension, the mean over 100 samples is shown.

3.3.3 Nonlinear, Nonsmooth Containers

We now proceed with non-polytopal, nonsmooth containers. The nonsmoothness may increase the number of iterations, but the algorithm is still stable (for sensible input) and achieves fast results.

In the first example (see Figure 3.5), P consists of 1000 normally distributed points in $[0, 1]^n$, with $n = 3, 10, 30$, $\varepsilon = 0.0001$, and C is the intersection of two Euclidean balls with equal radius. The value of q (here the ratio of the inner and outer radius of C) varies between 0.3 and 0.8. The dependence of the average number of iterations on the value of q is shown. The running times increase a bit with smaller q values, especially in higher dimensions. However, even this increase is by no means as steep as predicted in terms of the upper bound.

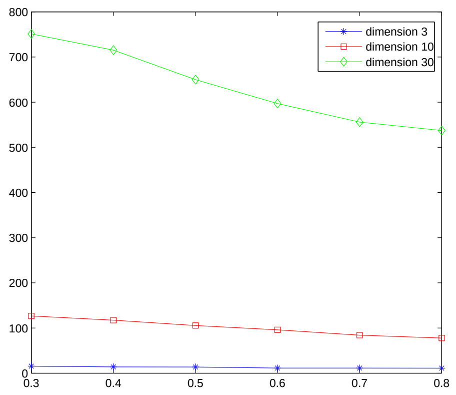


Figure 3.5: Number of iterations for different values of q and different dimensions. The value of q is indicated on the x-axis. For P, 100 samples of 1000 normally distributed points are used.

Figure 3.6 illustrates the average number of iterations where C is the intersection of one to five Euclidean balls. Again, P consists of 1000 normally distributed points and the accuracy is 0.0001. For moderate values of q, the number of balls forming the intersection does not seem to affect the number of iterations (of course, the computing time per iteration increases a bit). The sample size for Figures 3.5 and 3.6 is again 100.

3.3.4 Further Remarks on the Implementation

We use the solver Xpress-MP© with a dual simplex algorithm for the linear programs. Experiments show that this method is superior to interior-point methods here as it starts from the solution of the preceding iteration. Therefore, we do not have to reconsider the whole program again. Though the LP size grows in every iteration, the time consumed per iteration is almost constant.

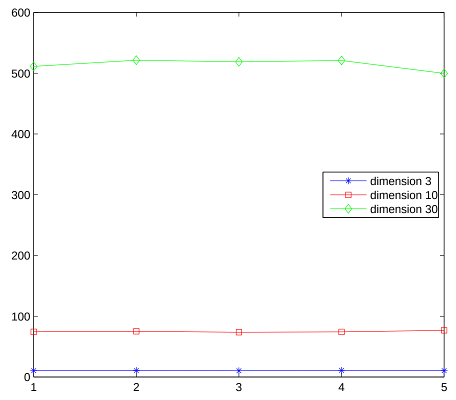


Figure 3.6: Number of iterations for different dimensions, where C is the intersection of 1 to 5 balls. The number of balls is indicated on the x-axis. For P, 100 samples of 1000 normally distributed points are used.

For the initial bounding polytope H of C, boxes, regular simplices, and polytopes with random normal vectors are suggestive choices. Experiments show that a finer initial approximation can reduce the number of iterations. However, though the box is usually a better approximation, the simplex performs slightly better in tests. The reason is probably that this choice allows less ambiguity in the optimal solutions of the linear programs. Box-shaped containers, in contrast, usually permit an $(n-1)$-dimensional set of centers for the optimal radius. Concerning the experiments with random directions, picking $n+1$ random normal vectors is comparable to choosing the simplex. Experiments suggest that choosing twice or even ten times as many vectors adds many unnecessary constraints to the LP, making the algorithm perform slower especially in higher dimensions. For 0-symmetric C, it seems reasonable to not only add the identified cutting plane but also its symmetric counterpart. This leads to a small reduction in the number of iterations in some cases, but on the other hand the sizes of the LPs grow faster.

In order to overcome ambiguities in optimal LP solutions which allow the centers to oscillate, one may try to minimize the distance (measured in the distance function induced by H) between the new center and its predecessor. This involves a second linear program in each iteration. The number of iterations reduces noticeably at least for normally distributed input data, but again, the observed running time deteriorates due to the effort spent for the additional linear programs.

Finally, instead of approximation by cutting planes, more sophisticated approximations, for instance by balls or ellipsoids, may provide faster convergence towards the optimum. This alternate approach results in subproblems where the containers are formed by intersections of ellipsoids and polytopes. Our experiments with such container shapes indicate that solving this type of problems directly using an SOCP solver is currently not competitive to the LP approach (at least, in the dimensions considered). This is apparently due to the fact that the SOCPs have many more constraints, as exclusively in the case of linear constraints, considering just one point (where the maximum is attained) in order to ensure that the whole set P satisfies the constraint is possible. More specialized subroutines than the general SOCP solvers might be helpful to improve the performance of the alternate approach. Yet, as stated before, we are aware of the existence of such methods only for the Euclidean case.

Another idea to improve the performance of cutting plane algorithms is reported in [115]. There, the computing times are improved using an interpolation technique. From our experiments with this technique, we can confirm that the interpolation step reduces the number of iterations a little. However, we observe that the overall running time deteriorates again due to the additional effort spent in each iteration. The reasons may be that the technique is not adequate for minimax approximation or for our point set sizes, since the positive examples mentioned in [115] involve other objective functions and point sets with less than 50 points.

4 k-Containment under Homothety

The major part of the results presented in this chapter is published in [38]. This is also joint work with René Brandenberg.

In the following, we consider the $\mathrm{MCP}^k_{\mathrm{Hom}}$, the k-containment problem under homothety. Recall that we want to cover a point set $P \subset \mathbb{R}^n$ with homothetic copies $c_i + \rho C_i$ of k given, fixed containers C_i, $1 \leq i \leq k$, such that the radius ρ is as small as possible. Here, it is crucial to find out which of the points in the input set P should go to which of the containers, that is, the partition of the point set P. The $\mathrm{MCP}^k_{\mathrm{Hom}}$ is a kind of clustering problem, and we refer to the particular subsets of $P_i \subset P$ clusters. As stated in Chapter 2, a special case of the $\mathrm{MCP}^k_{\mathrm{Hom}}$ is the k-center problem, where the point set P is covered with k identical containers. The cases when the containers are Euclidean balls and unit cubes have so far attracted most attention. It is known that the 2-center problem for balls and the 3-center problem for cubes are $\mathbb{NP}$-hard, whereas the 2-center for cubes can be solved in polynomial time [110]. As far as we know, it is not known whether the 2-center problem for cross polytopes is $\mathbb{NP}$-hard.

In Section 4.1, we summarize methods to partition a point set in order to achieve bounds for the optimal solution of the $\mathrm{MCP}^k_{\mathrm{Hom}}$. In particular, we develop and analyze a diameter partitioning method for the problem with general and even different containers. In some cases, we can even find an optimal partition in polynomial time and therefore solve the $\mathrm{MCP}^k_{\mathrm{Hom}}$ efficiently (see Section 4.2). In this context, we also consider problems where the container centers underlie additional constraints. In Section 4.3, we show how to approximate the general $\mathrm{MCP}^k_{\mathrm{Hom}}$ using a branch-and-bound scheme, and how to further accelerate this method. Finally, in Section 4.4, we consider a variant of the problem where not a discrete point set but its convex hull is to be covered. Of course, when $k > 1$, this is not equivalent to the problem for point sets.

4.1 Partitioning Procedures

Consider a k-containment problem for a point set P and assume that we know an optimal assignment of the points to k clusters. In that case, the problem reduces to k independent 1-containment problems of the type considered in Chapter 3. The problem can be treated efficiently for many container classes, and in addition to that, fast practical methods exist, too. Any partition of the point set into k

clusters can therefore be used to compute upper bounds. This section deals with partitioning procedures with provable approximation guarantees.

4.1.1 Previous Work on Partitioning

A simple partitioning procedure for general k-center problems is proposed in [62]. It is a greedy approach: one after another, k points from P, the so-called *heads* of clusters, are chosen, each in a way that it maximizes the distance to the heads already selected. The point set is then partitioned into k clusters, where each point is assigned to its nearest head. It is shown that this method yields an approximation quality of 2 for the general k-center problem. We come back to it in Lemma 4.16.

As stated before, besides k-center clustering, other objective functions are considered in the literature. One possibility is to minimize the diameter of the clusters. This kind of diameter clustering problems is addressed in [14] and [59], and the algorithms we consider are similar to those there, though our problem setting is different.

When the container is a cube, diameter and radius clustering are equivalent. This is because the Helly number of the cube is 2, and therefore, a point set is covered by a cube if and only if any two points are covered. Assume now that the optimal radius is known. Then any two points with a distance larger than this radius have to go to different clusters. Consider the graph with vertices P and edges between all point pairs with distance larger than the desired radius. If and only if this graph is bipartite, P can be covered by two cubes of the desired radius. This observation is used in [110] to show that the 2-center problem for cubes can be solved in polynomial time.

4.1.2 Diameter Partitioning

In the following, we address diameter partitioning for general k-containment problems and prove bounds for the approximation quality. Concerning the practical value of those bounds, see Sections 4.3.3 and 4.3.4.

Before proceeding with the diameter partitioning procedure, we define the generalized half-diameter of a convex body:

Definition 4.1 (half-diameter)**.** *Let B be a convex body, and C a container. The* half-diameter *of B with respect to the container C is defined as*

$$\operatorname{diam}(B,C) = \max\{R_{\text{Hom}}(\{b_1,b_2\},C),\ b_1,b_2 \in B\}.$$

Recall that the distance functional F_C for C satisfies norm properties for symmetric containers C. If and only if C is symmetric, $\operatorname{diam}(B,C)$ is just the usual half-diameter which of course satisfies

$$2\operatorname{diam}(B,C) = \max\{F_C(b_1-b_2),\ b_1,b_2 \in B\}.$$

Whenever C is not symmetric, there are b_1, $b_2 \in \mathbb{R}^n$ such that $F_C(b_1 - b_2) \neq F_C(b_2 - b_1)$.

The ρ-Distance Graph

Consider a k-center problem with container C. In order to partition the point set P into clusters, we consider the pairwise "distance" between points, that is, the dilatation factor needed to cover a point pair with a homothetic copy of C. This information is captured in the ρ-distance graph of the point set P.

Definition 4.2 (ρ-distance graph of (P, C)). *For every $\rho > 0$ we call the graph $G(\rho) = (P, E)$ with edges in E for every pair $\{p, q\}$ with $R_{\text{Hom}}(\{p, q\}, C) > \rho$ the* ρ-distance graph *of (P, C).*

This is merely the complete graph on the vertices P, where all edges of length $\leq \rho$ are removed. The maximal edge length occurring is of course $\operatorname{diam}(P, C)$.

Algorithm 4.1 now computes the minimal ρ such that $G(\rho)$ is k-colorable. Finding a k-coloring of $G(\rho)$ corresponds to partitioning the point set P into k subsets, where no pair of points with $R_{\text{Hom}}(\{p, q\}, C) > \rho$ is within one set. The aim is to prove that such a clustering can never be too bad.

Algorithm 4.1 Diameter partitioning for k-center

Input: P a point set, C a container in suitable representation
Output: ρ a lower bound for $R_{\text{Hom}}(P, C)$

```
let $l$ be the number of pairs $\{p, q\}$ of points in $P$
for $j = 1$ to $l$ do
    compute $\rho_j = R_{\text{Hom}}(\{p, q\}, C)$ where $\{p, q\}$ is the $j$th point pair
end for
sort and relabel such that $\rho_1 \geq \ldots \geq \rho_l$
for $j = 1$ to $l$ do
    if $G(\rho_j)$ is not $k$-colorable then
        break
    end if
    set $\rho = \rho_j$
end for
return $\rho$
```

Surely, if $k \geq 3$ deciding whether a graph is k-colorable is itself a hard problem and no method is known to make Algorithm 4.1 polynomial. Things look better when $k = 2$. In that case, an implementation of Algorithm 4.1 can maintain a 2-coloring of $G(\rho)$ and, with growing ρ, successively insert new edges. One should note that since $G(\rho)$ need not be connected, more than two labels (or colors) may

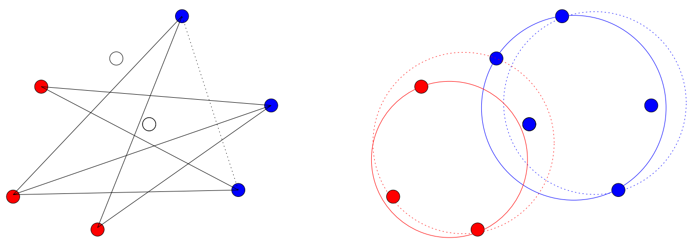


Figure 4.1: Example of diameter partitioning for $k = 2$ and Euclidean containers. On the left, the final ρ-distance graph where the dotted edge is the first to interfere with bipartiteness. The coloring is extended in a greedy manner on all the points, considering the distance to the centers for the points already assigned in the first step. In this example, this is not optimal, and we would get a better radius assigning the points as indicated by the dotted circles.

be necessary. When an edge is inserted which is not connected to the subgraph already built, a new pair of labels is created. When an edge joins two previously disconnected components, the relevant labels are merged.

In computations, an incomplete partition can be extended in a greedy manner upon all points in P, resulting in k clusters P_i such that $P_1 \cup P_2 \cup \ldots \cup P_k = P$. Besides the lower bound output ρ of Algorithm 4.1, an upper bound $\bar{\rho} = \max_i R_{\text{Hom}}(P_i, C)$ is obtained. See Figure 4.1 for an example.

Parallelotopes

If (and only if) C is a parallelotope, for instance the unit cube, the Helly number of C is 2; that is, any set of points P is contained in a translate of C, if and only if every pair of points in P is contained in a translate of C [30, 14.3], for short $R_{\text{Hom}}(P, C) = \operatorname{diam}(P, C)$ for all P. This implies that P can be packed into k translates of ρC if and only if $G(\rho)$ is k-colorable [14], [110]. Hence, Algorithm 4.1 solves the k-center problem for parallelotopes exactly. When the parallelotope is given in $\mathcal{H}$-representation $C = \bigcap_i \{x : a_i^T x \leq 1\}$ and especially when the container C is the unit cube,

$$R_{\text{Hom}}(\{p, q\}, C) = \max_i a_i^T (p - q)$$

and can easily be computed.

Of course, when C is not a parallelotope, and therefore $R_{\text{Hom}}(P, C) \neq \operatorname{diam}(P, C)$ for certain P, using diameter partitioning cannot guarantee optimal solutions. Yet, at least for $k = 2$, Algorithm 4.1 provides useful bounds and can even be altered into a pre-partitioning procedure (see Section 4.3.3).

Euclidean Containers

Let us now consider Euclidean containers.

Lemma 4.3. *Algorithm 4.1 computes a $\sqrt{2n/(n+1)}$-approximation of the optimal radius $R^k_{\mathrm{Hom}}(P,C)$ for any point set P and any ellipsoid C.*

Proof. Algorithm 4.1 outputs the smallest value of ρ such that $G(\rho)$ is k-colorable. Let $P_1,\dots,P_k$ be a partition of P obtained from a k-coloring of $G(\rho)$. Usually, such a partition is not unique. For each P_i, $\mathrm{diam}(P_i,C)$ is a lower bound for the optimal radius $R^k_{\mathrm{Hom}}(P,C)$. We get an upper bound for the optimal radius by computing the radii of the P_i, namely $\max_i R_{\mathrm{Hom}}(P_i,C) \geq R^k_{\mathrm{Hom}}(P,C)$, since $R^k_{\mathrm{Hom}}(P,C)$ is the radius for an optimal partition. Hence, by Jung's inequality [95], we obtain

$$\max_i \mathrm{diam}(P_i,C) \leq R^k(P,C) \leq \max_i R_{\mathrm{Hom}}(P_i,C) \leq \max_i \sqrt{\frac{2n}{n+1}}\,\mathrm{diam}(P_i,C),$$

finishing the proof. □

Remark 4.4. *Algorithm 4.1 is stated without addressing explicitly what happens when edges of equal length occur. In the worst case, when P is a regular simplex and all pairwise distances between vertices are equal, the algorithm might assign all points to the same cluster even when $k=n$. More precisely, consider the vertices of the standard simplex $\{e_1,\dots,e_{n+1}\}$ in $\mathbb{R}^{n+1}$ and let P be the image of these points under the canonical embedding of $\mathrm{aff}(e_1,\dots,e_{n+1})$ in $\mathbb{R}^n$. Then $G(\sqrt{2}/2)$ is the complete graph on $n+1$ vertices and certainly not n-colorable. When all points are assigned to the same cluster, the bound provided by Lemma 4.3 is then tight, since the Euclidean radius of this cluster is $\sqrt{n/(n+1)}$ but $\sqrt{2}/2$ is optimal. See Figure 4.2 for an illustration.*

General, Identical Containers

We proceed with bounds for general containers. For symmetric containers, the bounds get slightly weaker than in the Euclidean case. For nonsymmetric containers, the quality of our bound depends on the Minkowski symmetry of C (compare Section 2.2).

Lemma 4.5. *Algorithm 4.1 computes an $\frac{n}{n+1}(1+\frac{1}{s_C})$-approximation of $R^k(P,C)$ for any point set $P\subset\mathbb{R}^n$ and any container $C\subset\mathbb{R}^n$.*

Proof. Following the proof of Lemma 4.3 it suffices to show that

$$R_{\mathrm{Hom}}(P,C) \leq \frac{n}{n+1}\left(1+\frac{1}{s_C}\right)\mathrm{diam}(P,C)$$

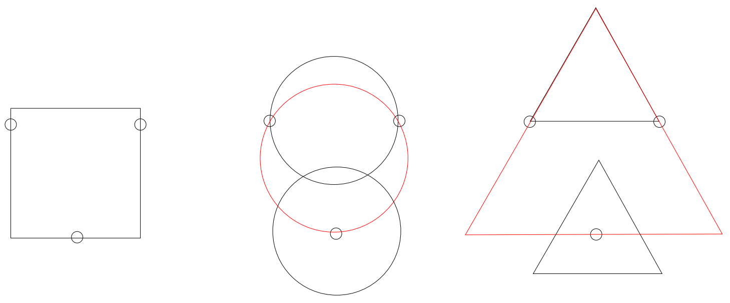


Figure 4.2: Worst case examples for diameter partitioning in 2D, where the container is a cube, Euclidean ball, or regular simplex. An optimal solution is indicated in black. A worst case solution for the diameter partitioning (if different) is indicated in red.

for any point set P. Suppose $\operatorname{diam}(P, C) = 1$, i.e. every two points in P can be covered by a translate of C. It easily follows that every two points of $P - P$ can be covered by $C - C$, and since both $P - P$ and $C - C$ are symmetric $R_{\text{Hom}}(P - P, C - C) = \operatorname{diam}(P - P, C - C) = 1$ [65]. Since $(1 + s_P)P$ can be covered by a translate of $P - P$ and $C - C$ by a translate of $(1 + \frac{1}{s_C})C$, we conclude with $s_P \geq \frac{1}{n}$ that P is contained in a translate of $\frac{n}{n+1}(1 + \frac{1}{s_C})C$. □

Remark 4.6. *If C is symmetric, $s_C = 1$ and a well known inequality about the ratio between the outer radius and the diameter of convex sets (or point sets) in arbitrary Minkowski spaces [29] can immediately be obtained from Lemma 4.5:*

$$\frac{R_{\text{Hom}}(P, C)}{\operatorname{diam}(P, C)} \leq \frac{2n}{n+1}.$$

When C is regarded as the unit ball of the norm of a Minkowski space, $R_{\text{Hom}}(P, C)$ and $\operatorname{diam}(P, C)$ here obviously correspond to the half-diameter and circumradius there.

Remark 4.7. *If the set of containers $\mathcal{C}_n$ is a special subset of the set of convex bodies in $\mathbb{R}^n$, the approximation error may actually be much better than predicted by Lemma 4.5. The bounds improve when a better guarantee on lower bounds for the Minkowski symmetry of the input point set P can be given, too. Yet, the worst case approximation error actually occurs when P is a regular simplex in $\mathbb{R}^n$, $C = -P$, and $k = n$ as in Remark 4.4. The optimal radius is then 1, but the algorithm returns a radius of n in case all points are put in the same cluster. See Figure 4.2 for an example.*

Different Containers

Regarding the general $\text{MCP}^k_{\text{Hom}}$, we also allow for different containers. Now two points p and q which are far apart in the (maybe nonsymmetric) "distance" induced by the function F_C for one container may be close by means of another. We need to pay credit to this by modifying the definition of the ρ-distance graph, Definition 4.2. We now consider edge-colored multigraphs with vertices in P. Each container C_i gets an individual edge set E_i.

Definition 4.8 (ρ-distance graph of $(P, C_1, \ldots, C_k)$). *For every $\rho > 0$ the edge-colored multigraph $G(\rho) = (P, E_1, \ldots, E_k)$ with edges in E_i for all pairs $\{p, q\}$ which satisfy $R_{\text{Hom}}(\{p, q\}, C_i) > \rho$ is called the* ρ-distance graph *of $(P, C_1, \ldots, C_k)$.*

We define a generalized k-coloring of $G(\rho)$ which has to respect the different edge sets:

Definition 4.9 (generalized k-coloring). *Let $G = (P, E_1, \ldots, E_k)$ be an (edge-colored) multigraph with vertex set P and k edge sets $E_1, \ldots, E_k$. A* generalized k-coloring *of G is a partition $P_1, \ldots, P_k$ of the vertices P such that for any $\{p, q\} \in E_i$ it follows $\{p, q\} \not\subset P_i$, $i = 1, \ldots, k$.*

Again, a solution of the generalized k-coloring problem for the ρ-distance graph $G(\rho)$ implies that ρ is a lower bound for the radius $R_{\text{Hom}}(P, C_1, \ldots, C_k)$. Figure 4.3 shows an example of a 2-containment problem with two different boxes and the corresponding ρ-distance graph. Algorithm 4.2 extends the diameter partitioning

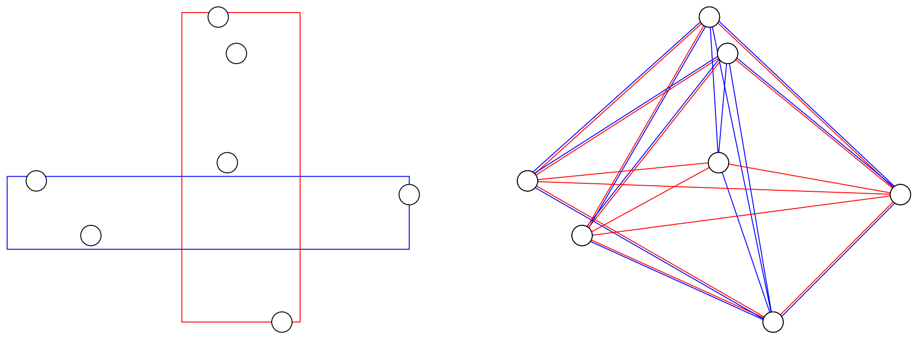


Figure 4.3: An example of an optimal containment with two boxes as containers and the corresponding edges in the final ρ-distance graph.

to k-containment problems. It uses generalized k-coloring as a subroutine.

Generalized k-coloring has stronger conditions than usual coloring as the different edge sets have to be respected. Yet, if $k = 2$, the problem can still be solved efficiently:

Lemma 4.10. *The generalized 2-coloring problem can be reduced to 2-SAT.*

Proof. Assigning boolean variables ζ_i, where $\zeta_i = 1 \Leftrightarrow (p_i \in P_1)$, shows that the generalized 2-coloring instance (P, E_1, E_2) is equivalent to the following instance of

Algorithm 4.2 Diameter partitioning for k-containment

Input: P a point set, containers $C_1, \dots, C_k$ in suitable representation
Output: ρ a lower bound for $R_{\mathrm{Hom}}(P, C_1, \dots, C_k)$

```
let $l$ be the number of combinations of pairs $\{p,q\}$ of points in $P$ and $i \in \{1,\dots,k\}$
for $j = 1$ to $l$ do
    compute $\rho_j = R_{\mathrm{Hom}}(\{p,q\}, C_i)$
end for
relabel such that $\rho_1 \geq \ldots \geq \rho_l$
for $j = 1$ to $l$ do
    if $G(\rho_j)$ has no valid generalized $k$-coloring then
        break
    end if
    set $\rho = \rho_j$
end for
return $\rho$
```

2-SAT:

$$\bigwedge_{\substack{(p_i,p_j)\\ E_1-\text{edges}}} (\neg\zeta_i \vee \neg\zeta_j) \quad \wedge \bigwedge_{\substack{(p_i,p_j)\\ E_2-\text{edges}}} (\zeta_i \vee \zeta_j).$$

A feasible assignment of the ζ_i implies a feasible coloring of the p_i. □

A valid assignment of a 2-SAT instance (or evidence that no valid assignment exists) can be found in time linear in the number of edges, e.g. by the BinSat algorithm [131]. Any coloring of the p_i obtained from such a valid assignment yields a partition into two sets P_1 and P_2 with the following property: $R_{\mathrm{Hom}}(\{p,q\}, C_i) \leq \rho$, $i = 1, 2$ for any pair of points $p, q \in P_i$.

Lemma 4.11. *Algorithm 4.2 computes*

(a) an $\frac{n}{n+1}\left(\max_{1\leq i\leq k}(1/s_{C_i}) + 1\right)$*-approximation for the general* MCP^k_{Hom}*,*

(b) a $\frac{2n}{n+1}$*-approximation for the* MCP^k_{Hom} *if all containers are 0-symmetric,*

(c) a $\sqrt{\frac{2n}{n+1}}$*-approximation for the* MCP^k_{Hom} *if all containers are ellipsoids or parallelotopes,*

(d) and an exact solution of the MCP^k_{Hom} *if all containers are parallelotopes.*

Proof. Algorithm 4.2 computes the smallest value ρ such that a generalized k-coloring for $G(\rho)$ exists. Let $P_1, \dots, P_k$ be a partition of P obtained from such a coloring. We have $R_{\text{Hom}}(P, C_1, \dots, C_k) \leq \max_i R_{\text{Hom}}(P_i, C_i)$. For the first statement, we use

$$R_{\text{Hom}}(P_i, C_i) \leq \frac{n}{n+1} \left(\max_{1 \leq i \leq k} \frac{1}{s_{C_i}} + 1 \right) \operatorname{diam}(P_i, C_i)$$

as in Lemma 4.5. The other three statements follow analogously using the approximation factors obtained for symmetric containers, ellipsoids, and parallelotopes. □

4.2 Polynomial k-Containment Problems

As already mentioned, the k-center problem is already $\mathbb{NP}$-complete for cubes when $k = 3$ and Euclidean balls when $k = 2$ [110]. Nevertheless, some specific k-containment problems can be solved in polynomial time.

4.2.1 2-Containment for Parallelotopes

In [110], it is shown that the 2-containment problem for cubes can be solved in polynomial time when the dimension is part of the input (see Section 4.1). Lemma 4.11 shows that a more general statement is possible: the containers may be two arbitrarily oriented parallelotopes.

Note that solving the 2-center problem for the cube via diameter partitioning is not optimal. A faster algorithm is proposed in [25]. It computes a minimal axis-parallel enclosing box for P and determines the position of the two cubes in this box by maximizing consecutively in the directions of the n coordinate axes. However, Algorithm 4.1 has the advantage of being adaptable to general 2-containment problems as a diameter partitioning scheme and, in addition to that, it works for different parallelotopes. The algorithm in [25], however, is limited to two identical parallelotopal containers.

4.2.2 2-Containment for Axis-Aligned Boxes Sharing a Common Translation Vector

We consider the modified problem where the translations of two containers depend on each other, that is, $t = c_2 - c_1$ is fixed. Equivalently, this is a nonconvex 1-containment problem where the container is the union of the first container C_1 and a translate $t + C_2$ of the second container with $t \in \mathbb{R}^n$. We show that the decision version of this problem can be solved in polynomial time when C_1 and C_2 are axis-aligned boxes. We now seek for a common translation vector c such that $P \subset c + (C_1 \cup t + C_2)$.

First observe that we can check which assignments are valid for any two points p_i, p_j from P. For instance, when p_i should go to the first and p_j to the second

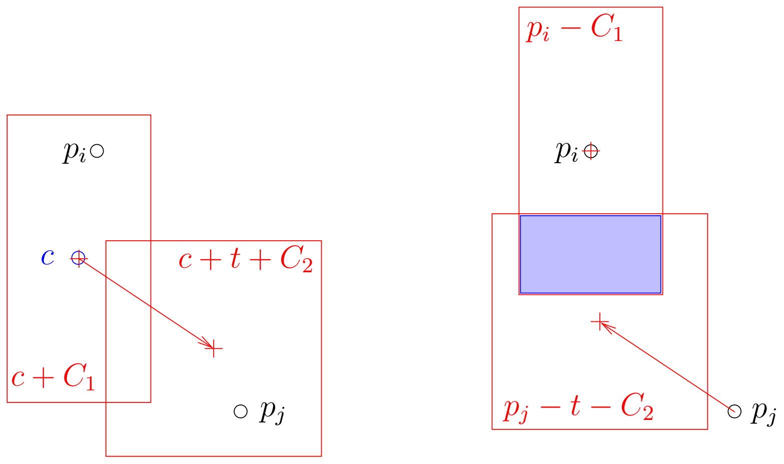


Figure 4.4: Covering two points p_i, p_j with a union of two boxes $c + (C_1 \cup t + C_2)$. We can place p_i in the first and p_j in the second box if and only if the intersection (in blue) of $p_i - C_1$ and $p_j - t - C_2$ is nonempty, and any c from the intersection is feasible.

container, this is equivalent to the existence of a translation vector c such that $p_i \in c + C_1$ and $p_j \in c + t + C_2$. Such a c exists if and only if the intersection of two boxes $p_i - C_1$ and $p_j - t - C_2$ is nonempty (see Figure 4.4).

Now introduce a variable ζ_i for each p_i, such that ζ_i is true if and only if p_i goes to the first container (compare Lemma 4.10). Algorithm 4.3 now checks for each point pair which assignments are valid and solves the resulting 2-SAT problem.

Note that this problem can again be captured as a variant of a graph coloring problem, yet it is more complicated since it involves directed edges. Generating a 2-SAT instance with a set of variables $\{\zeta_i\}$ and a set of clauses $\mathcal{K}$, however, is straightforward. Compared to the 2-SAT instance in Lemma 4.10, we have to consider two additional cases here.

Lemma 4.12. *Algorithm 4.3 solves the decision version of the 2-containment problem for two boxes sharing a common translation vector.*

Proof. Clearly, the conditions checked by Algorithm 4.3 are necessary for covering P with the two boxes. They are also sufficient. This is because any assignment of the variables ζ_i satisfying $\mathcal{K}$ assigns the points in P to two sets such that the intersection of the corresponding sets of feasible translation vectors c for any pair of points p_i, p_j is nonempty. Since the sets of feasible translation vectors are themselves boxes having Helly number 2, this implies that the overall intersection is nonempty. We conclude that a feasible translation vector c exists. ☐

Remark 4.13. *(a) Of course, the argument also works when C_1 and C_2 are identically aligned parallelotopes, that is, with mutually parallel sides.*

Algorithm 4.3 Containment for the union of two boxes

Input: P a point set, C_1, C_2 boxes, t translation vector

Output: **true** if P can be covered by a translate of $C_1 \cup t + C_2$, **false** otherwise

```
set $\mathcal{K} = \emptyset$
for all pairs of points $(p_i, p_j)$ do
    if $p_i - C_1 \cap p_j - C_1 = \emptyset$ then
        $\mathcal{K} = \mathcal{K} \cup \{\neg\zeta_i \vee \neg\zeta_j\}$
    end if
    if $p_i - C_1 \cap p_j - t - C_2 = \emptyset$ then
        $\mathcal{K} = \mathcal{K} \cup \{\neg\zeta_i \vee \zeta_j\}$
    end if
    if $p_i - t - C_2 \cap p_j - C_1 = \emptyset$ then
        $\mathcal{K} = \mathcal{K} \cup \{\zeta_i \vee \neg\zeta_j\}$
    end if
    if $p_i - t - C_2 \cap p_j - t - C_2 = \emptyset$ then
        $\mathcal{K} = \mathcal{K} \cup \{\zeta_i \vee \zeta_j\}$
    end if
end for
if $\mathcal{K}$ is satisfiable then
    return true
else
    return false
end if
```

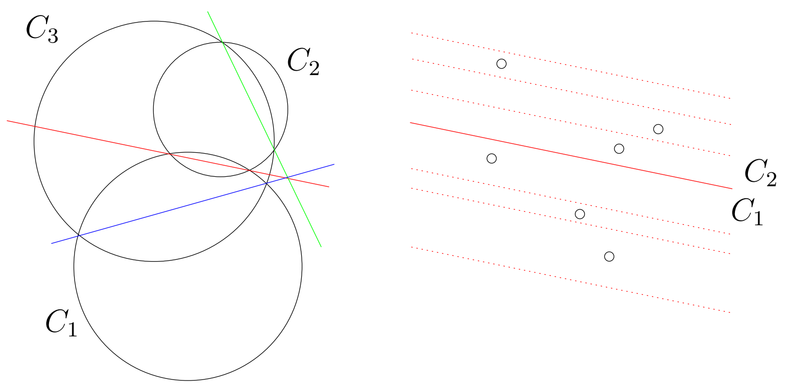


Figure 4.5: For some objects, for instance the union of a fixed number of balls, nonconvex containment can be solved efficiently. The nonconvex object is decomposed into convex subsets by hyperplanes. For each hyperplane, at most $m+1$ possible partitions of a point set P can occur. These are indicated by the dotted lines for the containers C_1 and C_2 in the picture on the right.

(b) If the containers satisfy $C_1 = C_2$, solving the intersection problems to set up $\mathcal{K}$ in Algorithm 4.3 reduces to computing F_C for all $p_i - p_j$, $p_i - p_j + t$ and $p_i - p_j - t$, where $p_i, p_j \in P$.

(c) As far as we know, no polynomial time algorithm is known for the case of two boxes (or parallelotopes) that are not identically aligned. Neither is it known whether this problem is $\mathbb{NP}$-hard.

4.2.3 Nonconvex Containment for Objects Decomposable by Hyperplanes

Nonconvex containment problems in general are hard. However, for special containers, a simple method exists. Assume that C can be decomposed by l hyperplanes to k convex objects $C_1, \ldots, C_k$ such that $C = \bigcup_{i=1}^{k} C_i$, where both k and l are bounded by a constant. This is the case for instance when C is the union of a fixed number of Euclidean balls, since for any two balls, a separating hyperplane can be given (compare Figure 4.5). The number of hyperplanes therefore depends only on the number of balls (and not on the input dimension). It also works when C is the union of k disjoint compact convex bodies for which separating hyperplanes are known.

We seek for a homothetic copy of C covering a given point set P. For each of the l hyperplanes, we have to decide which of the points in P are to lie above it and which below. Therefore, we get $m + 1$ possible partitions of the point set for each hyperplane, and a polynomial bound on the number of possible partitions

$(P_1, \dots, P_k)$ altogether, namely $O(m^l)$. Note that, in general, not all these partitions are feasible. We can now try all possible partitions, and for each partition and each point determine the corresponding P_i. Since the information about the relative position of a point to one of the hyperplanes excludes at least one possible container for this point, we are left with at most one possible container for each point in the end. If no container is left for some point, the partition considered is not feasible and can be disregarded. For each of the P_i, we now have a convex containment problem: minimize ρ subject to the constraints $P_1 \subset c + \rho C_1$, $P_2 \subset c + \rho C_2$, $\dots$, and $P_k \subset c + \rho C_k$. Overall, we can reduce the problem to a polynomial number of $\mathrm{MCP}_{\mathrm{Hom}}$ instances.

4.3 Approximating k-Containment Problems for Point Sets

We now address how to approximate the optimal k-containment radius up to a given constant ε. A number of approximation algorithms exists for k-center problems; see [5] and the surveys [9], [117]. In many cases, the aim is improving complexity bounds rather than practicability. For practical purposes, in contrast, numerous purely heuristic approaches are available (see e.g. [11], [78], [88], or [135] for a concrete application). Although they work well for many inputs in practice, these methods fail to provide provable guarantees.

The most popular problem in this context is arguably the Euclidean k-center problem. Until recently, bigger instances, i.e., $n \geq 3$ or $k \geq 3$, seemed to be out of reach even in this case (see e.g. [117]). This motivated studying the planar Euclidean 2-center problem separately, for instance in [8], [40], [50], [81], [91], and [122]. Progress is due to the existence of small core-sets for the Euclidean k-center problem, yielding a polynomial time approximation scheme (PTAS) in general dimension [19].

When the containers are cubes, fast algorithms are known when $k = 2$ (see Section 4.2.1). The case when $k = 3$ and $n = 2$ is addressed in [83]. In other cases, only super-polynomial methods are at hand.

4.3.1 A Basic Algorithm

In this section, a basic branch-and-bound algorithm for the Euclidean k-center problem is reviewed and adapted to the general case.

Core-Sets

The main idea of the algorithm is based on core-sets for the $\mathrm{MCP}^k_{\mathrm{Hom}}$ which are defined as follows:

Definition 4.14 (ε-core-set for $\mathrm{MCP}^k_{\mathrm{Hom}}$). *Let P, C_1, …, C_k be an instance of the MCP^k_{Hom}. Let $S \subset P$ and $(S_1, \dots, S_k)$ be a partition of S into k clusters such that $\max_i R_{Hom}(S_i, C_i) \le R_{\mathrm{Hom}}(P, C_1, \dots, C_k)$. Let c_1, …, c_k be optimal centers for the MCP_{Hom} with S_i and C_i. If it holds that*

$$P \subset \bigcup_{i=1}^{k} \big(c_i + (1+\varepsilon) \max_i R_{\mathrm{Hom}}(S_i, C_i) C_i\big),$$

S is called an ε-core-set for the MCP^k_{Hom} instance defined by P and the containers C_1, …, C_k.

Remark 4.15. *Let S be an ε-core-set for the MCP^k_{Hom} instance defined by P and the containers C_1, …, C_k. It holds that*

$$\max_i R_{\mathrm{Hom}}(S_i, C_i) \le R_{\mathrm{Hom}}(P, C_1, \dots, C_k) \le (1+\varepsilon) \max_i R_{\mathrm{Hom}}(S_i, C_i),$$

so $\max_i R_{\mathrm{Hom}}(S_i, C_i)$ is an ε-approximation of the optimal radius.

The condition $\max_i R_{\mathrm{Hom}}(S_i, C_i) \le R_{\mathrm{Hom}}(P, C_1, \dots, C_k)$ appears a little awkward at first glance. We need it to guarantee the approximation quality. The clue is that this condition can be satisfied by checking all possible labeled core-sets $S \subset P$ and choosing the labeling with minimal radius $\max_i R_{\mathrm{Hom}}(S_i, C_i)$ (compare Algorithm 4.4). Naturally, this labeling has at most the radius achieved by a labeling of S which is consistent with an optimal solution of the k-containment problem. Since $S \subset P$, the maximal radius for this labeling is surely at most as large as $R_{\mathrm{Hom}}(P, C_1, \dots, C_k)$.

In [19], the existence of small core-sets for the Euclidean k-center problem is proven. The argumentation works as follows. First, assume the existence of an oracle: for each point in P, it returns a cluster index from $\{1, \dots, k\}$ corresponding to a (fixed) optimal partition. The core-set is now constructed via an iterative procedure starting with an arbitrary point $p \in P$. The iteration step is as follows. Let $S = \bigcup_{1 \le i \le k} S_i$ denote a subset of points from P, partitioned into k sets each corresponding to a cluster. The smallest enclosing ball is then computed for each of the sets S_i, yielding centers c_i and a maximal radius ρ. As long as P is not covered by $c_i + (1+\varepsilon)\rho\mathbb{B}$, we find a point $p \in P$ which is not covered and call the oracle to determine to which cluster the point should go. We may choose a point maximizing the distance to the current centers c_i.

After adding $k+1$ points to the set S, the current radius ρ is at least $R^k_{\mathrm{Hom}}(P, \mathbb{B})/2$ ([62], compare Lemma 4.16). After this, we only choose points having distance at least $(1+\varepsilon)\rho$ from the current centers, so any set S_i containing more than one point satisfies $R_{\mathrm{Hom}}(S_i, \mathbb{B}) \ge (1+\varepsilon)R^k_{\mathrm{Hom}}(P, \mathbb{B})/4$. For every subsequent step, one can show that the radius of a cluster increases at least by $O(\varepsilon^2)R^k_{\mathrm{Hom}}(P, C)$ when a point is added, using the half-space lemma already mentioned in Section 3.2.1. For each cluster, at most $O(1/\varepsilon^2)$ steps can be performed until all points are covered, yielding an upper bound of $O(k/\varepsilon^2)$ overall. The oracle is removed by exhaustively labeling all the possible core-sets. The proof does not extend to other than Euclidean containers as the half-space property used does not hold there (see Section 3.2.1).

One should note that for cubes every diametrical pair of points is a 0-core-set and that Helly's theorem [79] implies the existence of 0-core-sets whose size is independent of the number of points in P for all containers (see Section 2.4.1). It is an open question whether core-sets whose size is independent of the dimension exist for general, symmetric containers (compare Section 3.2.1). The incremental algorithm, however, may compute a core-set of size $O(n)$ even when the container is the unit cube.

The developers of the core-set algorithms [19] seem to be mainly interested in proving new complexity bounds for Euclidean k-center. This may be the reason why neither in [19] nor in the following papers [17] and [102] (improving the 1-center algorithm), explicit procedures for solving the k-center problem are stated. However, the PTAS for Euclidean k-center indicated in these papers is easily combined with a B&B scheme. In [101], an implementation of such an algorithm for the Euclidean k-center problem is reported.

Branch-and-Bound Scheme

Algorithm 4.4 is a basic B&B scheme for the $\mathrm{MCP}^k_{\mathrm{Hom}}$. At each node in the B&B tree, we regard a set $S \subset P$ already partitioned into clusters S_i which have to be covered by homothetic copies of the corresponding containers C_i, that is $S_i \subset c_i + \rho_i C_i$. For the branching, a point $p \in P \setminus S$ not (yet) covered is chosen and added to each of the sets S_i consecutively. We choose a point p maximizing $\min_i F_{C_i}(p - c_i)$, the minimum of the values of the distance functionals F_{C_i} for $p - c_i$. In case the maximum is too expensive to compute, one may choose any point p with $F_{C_i}(p - c_i)$ bigger than the current $(1 + \varepsilon) \max_i \rho_i$.[1] The remaining points play no further role in this step of the basic B&B procedure. (This will be improved in Section 4.3.2.)

For the branching, the clusters are sorted according to the distances $F_{C_i}(p - c_i)$ and then p is assigned to the nearest cluster first. With this greedy-like strategy, good upper bounds are computed at an early stage of the algorithm, resulting in fast truncation of many branches and shorter overall running time. The radii of the 1-center instances for the C_i and their assigned core-set points generate first lower bounds on the optimal value for the subtree below the current node. Before doing this, computing the F_{C_i}-distances between the new point and the points already assigned to S_i is recommendable. This step helps preventing unnecessary radius computations.

[1]In [101], the next core-set point p maximizes $\min_i(F_{C_i}(p - c_i) - \rho_i)$, but our tests show that $\min_i F_{C_i}(p - c_i)$ yields better results, see Table 4.1.

Algorithm 4.4 returns an ε-core-set $S \subset P$ consisting of the points chosen at the nodes of an optimal branch, partitioned into subsets $S_1, \dots, S_k$, corresponding to the assignment of the points to the containers $C_1, \dots, C_k$. Here, the algorithm is written down recursively for better readability. However, to gain good running times, we do not use recursion in the implementation described in Section 4.3.4.

Algorithm 4.4 Branch-and-bound for k-containment

```
Input: $P$ a point set, containers $C_1, \dots, C_k$, $\varepsilon > 0$
Output: $S = S_1 \cup \dots \cup S_k$ subset $P$ an $\varepsilon$-core-set for $R_{\text{Hom}}(P, C_1, \dots, C_k)$ , $\rho_i$, $c_i$
  radius and center for $S_i$

  initialize:
  set $S_i = \emptyset$, $\rho_i = 0$, $c_i$ arbitrarily for all $i$
  set $\bar{\rho}$ to an upper bound for $R_{\text{Hom}}(P, C_1, \dots, C_k)$

  k-containment($S_i$, $\rho_i$, $c_i$):
  update the global upper bound $\bar{\rho}$
  compute $\delta = \max_{p_j \in P \setminus \bigcup S_i} \min_i F_{C_i}(p_j - c_i)$
  let $p$ be a point where the maximum is attained
  if $(1 + \varepsilon) \max_i \rho_i \geq \delta$ then
      return
  else
      sort the cluster indices in descending order according to $F_{C_i}(p - c_i)$
      for $j = i_1$ to $i_k$ do
          recompute $c_j$ and $\rho_j$ for $S_j = S_j \cup p^*$
          if $\max_i \rho_i \leq \bar{\rho}(1 + \varepsilon)$ then
              call k-containment($S_i$, $\rho_i$, $c_i$)
          end if
      end for
  end if
  return the best $S_i$, $\rho_i$, and $c_i$ found
```

The number of nodes in the branch-and-bound tree is bounded by $O(k^h)$, where h is the size of a maximal core-set constructed during the algorithm. It follows from [19] that for Euclidean k-center this B&B algorithm is a PTAS with an $O(2^{k \log k/\varepsilon^2} nm)$ worst case running time.

If an upper bound for the optimal radius is known, $\bar{\rho}$ can be initialized accordingly. Since the first k steps of Algorithm 4.4 (that is, when each cluster contains exactly one point), match the first k steps of the greedy algorithm in [62] (assuming that the distance to an empty cluster is set to zero), an approximation factor of at least 2 can be guaranteed in the symmetric case at that stage and $n + 1$ in the general case for identical containers:

Lemma 4.16. *For identical containers $C_i = C$ for all $1 \leq i \leq k$ which have been translated such that $s_C(-C) \subset C$ holds, Algorithm 4.4 computes a $(1 + \frac{1}{s_C})$-approximation of $R^k(P, C)$ in k steps.*

Proof. The proof follows the same lines as in [62], but is more involved since it allows for non-symmetric containers. Assume that P contains at least $k+1$ points (otherwise $R^k(P,C) = 0$). After k steps, each set S_i consists of exactly one point from P.[2] Denote those points by $p_1, \ldots, p_k$. Consider $\bar{\rho} = \max_{p \in P \setminus S} \min_{1 \leq i \leq k} F_C(p - c_i)$ and let p_{k+1} be a point where the maximum is attained. Since $\min_{1 \leq i \leq k} F_C(p - c_i) \leq \bar{\rho}$ for all $p \in P$, the value of $\bar{\rho}$ is an upper bound for the optimal radius. On the other hand, each of the p_i was chosen to maximize $\min_{1 \leq j < i} F_C(p - c_j)$ and $c_j = p_j$. Therefore, all distances $F_C(p_i - p_j)$ are at least $\bar{\rho}$ for all $1 \leq j < i \leq k$. Since the optimal solution has to assign two of those points to the same cluster (without loss of generality, p_i, p_j with $j < i$), the value of $R_{\text{Hom}}(\{p_i, p_j\}, C)$ is a lower bound for $R^k_{\text{Hom}}(P, C)$. Since $p_i, p_j \in c + R_{\text{Hom}}(\{p_i, p_j\}, C)C$ for some c, we get

$$p_i - p_j \in R_{\text{Hom}}(\{p_i, p_j\}, C)(C - C) \subset \left(1 + \frac{1}{s_C}\right) R_{\text{Hom}}(\{p_i, p_j\}, C)C.$$

Consequently,

$$R^k_{\text{Hom}}(P, C) \leq \bar{\rho} \leq F_C(p_i - p_j) \leq \left(1 + \frac{1}{s_C}\right) R_{\text{Hom}}(\{p_i, p_j\}, C)$$

which implies that

$$R^k_{\text{Hom}}(P, C) \leq \bar{\rho} \leq \left(1 + \frac{1}{s_C}\right) R^k_{\text{Hom}}(P, C),$$

proving the statement. □

The statement from Lemma 4.16 does not hold for different containers. We assign the first k points to the k clusters in any order, so the resulting factors may get arbitrarily large.

4.3.2 MICP Relaxation

Good lower bounds are particularly important for the performance of a B&B procedure. Whereas Algorithm 4.4 computes local lower bounds by determining the radii of the current clusters, we now propose lower bounds taking both assigned

[2]This is not stated explicitly in the algorithm description but it surely makes sense to put empty clusters first when sorting the clusters. Moreover, when we have identical containers, we do not need to distinguish between them as long as they are empty.

and unassigned points into account. The new bounds are at least as good as the old ones, but usually much better.

We derive a problem-specific convex relaxation of a mixed integer convex program (MICP) accelerating the B&B. In the following, a version of the $\mathrm{MCP}^k_{\mathrm{Hom}}$ with additional information is considered. It is assumed that the correct clusters are known for *some* of the points in P. This is a natural hypothesis in the context of a B&B scheme and enhances the chances to compute good upper and lower bounds for the optimal solution.

A Mixed Integer Convex Program

Recall that, during the run of the algorithm, the set $S = S_1 \cup \ldots \cup S_k$ denotes the assigned subset of P, that is, the containers are fixed for these points and we require $S_i \subset c_i + \rho C_i$ for some $c_i \in \mathbb{R}^n$ and $\rho > 0$, $i = 1, \ldots, k$. Now, let $S_0 \subset P \setminus S$ denote some of the unassigned points. Then the MCP^k_{Hom} with assigned points in $S_1, \ldots, S_k \neq \emptyset$ and unassigned points in S_0 can be formulated as a mixed integer convex program with variables ρ, c_i and $\lambda_{ij} \in \{0,1\}$ for each $1 \leq i \leq k$ and $p_j \in S_0$. In particular, for each $p_j \in S_0$ and each possible cluster S_i, a reference point $q_{ij} \in \mathrm{conv}(S_i)$ is fixed. A scalar λ_{ij} determines whether p_j is in cluster i.[3] Using the reference points q_{ij}, we require that $\lambda_{ij} p_j + (1 - \lambda_{ij}) q_{ij}$ is in the i-th cluster. Since the containers are convex, any such q_{ij} has to be covered by the i-th container anyway, together with the points in S_i. So, whenever $\lambda_{ij} = 0$ only the reference point q_{ij} has to be covered, a redundant condition. In contrast, if $\lambda_{ij} = 1$, p_j actually has to be contained in the homothetic copy of C_i .

$$\begin{aligned} &\min\ \rho \\ F_{C_i}(p_j - c_i) &\leq \rho && \forall p_j \in S_i,\ 1 \leq i \leq k \\ F_{C_i}\big(\lambda_{ij} p_j - c_i + (1-\lambda_{ij}) q_{ij}\big) &\leq \rho && \forall p_j \in S_0,\ 1 \leq i \leq k \\ \sum_{i=1}^{k} \lambda_{ij} &= 1 && \forall p_j \in S_0 \\ \lambda_{ij} &\in \{0,1\} && \forall p_j \in S_0,\ 1 \leq i \leq k \end{aligned} \tag{4.1}$$

Relaxation

Relaxing the $\{0,1\}$-condition on the multipliers λ_{ij} yields a convex program, providing a lower bound for the radius $R_{Hom}(P, C_1, \ldots, C_k)$. A geometric interpretation of the relaxation is including not the point p_j itself but a point on the line section between p_j and q_{ij} for all i, whereas the constraint $\sum_{i=1}^{k} \lambda_{ij} = 1$ enforces that not all of these points can be close to the reference points (see Figure 4.6).

[3] If it is known (e.g. from a pre-partitioning step) that p_j cannot be in cluster i, the corresponding constraint and variables λ_{ij} can simply be removed.

Picking $q_{ij} \in \operatorname{conv}(S_i)$ such that the distance between q_{ij} and p_j is small gives the best bounds. However, the projection of p_j onto the convex hull of S_i causes longer overall computing time. Balancing between fast computations and a good choice of q_{ij}, the most successful strategy seems choosing q_{ij} as a point in S_i as close as possible to p_j.

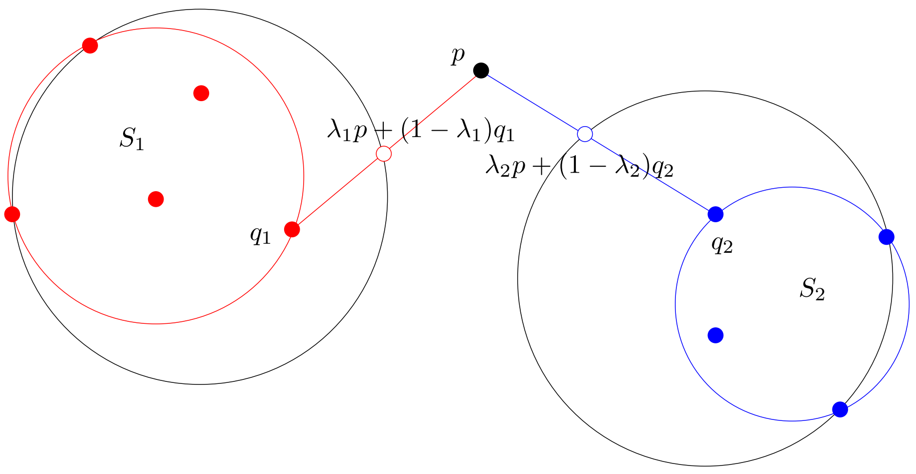

p
S_1
$\lambda_1 p + (1 - \lambda_1) q_1$
$\lambda_2 p + (1 - \lambda_2) q_2$
q_1
q_2
S_2

Figure 4.6: Geometric interpretation of the relaxed program for Euclidean 2-center. Optimal cluster radii with (black) and without (red respectively. blue) considering the unassigned point p.

For polytopal C_i, the relaxation of Program (4.1) is a linear program; for Euclidean containers, we get a second-order cone program. Some other cases can be cast as SOCPs, too (compare Section 3.1).

When S_0 is empty, we get the same bounds as before since we do not consider any unassigned points. The more points from $P \setminus S$ belong to S_0, the better we expect the lower bound on $R_{\mathrm{Hom}}(P, C_1, \ldots, C_k)$ to be. However, as each $p \in S_0$ results in at least $k-1$ additional variables and constraints, the relaxation of Program (4.1) is practicable only when both k and S_0 are not too big. Experiments show that even very small sets S_0 usually provide enough potential to reduce the running times significantly (compare Table 4.2, where 5 points are chosen).

There are different possible strategies to select points for S_0, e.g., randomly, maximizing the minimal distance to a current cluster, maximizing the distance to the most recent core-set point, or maximizing the minimal distance to the unassigned points already picked.

The solution of the convex program provides not only lower bounds. Upper bounds can easily be obtained by assigning the points $p_j \in P \setminus S$ to the clusters, using $\min_i F_{C_i}(p_j - c_i)$ or $\max_i \lambda_{ij}$ (for $p_j \in S_0$) as a criterion. The effectiveness of this approach is evaluated in experiments in Section 4.3.4.

4.3.3 Using Diameter Partitioning within the B&B

Let us now consider another method to accelerate the branch-and bound. We try to make use of the bounds obtained by the diameter partitioning in Section 4.1.2, at least when $k = 2$. When the number of points in P, m, is not too big, we can perform the diameter partitioning on the whole point set as a preprocessing step. Of course, we can initialize the upper bound for the branch-and-bound routine with the upper bound found by the diameter partitioning. Since the bounds obtained by the diameter partitioning frequently happen to be significantly better than predicted by the analysis in Section 4.1, this already prevents us from having to explore wide parts of the tree. Depending on the shape of the container and the dimension, different approximation qualities for the underlying 2-containment problem can be guaranteed.

Moreover, we can reuse the information captured in the pairwise distances. After completing the diameter partitioning, we simply rebuild the graph $G(\bar{\rho})$ for the upper bound and color it. Since $\bar{\rho}$ is an upper bound, we can be sure that the graph is 2-colorable. Again, we may need to use different label pairs since the resulting graph may not be connected. We call the resulting collection of label pairs a pre-partitioning of the point set P. During the B&B, for every new core-set point, we can now check whether it is labeled in the pre-partitioning. If it is not, we proceed as before. But if it has a label, we can immediately assign all points from this label pair to the two clusters by deciding where to put the new core-set point. In this setting, instead of picking an arbitrary point as first core-set point, choosing one from the largest labeled component is advantageous. In the following section, this method is tested in experiments.

4.3.4 Experiments

In the following, we present exemplary test results for the B&B algorithm, the MICP relaxation, and the diameter partitioning. The implementations used are based upon code from [13] and [54]. We use Matlab© R2006B for all tests in this section.

Improvements on the B&B

The experiments in [101] show that the B&B algorithm used there performs much better on practical data sets than the predicted worst case running times suggest. It is concluded that in dimensions 2 and 3, Euclidean k-center is practical for $\varepsilon \geq 0.01$ and $k \leq 4$, where computations in 3-space are significantly more expensive than in 2-space. The latter is caused in the fact that, though the upper bounds on core-set sizes are dimension independent, in practical computations the core-set sizes in lower dimensions are far from the upper bounds and grow noticeably

				running time (s)			
data set	m	n	k	original	(1)	(2)	final
cat	352	3	2	3.4	2.5	1.0	0.9
cat	352	3	3	62.2	29.3	10.0	9.0
cat	352	3	4	*	1952.1	552.7	505.6
shark	1744	3	2	3.3	2.4	1.0	0.6
shark	1744	3	3	248.8	15.8	6.7	3.8
seashell	18033	3	2	29.8	11.4	8.2	1.3
seashell	18033	3	3	169.9	81.5	65.4	11.5
dragon	437645	3	2	132.9	70.7	69.1	3.6
dragon	437645	3	3	5536.2	2468.1	2196.3	154.9
norm. dist.	1000	5	3	929.8	460.0	104.0	86.3
norm. dist.	10000	5	3	10843.5	7074.8	2840.1	1085.1

Table 4.1: Step-by-step comparison of the running time of the original algorithm proposed in [101] and our improvements for the Euclidean k-center. In column (1), we change the rule for picking core-set points, and in column (2), we additionally replace the 1-center computation. The last column contains the results for the final method, also using the modified Euclidean distance computation. In Tables 4.2 and 4.3, we refer to this program as "pure B&B". We use 3D geometric model data sets as well as two examples of (0,1) normally distributed points ("norm. dist."). We report running times in seconds for $\varepsilon = 0.01$. Concerning the entry *, the calculation was interrupted when it was still unfinished after 24 hours.

with the dimension (and so do the running times of the B&B procedure). Our experiments show that the running times of [101] can substantially be improved by our realization of Algorithm 4.4 and even further by the methods presented in Sections 4.3.2 and 4.3.3.

According to [101], the implementation reported there is the first to practically solve huge k-center instances. However, it is also reported that "some of the data sets [...] solved in 3D [...], ran for almost a week on an Intel Itanium system". The same holds for our original implementation of the algorithm on a 2.0 GHz Intel Core 2 system. In Table 4.1, we illustrate our modifications of the B&B algorithm proposed in [101] for the case of the Euclidean k-center. We use geometric model data sets comparable to the ones in [101] for these tests.[4]

We start from a straightforward implementation of the method, using the code [103] (as it is done in [101] and [102]) for the 1-center computations, and we implement a node stack for the B&B instead of using recursion since this is much faster.

[4]For data set sources, see: Large Geometric Models Archive, Georgia Institute of Technology `http://www.cc.gatech.edu/projects/large_models/`, ORC Incorporated `http://www.ocnus.com/models/`, and 3D Cafe `http://www.3dcafe.com/`

This original implementation can be improved in different ways. In a first step, we replace the rule for picking new core-set points. Instead of the maximal value of $\min_i(F_{C_i}(p - c_i) - \rho_i)$ we choose the maximal value of $\min_i F_{C_i}(p - c_i)$ as a criterion for the next core-set point. This results in significantly smaller B&B trees and running times. The reason seems to be that the new rule for picking core-set points is more likely to assign points to the clusters with larger radii, so less points are needed overall. We then decrease the time spent per node by replacing the 1-center with an alternate implementation relying on the solver SeDuMi [116], [126], as well as using "vectorized" code [39] for Euclidean distance computations. Naturally, this last step especially helps when m is large.

Note that, though the Euclidean case is considered in Table 4.1, our B&B method applies to *general* k-containment problems. In the following, we refer to this implementation as the "pure B&B". Since the cases where containers are Euclidean balls or cubes are those which have been studied most intensively, we present examples for these cases. We now consider the effects of the relaxation and the diameter partitioning on the B&B scheme.

MISOCP-relaxation for Euclidean k-Center

We consider the Euclidean k-center problem, and compare the pure B&B implementation with the modified method from Section 4.3.2. The relaxation of the mixed integer second-order-cone program is used to determine cluster centers and lower bounds at each node in the B&B tree.

The test results in Table 4.2 show that the MISOCP-relaxation significantly reduces the size of the tree for Euclidean k-center. Since the larger program with the additional constraints for the points from S_0 is time-consuming, the improvement in the running time is still considerable but not as striking as in the number of nodes. Therefore, further speedup should be possible by advanced strategies for the MISOCP-relaxation. In particular, we expect that improvements to the current method can be achieved through more elaborate techniques for determining the nodes at which to solve the program, the accuracy to which it should be solved, and the strategy to pick the points in S_0.

Diameter Partitioning for 2-Containment with Balls and Cubes

In Table 4.3, test results for diameter partitioning both with Euclidean balls and cubes as containers are listed.

For the Euclidean balls, we compare the pure B&B method to the B&B with diameter partitioning as described in Section 4.3.3. One can see from Table 4.3 that the bounds obtained by the diameter partitioning are already quite good. For point sets of 100 points, the diameter partitioning is very fast, too.

				pure B&B			B&B with relaxed MISOCP		
data set	m	n	k	nodes	leaves	time	nodes	leaves	time
cat	352	3	4	10353	2138	505.6	2380	144	207.7
shark	1744	3	4	649	126	26.1	225	27	13.6
seashell	18033	3	4	12718	2365	925.6	3266	479	371.0
dragon	437645	3	3	341	96	154.9	161	43	89.2
rand. box	1000	5	3	889.3	57.8	44.6	623.9	35.7	45.6
rand. box	1000	5	4	20919.9	3249.8	1272.6	6544.0	238.4	611.2
rand. box	10000	5	3	2595.1	167.6	166.6	1577.7	84.3	139.0
rand. box	10000	5	4	32611.9	3021.6	2273.7	13768.3	808.1	1459.4
trunc. cube	1000	5	3	1146.9	96.8	111.7	690.9	43.7	60.5
trunc. cube	1000	5	4	17407.5	689.8	1783.3	9313.0	165.8	942.8
trunc. cube	10000	5	3	2282.8	148.3	258.5	1380.2	70.2	145.3
trunc. cube	10000	5	4	62343.6	2771.5	8087.4	32311.5	965.0	4096.1

Table 4.2: The B&B algorithm with and without SOCP bounds for Euclidean k-center and an approximation error of 0.01. The 3D geometric model data sets are comparable to the ones used in [101]. The 5D "rand. box" data sets refer to equally distributed points within boxes with randomly scaled axes. The 5D "trunc. cube" data sets refer to equally distributed points within the unit cube, truncated by randomly generated hyperplanes. (We assume that these data sets are more appropriate for k-center problems than, e.g., equally distributed points within the unit cube.) Sizes of the B&B tree and running times (in seconds) are listed – in case of the random data sets, the mean over samples of 20.

			pure B&B	B&B with diameter partitioning			
containers	data set	n	time	DP error	DP time	B&B time	overall time
Euclidean	rand. box	10	7.2	0.06	0.5	2.6	3.1
Euclidean	rand. box	20	26.2	0.11	0.5	14.1	14.6
Euclidean	rand. box	30	88.3	0.15	0.8	67.3	68.2
Euclidean	norm. dist.	10	10.9	0.12	0.5	7.1	7.6
Euclidean	norm. dist.	20	56.1	0.15	0.6	32.5	33.1
Euclidean	norm. dist.	30	135.5	0.17	0.8	89.0	89.9
Euclidean	trunc. cube	10	25.7	0.12	0.4	12.8	13.2
Euclidean	trunc. cube	20	129.8	0.16	0.5	93.8	94.3
Euclidean	trunc. cube	30	568.2	0.19	0.5	425.0	425.5
rot. cubes	rand. box	10	12.8	$< \varepsilon$	2.1	-	2.1
rot. cubes	rand. box	20	103.2	$< \varepsilon$	2.4	-	2.4
rot. cubes	rand. box	30	639.9	$< \varepsilon$	3.1	-	3.1
rot. cubes	norm. dist.	10	21.5	$< \varepsilon$	1.1	-	1.1
rot. cubes	norm. dist.	20	210.9	$< \varepsilon$	2.1	-	2.1
rot. cubes	norm. dist.	30	1153.1	$< \varepsilon$	2.7	-	2.7
rot. cubes	trunc. cube	10	34.6	$< \varepsilon$	1.7	-	1.7
rot. cubes	trunc. cube	20	744.8	$< \varepsilon$	2.7	-	2.7
rot. cubes	trunc. cube	30	2832.7	$< \varepsilon$	3.7	-	3.7

Table 4.3: Test results for diameter partitioning where the containers are either two Euclidean balls or two arbitrarily, independently rotated unit cubes. The "rand. box" data sets refer to 100 equally distributed points within boxes with randomly scaled axes. The "norm. dist." data sets refer to 100 $(0, 1)$-normally distributed points. The "trunc. cube" data sets refer to 100 equally distributed points within the unit cube, truncated by randomly generated hyperplanes. The accuracy is $\varepsilon = 0.01$ for all tests.

We also list test results for the 2-containment problem with two arbitrarily rotated cubes. Note that the usual 2-center algorithm for cubes from [91] does not work here since we have non-identical containers. The diameter partitioning, however, solves the problem exactly, and it is much faster than the B&B.

Note that the core-sets generated by the B&B method are usually larger for cubes than for Euclidean balls (and so are the running times). The reason are possibly ambiguities in the center locations when the containers are cubes, so, frequently, adding a point to the core-set only results in a translation of the corresponding container center but not in an increase of the radius. The increase in the radius is guaranteed only for Euclidean containers by means of the half-space lemma already mentioned, see also Sections 3.2.1 and 4.3.1.

4.4 Approximating Euclidean k-Center Problems for $\mathcal{V}$-Polytopes

In this section, we consider a different variant of the k-containment problem under homothety. Instead of a finite point set P, the task is now to cover the convex hull of P, i.e., we want to cover a non-discrete object. In some applications, this setting is more appropriate than a discrete point set (see [128]). For instance, in a 2D location planning problem when placing emergency sirens, the whole city area is to be covered [135]. The 2D variant of the problem is also addressed in [124]. Note that "clustering for geometric objects" usually refers to a different problem [138], since in that setting, each of the objects (polytopes, for instance) goes to one of the clusters as a whole.

We restrict ourselves to the Euclidean k-center in this section, since the main result is a PTAS for this specific problem. The argumentation uses the aforementioned core-set results for Euclidean k-center on point sets.

Let V be a n-dimensional polytope in vertex representation, and let P be a finite point set, $P = \{p_1, \dots, p_m\}$, such that $\operatorname{conv}(P) = V$. The task is to cover V with k Euclidean balls $c_i + \rho_i\mathbb{B}$, $1 \leq i \leq k$ such that the maximal radius $\rho = \max_i \rho_i$ of the balls is minimized.

4.4.1 Core-Set Points from $\mathcal{V}$-Polytopes

In case of the Euclidean k-center problem for point sets, only a discrete set of points has to be covered. An algorithm for covering P is introduced in Section 4.3.1. The algorithm is a PTAS when the containers are Euclidean balls [19]. Now observe that the result about the core-set size is independent of the representation of the object to be covered. When transferring it to other types of input data, the crucial step is finding a point which is far enough from the current centers as a new core-set point. This is sufficient for an increase of the radius in the next iteration. Therefore, the same result on the size of the core-set holds when $V = \operatorname{conv}(P)$ is to be covered. The only problem arising here is actually checking whether V is covered yet and finding a new core-set point.

Whenever $\bigcup_i \big(c_i + (1+\varepsilon)\rho\mathbb{B}\big)$ does not cover the whole set V, naturally, any point $p \in V$ maximizing $\min_{1\le i\le k} \|p - c_i\|$ is not covered yet. Possible candidates for such points are, of course, the points $p_i \in P$, but there may be more. Consider the Voronoi cells induced by the centers c_i and intersect them with the polytope V. A point furthest from the current centers can be found by norm-maximization on the resulting finite cells (compare Figure 4.7).

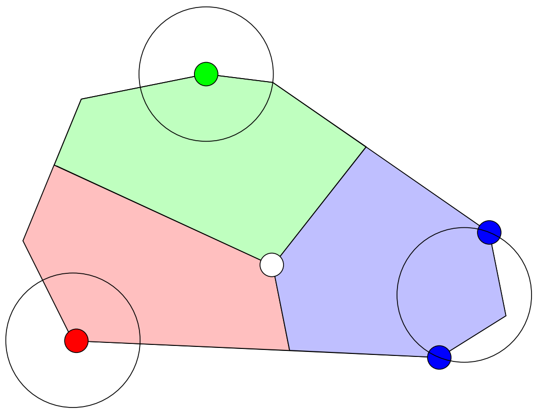


Figure 4.7: Intersection of the polytope V with the Voronoi cells induced by the centers of a Euclidean 3-center solution for a set S of four points. The points in the clusters S_1, S_2, and S_3 are colored red, blue, and green. The radius ρ of the current solution is also shown. The point $p \in V$ maximizing $\min_{1\le i\le k} \|p - c_i\|$, indicated in white, is here at the common vertex of the three Voronoi cells.

The key observation here is the following lemma, showing that considering low-dimensional faces of V is sufficient in order to find all relevant points:

Lemma 4.17. *A polytope $V \subset \mathbb{R}^n$ is covered by k Euclidean balls of unit radius if and only if all j-faces of V with $j < k$ are covered.*

Proof. Let $n > k$ since otherwise the statement is trivial. Assume that all j-faces with $j < k$ of P are covered. Consider the centers of the balls $c_1, \dots, c_k$. Their affine hull $\operatorname{aff}(c_1, \dots, c_k)$ has dimension at most $k-1$. All faces of the Voronoi cells induced by the centers c_i are perpendicular to $\operatorname{aff}(c_1, \dots, c_k)$. Therefore, all those faces have at least dimension $\dim(\operatorname{aff}(c_1, \dots, c_k)^\perp) \ge n - (k-1)$. Consider now the intersection of V with one of the (closed) Voronoi cells. This intersection is a polytope whose vertices are either vertices of V or formed by intersections of faces of V with faces of the Voronoi cell. In order to obtain a vertex, we need a face of V whose dimension is less than or equal to $k-1$. From the assumption, all these faces are completely covered by the k balls. This implies that the distance between the vertex and any of the centers of the adjacent cells is less than 1. Therefore, for each Voronoi cell, its intersection with V is completely covered by the single ball centered there. Since V is the union of those k polytopes, V is also covered. □

The above lemma implies that searching for new core-set points on the j-faces of V where $j < k$ is enough. This is surely an improvement since k, in contrast to n, is fixed.

4.4.2 Complexity of the Algorithm

With help of Lemma 4.17, we have a PTAS for the Euclidean k-center problem for $\mathcal{V}$-polytopes. Using the core-set approach from [19], we can determine the complexity of the method. We get the following theorem:

Theorem 4.18. *Euclidean k-center for vertex-represented polytopes can be approximated up to an accuracy of ε in $O(k^{\varepsilon^{-2}} n m^k \varepsilon^{-k})$ time, where n denotes the dimension and m the number of input points.*

Proof. Since there are $O(1/\varepsilon^2)$ points in the core-set, we have to consider $O(k^{1/\varepsilon^2})$ possible labelings. The core-set points and labelings can be processed in a B&B scheme as described in Section 4.3.1. At each node of the B&B tree, we have to find a new core-set point or decide that V is already covered. This involves checking $O(m^k)$ potential faces of V. We can now compute the vertices explicitly or use an a-priori computed discretization of the low dimensional faces. Such a discretization requires $O(1/\varepsilon^k)$ points. Computing the Euclidean distances takes $O(n)$ operations. The Euclidean 1-center problems can be approximated up to the required accuracy in time linear in n (see, e.g., [102]). □

Remark 4.19. *Assume we want to cover a nonconvex non-discrete region presented as the union of a set of $\mathcal{V}$-polytopes, $\bigcup_{i=1}^{l} V_i$, where each V_i is the convex hull of m_i vertices. We can use the same approach, checking for each of the V_i whether it is covered. In that case, we need to consider $O(\sum_{i=1}^{l} m_i^k)$ faces in each iteration.*

4.4.3 Euclidean 2-Center for $\mathcal{V}$-Polytopes

When $k = 2$, some simplifications are possible. The polytope V is covered by two balls exactly when all edges of V are covered (see Figure 4.8).

Remark 4.20. *When an assignment of the points in V to two clusters is given, the Euclidean 2-center problem for $V = \operatorname{conv}(P)$ can be formulated as a second-order cone program. This is due to the fact that the requiring to cover the whole edge from*

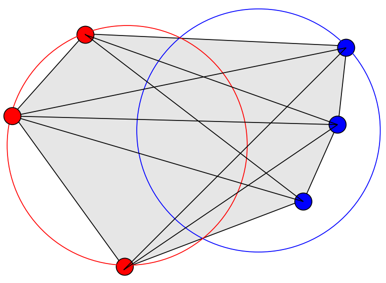


Figure 4.8: Example of the 2-center problem for a $\mathcal{V}$-polytope. For points assigned to different clusters, we need to make sure that the connecting edges are covered.

p_1 to p_2 such that p_1 is in $c_1 + \rho\mathbb{B}$ and p_2 is in $c_2 + \rho\mathbb{B}$ is equivalent to satisfying the following constraints with $\lambda \in [0, 1]$:

$$\|\lambda p_1 + (1 - \lambda)p_2 - c_1\| \leq \rho$$
$$\|\lambda p_1 + (1 - \lambda)p_2 - c_2\| \leq \rho$$

In general, the situation is much easier when $k = 2$, since any point in V maximizing the distance to the current centers is either a vertex of V or it has equal distance to both centers. A new core-set point can therefore be found by computing the distances from the two centers to all vertices of V and by solving the above SOCP for each pair of points p_1, p_2 with p_1 in the first and p_2 in the second cluster.

5 Rotational Containment

A natural generalization of containment under homothety is the minimal containment problem under similarity, the $\mathrm{MCP}_{\mathrm{Sim}}$. It allows rotations of the concerned bodies and consequently increases the number of degrees of freedom. We also consider the anchored version where the container rotates about the origin, and no translation is permitted, the $\mathrm{MCP}_{\mathrm{Rot}}$. We first discuss some general properties of rotational containment problems. The main part of this chapter deals with approximation algorithms for smallest enclosing cylinders and related problems. Finally, we introduce a rotational containment problem derived from an application in human medicine.

5.1 Complexity and Basic Properties

Research on rotational containment problems is focused mainly on the 2D and 3D cases, see, for instance, [1], [16], [42], or [111]. In higher dimensions, less is known.

In case the container in an $\mathrm{MCP}_{\mathrm{Sim}}$ instance has rotational symmetries, many possible rotations are equivalent. For instance, when the container is invariant under rotation about an axis, the problem of finding an optimal rotation can be cast as finding an optimal axis direction and considering the resulting instance of the $\mathrm{MCP}_{\mathrm{Hom}}$. The smallest enclosing cylinder problem is an example of such a problem which we discuss in detail in Sections 5.1.2 and 5.2. It is proven to be $\mathbb{NP}$-hard [110].

Without rotational symmetries, the problem is suspected to be even harder, since the number of degrees of freedom is higher. However, it is an open question whether containment under similarity is $\mathbb{NP}$-hard when the container is the unit cube [110]. Note that this problem subsumes the Hadamard determinant problem in the sense that the $\mathrm{MCP}_{\mathrm{Rot}}$ where the unit cross polytope is to be covered with a cube in dimension n has an optimal radius of $1/\sqrt{n}$ if and only if a Hadamard matrix with n columns exists (see also [85]).

5.1.1 Approximation via Discretizations

An option to approximate general rotational containment problems is discretizing the space of possible container orientations. Finding an approximation of the rotational containment problem, the $\mathrm{MCP}_{\mathrm{Sim}}$ is thereby reduced to a number of

$\mathrm{MCP}_{\mathrm{Hom}}$ instances. The same can be done for the $\mathrm{MCP}^k_{\mathrm{Sim}}$, though this results in as many as $O(\varepsilon^{-k(n-1)!})$ instances of the $\mathrm{MCP}^k_{\mathrm{Hom}}$ in the general case, which is practicable only when both the number of containers k and the dimension n are small.

Figure 5.1 depicts the approximate solution of such a 4-containment problem with 18512 data points[1] and identical 2-dimensional containers being conical sections of circles. These 'pie slice' shapes arise in applications when points should be within the sight of cameras, in the transmission range of oriented senders, or reachable by robot arms with joint limits (see also Section 1.1.2 and [72]). In the example in Figure 5.1, a discretization of the possible space of rotations is combined with the B&B algorithm for the $\mathrm{MCP}^k_{\mathrm{Hom}}$ as presented in Section 4.3. In the planar case, the number of container orientations considered is only linear in ε. Even then, the computational effort increases severely compared to the $\mathrm{MCP}^k_{\mathrm{Hom}}$ as the rotations of the four containers have to be addressed independently of each other. Still, the computation is finished within some hours due to the fast methods for containment under homothety.

We also consider discretizations of the orientation space in Sections 5.2 and 5.3. The objects considered there have an axis of symmetry, so the number of possible container orientations does not get too large, at least in small dimensions. Still, in order to consider all possible orientations of the axis, we have to discretize the unit sphere. In general dimensions, the barycentric subdivision method [139] is available. In 3-space, it is recommendable to use an equilateral triangulation, for instance obtained by successively refining the faces of the 3-dimensional cross polytope, the octahedron.

The number of direction vectors in the discretization is determined by the desired accuracy, but also by the shape of the container. We can determine a bound on the quality of the discretization by considering a smallest annulus containing the boundary of C. The smaller the ratio q of the inner and outer radius of the annulus, the higher are the demands on the accuracy of the discretization.

Remark 5.1. *In the following section, we consider infinite cylinders as containers, where we cannot find an enclosing ball of finite radius. However, since P is finite, we can find a bound depending on the ratio of the optimal radius and the maximum norm of a point in P, $\max_i \|p_i\|$. Obviously, when this value gets small, we need to find a very good approximation for the axis since then, even a small perturbation in the angle can cause large errors.*

On the other hand, in such cases, good approximations can only be achieved with axes enclosing a small angle with an optimal solution. Compared to more ball-like point sets where the ratio is close to one, it is easier to identify unnecessary directions (which can only provide large radius values) and discard them.

[1]See TSPLIB, Ruprecht-Karls-Universität Heidelberg for the data set, `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95`.

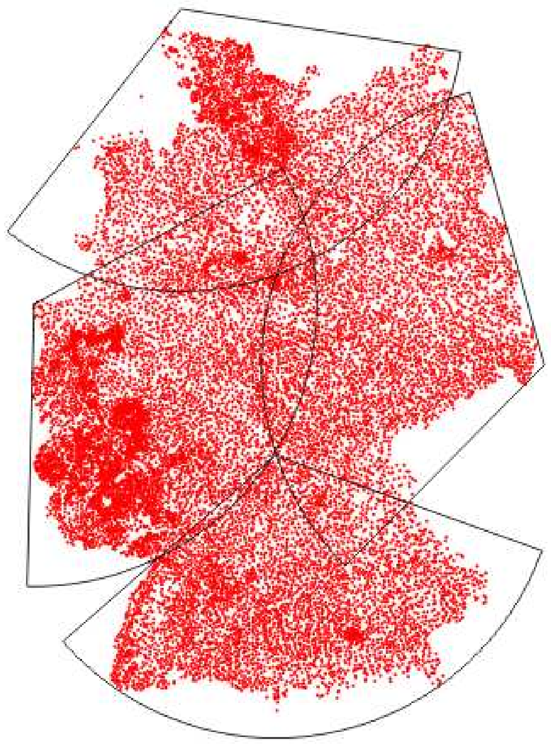


Figure 5.1: Approximate solution of a 4-containment problem under similarity with 18512 data points and accuracy $\varepsilon = 0.02$. The full computation takes less than 5 hours using the same environment as in Section 4.3. The implementation used here is due to [13].

5.1.2 Containers with axial symmetry

We formulate rotational containment problems where the containers are cylinders and cones which are rotationally symmetric about an axis. Note that these containers are not bounded, different to the definition in Section 2.3. However, since the following problems are outer containment problems for finite point sets, this is not a problem. Once the point set is fixed, one can always find an equivalent finite container. We therefore simply use the notation from Section 2.3 for infinite containers, too.

Cones

Containment problems involving circular cones arise for instance in robotics or transmitter location, compare Sections 1.1.1 and 1.1.2. The complexity of these problems in general dimension is regarded in the following.

Consider the problem of covering a point set with a circular cone $C = \operatorname{pos}(t + \mathbb{B})$, $t \in \mathbb{R}^n \setminus \mathbb{B}$ of minimal angle[2]. This is a containment problem according to Section 2.1, but not an instance of $\mathrm{MCP}_{\mathrm{Rot}}$. Nevertheless, when we fix the angle of the cone, we can solve the decision problem by checking whether a given point set can be covered by a rotated copy of $C = \operatorname{pos}(t + \mathbb{B})$, $t \in \mathbb{R}^n \setminus \mathbb{B}$. (Note that this "container" C has the origin on its boundary.)

Covering a point set with a cone of minimal angle can be reduced to solving a 1-center problem for points on a sphere, at least in case a solution exists where the angle is not as large as π [20], [105] (see the applications section in [66], too).

When we consider a circular double cone $C = \operatorname{pos}(t+\mathbb{B}) \cup \operatorname{pos}(-t+\mathbb{B})$, $t \in \mathbb{R}^n \setminus \mathbb{B}$ instead, the problem is $\mathbb{NP}$-complete:

Theorem 5.2. *Deciding whether a set of points P can be covered with a double cone of fixed angle is $\mathbb{NP}$-complete.*

Proof. The proof uses a related construction to those in the proofs in [68], [110] to obtain a reduction from 3-SAT. Consider a 3-SAT problem with l variables ζ_i and a set $\mathcal{K}$ of clauses k.

Let us now construct the point set P. We choose the dimension n such that $n = l + 9$.

We use the following condition to specify the cone: A point p is contained in the cone if and only if $\langle v, p\rangle^2 \geq \frac{1}{n}\|p\|^2\|v\|^2$, where v denotes a vector spanning the cone axis. (This corresponds to an angle of $\alpha = \arccos \frac{1}{\sqrt{n}}$, which we do not use for specification since it may be non-rational.)

We use a set $P_S = \{\pm e_i,\ i = 1, \ldots, n\}$ to get a discrete structure for the set of solutions, that is, the feasible axis directions to cover the point set with a double

[2]We are not interested in the extreme cases here where the cone is a line or a half-space.

cone. Let $v \in \{\pm 1\}^n$ and consider the partition of the points in P_S induced by the hyperplane perpendicular to v. Let $p \in P_S$ and $\langle v, p \rangle \geq 0$, then $\langle v, p \rangle = 1$ holds. For points p in the negative half-space, $\langle v, p \rangle = -1$. Therefore, all points in P_S are covered, and moreover, the $v \in \{\pm 1\}^n$ are the only possible solutions. We now require $v \in (\{\pm 1\}^l \times \{1\}^9)$. In order to avoid two representations of an axis, fixing one additional coordinate would be sufficient. However, we need the extra coordinates in order to represent the clauses. Therefore, we have to ensure that the last nine coordinates of v have the same sign. For this purpose, we use another set of two points,

$$P_O = \left\{\pm \frac{4}{5}\right\} \times \{0\}^{l-1} \times \left\{\frac{1}{5}\right\}^9,$$

to eliminate unwanted axis orientations. For $v \in \{\pm 1\}^n$, $\langle v, p \rangle^2 \geq \frac{1}{n}\|p\|^2\|v\|^2 = 1$ for both points $p \in P_O$ if and only if the last nine coordinates of v have the same sign. There are now 2^l possible axes left to cover the point set $P_S \cup P_O$ with an anchored double cone of the specified angle.

The point set P_K representing the clauses is constructed as follows. We assume without loss of generality that each clause consists of three distinct variables. For each clause $k \in \mathcal{K}$, include a point $u = (\upsilon_1, \ldots, \upsilon_n)$ in P_K:

$$\upsilon_i = \begin{cases} 1/2 & i \leq l \text{ and } \zeta_i \text{ occurs in } k \\ -1/2 & i \leq l \text{ and } \neg\zeta_i \text{ occurs in } k \\ 0 & i \leq l \text{ and neither } \zeta_i \text{ nor } \neg\zeta_i \text{ occurs in } k \\ 1/6 & i > l \end{cases}$$

Let $P = P_S \cup P_O \cup P_K$. Any axis direction v now defines an assignment of l values ± 1 to the variables in the 3-SAT formula. Let τ denote the number of literals satisfied in clause $k \in \mathcal{K}$ under the assignment defined by v, i.e., $\tau \in \{0, 1, 2, 3\}$ and the clause k is false under this assignment if and only if $\tau = 0$. We have

$$\langle v, u \rangle = \frac{1}{2}(\tau - (3 - \tau)) + \frac{9}{6} = \tau.$$

In case $\tau = 0$, we get

$$\langle v, u \rangle^2 = 0 < 1 = \frac{1}{n}\|u\|^2\|v\|^2.$$

Otherwise, $\tau \geq 1$ and we get

$$\langle v, u \rangle^2 = \tau^2 \geq 1 = \frac{1}{n}\|u\|^2\|v\|^2.$$

Therefore, a feasible assignment to the 3-SAT problem exists if and only if the constructed point set P can be covered by an anchored double cone of angle α.

The problem is in $\mathbb{NP}$ since we can verify the containment of a point set in the cone in polynomial time. $\square$

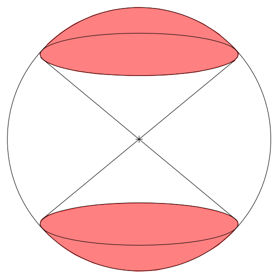


Figure 5.2: Illustration of a circular double cone in 3D, and the caps it covers on the unit sphere.

All the points in P have norm 1, that is, $P \subset \mathbb{S}$. Note that covering a point set on the unit sphere with an anchored double cone (see Figure 5.2) is equivalent to finding an empty anchored slab. See [45] on computing largest empty slabs in 3D.

Remark 5.3. *When a point set P on the unit sphere can be covered with an anchored double cone of angle α, we can find an empty anchored slab of width at least $2\cos\alpha$ within the point set, too. The axis direction of the cone can be used as normal vector for the slab.*

Cylinders

Let us recall the smallest enclosing Euclidean cylinder of a point set P. The smallest enclosing cylinder radius is the outer $(n-1)$-radius of a point set, compare Section 2.4.1. Let l denote a line in $\mathbb{R}^n$, passing through the origin, e.g., $l = \text{lin}(e_1)$.

Definition 5.4. *The* smallest enclosing cylinder *of P is an optimal solution of the MCP$_{\text{Sim}}$ for P and the container $C = l + \mathbb{B}$. Equivalently, we seek for the smallest cylinder containing P, that is, among all 1-dimensional linear subspaces a and points c in $\mathbb{R}^n$, we choose a, c such that $P \subset c + a + \rho\mathbb{B}$ and ρ is minimal. For any cylinder $c + a + \rho\mathbb{B}$ we call $c + a$ the* axis *and ρ the* radius.

Definition 5.5. *The* smallest enclosing anchored cylinder *of P is defined alike the smallest enclosing cylinder, with the additional condition that the axis has to pass through the origin, i.e., $c = 0 \in \mathbb{R}^n$. It is therefore an optimal solution of the MCP$_{\text{Rot}}$ for P and the container $C = l + \mathbb{B}$.*

The decision version of the smallest enclosing anchored cylinder problem is $\mathbb{NP}$-hard [110]. Naturally, the smallest enclosing anchored cylinder radius is at least as large as the smallest enclosing cylinder radius (with equality for 0-symmetric P). For point sets P with $0 \in \text{conv}(P)$, the optimal radius for the anchored problem is at most twice as large as the smallest enclosing cylinder radius.

We now consider another related problem, the anchored 1-ray problem, where the point set is to be covered by the union of an anchored semi-infinite cylinder and a ball centered at the origin. The non-anchored variant of this problem is not so interesting. As the starting point for the ray can move anywhere, one can as well consider the smallest enclosing cylinder of the point set. Let r be a fixed ray in $\mathbb{R}^n$, e.g., $r = \text{pos}(e_1)$.

Definition 5.6. *The* anchored 1-ray *problem for P is the MCP_{Rot} with $C = r + \mathbb{B}$. For any outer parallel body $a + \rho\mathbb{B}$ of a ray a emanating from the origin, a is called the axis and ρ the radius.*

Of course, the optimal radius for the anchored 1-ray problem may be arbitrarily larger than the radius of the smallest enclosing anchored cylinder. An algorithm for the 3D anchored 1-ray problem can be found in [53].

In contrast to the smallest enclosing (anchored) cylinder problem, the 1-ray problem is a convex problem in the following sense: The decision problem where ρ is fixed can be formulated as a convex quadratically constrained feasibility problem (see also [19], [68], and Figure 5.3). Let ρ be fixed, and v a variable representing the axis direction. When $\|p_i\| \leq \rho$, the point is covered no matter what the axis direction is. For any $p_i \in P$ with $\|p_i\| > \rho$, p_i is covered whenever the line spanned by v passes within distance ρ of p_i, or, equivalently, v is in the conic hull of a ball with radius ρ and center p_i. Therefore, the following SOCP constraint has to be satisfied in order to cover p_i with $\text{pos}(v) + \rho\mathbb{B}$ for $\|p_i\| > \rho$:

$$\|v\|\sqrt{\|p_i\|^2 - \rho^2} \leq \langle v, p_i \rangle \tag{5.1}$$

We still have to prevent $v = 0$ in this formulation, which, for instance, may be done by adding a linear constraint $\langle v, p_i \rangle \geq \nu$ for some $\nu > 0$. We then have an SOCP feasibility problem. SOCP feasibility is a subclass of SDP feasibility problems. Their complexity is unknown (see [99] for a summary), but polynomial time methods exist when we settle for approximations of SOCPs or SDPs. See Section 5.2.3 on approximating the anchored 1-ray problem using a modified formulation of (5.1).

We now consider some basic structural properties of the anchored 1-ray problem. Setting $\|v\| = 1$ in (5.1), we see that a feasible anchored 1-ray for a given radius ρ exists if and only if the spherical caps formed by the intersection of the conic hulls of the balls and the unit sphere have a common point (compare Figure 5.3).

Remark 5.7. *If the spherical diameter[3] of the caps on the unit sphere is less than $2\pi/3$, we have a Helly-type theorem as shown in [118] (for more general sets): If each n members of the set of caps intersect, they all have a common intersection. Or, equivalently, if each set of n points in P admits an anchored 1-ray for radius*

[3]the maximum angle between two points in the cap, seen from the origin

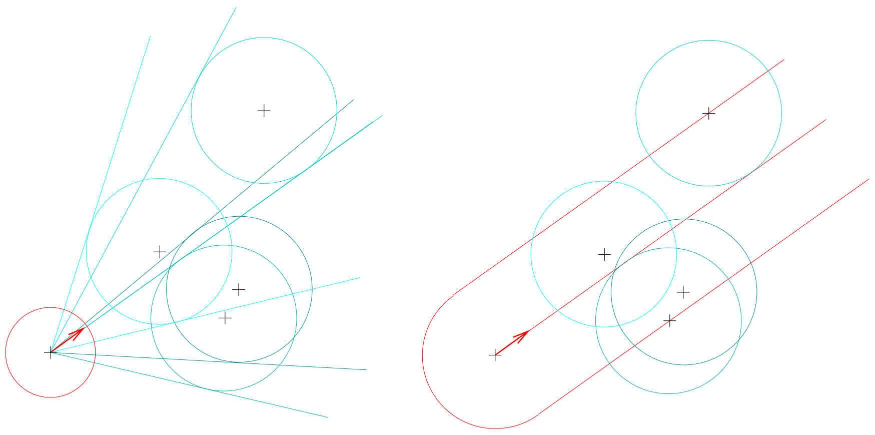


Figure 5.3: An example of an anchored 1-ray problem in 2D with 4 points. The balls centered at the points have the optimal cylinder radius. Each ball spans a cone, and the intersection of these cones determines the axis direction indicated by the red arrow. The intersection of the cones with the surface of the unit ball (in red) results in spherical caps with radii depending on the distance of the input points to the origin. On the right, the optimal solution is shown in red.

ρ, all points do. The same holds if we consider point sets P with $\langle p, v\rangle \geq 0$ for all $p \in P$ and some $v \neq 0$, since we may then restrict the problem to a half sphere.

Remark 5.8. *The anchored 1-ray problem is an LP-type problem (see also Section 2.4.3), that is, it satisfies the properties of* monotonicity *and* locality*. Let $P' \subset P'' \subset P$.*

(a) The optimal radius for P'' is at least as large as for P'.

(b) In case that the radii for P' and P'' are equal, and there is a $p \in P$ such that the radius of $P'' \cup \{p\}$ is larger than the radius of P'', the radius of $P' \cup \{p\}$ is larger than the radius of P', too.

LP-type problems admit the use of simplex-like algorithms, see [57] for details. Such a method, however, also requires a routine solving the base case, that is, computing an optimal 1-ray for sets $P' \subset P$.

We can formulate a k-containment version for the cylinder problems considered here, too. Covering a point set with k cylinders is known as the k-line center problem, an example of the $\mathrm{MCP}^k_{\mathrm{Sim}}$. See [6], [7], [8], and [92] on 2-line center and k-line center. We may also consider the problem variant where the point set is to be covered with cylindrical containers spanned by k rays. The case when $k = 2$ and the rays emit from a common, yet not fixed source is known as the double-ray center problem [60]. See Section 5.2.2 for some results on the anchored version of this problem where the center is fixed, an example of the $\mathrm{MCP}^k_{\mathrm{Rot}}$.

5.2 Approximating Smallest Enclosing Cylinders and Related Problems

We now consider approximating smallest enclosing cylinders, smallest enclosing anchored cylinders, anchored 1-rays, and the corresponding k-containment variants.

5.2.1 A-priori Bounds

In many applications, an approximation of the smallest enclosing cylinder is enough and no exact solution is needed. A typical approach in such approximation algorithms is generating, and then solving or approximating a huge number of subproblems, e.g., of type $\mathrm{MCP}_{\mathrm{Hom}}$. In order to downsize the number of computations for subproblems in such an approximation algorithm, even rough estimates for the radius can provide useful a-priori lower or upper bounds. In the following, we discuss possibilities to obtain such bounds for cylinders. Even without reducing the overall worst-case complexity of the algorithm, these methods can be very useful in practice.

Geometric Inequalities

Geometric inequalities are one possibility to get bounds for the smallest enclosing cylinder radius. Computing other geometric measures of P can help narrowing the interval for the radius whenever suitable bounds on the ratio of the two measures are known. The measure of choice should provide significant information but must not be too expensive to compute.

The following simple construction yields a 4-approximation of the smallest enclosing cylinder radius as follows [77]: We choose an arbitrary point $p_1 \in P$ and a second point p_2 from $\arg\max_i \|p_i - p_1\|$. Choosing the line defined by p_1 and p_2 as an axis, we get a cylinder with a radius at most twice as large as the best cylinder through p_1, since due to $\|p_i - p_1\| \leq \|p_2 - p_1\|$,

$$\mathrm{dist}(p_i, p_1 + \mathrm{lin}(p_2 - p_1)) \leq \mathrm{dist}(p_i, p_1 + a) + \mathrm{dist}(p_2, p_1 + a)$$

holds for any $p_i \in P$ and any line a through the origin. Another factor of 2 in the approximation quality originates from fixing p_1. In the anchored version of the problem, simply taking the anchor point for p_1 yields a factor 2 approximation overall.

We can still do better for the free cylinder when we choose p_1 and p_2 as a pair of points spanning the diameter of P (compare [2]). Now consider a point $q \in P$ maximizing the distance to the axis spanned by p_1 and p_2. The three points form a triangle with its longest edge between p_1 and p_2. Now consider the projection of an optimal cylinder axis in $\mathrm{aff}(p_1, p_2, q)$. Each of the points has a distance less or equal than the optimal cylinder radius from the projected axis. Therefore, the smallest height of the triangle can be at most twice the optimal radius and we have found a factor 2-approximation. In case we consider computing the exact diameter as too expensive in terms of running time, we may replace it by an approximation, resulting in a slightly worse factor than 2.

A good choice to get bounds for the cylinder is the minimum volume enclosing ellipsoid of the point set P. The radius of the smallest enclosing cylinder of P is bounded from above by the value of the second largest semi-axis of the smallest enclosing ellipsoid of P [35]. Moreover, it is shown in [35], that a lower bound is available by means of a scaled volume of the $(d-1)$-dimensional ellipsoid obtained from projecting the smallest enclosing ellipsoid along its largest semi-axis. There are fast methods to approximate the smallest enclosing ellipsoid in practice, and in addition to the bounds described above, its axes also can provide valuable information about the cylinder axis. See also [100] for more on this topic.

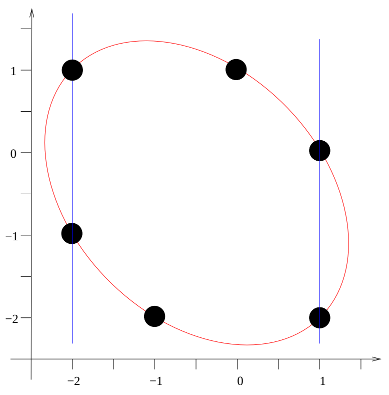
Semidefinite Programming

The following SDP [133] in the variables X and ρ^2 yields a lower bound for the smallest enclosing anchored cylinder radius. Adding a (nonconvex) rank constraint $\operatorname{rank}(X) = n-1$ would yield the exact solution.

$$\begin{aligned} \min\ & \rho^2 \\ \text{s.t. } \operatorname{trace}(p_i p_i^T X) &\leq \rho^2 \qquad\qquad 1 \leq i \leq m \\ \operatorname{trace}(X) &= n-1 \\ X &\succeq 0 \\ I - X &\succeq 0 \end{aligned} \tag{5.2}$$

Any matrix X satisfying the constraints from Program (5.2) and the rank constraint can be decomposed as $X = YY^T$ with an $n \times (n-1)$ matrix Y whose columns form an orthonormal basis of an $(n-1)$-dimensional linear subspace. The value of $\operatorname{trace}(p_i p_i^T X)$ is just the squared length of the projection of p_i into this subspace. Therefore, an enclosing cylinder with an axis perpendicular to this subspace needs at least this radius.

In [133], the SDP-solution for the Euclidean outer j-radius is rounded by a randomized routine resulting in a $\sqrt{12\log(m)}$-approximation with probability at least $1-2/m$. This is not very useful for the smallest enclosing anchored cylinder as a 2-approximation is easily obtained. Yet, it is straightforward to show that for the special case $j = n-1$, the same rounding routine actually computes a $\sqrt{2}$-approximation of the optimal radius with probability 1. More precisely, the rounded upper bound for the squared radius is within 2 times the optimal value of the SDP. See Figure 5.4 for a geometric interpretation of the SDP bound in 2D.



$$X = \begin{pmatrix} \frac{1}{2} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{2} \end{pmatrix}$$

$$X = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

Figure 5.4: Example for SDP (5.2) in 2D. The set P consists of the black points. Minimizing yields $\max_i \operatorname{trace}(p_i p_i^T X) = 3$ and an optimal matrix X with full rank. The matrix X indicated on the upper right is optimal. Geometrically, the condition $\{x \in \mathbb{R}^n : \operatorname{trace}(xx^T X) = 3\}$ defines the red ellipsoid containing the point set. A matrix X corresponding to an optimal cylinder has rank 1 and gives $\max_i \operatorname{trace}(p_i p_i^T X) = 4$ here. Such a matrix X is indicated on the lower right hand side. The corresponding cylinder (or ellipsoid with one infinite axis) is indicated in blue.

Note that the ellipsoid defined by an optimal matrix from Program (5.2) is generally not the minimum volume enclosing ellipsoid of P. Obviously, the objective functions are different.

By a similar approach, we get a convex relaxation for the free cylinder problem:

$$\begin{aligned} \min\ & \rho \\ \|Xp_i - c\| &\le \rho \qquad\qquad 1 \le i \le m \\ \operatorname{trace}(X) &= n-1 \\ X &\succeq 0 \\ I - X &\succeq 0 \end{aligned} \tag{5.3}$$

Again, adding a rank constraint $\operatorname{rank}(X) = n-1$ would yield the exact solution. Comparing this relaxation to the relaxation in (5.2) when both are used for the anchored cylinder problem, we see that Program (5.3) results in weaker bounds. This is because, assuming that $c = 0$, we minimize $\max_i \operatorname{trace}(p_i p_i^T X)$ in (5.2) and $\max_i \operatorname{trace}(p_i p_i^T X^2)$ in (5.3), under the same constraints. Since $0 \preceq X \preceq I$, we have $\operatorname{trace}(p_i p_i^T X^2) \le \operatorname{trace}(p_i p_i^T X)$ for any $p_i \in P$.

Remark 5.9. *One may also try to fix the projection direction and rotate the point* *set, yielding a formulation of the following type:*

$$\begin{aligned} &\min\ \rho \\ &\|\operatorname{proj}_{e_1^\perp}(Xp_i + c)\| \le \rho \qquad\qquad 1 \le i \le m \\ &X \text{ a rotation matrix} \end{aligned}$$

However, it is difficult to find a suitable convex relaxation for this kind of formulation. When we simply relax the nonconvex condition, we get at least the convex hull of the feasible set. Since the matrix with all entries 0 is in the convex hull of the set of rotation matrices [130], this yields nothing but trivial bounds. This observation may also be a reason why so far, there is no simple, suggestive SDP relaxation known for general rotational containment.

5.2.2 Core-Sets for Cylinders

As for the containment problems in Sections 3 and 4, the concept of core-sets can also be applied to cylinders.

Previous Work

An easy method to compute a-priori core-sets is to put a grid on the input set P and round the points to grid points. A more elaborate variant is the computation of an ε-kernel as described in [3], [4], and [141]. An ε-kernel is a subset S of the points in P such that any slab containing the set S, when expanded by a factor of $(1+\varepsilon)$ also contains the set P. Such kernels help approximating a wide range of extent measures of point sets, including the smallest enclosing cylinder, the smallest bounding box, the width, and the diameter. Alternatively, one can try to apply the incremental approach as described for the smallest enclosing ball in Section 3.1.2 to construct a core-set. The analysis of the grid technique, the ε-kernels, and the incremental algorithm yields bounds on the core-set size depending on the accuracy ε and the dimension n [4], [141] for smallest enclosing cylinders. The question is still open whether dimension-independent core-sets for this problem exist.

It is known that all the points in P may be necessary to compute the exact cylinder radius. One can find a set of m points such that each subset of $m-1$ points can be covered by a cylinder of given radius ρ, yet not the whole set. So point sets P exist where choosing less than m points can never provide an optimal cylinder. This observation is originally formulated for line transversals of sets of balls of equal radius ([44], see also [84]), as the existence of a line transversal for a set of balls with radius ρ situated at the points in P is equivalent to the existence of an enclosing cylinder with radius ρ for P.[4]

[4]Another way to phrase this statement is therefore that there is no Helly-type theorem for line

An even stronger negative result is stated for the 2-line center problem, i.e., the $\mathrm{MCP}^2_{\mathrm{Sim}}$ where the containers are cylinders [74].

The smallest enclosing cylinder algorithms in [19], [76], and [77] do not compute core-sets but use similar ideas to generate an approximate cylinder axis. In the following, we show that core-sets whose size is independent of the dimension and the number of points in P exist for the *anchored* 1-ray and cylinder problems. Our method does not extend to "free" cylinders. Nevertheless, it can be used to simplify the approach presented in [19]. See Section 5.2.4 for more on this topic.

Core-Sets for the Anchored 1-Ray Problem

For shortness, let $\rho(P) := R_{\mathrm{Rot}}(P, \mathrm{pos}(e_1) + \mathbb{B})$ denote the optimal radius for the anchored 1-ray problem for P and let $a(P)$ denote a corresponding axis. First, let us formally define core-sets for the anchored 1-ray problem.

Definition 5.10. *Let $P \subset \mathbb{R}^n$. We call $S \subset P$ an* ε-core-set for the anchored 1-ray problem *if $P \subset a(S) + (1+\varepsilon)\rho(P)\mathbb{B}$.*

Even when $\rho(P)$ is unknown, one can still check the following condition:

Remark 5.11. *A sufficient, but not necessary condition for $S \subset P$ to be an ε-core-set is that $P \subset a(S) + (1+\varepsilon)\rho(S)\mathbb{B}$, since $\rho(S) \leq \rho(P)$.*

The points close enough to the origin do not contribute to the axis direction and are not needed in the core-set. More precisely, it suffices to consider P', where P' is the set P without the points that are within distance $(1+\varepsilon)\rho(P)$ of the origin.

Lemma 5.12. *For any $\varepsilon > 0$, when S is an ε-core-set for P', it is also an ε-core-set for P. Moreover, for any point p from such a point set P', $\langle p, v\rangle > 0$ where v denotes the unit direction vector of $a(P')$.*

Proof. The first assertion follows directly from the core-set definition. Note that, in case $P' = \emptyset$, we get an empty core-set S, and $a(S)$ may be an arbitrary axis. This happens when P is within a ball of radius $(1+\varepsilon)\rho(P)$.

For the second assertion, verify that any optimal axis is in the intersection of the cones spanned by the balls $p + \rho(P')\mathbb{B}$, $p \in P'$ and none of those balls contains the origin. Therefore, an optimal axis encloses an angle of at most $\arctan(1/(1+\varepsilon)) < \pi/2$ with any of the points in P'. □

transversals of balls. Consequently, modifications of the problem where stronger conclusions are possible are subject to research. One option are generalizations of Helly's theorem which are of the same type as Hadwiger's transversal theorem (see [71]) for this problem with additional information. In case the balls centered at the points in P are all disjoint (that is, the points in P are far enough apart), the existence of an order-respecting line transversal for each subset of $2n$ balls implies the existence of a line transversal for the whole set [32].

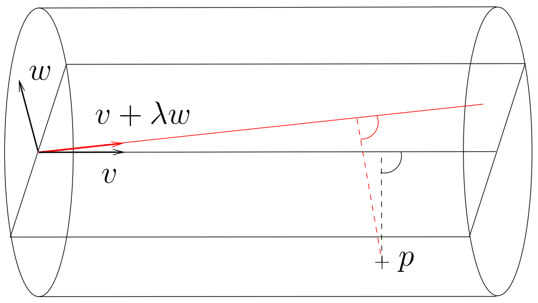


Figure 5.5: Illustration of the construction in the proof of the conic half-space lemma. The axis and cylinder spanned by v are indicated in black, and so is the half-space perpendicular to w. The new axis spanned by $v + \lambda w$ is indicated in red. For any point $p \in P$, we calculate the distance to the new axis.

Before proceeding with the existence of core-sets, we observe that the following half-space lemma holds. It is strongly related to a similar property for smallest enclosing balls used in core-set proofs for the Euclidean 1-center problem [31], [19, Lemma 2.2], see also Section 3.1.2. However, here, we formulate a “conic” version.

Lemma 5.13. *Let $P \subset \mathbb{R}^n$ with $\min_i \|p_i\| \geq \rho(P)$. For each direction vector w in $a(P)^\perp$, there is at least one point $q \in P$ in the closed half-space containing $a(P)$ and perpendicular to w satisfying* $\operatorname{dist}(a(P), q) = \rho(P)$.

Proof. From the assumption, we know that an optimal axis encloses an angle of at most $\pi/2$ with any of the points in P. Let therefore $v \in \mathbb{R}^n$, $\|v\| = 1$ be the direction of an arbitrary axis a with $\langle v, p\rangle \geq 0$ for all points p. Furthermore, let $w \in \mathbb{R}^n$, $\|w\| = 1$ be another direction vector with $\langle w, v\rangle = 0$. For a scalar $\lambda > 0$, we now compute the distance of a point p to the axis spanned by $v + \lambda w$ (see Figure 5.5):

$$\begin{aligned}\|\operatorname{proj}_{(v+\lambda w)^\perp}(p)\|^2 &= \left\| p - \frac{1}{\|v+\lambda w\|^2}\langle v+\lambda w, p\rangle (v+\lambda w)\right\|^2 \\ &= \|p\|^2 - \frac{\langle v,p\rangle^2 + 2\lambda\langle v,p\rangle\langle w,p\rangle + \lambda^2\langle w,p\rangle^2}{1+\lambda^2} \\ &= \|p - \langle v,p\rangle v\|^2 + \left(1 - \frac{1}{1+\lambda^2}\right)\langle v,p\rangle^2 - \frac{2\lambda\langle v,p\rangle\langle w,p\rangle + \lambda^2\langle w,p\rangle^2}{1+\lambda^2} \\ &= \|p - \langle v,p\rangle v\|^2 + \frac{\lambda}{1+\lambda^2}\big(\langle v,p\rangle^2\lambda - \langle w,p\rangle^2\lambda - 2\langle v,p\rangle\langle w,p\rangle\big)\end{aligned}$$

We now assume that all points in P with maximal distance to the axis a satisfy $\langle w, p\rangle > 0$, i.e. they are not in the closed half-space $\{x : \langle w, x\rangle \leq 0\}$.

Of course, $\lambda/(\lambda^2+1)$ is always positive. The remaining term $(\langle v,p\rangle^2 - \langle w,p\rangle^2)\lambda - 2\langle v,p\rangle\langle w,p\rangle$ is a linear function in λ. Consider the case when $\langle w,p\rangle > 0$. When

$\langle v, p\rangle = 0$, this is negative for all $\lambda > 0$. Otherwise, the expression is surely negative for $\lambda = \langle w, p\rangle / \langle v, p\rangle$, and therefore for all $\lambda \in]0, \langle w, p\rangle/\langle v, p\rangle]$. Any such λ will decrease the distance to the axis for such a point p.

Now consider a point with $\langle w, p\rangle \leq 0$. From the assumption, we know that $\|p - \langle v, p\rangle v\|$ is strictly smaller than $\max_i \operatorname{dist}(p_i, a)$. We can now choose $\lambda > 0$ small enough such that the distance to the new axis gets larger, but is still smaller than $\max_i \operatorname{dist}(p_i, a)$. This completes the proof of the lemma, as it implies that a is not an optimal axis. □

Now we are ready to prove the main theorem of the core-set section. The good news is that we can find core-sets whose size is independent of both the number of points in P and the dimension of the space $\mathbb{R}^n$. The drawback of the following statement, however, is that the core-set size depends on the shape of the point set. Intuitively, one might expect that the computation of anchored 1-rays for "long and skinny" point sets is easier, since the axis direction seems to be almost settled. However, we get larger core-sets when the ratio between the optimal radius and the maximal distance of a point in P to the origin is small. It is obvious that when p is far from the origin, a slight change of the axis may have a huge impact on the distance between the axis and p. This issue is also discussed in Remark 5.1 concerning discretizations. We do not know whether smaller core-sets exist for such point sets in general. However, when the radius is 0 and all points are on a line, any point from P is a 0-core-set. So, of course, it is possible that our way of constructing core-sets or the analysis here can still be improved for such point sets.

Theorem 5.14. *For any $0 < \varepsilon < 1$, there exists an ε-core-set for the anchored 1-ray problem for P whose size depends only on ε and the ratio $\max_i \|p_i\|/\rho(P)$.*

Proof. We may assume $\rho(P) > 0$ and $\min_i \|p_i\| > (1+\varepsilon)\rho(P)$ in the following. We show that Algorithm 5.1 computes the desired core-set under the assumption that $\rho(S)$ and $a(S)$ can be computed exactly for $S \subset P$.[5]

Before entering the loop in Algorithm 5.1, we choose two points p, p' as initial set S. We know that $\operatorname{dist}(\operatorname{pos}(p), p') \geq \rho(P)$, and therefore

$$\rho(\{p, p'\}) = \operatorname{dist}\left(\operatorname{pos}\left(\frac{p+p'}{2}\right), p'\right) \geq \operatorname{dist}(\operatorname{pos}(p), p')/2 \geq \rho(P)/2.$$

Since $\rho(S)$ is non-decreasing during the course of the algorithm, we have that $\rho(S) \leq \rho(P) \leq 2\rho(S)$ (compare Section 5.2.1). Set

$$\gamma = (\log \rho(P) - \log \max_i \|p_i\| - \log 2)/\log \varepsilon.$$

[5]We are mainly interested in the structural result here, i.e., the existence of core-sets. In order to improve readability, we discuss the computational details of an approximation algorithm in Section 5.2.4.

Algorithm 5.1 Core-set algorithm for anchored 1-ray

Input: P a point set, $\varepsilon > 0$

Output: $S \subset P$ a core-set, ε-approximation of optimal anchored 1-ray $\rho(S)$ radius, $a(S)$ axis

```
let $p$ be a point in $P$ maximizing $\|p\|$
let $p'$ be a point in $P$ maximizing the distance to $\mathrm{pos}(p)$
set $S = \{p, p'\}$, compute $a(S)$, $\rho(S)$
while $P \not\subset a(S) + (1+\varepsilon)\rho(S)\mathbb{B}$ do
    let $p \in P$ and $p \notin a(S) + (1+\varepsilon)\rho(S)$
    set $S$ to $S \cup \{p\}$
    recompute $a(S)$, $\rho(S)$
end while
return $S$, $a(S)$, $\rho(S)$
```

Therefore, $\gamma \geq 0$ and for any $p_i \in P$, it holds that $2\|p_i\|/\rho(P) \leq 1/\varepsilon^\gamma$. We now claim that in every following step of Algorithm 5.1, the radius increases by at least $\rho(P)\varepsilon^{4\gamma+3}/20$.

Let $p \in P$ be the new core-set point. Let v denote the unit length direction of the axis $a(S)$ for the previous set S, and v' the unit direction of the new axis $a(S \cup \{p\})$. Note that $\langle v, v' \rangle \geq 0$ always holds (see Lemma 5.12). We distinguish two cases, namely whether the "old" and the "new" axis are far or close, respectively. For this purpose, we choose a threshold $\delta = \varepsilon^{4\gamma+2}/16$. This choice is justified in the calculation.

After computing the new axis, one of the following two cases occurs:

Case 1: $\langle v, v' \rangle \geq 1 - \delta$. When v and v' are close, the distance of p to the new axis is almost as large as its distance to the old axis. We first estimate the new squared radius which is bounded from below by the distance of p to the new axis:

$$\begin{aligned}
\rho(S \cup \{p\})^2 &\geq \|p - \langle p, v' \rangle v'\|^2 \\
&= \|p - \langle p, v \rangle v + \langle p, v \rangle v - \langle p, v' \rangle v'\|^2 \\
&= \|p - \langle p, v \rangle v\|^2 + |\langle p, v \rangle^2 - \langle p, v' \rangle^2| \\
&\geq (1+\varepsilon)\rho(S)^2 + \langle p, v + v' \rangle \langle p, v - v' \rangle \\
&\geq (1+\varepsilon)\rho(S)^2 - \|p\|^2 \|v + v'\| \|v - v'\| \\
&\geq (1+\varepsilon)\rho(S)^2 - \|p\|^2 \sqrt{(2 + 2\langle v, v' \rangle)(2 - 2\langle v, v' \rangle)} \\
&\geq \left(1 + \varepsilon - \frac{2\sqrt{2\delta - \delta^2}}{\varepsilon^{2\gamma}}\right) \rho(S)^2
\end{aligned}$$

We use the inequality $\|p\|/\rho(S) \leq 1/\varepsilon^\gamma$ which holds for all $p \in P$ since γ is chosen to meet $2\|p\|/\rho(P) \leq 1/\varepsilon^\gamma$ here. We now get the following bound for the increase in the radius:

$$\begin{aligned}
\rho(S \cup \{p\}) - \rho(S) &\geq \sqrt{1 + \varepsilon - \frac{2\sqrt{2\delta - \delta^2}}{\varepsilon^{2\gamma}}}\rho(S) - \rho(S) \\
&\geq \frac{2}{5}\left(\varepsilon - \frac{2\sqrt{2\delta - \delta^2}}{\varepsilon^{2\gamma}}\right)\rho(S) \\
&\geq \frac{2}{5}\left(\varepsilon - \frac{\varepsilon}{\sqrt{2}}\right)\rho(S) \\
&\geq \frac{\varepsilon}{10}\rho(S)
\end{aligned}$$

We see that when $\|p\|$ is large compared to $\rho(P)$, a slight move of the axis may change the distance of p considerably. In order to ensure that the radius increases in such a step, we have to make the threshold very small.

Case 2: $\langle v, v' \rangle \leq 1 - \delta$. When v and v' are far, clearly, the new axis is closer to p. Different from the first case, it is no use considering the distance from p to the new axis in order to bound the increase in the radius. We need to find another point q from the set S with sufficient distance from the new axis. For this purpose, we use Lemma 5.13, the conic half-space lemma. It implies that a point $q \in S$ exists, such that $\|\operatorname{proj}_{v^\perp}(q)\| = \rho(S)$ and q is in the half-space perpendicular to $\operatorname{proj}_{v^\perp}(v')$. Since $0 \geq \langle q, \operatorname{proj}_{v^\perp}(v') \rangle = \langle q, v' - \langle v', v \rangle v \rangle$, it holds that $\langle q, v' \rangle \leq \langle v, v' \rangle \langle q, v \rangle$. Therefore, we get the following bound for the squared radius of the new set S.

$$\begin{aligned}
\rho(S \cup \{p\})^2 &\geq \|q - \langle q, v' \rangle v'\|^2 \\
&= \|q\|^2 - \langle q, v' \rangle^2 \\
&\geq \|q\|^2 - \langle v, v' \rangle^2 \langle q, v \rangle^2 \\
&\geq \|q\|^2 - (1 - \delta)^2 \langle q, v \rangle^2 \\
&= \|q\|^2 - (1 - \delta)^2 \big(\|q\|^2 - \|q - \langle q, v \rangle v\|^2\big) \\
&= (2\delta - \delta^2)\|q\|^2 + (1 - \delta)^2 \rho(S)^2 \\
&\geq \big(1 + (2\delta - \delta^2)(1 + \varepsilon)^2 - 2\delta + \delta^2\big)\rho(S)^2 \\
&= \big(1 + (2\delta - \delta^2)(2\varepsilon + \varepsilon^2)\big)\rho(S)^2.
\end{aligned} \tag{5.4}$$

Note that we are using $\|q\| \geq (1 + \varepsilon)\rho(P) \geq (1 + \varepsilon)\rho(S)$ here. Altogether, we have

$$\begin{aligned}
\rho(S \cup \{p\}) - \rho(S) &\geq \sqrt{\rho(S)^2 + (2\delta - \delta^2)(2\varepsilon + \varepsilon^2)\rho(S)^2} - \rho(S) \\
&\geq \frac{2}{5}(2\delta - \delta^2)(2\varepsilon + \varepsilon^2)\rho(S) \\
&\geq \frac{\varepsilon^{4\gamma+3}}{10}\rho(S).
\end{aligned} \tag{5.5}$$

Taking the two cases together proves an increase in the radius in each iteration. Since we have to choose the threshold very small for “long and skinny” point sets in order to deal with the first case, the bound for the increase of the radius gets small for such point sets in the second case.

As we start from a 2-approximation of $\rho(P)$, and each step increases the radius $\rho(S)$ by at least

$$\frac{\varepsilon^{4\gamma+3}}{10}\rho(S) \geq \frac{\varepsilon^{4\gamma+3}}{20}\rho(P),$$

we are done after incrementing the set S at most $\lceil 20/\varepsilon^{4\gamma+3}\rceil$ times. As only one point is added in each step, we have found a core-set of size $O(1/\varepsilon^{4\gamma+3})$. With

$$\varepsilon^{4\gamma+3} = \varepsilon^{4(\log\rho(P)-\log\max_i\|p_i\|-\log 2+\log\varepsilon)/\log\varepsilon+3} = \left(\frac{\rho(P)}{2\max_i\|p_i\|}\right)^4\varepsilon^7,$$

we can conclude for the final core-set S, that

$$|S| = O\left(\left(\frac{\max_i\|p_i\|}{\rho(P)}\right)^4\varepsilon^{-7}\right).$$

Of course, this is a bound on the number of steps taken by Algorithm 5.1, too. □

Though we introduce an incremental algorithm constructing a core-set in the proof of Theorem 5.14, there are still some questions remaining open in order to obtain a practicable method. So far, we do not know how to actually compute an optimal axis and radius for the set S. We also assume that P does not contain any points with norm smaller than $(1+\varepsilon)\rho(P)$, but do not know the optimal radius either. In Section 5.2.4, we address how to deal with these issues in an approximation algorithm. But before, we generalize our core-set results to cylinders.

Core-Sets for Anchored Cylinders

The following observation shows how to relate smallest enclosing anchored cylinders and anchored 1-rays of point sets. Let now $a(P)$ denote the axis of a smallest enclosing anchored cylinder for the point set P with optimal radius $R_{\mathrm{Rot}}(P, l+\mathbb{B})$. The point set P is separated by the hyperplane perpendicular to an optimal axis. Reflect the points on the negative side of this hyperplane at the origin. Let v denote a direction vector spanning the axis and $P^+ = \{p : p \in P, \langle v, p\rangle \geq 0\}$ and $P^- = \{-p : p \in P, \langle v, p\rangle \leq 0\}$ (see also Figure 5.6).

Lemma 5.15. *The smallest enclosing anchored cylinder problem can be reduced to an anchored 1-ray problem where the point set $P^+ \cup P^-$ is to be covered.*

Proof. On the one hand, for each $p \in P$, any cylinder containing p also contains $-p$. Therefore, the radius does not get any larger when we swap points to the other side of the origin. On the other hand, $(P^+ \cup P^-) \cup -(P^+ \cup P^-)$ is a symmetric point set whose smallest enclosing cylinder is just the symmetric extension of an optimal 1-ray for $(P^+ \cup P^-)$. Since $P \subset (P^+ \cup P^-) \cup -(P^+ \cup P^-)$ the radius does not get any smaller either. □

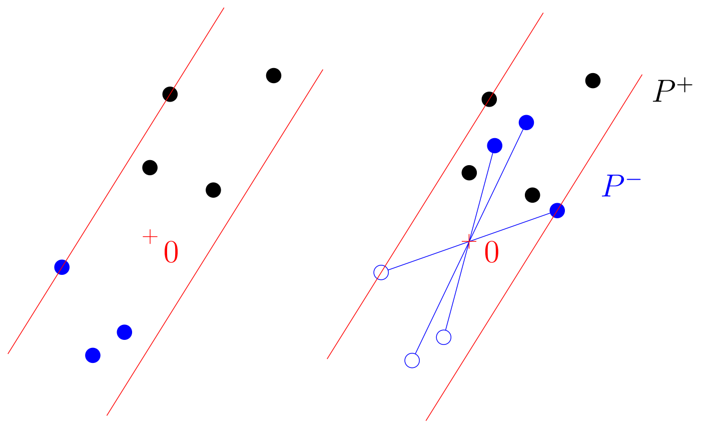


Figure 5.6: Example of an anchored cylinder problem in 2D, where the optimal cylinder is indicated in red. On the right, the point set $P^+ \cup P^-$ for the corresponding anchored 1-ray problem is shown. The black points form the set P^+ and the blue points the set P^-.

Consequently, the anchored cylinder problem can be reduced to an anchored 1-ray problem assuming the existence of an oracle: For any point $p \in P$, the oracle reports whether p is in P^+ or in P^-. We call such an assignment a labeling[6] of the points in P. By enumerating all possible core-set labelings, the oracle can be removed. Therefore, we immediately get the following corollary:

Corollary 5.16. *For any $0 < \varepsilon < 1$, a labeled ε-core-set S of size*

$$|S| \leq O\left(\left(\frac{\max_i \|p_i\|}{R_{\text{Rot}}(P, l + \mathbb{B})}\right)^4 \varepsilon^{-7}\right)$$

for the anchored cylinder problem exists. It can be found by checking at most $O(2^{(\max_i \|p_i\|/R_{\text{Rot}}(P,l+\mathbb{B}))^4 \varepsilon^{-7}})$ possible labeled core-sets.

The labeling strategy is similar to the approach for Euclidean k-center core-sets in [19], [101]. We also observe that the same approach works for the corresponding k-containment problems.

[6]compare the labeled core-sets for the k-containment problem in Section 4.3

Corollary 5.17. *(a) For any $0 < \varepsilon < 1$, a labeled ε-core-set S of size*

$$|S| \leq O\left(k\left(\frac{\max_i \|p_i\|}{R_{\mathrm{Rot}}(P, l+\mathbb{B})}\right)^4 \varepsilon^{-7}\right)$$

for the anchored k-ray center problem exists. It can be found by checking $O(k^{k(\max_i \|p_i\|/R_{\mathrm{Rot}}(P,l+\mathbb{B}))^4 \varepsilon^{-7}})$ possible labeled core-sets.

(b) For any $0 < \varepsilon < 1$, a labeled ε-core-set S of size

$$|S| \leq O\left(k\left(\frac{\max_i \|p_i\|}{R_{\mathrm{Rot}}(P, l+\mathbb{B})}\right)^4 \varepsilon^{-7}\right)$$

for the anchored k-line center problem exists. It can be found by checking $O((2k)^{k(\max_i \|p_i\|/R_{\mathrm{Rot}}(P,l+\mathbb{B}))^4 \varepsilon^{-7}})$ possible labeled core-sets.

These results rely essentially on the fact that we consider the $\mathrm{MCP}_{\mathrm{Rot}}$ here and not the $\mathrm{MCP}_{\mathrm{Sim}}$. They do not generalize to non-anchored objects. As already mentioned in Section 5.2.2, it is known that core-sets for the general 2-line center problem do not exist. In practice, we recommend a B&B scheme to approximate the smallest enclosing anchored cylinder as described in Section 5.2.4. In the following, we address how to obtain core-set based approximation algorithms for the problems considered here.

5.2.3 Approximating Anchored 1-Rays by Convex Programming

With Algorithm 5.1, we already have a framework for an approximation algorithm for the anchored 1-ray problem. Firstly, we address how to approximate the anchored 1-ray using convex quadratic programming. This is then used for the sets $S \subset P$ within Algorithm 5.1 and a core-set based B&B method for the smallest enclosing anchored cylinder (see Section 5.2.4).

Convex Programming Formulation

As stated in Section 5.1, finding an anchored 1-ray is a convex problem for a fixed radius ρ. We now modify the constraints in the formulation in (5.1) to obtain the following optimization problem:

$$\begin{aligned} &\min \|v\| \\ &\langle v, p_i\rangle \geq \sqrt{\|p_i\|^2 - \rho^2} \qquad 1 \leq i \leq m,\ \|p_i\| > \rho \end{aligned} \tag{5.6}$$

Recall that a given radius ρ is feasible if and only if the intersection of the cones spanned by balls of radius ρ centered at the points from P contains a line. Equivalently, the spherical caps formed by the intersection of the cones with the unit sphere intersect in at least one point. The linear constraints in Program (5.6) now define hyperplanes perpendicular to the p_i with $\|p_i\| > \rho$. The right hand sides are chosen such that each hyperplane exactly separates the corresponding spherical cap from the rest of the unit sphere. See Figure 5.7 for an example. The optimal value of Program (5.6) is 1 if and only if those spherical caps intersect in one point.

Remark 5.18. *Let us resume some important properties of Program (5.6):*

(a) *In case that ρ is larger than the optimal radius, we can find v with norm $\|v\| < 1$, and in case ρ is too small to admit an enclosing cylinder, the optimal value of Program (5.6) is larger than 1 or the program is even infeasible.*

(b) *When the points in P with $\|p_i\| > \rho$ are strongly separable from the origin by a hyperplane, or, equivalently, $0 \notin \operatorname{conv}(P)$, the ray defined by the normal vector of the separating hyperplane (pointing away from 0) intersects each of the hyperplanes for the linear inequalities in Program (5.6) and therefore, a feasible point exists on this ray.*

(c) *If it exists, the optimal solution v is unique since the squared objective $\|\cdot\|^2$ is strictly convex.*

Comparing (5.1) to (5.6), the second version has the advantage that we do not need to deal with the scaling of v by adding additional constraints. Moreover, we only have linear constraints here which are generally easier to handle.

Program (5.6) is a linearly constrained convex optimization problem. Note that the square roots on the left hand side of the linear constraints and in the objective function cannot be represented exactly in the bit model of computation but have to be approximated. Strictly speaking, the program is not a QP as we do not use the squared norm in the objective (since this simplifies some of the calculations later). Anyhow, we can use any QP solver to compute an approximate solution for this program, too.

Binary Search Algorithm

We now approximate the anchored 1-ray problem by using the bisection method (see, e.g., [34]), that is, performing binary search to determine the radius. We approximate the optimal solution of Program (5.6) and get a solution $\hat{v}$ which is feasible and has $\|\hat{v}\|$ within a given (additive) error of the optimum. Moreover, we do not consider the exact constraints involving square roots but approximate values instead, adding to the overall error. In order to find an approximation of the optimal radius $\rho(P)$, we only need to know whether a given value ρ is feasible. Without the exact solution of Program (5.6) it may occur that our decision on the feasibility of ρ is incorrect. Nevertheless, the goal is to show that an approximation of the optimal solution of Program (5.6) is sufficient for our purpose.

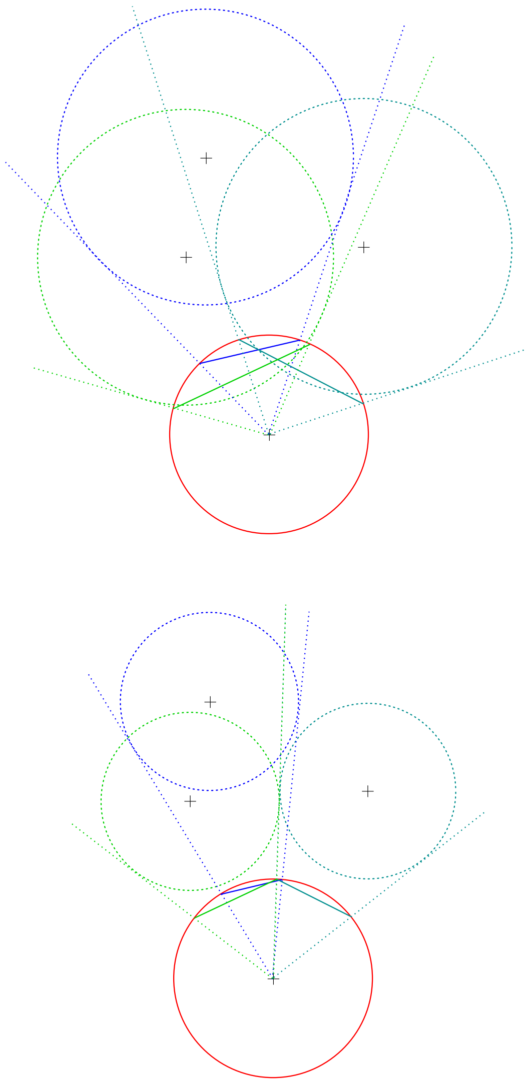


Figure 5.7: Program (5.6) in a 2D example with three points. The unit sphere is drawn in red. On the upper left, the cones spanned by the balls centered at the input points for a given radius ρ are shown. Clearly, ρ is larger than the optimal radius here. Each point in P adds one linear constraint to the program. Geometrically, these are hyperplanes which separate the intersection of the corresponding cone and the sphere from the rest of the sphere. These linear constraints are shown again on the upper right. Since ρ is too large, the optimal solution v has $\|v\| < 1$ (indicated by the red dotted circle). The axis spanned by v is shown, too. Below, the same example, but this time for the optimal radius ρ. The optimal cylinder is shown on the lower right.

Parameterized, linearly constrained programs with convex quadratic objective function are studied for instance in [27], [28], and [70], and in particular, the stability of the solution under small perturbations of the parameters is analyzed. By all means, in our case, the parameter ρ is only affecting the right hand side of the linear constraints, and the overall problem structure is simple. Though technical, the following analysis is straightforward and directly shows that an optimal value close enough to 1 implies that the radius is close to optimal.

Lemma 5.19. *Let $P \subset \mathbb{R}^n$ and $0 \notin \operatorname{conv}(P)$. Let $0 \leq \rho_1 < \rho_2 < \max_i \|p_i\|$, and let v_1, v_2 be optimal solutions of Program (5.6) for ρ_1, ρ_2 respectively. It holds that*

$$\rho_2 - \rho_1 \leq (\|v_1\| - \|v_2\|)\frac{2\max_i \|p_i\|^2}{\rho_1 + \rho_2}.$$

Proof. We know that neither v_1 nor $v_2 = 0 \in \mathbb{R}^n$ as both radii are strictly smaller than $\max_i \|p_i\|$. Since $\rho_1 < \rho_2$, v_1 is feasible for ρ_2:

$$\langle v_1, p_i\rangle \geq \sqrt{\|p_i\|^2 - \rho_1^2} > \sqrt{\|p_i\|^2 - \rho_2^2} \qquad \rho_2 < \|p_i\|$$

When we scale v_1 by a factor of $\max_i\big(1 - (\sqrt{\|p_i\|^2 - \rho_1^2} - \sqrt{\|p_i\|^2 - \rho_2^2})/\langle v_1, p_i\rangle\big)$, all these constraints stay feasible. We can bound this factor by

$$\begin{aligned}
&\max_i \left(1 - \frac{\sqrt{\|p_i\|^2 - \rho_1^2} - \sqrt{\|p_i\|^2 - \rho_2^2}}{\langle v_1, p_i\rangle}\right) \\
&= 1 - \frac{\rho_2^2 - \rho_1^2}{\max_i \langle v_1, p_i\rangle(\sqrt{\|p_i\|^2 - \rho_1^2} + \sqrt{\|p_i\|^2 - \rho_2^2})} \\
&\leq 1 - \frac{\rho_2^2 - \rho_1^2}{2\|v_1\| \max_i \|p_i\|^2}.
\end{aligned}$$

Since v_2 is optimal for ρ_2, we have that

$$\|v_2\| \leq \|v_1\| \left(1 - \frac{\rho_2^2 - \rho_1^2}{2\|v_1\| \max_i \|p_i\|^2}\right),$$

proving the claimed statement. □

Remark 5.20. *Note that an inverse statement of Lemma 5.19 does not hold without further assumptions. For instance, we can choose P such that $\|p_i\| = 1$ for all p_i and $\rho(P)$ is strictly smaller, but arbitrarily close to 1. When $\rho = \rho(P)$, we get a solution v with $\|v\| = 1$. When $\rho = 1$, however, the optimal solution of Program (5.6) is $v = 0$ with optimal value 0.*

In the following, we address the required approximation quality. Let $\rho > 0$ be a fixed radius, and for all $\|p_i\| > \rho$ let $\pi_i = \sqrt{\|p_i\|^2 - \rho^2}$, and let $\hat{\pi}_i$ denote an approximation of π_i. We require that

$$\frac{|\hat{\pi}_i - \pi_i|}{\pi_i} \leq \delta_1$$

and $\hat{\pi}_i \geq 0$ for all i with $\|p_i\| > \rho$. Let v denote the optimal solution of the convex program with right hand sides π_i, and let $\hat{v}$ denote an approximation of the optimal solution of the program with right hand sides $\hat{\pi}_i$. We may assume that $\hat{v}$ is feasible for this program and that the value of the objective function is within an additive error δ_2 of the optimum. Such a $\hat{v}$ can be found in polynomial time e.g. by an interior-point method. Since we know that $\pi_i \leq \langle v, p_i \rangle$ for all p_i, we also have that

$$\hat{\pi}_i \leq \left(1 + \frac{\hat{\pi}_i - \pi_i}{\langle v, p_i \rangle}\right) \langle v, p_i \rangle \leq \left(1 + \frac{|\pi_i - \hat{\pi}_i|}{\pi_i}\right) \langle v, p_i \rangle,$$

implying

$$\|\hat{v}\| - \delta_2 \leq \|v\| \max_i \left(1 + \frac{|\pi_i - \hat{\pi}_i|}{\pi_i}\right). \tag{5.7}$$

Of course, we can use the same argument again with the roles of v and $\hat{v}$ exchanged, yielding

$$\|v\| \leq \|\hat{v}\| \max_i \left(1 + \frac{|\pi_i - \hat{\pi}_i|}{\hat{\pi}_i}\right). \tag{5.8}$$

Putting (5.7) and (5.8) together, we get

$$\big|\|v\| - \|\hat{v}\|\big| \leq \max_i \frac{|\pi_i - \hat{\pi}_i|}{\pi_i - |\pi_i - \hat{\pi}_i|} \left(\|v\| + \big|\|v\| - \|\hat{v}\|\big|\right) + \delta_2,$$

and with

$$\max_i \frac{|\pi_i - \hat{\pi}_i|}{\pi_i - |\pi_i - \hat{\pi}_i|} \leq \frac{\delta_1}{1 - \delta_1}$$

this implies

$$\big|\|v\| - \|\hat{v}\|\big| \leq \frac{\delta_1 \|v\| + (1 - \delta_1)\delta_2}{1 - 2\delta_1}. \tag{5.9}$$

Now assume that the approximate solution of Program (5.6) has a value larger than 1 for the radius ρ, but ρ is feasible, or vice versa. This implies that 1 is in the interval between the exact optimal value $\|v\|$ and our approximation $\|\hat{v}\|$, and we can ensure that this interval is sufficiently small by our choice of δ_1 and δ_2. In particular, in case of such a wrong decision, it holds that

$$\big|\|v\| - 1\big| \leq \frac{\delta_1 + (1 - \delta_1)\delta_2}{1 - 3\delta_1}. \tag{5.10}$$

With the statement from Lemma 5.19, we can use this bound to make sure that a wrong decision can only happen when ρ is already very close to $\rho(P)$. We are now ready to consider the binary search algorithm for the anchored 1-ray problem, Algorithm 5.2.

Algorithm 5.2 Binary search for anchored 1-ray radius

Input: $P = \text{conv}\{p_1, \ldots, p_m\}$, $\varepsilon' > 0$, ρ_1 a lower and ρ_2 an upper bound for the optimal radius

Output: ρ radius within $(1 + \varepsilon')$ times the optimum

```
set $\delta_0 = \rho_1\varepsilon'/2$
set $\delta_1 = (\rho_1/\max_i \|p_i\|)^2\varepsilon'/16$
set $\delta_2 = (\rho_1/\max_i \|p_i\|)^2\varepsilon'/8$
while $\rho_2 - \rho_1 > \delta_0$ do
    set $\rho = (\rho_1 + \rho_2)/2$
    for $i = 1, \ldots, m$ do
        let $\hat{\pi}_i$ be an approximation of $\pi_i$ up to a relative accuracy of $\delta_1$
    end for
    solve Program (5.6) with $\hat{\pi}_i$ up to an accuracy of $\delta_2$
    if program feasible and approximated optimal value $\|\hat{v}\| < 1$ then
        set $\rho_2 = \rho$
    else
        set $\rho_1 = \rho$
    end if
end while
```

Lemma 5.21. *For any given accuracy ε', Algorithm 5.2 finds an ε'-approximation for the optimal radius of the anchored 1-ray problem taking at most $O(\log(1/\varepsilon'))$ iterations.*

Proof. We use Lemma 5.19 and the bound from the inequality in (5.10). By a simple calculation, setting the values of δ_1 and δ_2 as in Algorithm 5.2, we see that $|\rho(P) - \rho| \leq \varepsilon'/2\rho(P)$ in case of a wrong decision. By the value of δ_0, we assure that this does no harm to the overall approximation quality. The number of iterations depends only on ε' and the quality of the initial bounds ρ_1 and ρ_2. We know that we can easily find such bounds that are at most by a factor of 2 apart (see Section 5.2.1). $\square$

5.2.4 Core-Set Based Approximation Algorithms for Cylinders

We now show that we can use Algorithm 5.2 within Algorithm 5.1 in order to approximate the anchored 1-rays, that is, we can rely on an approximation of $\rho(S)$

and $a(S)$ instead of the exact values. The analysis of the core-set size still holds up to a constant factor.

Details for the Anchored 1-Ray Core-Set Algorithm

In order to obtain the desired approximation quality overall, we need to set the accuracy both in Algorithm 5.1 and in Algorithm 5.2 appropriately. Note that we use ε for Algorithm 5.1 and ε' for Algorithm 5.2.

Within Algorithm 5.1, we do not only need the radius for the set S, but also an axis in order to check whether the current cylinder already approximately covers the point set P. We consider the axis $\hat{v}$ computed within Algorithm 5.2 for the return value ρ. In particular, we show that this axis satisfies an approximate version of the half-space lemma which we need to adapt the proof of Theorem 5.14. We derive some properties of the output of Algorithm 5.2 for a given point set P.[7] We again assume $\rho(P) > 0$ in the following.

The first lemma considers the optimal solution of Program (5.6) which satisfies a kind of half-space property, too.

Lemma 5.22. *Let P be a point set, $\rho > 0$ a given radius such that $0 \notin \operatorname{conv}(P \backslash \rho\mathbb{B})$, and let v be the exact optimal solution of Program (5.6) for ρ. For any vector $w \perp v$, there is a point p_i in P such that $\langle w, p_i \rangle \leq 0$ and $\langle v, p_i \rangle = \pi_i$.*

Proof. It follows from the premises that $\langle v, p_i \rangle > 0$ for all p_i and that $\|v\| > 0$, too. Consider the dual[8] of Program (5.6), that is,

$$\begin{aligned} \max \sum_{i=1}^{m} \pi_i \omega_i & \\ \left\| \sum_{i=1}^{m} \omega_i p_i \right\| &\leq 1 \\ \omega_i &\geq 0 \qquad 1 \leq i \leq m \end{aligned}$$

where ω_i denote the dual variables. Both the primal and the dual problem are strictly feasible due to our assumptions. Using the strong duality theorem, we get that

$$\omega_i > 0 \Leftrightarrow \langle v, p_i \rangle = \pi_i \quad \text{and} \quad v = \|v\| \sum_{i=1}^{m} \omega_i p_i$$

hold for primal and dual optimal solutions (see, e.g., [106]). This means that v is in the positive hull of the points p_i whose constraints are satisfied with equality.

[7]We later apply these results to the point sets S within Algorithm 5.1.

[8]One can also prove Lemma 5.22 without using duality theory, but by explicitly constructing a better axis in case the claim is not satisfied. The elementary proof is, however, longer and more technical.

When we project these points p_i onto $v^\perp$, we get that

$$0 \in \operatorname{conv}(\operatorname{proj}_{v^\perp}(p_i)).$$

Therefore, the claim is satisfied for any $w \in v^\perp$. □

The aim is now to derive an approximate version of the half-space lemma, which can be achieved by Lemma 5.22 and some technical calculations, resulting in Lemma 5.26. In order to get a half-space property of the type we need, we consider the distance of the points in P to the axis spanned by $\hat{v}$, that is, $\|p - \langle p, \hat{v}\rangle \hat{v}/\|\hat{v}\|^2\|$. We therefore need to bound $\|\hat{v}\|$. As a first step, we consider $\|v\|$.

Lemma 5.23. *Let P such that $\min_i \|p_i\| \geq (1+\varepsilon)\rho(P)$, let ρ be the radius computed by Algorithm 5.2 for accuracy $\varepsilon' \leq \frac{1}{4}\varepsilon^2$, and let v be the exact optimal solution of Program (5.6) for ρ. It holds that*

$$\big|1 - \|v\|\big| \leq \frac{\varepsilon'}{\varepsilon}.$$

Proof. In order to bound $|\|v\| - 1|$, we again apply the method used for Lemma 5.19 as well as (5.7) and (5.8). Here, we get

$$\begin{aligned}
\big|1 - \|v\|\big| &\leq \max_i \frac{\big|\sqrt{\|p_i\|^2 - \rho(P)^2} - \sqrt{\|p_i\|^2 - \rho^2}\big|}{\sqrt{\|p_i\|^2 - \rho(P)^2} - \big|\sqrt{\|p_i\|^2 - \rho(P)^2} - \sqrt{\|p_i\|^2 - \rho^2}\big|} \\
&\leq \max_i \frac{|\rho(P)^2 - \rho^2|}{\|p_i\|^2 - \rho(P)^2 - |\rho(P)^2 - \rho^2|} \\
&\leq \frac{2\varepsilon' + \varepsilon'^2}{2\varepsilon + \varepsilon^2 - (2\varepsilon' + \varepsilon'^2)} \\
&\leq \frac{\varepsilon'}{\varepsilon}\,\frac{2 + \varepsilon^2/4}{2 + \varepsilon/4} \leq \frac{\varepsilon'}{\varepsilon},
\end{aligned}$$

since $\varepsilon' \leq \varepsilon^2/4$. □

The assertion here is inverse to the statement in Lemma 5.19. However, compared to the proof there, we need the additional assumption $\min_i \|p_i\| \geq (1+\varepsilon)\rho(P)$, compare Remark 5.20. We can now easily bound $\|\hat{v}\|$.

Corollary 5.24. *Let P such that $\min_i \|p_i\| \geq (1+\varepsilon)\rho(P)$, let ρ and $\hat{v}$ be the radius and axis direction computed by Algorithm 5.2 for accuracy $\varepsilon' \leq \varepsilon^2/4$, and let v be the exact optimal solution of Program (5.6) for ρ. It holds that*

$$|1 - \|\hat{v}\|| \leq (1+\delta_1)\frac{\varepsilon'}{\varepsilon} + 4\delta_1.$$

Proof. With Lemma 5.23, the inequality in (5.9), and the values for δ_1 and δ_2 from Algorithm 5.2, we get

$$\begin{aligned}|1 - \|\hat{v}\|| &\leq \big|1 - \|v\|\big| + \big|\|v\| - \|\hat{v}\|\big| \\ &\leq \frac{\varepsilon'}{\varepsilon} + \frac{\delta_1\|v\| + (1-\delta_1)\delta_2}{1-2\delta_1} \\ &\leq \frac{1-\delta_1}{1-2\delta_1}\frac{\varepsilon'}{\varepsilon} + \frac{3\delta_1 - 2\delta_1^2}{1-2\delta_1} \\ &\leq (1+\delta_1)\frac{\varepsilon'}{\varepsilon} + 4\delta_1\end{aligned}$$

as claimed. □

In order to simplify the final calculation, we also show that v and $\hat{v}$ are close.

Lemma 5.25. *Let P such that $\min_i \|p_i\| \geq (1+\varepsilon)\rho(P)$, let ρ and $\hat{v}$ be computed by Algorithm 5.2 for accuracy $\varepsilon' \leq \varepsilon^2/4$, and let v be the exact optimal solution of Program (5.6) for ρ. It holds that*

$$\|v - \hat{v}\| \leq \left(3\frac{\varepsilon'}{\varepsilon} + 5\right)\sqrt{\delta_1}.$$

Proof. Adding up the inequalities in Program (5.6) for the exact solution and the approximation, we get

$$\langle p_i, v + \hat{v}\rangle \geq \pi_i + \hat{\pi}_i,$$

and with the same argument as before,

$$\|v\| \leq \|v + \hat{v}\| \max_i \frac{\pi_i}{\pi_i + \hat{\pi}_i} \leq \|v + \hat{v}\|\frac{1}{2-\delta_1}.$$

Consequently,

$$(3 - 4\delta_1 + \delta_1^2)\|v\|^2 - \|\hat{v}\|^2 \leq 2\langle v, \hat{v}\rangle$$

and therefore we get for the distance of v and $\hat{v}$

$$\|v - \hat{v}\|^2 = \|v\|^2 + \|\hat{v}\|^2 - 2\langle v, \hat{v}\rangle \leq 2\|\hat{v}\|^2 - (2 - 4\delta_1 + \delta_1^2)\|v\|^2.$$

In particular, using (5.7), the values for δ_1 and δ_2, and Lemma 5.23, it holds that

$$\begin{aligned}\|v - \hat{v}\| &\leq \sqrt{2\big(\|v\|(1+\delta_1) + \delta_2\big)^2 - \big(2 - 4\delta_1 + \delta_1^2\big)\|v\|^2} \\ &= \sqrt{(8\delta_1 + \delta_1^2)\|v\|^2 + 4\delta_2(1+\delta_1)\|v\| + 2\delta_2^2} \\ &\leq \sqrt{(8\delta_1 + \delta_1^2)\frac{\varepsilon'^2}{\varepsilon^2} + (24\delta_1 + 10\delta_1^2)\frac{\varepsilon'}{\varepsilon} + 16\delta_1 + 17\delta_1^2} \\ &\leq \left(3\frac{\varepsilon'}{\varepsilon} + 5\right)\sqrt{\delta_1}\end{aligned}$$

finishing the proof. □

We are now ready to proof the approximation version of the conic half-space lemma, Lemma 5.13.

Lemma 5.26. *Let P such that $\min_i \|p_i\| \geq (1+\varepsilon)\rho(P)$, and let $\hat{v}$ be an approximate axis computed by Algorithm 5.2 for accuracy $\varepsilon' \leq \varepsilon^2/4$. For any $w \perp \hat{v}$, $\|w\| = 1$, there is a point p_i in P close to the half-space opposed to w*

$$\langle w, p_i \rangle \leq 2\sqrt{\varepsilon'}\left(\frac{\varepsilon'}{\varepsilon}+1\right)\left\langle \frac{\hat{v}}{\|\hat{v}\|}, p_i \right\rangle$$

and close to the boundary of the enclosing cylinder

$$|\operatorname{dist}(p_i, \operatorname{pos}(\hat{v}))^2 - \rho(P)^2| \leq \left(\frac{\max_i \|p_i\|^2}{\rho(P)^2}\left(5\frac{\varepsilon'}{\varepsilon}+3\sqrt{\varepsilon'}\right)+2\varepsilon'+\varepsilon'^2\right)\rho(P)^2.$$

Proof. Let w be a direction vector with $\|w\| = 1$ and $\hat{v} \perp w$.

Consider the projection of w onto $v^\perp$, the hyperplane perpendicular to the optimal solution of Program (5.6) for the computed radius ρ, that is, $w - \langle v, w\rangle v$. From Lemma 5.22, we know that a point p_i exists with $p_i \in P$, $\langle w - \langle v, w\rangle v, p_i\rangle \leq 0$ and $\langle v, p_i\rangle = \pi_i$. This implies that $\langle w, p_i\rangle \leq \langle v, w\rangle\langle v, p_i\rangle$. It holds that $\langle v, w\rangle = \langle v - \hat{v}, w\rangle$ since $w \perp \hat{v}$, so

$$\langle w, p_i\rangle \leq \langle v - \hat{v}, w\rangle\langle v, p_i\rangle = \langle v - \hat{v}, w\rangle\pi_i \leq \|v - \hat{v}\|\frac{\hat{\pi}_i}{1-\delta_1} \leq \|v - \hat{v}\|\frac{\langle \hat{v}, p_i\rangle}{1-\delta_1}$$

With Corollary 5.24 and Lemma 5.25, we have

$$\begin{aligned}\langle w, p_i\rangle &\leq \frac{\left(3\frac{\varepsilon'}{\varepsilon}+5\right)\sqrt{\delta_1}}{(1-\delta_1)\left(1-(1+\delta_1)\frac{\varepsilon'}{\varepsilon}-4\delta_1\right)}\left\langle \frac{\hat{v}}{\|\hat{v}\|}, p_i\right\rangle \\ &\leq \frac{\left(3\frac{\varepsilon'}{\varepsilon}+5\right)\frac{\sqrt{\varepsilon'}}{4}}{1-\frac{5}{16}\varepsilon'-\frac{\varepsilon'}{\varepsilon}} \\ &\leq 2\sqrt{\varepsilon'}\left(\frac{\varepsilon'}{\varepsilon}+1\right)\left\langle \frac{\hat{v}}{\|\hat{v}\|}, p_i\right\rangle.\end{aligned}$$

and the first statement holds. Now consider the distance of p_i to the axis spanned by $\hat{v}$:

$$\begin{aligned}\operatorname{dist}(p_i, \operatorname{pos}(\hat{v}))^2 &= \|p_i\|^2 - \left\langle \frac{\hat{v}}{\|\hat{v}\|}, p_i\right\rangle^2 \\ &= \|p_i\|^2 - \langle v, p_i\rangle^2 + \langle v, p_i\rangle^2 - \left\langle \frac{\hat{v}}{\|\hat{v}\|}, p_i\right\rangle^2 \\ &= \rho^2 + \langle v, p_i\rangle^2 - \left\langle \frac{\hat{v}}{\|\hat{v}\|}, p_i\right\rangle^2\end{aligned}$$

Consequently, we look for an upper bound for

$$\begin{aligned}\left|\langle v,p_i\rangle^2-\left\langle\frac{\hat v}{\|\hat v\|},p_i\right\rangle^2\right| &= \left(\langle v,p_i\rangle+\left\langle\frac{\hat v}{\|\hat v\|},p_i\right\rangle\right)\left\langle v-\frac{\hat v}{\|\hat v\|},p_i\right\rangle\\ &\le \big(\|v\|+1\big)\|p_i\|^2\left\|v-\frac{\hat v}{\|\hat v\|}\right\|\\ &\le \left(2+\frac{\varepsilon'}{\varepsilon}\right)\|p_i\|^2\big(\|v-\hat v\|+|\|\hat v\|-1|\big)\\ &\le \left(2+\frac{\varepsilon'}{\varepsilon}\right)\left(\left(3\frac{\varepsilon'}{\varepsilon}+5\right)\sqrt{\delta_1}+(1+\delta_1)\frac{\varepsilon'}{\varepsilon}+4\delta_1\right)\|p_i\|^2\\ &\le \left(5\frac{\varepsilon'}{\varepsilon}+3\sqrt{\varepsilon'}\right)\|p_i\|^2\end{aligned}$$

which holds with Corollary 5.24, Lemma 5.25, and since $\delta_1\le\varepsilon'/16$. Altogether,

$$\begin{aligned}|\operatorname{dist}(p_i,\operatorname{pos}(\hat v))^2-\rho(P)^2| &\le |\operatorname{dist}(p_i,\operatorname{pos}(\hat v))^2-\rho^2|+|\rho^2-\rho(P)^2|\\ &\le \left(\frac{\max_i\|p_i\|^2}{\rho(P)^2}\left(5\frac{\varepsilon'}{\varepsilon}+3\sqrt{\varepsilon'}\right)+2\varepsilon'+\varepsilon'^2\right)\rho(P)^2\end{aligned}$$

proving the second statement. □

Let us now reconsider the proof of Theorem 5.14. The calculations for the first case in the proof do not use any specific properties of the computed cylinder. The second case, however, uses Lemma 5.13 which is now replaced by the approximative version from Lemma 5.26.

Recall the following terms from the proof of Theorem 5.14: We consider a set $S\subset P$ and add a new core-set point p. The new axis direction is denoted by v'. It holds that $\max_i\|p_i\|^2/\rho(P)^2\le\varepsilon^{-2\gamma}$ and $\delta=\varepsilon^{4\gamma+3}/16$.

Instead of the exact axis for S, we now have an approximation of the old axis direction $\hat v/\|\hat v\|$. Moreover, a point $q\in P$ exists, satisfying the conditions from Lemma 5.26. We choose $\varepsilon'=\delta^3/4$. With this choice, we have

$$2\sqrt{\varepsilon'}\left(\frac{\varepsilon'}{\varepsilon}+1\right)\le\frac{\delta}{2}\quad\text{and}\quad\frac{\max_i\|p_i\|^2}{\rho(P)^2}\left(5\frac{\varepsilon'}{\varepsilon}+3\sqrt{\varepsilon'}\right)+2\varepsilon'+\varepsilon'^2\le\varepsilon\delta$$

for the two factors from Lemma 5.26. Consequently, two new terms come up in the calculation in (5.4). We get

$$\begin{aligned}\rho(S\cup\{p\})^2 &\ge \|q\|^2-\left(1-\delta+\frac{\delta}{2}\right)^2\left\langle q,\frac{\hat v}{\|\hat v\|}\right\rangle^2\\ &= \|q\|^2-\left(1-\frac{\delta}{2}\right)^2\left(\|q\|^2-\left\|q-\left\langle q,\frac{\hat v}{\|\hat v\|}\right\rangle\frac{\hat v}{\|\hat v\|}\right\|^2\right)\\ &\ge \left(1-\left(1-\frac{\delta}{2}\right)^2\right)(1+\varepsilon)^2\rho(S)^2+\left(1-\frac{\delta}{2}\right)^2(1-\varepsilon\delta)\,\rho(S)^2.\end{aligned}$$

This results in a change by a constant factor in (5.5). More precisely, we get the following:

Remark 5.27. *Using an approximation for $\rho(S)$ and $a(S)$ as described above, it holds that the (exact) radius for S increases by at least*

$$\rho(S \cup \{p\}) - \rho(S) \geq (\sqrt{1+\varepsilon\delta} - 1)\rho(S) \geq \frac{\varepsilon^{4\gamma+3}}{40}\rho(S)$$

when p is added to the set S.

The fact that we have an ε'-approximation of the optimal radius in Algorithm 5.1 adding a little to the overall error implies that we should choose the accuracy ε in Algorithm 5.1 slightly smaller than the desired accuracy. Again, this results in a constant factor close to one.

In order to save steps when calling Algorithm 5.2, we recommend to use the best values for ρ_1 and ρ_2 available, that is the best upper bound found so far and the lower bound obtained for the preceding set S.

Finally, we discuss how to deal with the condition $\min_i \|p_i\| \geq (1+\varepsilon)\rho(P)$. One possibility is to perform another binary search at the top level, calling the core-set algorithm to decide whether the current radius is feasible, resulting in $O(\log(1/\varepsilon))$ executions of Algorithm 5.1. This approach has the drawback that it is likely that similar core-sets are computed over and over, which is an undesirable overhead from a practical viewpoint. It seems preferable to reuse already computed core-sets. When we use Algorithm 5.1 for the original point set P, simply ignoring the condition, the method is still correct. However, we cannot prove that the lower bound increases by at least a fixed positive quantity in each iteration step. A third possibility is combining the two ideas: For a given radius ρ, we may use all the core-set points computed so far (for other values of ρ) as an initial core-set. Summing up all the sizes of all these sets, we immediately get an upper bound of $O(\log(1/\varepsilon))$ times the original bound for the core-set size. In experiments, the computed core-sets turn out to be small even when we simply drop the condition $\min_i \|p_i\| \geq (1+\varepsilon)\rho(P)$ (see Section 5.2.5).

Branch-and-Bound for Anchored Cylinders

Labeling the core-sets as described in Section 5.2.2, we can approximate anchored cylinders, too. In order to explore all possible labelings, we use a B&B algorithm related to the one described in Section 4.3.1. Naturally, we can use the same approach for anchored k-line or k-ray center problems, too. The resulting algorithm is a PTAS for fixed k and point sets with bounded ratio $\max_i \|p_i\|/\rho(P)$. As far as we know, this is the first approximation algorithm with polynomial running time for these problems in general dimension.

Algorithm 5.3 identifies a core-set $S \subset P \cup -P$ as well as an axis direction v and a radius ρ of an enclosing cylinder. The algorithm is written down recursively since this simplifies the presentation, though the implementation used in Section 5.2.5 does not use recursion. We also omit the accuracies determined in Section 5.2.4 in order to improve the readability.

Algorithm 5.3 Branch-and-bound for anchored cylinders

Input: P a point set, $\varepsilon > 0$

Output: a core-set with $S \subset P \cup -P$, v, ρ approximate axis direction and radius

```
initialize:
set $S = \arg\max_i \|p_i\|$, $\rho = 0$
set the global upper bound $\bar{\rho} = \max_i \|p_i\|$

anchored_cylinder($S$, $\rho$, $\bar{\rho}$):
approximate the anchored 1-ray of $S$
let $v$ be the axis direction and $\rho$ the radius
if $\rho \leq \bar{\rho}$ then
    if $P \subset \text{lin}(v) + (1+\varepsilon)\rho\mathbb{B}$ then
        update the global upper bound $\bar{\rho}$
    else
        let $p$ be a point in $P$ with maximal distance to $\text{lin}(v)$
        if $\langle v, p \rangle \geq 0$ then
            set $\sigma = 1$
        else
            set $\sigma = -1$
        end if
        call anchored_cylinder($S \cup \{\sigma p\}$, $\rho$, $\bar{\rho}$)
        call anchored_cylinder($S \cup \{-\sigma p\}$, $\rho$, $\bar{\rho}$)
    end if
end if
return the best $S$, $v$, and $\rho$ found
```

When approximating the anchored 1-ray of S in Algorithm 5.3, it can of course happen that a set S generated in the course of the algorithm does not admit a 1-ray solution with a radius between ρ and $\bar{\rho}$. In this case, we do not need to further consider the current node since the subtree below does not provide any better solutions than the best one already found. Also note that the new nodes are sorted such that the more promising node is considered first. Therefore, we check whether the new core-set point is on the "+" or "−" side according to the current axis, and use this sign first. This helps to find a good upper bound at an early stage of the B&B process.

Concerning "ball-like" versus "long and skinny" point sets, we get larger core-sets in the second case (see Section 5.2.2). However, the B&B algorithm honors a large ratio of $\max_i \|p_i\|$ and the optimal radius. The assignment of "+" and "−" labels to points is obvious in many cases, and therefore, a lot of branches of the tree can be cut at a very early stage.[9] In this sense, finding smallest enclosing anchored cylinders for "long and skinny" point sets is in fact easier.

The number of nodes in the B&B tree is bounded by $2^{|S|}$, where $|S|$ is the size of the largest core-set occurring during the run of the algorithm. In practice, the computed core-sets turn out to be much smaller than predicted by the worst-case bounds for $|S|$.

Non-Anchored Cylinders

Considering "free", that is non-anchored cylinders, the core-sets constructed so far do not provide new insights for this problem. It remains an open problem whether dimension-independent core-set sizes are possible for non-anchored cylinders. Yet, we can still make use of the algorithm described for the anchored cylinder problem. As described in [19], one can find a point sufficiently close to an optimal axis of a smallest enclosing free cylinder by checking an ε-net of size $O(2^{-\log(\varepsilon)/\varepsilon^2})$ for all subsets of P containing $O(1/\varepsilon^2)$ points. This is improved to $O(1/\varepsilon)$ in [18] and [102], resulting in ε-nets of size $O(2^{-\log(\varepsilon)/\varepsilon})$. We can translate this point to the origin and then solve the anchored version of the problem.

A number of smallest enclosing cylinder algorithms exist [19], [76], and [77] which do not compute core-sets but use related concepts to generate an axis close to optimal. In [19] and [76], the point set P is embedded in a lower dimensional subspace. An optimal solution of the lower dimensional problem does not necessarily correspond to a solution of the original problem. However, one can determine a number of possible candidate lines in the subspace, and at least one of them has to be close to an optimal axis. The stated running times are polynomial in the dimension n, but involve a term of $m^{\varepsilon^{O(1)}}$. In [77], however, a linear time approximation algorithm for the smallest enclosing cylinder is presented. It can be seen as an extension of the incremental approach (see also Sections 3.1.2 and 4.3.1). In each iteration, a set of lines is generated such that, checking all lines, one finds at least one that is close to an optimal axis.

We resume that, combining the core-set algorithm developed here with "guessing" a point near the axis is similar to the method presented in [19] for the cylinder problem, but the computation of the anchored cylinder here has a much simpler structure. Neither of the two methods is competitive in terms of worst case running time, since an algorithm for the free cylinder that is linear both in the number of points and the dimension exists [77]. We are, however, not aware of any practical implementation of this algorithm. (The running time involves quite a large constant

[9]This is also supported by the experimental results in Section 5.2.5.

depending on ε.) Our method seems to be quite practicable for anchored cylinder problems (see Section 5.2.5), but especially in higher dimensions, finding an anchor point gets computationally challenging. Practical algorithms for the smallest enclosing cylinder problem in *small* dimensions can be found in [41] and [141]. We therefore restrain the experiments in the following section to the anchored version, with a special focus on the practical core-set sizes.

5.2.5 Implementation and Experiments

In the following, we present and discuss some exemplary tests of the anchored ray and anchored cylinder algorithms.

Implementation

We use Matlab© implementations of Algorithms 5.1 and 5.3, calling the solver Xpress-MP© from a C++ program via the Matlab MEX interface[10] for the convex programs in Algorithm 5.2. Again, we use code provided by [39] for Euclidean distance computations. As stated before, the implementation of Algorithm 5.3 avoids recursion which is replaced by a node stack. The upper bound $\bar{\rho}$ is a global variable, ensuring that we always use the best value available no matter whether the node considered was generated at an earlier stage of the algorithm.

The reported running times are obtained for a SUNW SPARC Sun Fire 440 Workstation with a 1.3 GHz CPU and 1.6 GB RAM.

Note that we do not use the value for ε' derived in the previous section in the implementation. In theory, we choose ε' very small in order to prove an upper bound on the core-set size. In practice, we see that the core-sets stay small anyway. It also turns out that ignoring the condition of $\min_i \|p_i\| \geq (1+\varepsilon)\rho(P)$ does no harm to the size of the computed core-sets.[11]

Experimental Results for Anchored 1-Rays

In a first test, we compare the core-set approach for the anchored 1-ray problem to directly solving Program (5.6). The test results are resumed in Table 5.1. For point sets consisting of 1000 points, the direct solution is faster than the core-set

[10] see `http://www.mathworks.com` for details on Matlab MEX

[11] In [102], the authors mention a similar issue concerning the 1-center core-set algorithm (see Section 3.1.2). The core-set algorithm there is iterative as well, and the subproblems are Euclidean 1-center problems. The authors derive an accuracy of ε^2 for the subproblems. They write that they assumed an accuracy of ε in the first place (and suggest that the same holds for the results reported in [19]), which is insufficient in theory. However, they had already conducted their experiments with the accuracy for the subproblems set to ε only. They observe that the core-sets computed for the examples do not differ no matter whether ε or ε^2 is used for the subproblems.

		core-set algorithm		direct algorithm
n	m	core-set size	time (s)	time (s)
3	1000	3.58	0.14	(∗) 0.13
3	5000	3.45	0.16	0.36
3	10000	3.31	0.37	0.76
3	50000	3.19	0.50	4.26
10	1000	6.20	0.38	0.28
10	5000	6.52	0.50	0.86
10	10000	6.28	0.59	(∗) 1.60
10	50000	6.25	1.61	10.27
20	1000	8.58	0.83	0.60
20	5000	8.96	1.09	1.89
20	10000	9.46	1.42	4.24
20	50000	9.71	3.74	27.91
30	1000	10.48	1.41	1.04
30	5000	11.48	1.95	4.43
30	10000	11.26	2.34	8.86
30	50000	12.50	6.27	56.06

Table 5.1: Test results for the anchored 1-ray problem, both for the direct solution via quadratic programming and the core-set algorithm. We list m and n and report the average running time for both algorithms and the average core-set size for $\varepsilon = 0.01$. The data sets are generated as $(0, 1)$ normally distributed data, and then randomly scaled and shifted (such that the anchor point is in P). The sample size is 100, except for the rows indicated by $(*)$, where the direct algorithm fails for one instance and the average running time of the direct algorithm is indicated for the remaining 99 instances.

algorithm but gets slower for larger m. This is not surprising when we recall the results from Section 3.3. The standard quadratic programming solvers are designed for cases where the numbers of rows and columns in the constraint matrix are comparable in size, or, in case of large input dimension, when the constraint matrix is sparse. Neither of the two is applicable in our case. The same observation is made in [58] where an exact LP-type solver for convex quadratic programs is compared to other approaches, one of them a standard quadratic programming solver. As in an LP-type algorithm, the core-set approach can take advantage of the small number of points necessary to find a good solution. The results in Table 5.1 corroborate that the sizes of the core-sets in practice grow a little with the dimension, and are much smaller than predicted by the worst case analysis, just as in the case of the k-center core-sets in Section 4.3.4 and in [101]. In the dimensions considered here, the average core-set size gets only slightly larger when we make ε smaller than 0.01, and in many cases, the core-set points stay the same. The core-set algorithm is stable in all testes instances, but it happens that the direct algorithm fails, albeit rarely. In these cases, the solver terminated without finding a satisfactory solution. We resume that the direct approach is preferable only when the point set is not too large compared to the dimension.

Experimental Results for Anchored Cylinders

We proceed with the smallest enclosing anchored cylinder. In Table 5.2, the core-set size, the size of the branch-and-bound tree, and the running time for the smallest enclosing anchored cylinder of some exemplary input sets are listed. The algorithm is extremely fast for the geometric model data sets. We observe that the B&B trees generated for these data sets only have one leaf, since all other branches are discarded at an early stage. The sphere, however, is a kind of worst case input. Though the core-set stays small, many branches of the tree have to be explored. Nevertheless, the number of nodes in the tree is far from the worst-case bound here. We also consider cross polytopes in dimensions 5 and 10. For the 5-dimensional cross polytope, we get the exact solution for both values of ε considered.[12] In dimension 10, only the accuracy of 0.01 is small enough to make the algorithm return the exact solution. In this example, we have 2^n optimal axis directions. By fixing the sign of the first core-set point, we can discard half of them. We therefore get 2^{n-1} possible labeled core-sets of n points and $2^{n-1}-2$ inner nodes in the B&B tree. Of course, in higher dimensions, we can do with less than n core-set points for $\varepsilon = 0.01$.

In Figure 5.8, test results for the core-set algorithm in different dimensions are shown. The practical core-set sizes grow with the dimension and the number of points in P, but are still small, even in dimension 30. For the type of data set

[12]The running time for $\varepsilon = 0.1$ is smaller due to the faster calculations for the subproblems.

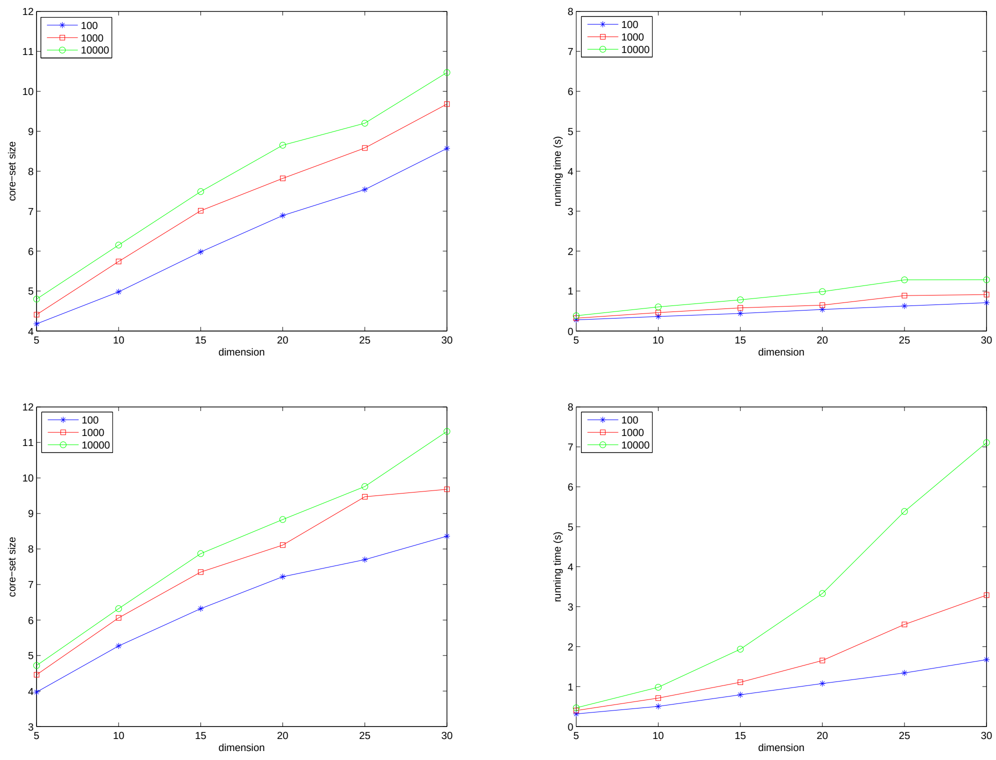


Figure 5.8: Test results for the smallest enclosing anchored cylinder algorithm. The average core-set size and running time for different dimensions is shown. The numbers of 100, 1000, and 10000 refer to m, the number of points in P and $\varepsilon = 0.01$. The data sets in the upper two graphs are generated as described in Table 5.1. For the data sets in the lower two graphs, we skipped the translation, so they have mean 0.

				B&B algorithm		
data set	n	m	ε	core-set size	nodes	time (s)
cat	3	352	0.01	6	10	2.32
shark	3	1744	0.01	4	6	1.40
seashell	3	18033	0.01	4	12	2.78
dragon	3	437645	0.01	4	6	2.03
sphere	3	1000	0.01	7	122	27.74
sphere	5	1000	0.01	11	1188	42.70
cross polytope	5	10	0.1	5	30	0.60
cross polytope	5	10	0.01	5	30	1.03
cross polytope	10	20	0.1	0	62	1.46
cross polytope	10	20	0.01	10	1022	28.07

Table 5.2: Test results for the smallest enclosing anchored cylinder algorithm on exemplary data sets. See Section 4.3.4 for the origin of the geometric model data sets. The “sphere” data sets refer to points on the surface of the unit sphere, and “cross polytope” indicates that P consists of the vertices of the unit cross polytope. We report the size of the core-set computed, the number of nodes processed during the B&B, and the running time in seconds.

in the upper two graphs, an approximate cylinder axis is usually not hard to find. The data sets are generated as $(0, 1)$ normally distributed data, and then randomly scaled and translated by choosing a random anchor point in P. The translation frequently makes finding the axis direction a lot easier, resulting in small B&B trees and running times. When we consider data sets with mean 0 as in the lower two graphs, the computed core-sets have almost the same size, but many more possible labelings have to be considered. This results in larger B&B trees and longer running times. Still, even in this case, we can find a good approximation for the smallest enclosing anchored cylinder within a few seconds for 10000 points in dimension 30.

In the next example, we consider smallest enclosing anchored cylinders of cylindrical data sets of different aspect ratio in order to have a look at the dependence of the core-set size and the shape of P. For this purpose, we generate data points equally distributed on a circular cylindrical surface for different ratios of cylinder height and radius. Using points exactly on the surface frequently results in core-sets consisting of two points only, independent of the dimension. We therefore slightly perturb the point sets. Figure 5.9 depicts the results of the calculation. We see a slow rise in the core-set size, especially in higher dimensions, but clearly, it is far from the theoretical upper bound. The running times are actually smaller for a larger ratio of cylinder height and radius in spite of the larger core-sets. This is again because the B&B tree can be cut down to almost a single branch for “long and skinny” point sets. For a ratio close to one, however, larger parts of the tree have be explored.

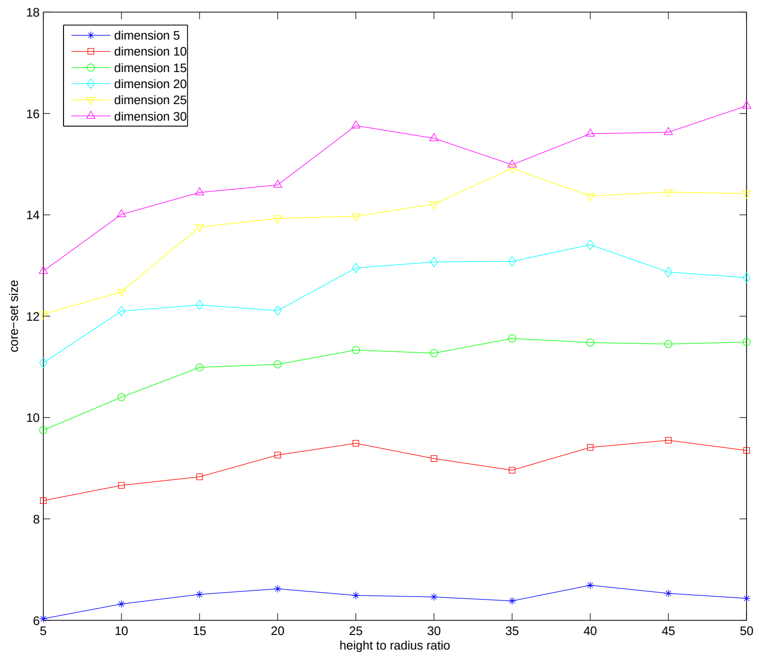


Figure 5.9: Test results for cylindrical data sets in dimensions 5 to 30. For each data point, 100 point sets are sampled. Point sets consist of 1000 points equally distributed on a cylindrical surface with a $(0, \varepsilon)$ normally distributed error. On the x-axis, the height to radius ratio of the cylindrical surface for the input data is shown, varying from 5 to 50. Again, the accuracy $\varepsilon = 0.01$.

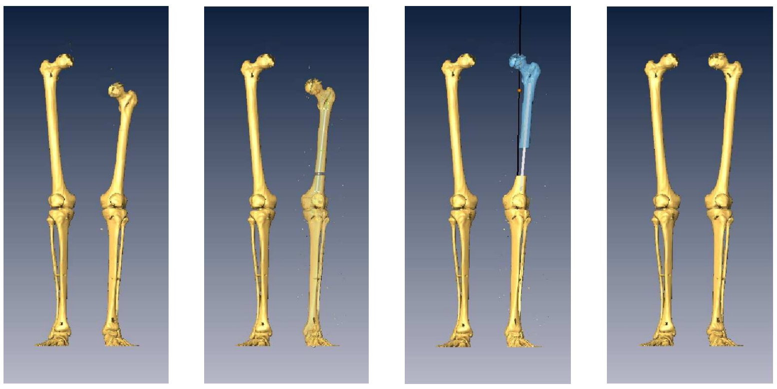


Figure 5.10: Schematic illustration of a deformity correction procedure: initial state, post-operational state, projected post-treatment state, and consolidated state.

5.3 A Cylinder Problem in Extremity Surgery

As announced in Section 1.1.3, we introduce an application in extremity surgery. It yields a containment problem involving the placement of a cylinder. This section differs from the preceding ones since it involves modeling for a specific application. Moreover, the resulting containment question is a case of inner containment, and the input is not a point set. The following is joint work with René Brandenberg, Tobias Gerken, and Peter Gritzmann, see also the publication [36].

5.3.1 Motivation

The problem setting from extremity surgery is the following: A patient suffers from a malalignment of an extremity which may be congenital or post-traumatic. A 3D model of the concerned bones is available from a CT scan. Such a scan consists of a number of layers in the direction of the body axis. The surgist determines a goal state of the extremity to be reached after treatment. Based on the goal state, a corresponding placement of the intramedullary nail is to be found.

The treatment itself consists of several steps: First, the bone is cut, the bone parts are aligned and the intramedullary nail is implanted. The nail can be lengthened using a built-in electric motor which is controlled from the outside. After the operation, the nail is slowly extended, making the bone tissue grow in the gap between the bone parts. This results in an overall lengthening of the bone in the direction of the implanted nail. Note that the precise placement of the nail is crucial since, once implanted, its position can only be changed by another operation. Figure 5.10 depicts the basic steps of the procedure. Here, we merely address the placement of the intramedullary nail. For other steps of the operation planning procedure and the medical background, see [36] and the references therein.

We model the intramedullary nail by a cylinder (see also Figure 1.3). In order to describe that the nail has to pass through the interior of the bone, we use ellipses to approximate the circumference of the bone in relevant CT layers. Note that these layers are usually not parallel any more since in order to correct the deformity, the bone parts are newly aligned before implanting the nail. Cutting the bone, aligned the parts, and implanting the lengthening device can be simulated in the planning procedure using the CT data (see [36] for the description of a software tool). Any feasible placement of the cylindrical nail has to intersect each layer within the interior of the ellipse defining the bone. The bone is therefore approximated by a piecewise convex object determined by the convex hulls of pairs of adjacent ellipses, and the cylinder has to be contained in this object.

For a good nail placement, we look for a corridor of maximal radius through the ellipses (compare Figure 5.11). Centering the nail in this corridor ensures that sufficient margins are left on all sides.

5.3.2 Problem Treatment

Though the problem originates from a 3D application, we consider the cylinder problem in general dimension. If appropriate, we address specific properties of the 3D case.

Problem Statement

Given a set of $(n-1)$-dimensional ellipsoids $E_1, \dots, E_m \subset \mathbb{R}^n$, we seek for a cylinder of maximal radius passing through all ellipsoids such that the intersection of an ellipsoid and the cylinder is contained in the ellipsoid. We therefore have the following problem

$$\begin{aligned} &\max\ \rho \\ &(a + \rho\mathbb{B}) \cap E_i \subset E_i \qquad 1 \le i \le m \end{aligned}$$

where the cylinder axis a may be an arbitrary line in $\mathbb{R}^n$. Note that this is an inner containment problem in the notion introduced in Section 2.1. In the following, we address the feasible cylinders as traversing cylinders of the set of ellipses.

As we try to find a maximal cylindrical corridor, this problem is related to some of the facility location and motion planning problems mentioned in Section 1.2.3.

Algorithm

We introduce a method to approximate a maximal traversing cylinder. We tackle the problem using a discretization of the direction space. It turns out that, when the direction of the axis is known, the problem is convex and can be formulated as an SDP. Using a discretization is justified here since we seek for an approximation of the maximal cylinder radius and are not interested in cylinders of very small radius.

Projecting the ellipsoids along a given axis, we get full dimensional ellipsoids in $\mathbb{R}^{n-1}$. In order to find a maximal cylinder with this axis direction, we have to find the largest ball in the intersection of the ellipsoids. This can be done as stated in [33]. Let $E_i = \{(x - z_i)^T A_i^{-1}(x - z_i) \leq 1\}$ denote a representation of the projection of the i-th ellipsoid, $1 \leq i \leq m$. We require that for any x with $\|x\| = 1$, $\rho x + c$ is contained in E_i. We get the following program, where $\rho,\ \lambda_i \in \mathbb{R}$, $c \in \mathbb{R}^{n-1}$:

$$\begin{aligned} &\max\ \rho \\ &\begin{pmatrix} A_i & -\rho\,\mathrm{Id} & c - z_i \\ -\rho\,\mathrm{Id} & \lambda_i\,\mathrm{Id} & 0 \\ (c - z_i)^T & 0 & 1 - \lambda_i \end{pmatrix} \succeq 0 \qquad 1 \leq i \leq m \\ &\lambda_i \geq 0 \qquad 1 \leq i \leq m \end{aligned} \tag{5.11}$$

It is not obvious that the program actually formulates the problem. Using Schur complement formula, we see that the matrix inequality in Program (5.11) is equivalent to

$$\begin{pmatrix} -\rho^2 A_i^{-1} & -\rho A_i^{-1}(c - z_i) \\ (c - z_i)^T(-\rho A_i^{-1}) & 1 - (c - z_i)^T A_i^{-1}(c - z_i) \end{pmatrix} - \lambda_i \begin{pmatrix} -\mathrm{Id} & 0 \\ 0 & 1 \end{pmatrix} \succeq 0.$$

This means that c and ρ are feasible for Program (5.11) if and only if $\lambda_i \geq 0$ exist satisfying the above matrix inequality. By terms of the S-procedure[13], the existence of such a multiplier λ_i is equivalent to the following condition which occurs in this form in [94, Theorem 3] and is due to [89], [90]:

$$\begin{gathered} -\rho^2 x^T A_i^{-1} x - 2x^T(\rho A_i^{-1})(c - z_i) + 1 - (c - z_i)^T A_i^{-1}(c - z_i) \geq 0 \\ \text{for all } x \text{ with } \ -x^T x + 1 \geq 0. \end{gathered}$$

Rewriting the first equation, we get

$$(c + \rho x - z_i)^T A_i^{-1}(c + \rho x - z_i) \leq 1,$$

which is just the inequality defining E_i, when x is replaced by $c + \rho x$. This holds for any x in the $(n-1)$-dimensional unit ball by the second equation, so the condition actually describes that the ball $c + \rho\mathbb{B}$ is covered by the ellipsoid.

We may of course assume that the set of ellipsoids is bounded, so we can use a discretization of the direction space to find an approximately optimal axis for any given accuracy ε. The SDP (5.11) has to be considered for each direction in the discretization in order to approximate the corresponding cylinder radius as described in Algorithm 5.4. See Figure 5.11 for an example. The implementation of the above algorithm used for Figure 5.11 is due to [61].

[13]The S-procedure is a sort of Lagrange relaxation method frequently used in problems with quadratic constraints, see [33], [94] for an introduction.

Algorithm 5.4 Largest traversing cylinder

Input: $E_1, \ldots, E_m \subset \mathbb{R}^n$ ellipsoids, $\varepsilon > 0$
Output: ε-approximation of the largest traversing cylinder (if existent)

```
discretize the upper half-sphere guaranteeing accuracy $\varepsilon$
set $\rho' = \infty$
**for** each direction vector $v$ in the discretization **do**
    project the ellipsoids into $v^\perp$
    **if** feasible point found in SDP (5.11) for the projected ellipsoids **then**
        let $\rho$, $c$ be an approximate solution
        **if** $\rho > \rho'$ **then**
            set $\rho' = \rho$, $c' = c$, $v' = v$
        **end if**
    **end if**
**end for**
**if** $\rho' < \infty$ **then**
    use $c'$ and $v'$ to obtain the corresponding axis $a$ in $\mathbb{R}^n$
    **return** $\rho'$ and $a$
**end if**
```

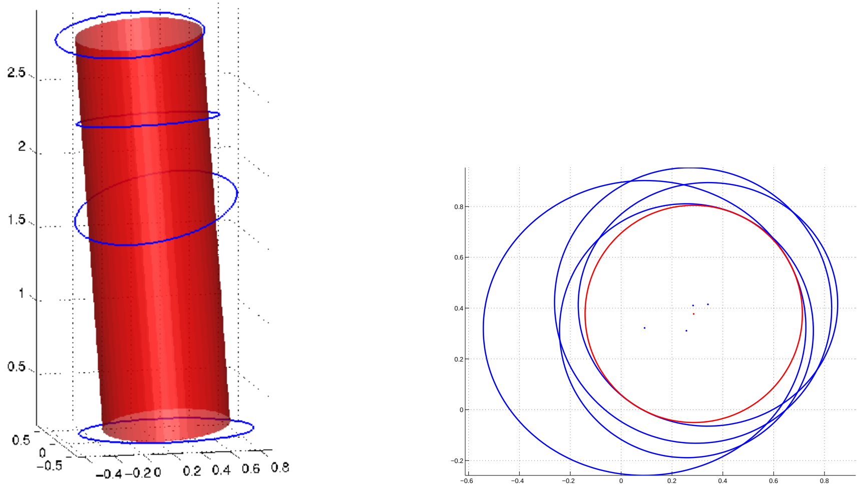


Figure 5.11: Four ellipses in 3D and the largest cylinder found by Algorithm 5.4. On the right is the 2D projection along the corresponding axis.

Practical Considerations

In order to achieve a practical algorithm, it is convenient to consider only parts of the discretization, not the full sphere. For inputs from the application, the area of the sphere admitting feasible cylinder directions is usually quite small. Moreover, from a practical viewpoint, it is justified to consider the 3D case only and impose some additional constraints on the input. Naturally, in the application, the order in which the ellipsoids are to be traversed by the cylinder is known. We can now, for instance, use the lower and the upper most ellipsoid to bound the cone of feasible directions.

Let E_1 and E_m denote the lower most and upper most ellipses, respectively. Any plane separating the two ellipses can be used to bound the cone of feasible directions. Orient the normal vector of the plane such that E_m is in the positive half-space. Any feasible line traversing E_1 before traversing E_m has to intersect the separating plane, enclosing an angle $\leq \pi$ with the normal vector. We can therefore get a polytopal approximation of the cone of feasible directions for E_1 and E_m. Let $v_1, \ldots, v_k$ be an outer approximation of E_1 by a convex polygon in $\operatorname{aff}(E_1)$, and $w_1, \ldots, w_k$ for E_l. In an optimal solution of the LP

$$\begin{aligned} &\max \delta \\ &\langle a, v_i \rangle \leq \delta \qquad 1 \leq i \leq k \\ &\langle a, w_i \rangle \geq \delta \qquad 1 \leq i \leq k \\ &\langle a, v_1 \rangle = \delta \\ &\langle a, v_2 \rangle = \delta \end{aligned}$$

equality holds for at least one w_i, and $\{x : \langle a, x \rangle = \delta\}$ defines a separating plane with v_1, v_2 on it. We can do this for each consecutive pair of v_i (or w_i, of course) and get a number of bounding planes for the cone of possible cylinder axes.

5.3.3 Background

The decision version of the traversing cylinder problem introduced here is a special stabbing or transversal problem.

$\mathbb{NP}$-Completeness

We show that the problem is $\mathbb{NP}$-complete in general dimension using a construction similar to those already considered in Section 5.1. Again, we reduce 3-SAT to the geometric problem.

Let ζ_i denote the variables in the set $\mathcal{K}$ of l clauses k. Set $n = l + 1$. In order to get a discrete structure for the solution space, we place $2n$ $(n-1)$-dimensional disks at the points $\pm e_i$, $1 \leq i \leq n$, such that the normal vector of the disk placed at e_i is also e_i (and $-e_i$ for $-e_i$). The squared radius of all the disks is set to $n-1$. Lines passing through all those disks are (as in Section 5.1.2) determined by the $v \in \{\pm 1\}^n$, and to avoid having two different representations for the same line we require the last coordinate of v to be 1.

Now we place a disk for each clause k. Without loss of generality, we may assume that each clause has three distinct variables. The centers of the disks are placed at points u, where

$$v_i = \begin{cases} 1 & i \le l \text{ and } \zeta_i \text{ occurs in } k \\ -1 & i \le l \text{ and } \neg\zeta_i \text{ occurs in } k \\ 0 & i \le l \text{ and neither } \zeta_i \text{ nor } \neg\zeta_i \text{ occurs in } k \\ 3 & i = n \end{cases}$$

The normal vector of the disk centered at u is also set to u. The squared radius of the disks representing clauses is chosen to be $36n$.

Let τ denote the number of true literals in clause k when the values defined by v are assigned to the variables. If $\tau = 0$, clause k is not satisfied and $\langle v, u\rangle = 0$, implying that the line spanned by v is parallel to the disk representing k and therefore does not intersect it (since the line spanned by v passes through the origin).

Let now $\tau \geq 1$. The line determined by v intersects the disk at u in the point $6v/\tau$. Since

$$\left\| \frac{6}{\tau} v - u \right\|^2 = \frac{36}{\tau^2}\left(n - \frac{2}{3}\tau + \frac{1}{3}\tau^2 + \frac{1}{9}\tau^3\right)$$

this intersection point is within the disk for $\tau \in \{1, 2, 3\}$. Therefore, we have a line transversal if and only if the assignment defined by v satisfies all clauses in $\mathcal{K}$.

Links to Transversal Theory

When we consider the question whether at least a cylinder of radius 0 (i.e. a traversing line) exists, we get a geometric transversal problem. Such geometric line transversal problems may yield solution spaces of complicated structure, consisting of several connected components (see [137] for a survey).

In some settings of line transversal problems, stronger statements about the structure of the solution space are possible. Recent results address line transversals for disjoint balls [32]. In case the balls are disjoint, the space of feasible directions is convex for fixed traversion orders of the balls, yielding bounds on the number of connected components of the solution space.

In our application, we deal with flat objects. It is shown in [96], that provided the case that the affine hull $\operatorname{aff}(E_i)$ of any ellipsoid does not intersect any other ellipsoids, the number of geometric permutations (that is, the number of traversion orders of the sets) is at most one. In case the ellipsoids are parallel, one can even show that the feasible line transversals can be parameterized to form a convex set in a higher dimensional space [10].

However, there are also negative results. For instance, one can easily verify that three line segments in 3D already may have a nonconvex cone of feasible directions (see Figures 5.12, 5.13).

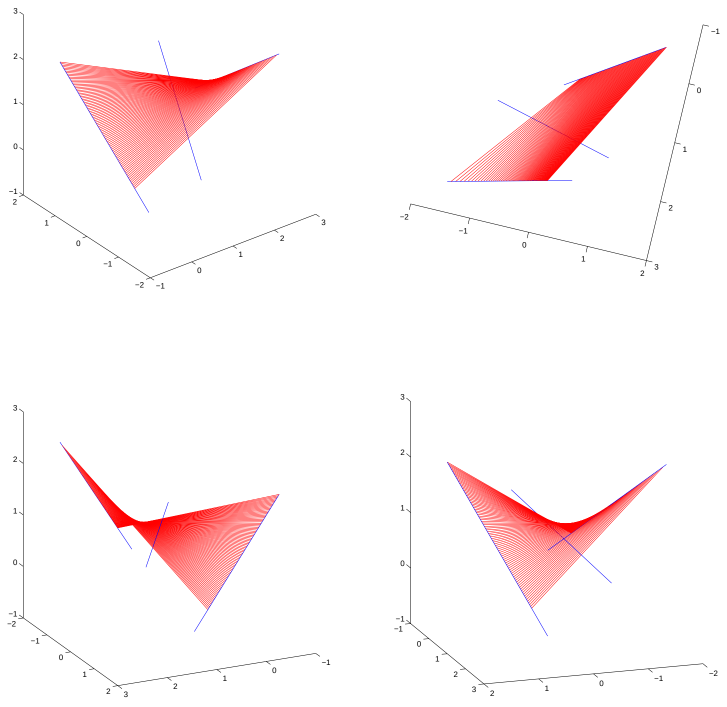


Figure 5.12: Four different views of the set of line transversals for three line segments in 3D.

It is not clear whether the cone is still nonconvex when we bound the eccentricity of the ellipses or even consider circular disks only. In [32], some fundamental properties of the problem for balls are used which do not apply here, so the method used there does not extend to circular disks.

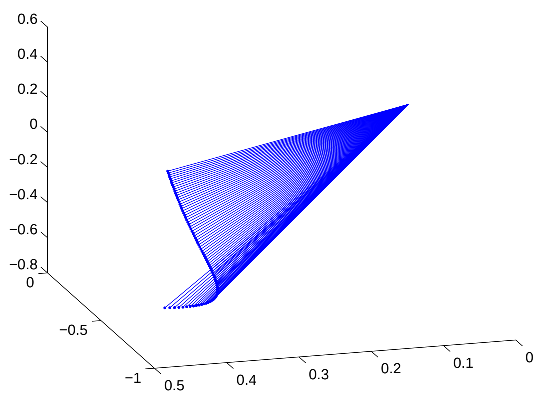


Figure 5.13: The (nonconvex) cone of feasible directions for the line transversals of the line segments in Figure 5.12.

6 Open Questions

Finally, we point out open problems related to the topics of this thesis and recall some unanswered questions.

In Chapter 3, the 1-containment problem under homothety is considered. As stated there, it is known that core-sets whose size is independent of the dimension exist for special cases of the MCP_{Hom}. Moreover, non-symmetric containers exist which do not permit such core-sets. As far as we know, the question is open for general, symmetric containers. It is interesting merely for investigating fundamental structural properties of convex bodies, but also for designing approximation algorithms for the problem.

The k-containment problem under homothety is considered in Chapter 4. We address a number of special cases of 2-containment problems under homothety. Firstly, as far as we know, it is an open problem whether the $\text{MCP}^2_{\text{Hom}}$ for cross-polytopes is $\mathbb{NP}$-hard. Defining an instance with an appropriate discrete structure of the set of solutions (as it is done in the Euclidean case) seems to be more difficult here (see also [68]). The complexity of the $\text{MCP}^2_{\text{Hom}}$ for a cube and a ball remains open, too, and the same holds for some examples of containment problems where the containers share a common translation vector, for instance 2-containment for arbitrarily aligned parallelotopes and 3-containment for cubes.

Concerning the rotational containment problems considered in Chapter 5, the most famous open question is probably Megiddo's cube cover problem [110].

The question whether dimension-independent core-sets for the (non-anchored) smallest enclosing cylinder exist is also of fundamental interest. Such core-sets may provide simple and fast algorithms for the problem. Moreover, the question is closely related to finding line transversals of sets of balls.

We considered core-sets for circular anchored cylinders, and it would be interesting to see whether the statements can be extended to smallest enclosing anchored elliptical cylinders, that is, where $C = a + E$ and $E \subset a^{\perp}$ is an $(n-1)$-dimensional ellipsoid, and the objective is to minimize $\text{vol}_{n-1}(E)$ (compare [100]). Core-sets for smallest enclosing ellipsoids exist [104].

For the transversal questions arising in the extremity surgery problem, it would be interesting to gain further insight into the structure of the transversal space of (ordered) subdimensional objects, such as disks.

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] P.K. Agarwal, N. Amenta, and M. Sharir. Largest placement of one convex polygon inside another. *Discrete and Computational Geometry*, 19(1), 1998.

[2] P.K. Agarwal, B. Aronov, and M.Sharir. Line transversals of balls and smallest enclosing cylinders in three dimensions. *Discrete and Computational Geometry*, 21:473–388, 1999.

[3] P.K. Agarwal, S. Har-Peled, and K.R. Varadarajan. Approximating extent measures of points. *Journal of the ACM*, 51(4):606–635, 2004.

[4] P.K. Agarwal, S. Har-Peled, and K.R. Varadarajan. Geometric approximation via coresets. In J.E. Goodman, J. Pach, and E. Welzl, editors, *Combinatorial and Computational Geometry*, volume 52 of *MSRI publications*, pages 1 –30. Cambridge University Press, 2005.

[5] P.K. Agarwal and C.M. Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33(2):201–226, 2002.

[6] P.K. Agarwal, C.M. Procopiuc, and K.R. Varadarajan. A $(1+\epsilon)$-approximation algorithm for 2-line-center. *Computational Geometry: Theory and Applications*, 26(2):119–128, 2003.

[7] P.K. Agarwal, C.M. Procopiuc, and K.R. Varadarajan. Approximation algorithms for a k-line center. *Algorithmica*, 42:221–230, 2005.

[8] P.K. Agarwal and M. Sharir. Planar geometric location problems. *Algorithmica*, 11(2):185–195, 1994.

[9] P.K. Agarwal and M. Sharir. Efficient algorithms for geometric optimization. *ACM Computing Surveys*, 30(4):412–458, 1998.

[10] N. Amenta. K-transversals of parallel convex sets. In *Proceedings of the 8th Canadian Conference on Computational Geometry*, pages 80–86, 1996.

[11] M.R. Anderberg. *Cluster analysis for applications.* Probability and mathematical statistics. Academic Press, London, 1973.

[12] E.M. Arkin, G. Barequet, and J.S.B. Mitchell. Algorithms for two-box covering. In *Proceedings of the 22nd Annual Symposium on Computational geometry*, pages 459–467. ACM, 2006.

[13] A. Arnold. Approximationsalgorithmen zur Lösung von allgemeinen k-Containment Problemen unter Homothetie. Diplomarbeit, Zentrum Mathematik, Technische Universität München, in preparation.

[14] D. Avis. Diameter partitioning. *Discrete and Computational Geometry*, 1:265–276, 1986.

[15] F. Avnaim and J.-D. Boissonnat. Simultaneous containment of several polygons. In *Proceedings of the 3rd Annual Symposium on Computational Geometry*, pages 242–247, 1987.

[16] F. Avnaim and J.-D. Boissonnat. Polygon placement under translation and rotation. In *Proceedings of the 5th Annual Symposium on Theoretical Aspects of Computer Science*, pages 322–333, 1988.

[17] M. Bădoiu and K.L. Clarkson. Smaller coresets for balls. In *Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms*, pages 801–802, 2003.

[18] M. Bădoiu and K.L. Clarkson. Optimal core-sets for balls. *Computational Geometry: Theory and Applications*, 40(1):14–22, 2008. Preliminairy versions: 2002, 2003.

[19] M. Bădoiu, S. Har-Peled, and P. Indyk. Approximate clustering via coresets. In *Proceedings of the 34th Annual ACM Symposium on the Theory of Computing*, pages 250 – 257. ACM Press, 2002.

[20] G. Barequet and G. Elber. Optimal bounding cones of vectors in three dimensions. *Information Processing Letters*, 93:83–89, 2005.

[21] G. Barequet and S. Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *Journal of Algorithms*, 38(1):91 – 109, 2001.

[22] R. Baumgart, P. Thaller, S. Hinterwimmer, M. Krammer, T. Hierl, and W. Mutschler. A fully implantable, programmable distraction nail (fitbone) – new perspectives for corrective and reconstructive limb surgery. In K.S. Leung, G. Taglang, and R. Schnettler, editors, *Practice of Intramedullary Locked Nails. New developments in Techniques and Applications*, pages 189–198. Springer Verlag Heidelberg, New York, 2006.

[23] B. Ben-Moshe, M.J. Katz, and M. Segal. Obnoxious facility location: Complete service with minimal harm. *International Journal of Computational Geometry and Applications*, 10(6):581–592, 2000.

[24] S. Bereg, J.M. Díaz-Báñez, C. Seara, and I. Ventura. On finding widest empty curved corridors. *Computational Geometry: Theory and Applications*, 38(3):154–169, 2007.

[25] S. Bespamyatnikh and D. Kirkpatrick. Rectilinear 2-center problems. In *Proceedings of the 11th Canadian Conference on Computational Geometry*, pages 68–71, 1999.

[26] S. Bespamyatnikh and M. Segal. Covering a set of points by two axis-parallel boxes. *Information Processing Letters*, 75(3):95–100, 2000.

[27] M.J. Best and N. Chakravarti. Stability of linearly constrained convex quadratic programs. *Journal of Optimization Theory and Applications*, 64(1):43 – 53, 1990.

[28] M.J. Best and B. Ding. On the continuity of the minimum in parametric quadratic programs (technical note). *Journal of Optimization Theory and Applications*, 86(1):245 – 250, 1995.

[29] H.F. Bohnenblust. Convex regions and projections in Minkowski spaces. *Annals of Mathematics*, 39:301–308, 1938.

[30] V. Boltyanski, H. Martini, and P.S. Soltan. *Excursions into Combinatorial Geometry.* Springer, 1997.

[31] T. Bonnesen and W. Fenchel. *Theorie der konvexen Körper.* Springer, Berlin, 1974.

[32] C. Borcea, X. Goaoc, and S. Petitjean. Line transversals to disjoint balls. *Discrete and Computational Geometry*, 39(1-3):158–173, 2008.

[33] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*, volume 15 of *SIAM studies in applied mathematics.* SIAM, Philadelphia, 1994.

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

[35] R. Brandenberg. An affine invariant geometric inequality on minimal enclosing ellipsoids and an application in computing minimal enclosing elliptical cylinders. In preparation.

[36] R. Brandenberg, T. Gerken, P. Gritzmann, and L. Roth. Modeling and optimization of correction measures for human extremities. In W. Jäger and H.-J. Krebs, editors, *Mathematics – Key Technology for the Future. Joint Projects between Universities and Industry 2004-2007*, pages 131–148. Springer, 2008.

[37] R. Brandenberg and L. Roth. Minimal containment under homothetics: A simple cutting plane approach. *Computational Optimization and Applications*, to appear, 2009.

[38] R. Brandenberg and L. Roth. New algorithms for k-center and extensions. *Journal of Combinatorial Optimization*, 18(4):376–392, 2009. Preliminairy version in Combinatorial Optimization and Applications, LNCS 5165, pages 64-78, 2008.

[39] R. Bunschoten. A fully vectorized function that computes the Euclidean distance matrix between two sets of vectors. Via `http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=71`, 1999.

[40] T.M. Chan. More planar two-center algorithms. *Computational Geometry: Theory and Applications*, 13(3):189–198, 1999.

[41] T.M. Chan. Approximating the diameter, width, smallest enclosing cylinder, and minimum-width annulus. In *Proceedigs of the 16th Annual Symposium on Computational Geometry*, pages 300 – 309, 2000.

[42] B. Chazelle. The polygon containment problem. In F. Preparata, editor, *Computational Geometry*, volume 1 of *Advances in Computing Research*, pages 1–33. JAI Press, 1983.

[43] K. Chen. On k-median clustering in high dimensions. In *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1177–1185, 2006.

[44] L. Danzer. Über ein Problem aus der kombinatorischen Geometrie. *Archiv der Mathematik*, 8:347–351, 1957.

[45] J.M. Díaz-Báñez, M.A. López, and T. Sellarès. Computing largest empty slabs. In *ICCSA (3)*, volume 3045 of *LNCS*, pages 99–108, 2004.

[46] J.M. Díaz-Báñez, M.A. López, and T. Sellarès. Locating an obnoxious plane. *European Journal of Operational Research*, 173:556–564, 2006.

[47] Z. Drezner. The p-centre problem — heuristic and optimal algorithm. *Journal of the Operational Research Society*, 35(8):741–748, 1984.

[48] Z. Drezner, editor. *Facility Location*. Springer, 2004.

[49] B.C. Eaves and R.M. Freund. Optimal scaling of balls and polyhedra. *Mathematical Programming*, 23:138 – 147, 1981.

[50] D. Eppstein. Faster construction of planar two-centers. In *Proceedings of the 8th ACM-SIAM Symposium on Discrete Algorithms*, pages 131–138, 1997.

[51] T. Feder and D.H. Greene. Optimal algorithms for approximate clustering. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 434–444, 1988.

[52] K. Fischer, B. Gärtner, and M. Kutz. Fast smallest-enclosing-ball computation in high dimensions. In *Algorithms - ESA 2003*, volume 2832 of *LNCS*, pages 630–641, 2003.

[53] F. Follert. Maxmin location of an anchored ray in 3-space and related problems. In *Proceedings of the 7th Canadian Conference on Computational Geometry*, pages 7–12, 1995.

[54] K. Frankl. Praktische Methoden zur Lösung minimaler Multi-Containment Probleme unter Homothetie. Projektarbeit, Zentrum Mathematik, Technische Universität München, 2007.

[55] J. Gabarro-Arpa and R. Revilla. Clustering of a molecular dynamics trajectory with a Hamming distance. *Computers & Chemistry*, 24:693–698, 2000.

[56] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[57] B. Gärtner. A subexponential algorithm for abstract optimization problems. *SIAM J. Comput.*, 24:1018–1035, 1995.

[58] B. Gärtner and S. Schönherr. An efficient, exact, and generic quadratic programming solver for geometric optimization. In *Proceedings of the 16th Annual Symposium on Computational Geometry*, pages 110–118, 2000.

[59] L. Gąsieniec, J. Jansson, and A. Lingas. Approximation algorithms for Hamming clustering problems. *Journal of Discrete Algorithms*, 2(2):289–301, 2004.

[60] T. Gerken. On the double-ray center problem in 3-space with an application to surgical operation planning. Diplomarbeit, Zentrum Mathematik, Technische Universität München, 2003.

[61] B. Gölles. Ein Problem der Transversalentheorie und seine Anwendung in der medizinischen Operationsplanung. Projektarbeit, Zentrum Mathematik, Technische Universität München, 2007.

[62] T.F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[63] M. Grant, S. Boyd, and Y. Ye. Disciplined convex programming. In L. Liberti and N. Maculan, editors, *Global optimization: From theory to implementation*, Nonconvex Optimization and its Applications. Kluwer, 2005.

[64] M. Grant, S. Boyd, and Y. Ye. `cvx` Users' guide, version 1.0. `http://www.stanford.edu/~boyd/cvx/cvx_usrguide.pdf`, 2006.

[65] P. Gritzmann and V. Klee. Inner and outer j-radii of convex bodies in finite-dimensional normed spaces. *Discrete and Computational Geometry*, 7:255–280, 1992.

[66] P. Gritzmann and V. Klee. Computational complexity of inner and outer j-radii of polytopes in finite-dimensional normed spaces. *Mathmatical Programming*, 59:163–213, 1993.

[67] P. Gritzmann and V. Klee. On the complexity of some basic problems in computational convexity I: Containment problems. *Discrete Mathematics*, 136:129–174, 1994.

[68] P. Gritzmann and T. Theobald. On algorithmic stabbing problems for polytopes. Manuscript.

[69] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer-Verlag, Berlin, Heidelberg, 2nd edition, 1993.

[70] J. Guddat. Stability in convex quadratic parametric programming. *Mathematische Operationsforschung und Statistik*, 7(2):223 – 245, 1976.

[71] H. Hadwiger. Ueber Eibereiche mit gemeinsamer Treffgeraden. *Portugaliae mathematica*, 16(1):23–29, 1957.

[72] D. Halperin, M. Sharir, and K. Goldberg. The 2-center problem with obstacles. *Journal of Algorithms*, 42(1):109–134, 2002.

[73] S. Har-Peled. Clustering motion. *Discrete and Computational Geometry*, 31(4):545–565, 2004.

[74] S. Har-Peled. No coreset, no cry. In *FSTTCS 2004: Foundations of Software Technology and Theoretical Computer Science*, volume 3328 of *LNCS*, pages 324–335, 2004.

[75] S. Har-Peled and A. Kushal. Smaller coresets for k-median and k-means clustering. *Discrete and Computational Geometry*, 37(1):3–19, 2007.

[76] S. Har-Peled and K.R. Varadarajan. Projective clustering in high dimensions using core-sets. In *Proceedings 18th Annual Symposium on Computational Geometry*, pages 312–318, 2002.

[77] S. Har-Peled and K.R. Varadarajan. High-dimensional shape fitting in linear time. *Discrete and Computational Geometry*, 32:269–288, 2004.

[78] J.A. Hartigan. *Clustering algorithms.* Wiley series in probability and mathematical statistics. John Wiley and Sons, New York, 1975.

[79] E. Helly. Über Mengen konvexer Körper mit gemeinschaftlichen Punkten. *Jahresbericht der Deutschen Mathenatiker-Vereinigung*, 32:175–176, 1923.

[80] C. Helmberg. Semidefinite programming for combinatorial optimization. Habilitationsschrift, TU Berlin, 2000.

[81] J. Hershberger. A faster algorithm for the two-center decision problem. *Information Processing Letters*, 47(1):23–29, 1993.

[82] M. Hoffmann. Covering polygons with few rectangles (extended abstract). In *17th European Workshop on Computational Geometry (EuroCG '01)*, pages 39–42, 2001.

[83] M. Hoffmann. A simple linear algorithm for computing rectilinear 3-centers. *Computational Geometry: Theory and Applications*, 31(3):150–165, 2005.

[84] A.F. Holmsen. Recent progress on line transversals to families of translated ovals. In J.E. Goodman, J. Pach, and R. Pollack, editors, *Surveys on discrete and computational geometry: twenty years later*, volume 453 of *Contemporary Mathematics*, pages 283–297. AMS, 2008.

[85] M. Hudelson, V. Klee, and D. Larman. Largest j-simplices in d-cubes: Some relatives of the Hadamard maximum determinant problem. *Linear Algebra and its Applications*, 241/243(1–3):519–598, 1996.

[86] G. Ilizarov. Clinical application of the tension-stress effect for limb lengthening. *Clinical Orthopaedics and Related Research*, 250:8 – 26, 1990.

[87] G. Ilizarov. *Transosseous osteosynthesis: Theoretical and clinical aspects of the regeneration and growth of tissue.* Springer Verlag, Berlin, Heidelberg, New York, 1992.

[88] A.K. Jain and R.C. Dubes. *Algorithms for clustering data.* Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[89] V.A. Jakubovič. Minimization of quadratic functionals under quadratic constraints and the necessity of a frequency condition in the quadratic criterion for absolute stability of nonlinear control systems. *Soviet Mathematics Doklady*, 14(2):593–597, 1973.

[90] V.A. Jakubovič. S-procedure in nonlinear control theory. *Vestnik Leningrad University*, 4(1):73–93, 1977.

[91] J.W. Jaromczyk and M. Kowaluk. An efficient algorithm for the Euclidean two-center problem. In *Proceedings of the 10th Annual Symposium on Computational Geometry*, pages 303–311. ACM Press, 1994.

[92] J.W. Jaromczyk and M. Kowaluk. The two-line center problem from a polar view: a new algorithm and data structure. In *Algorithms and Data Structures*, volume 955 of *LNCS*, pages 13–25, 1995.

[93] F. John. Extremum problems with inequalities as subsidiary conditions. In *Courant Anniversary Volume*, pages 187–204. Interscience, 1948.

[94] U.T. Jönsson. A lecture on the S-procedure. KTH Lecture Notes, Royal Institute of Technology, Stockholm, Sweden, `http://www.math.kth.se/~ulfj/5B5746/Lecture.ps`, 2006.

[95] H.W.E. Jung. Über die kleinste Kugel, die eine räumliche Figur einschließt. *Journal für die reine und angewandte Mathematik*, 123:241–257, 1901.

[96] M. Katchalski. Thin sets and common transversals. *Journal of Geometry*, 14(2):103 – 107, 1980.

[97] M.J. Katz, K. Kedem, and M. Segal. Improved algorithms for placing undesirable facilities. *Computers & Operations Research*, 29(13):1859–1872, 2002.

[98] V. Klee. Circumspheres and inner products. *Mathematica Scandinavica*, 8:363–370, 1960.

[99] E. de Klerk. *Aspects of semidefinite programming.* Kluwer Academic Publishers, 2002.

[100] S. König. Containment mit Ellipsoiden, elliptischen Zylindern und Kegeln. Projektarbeit, Zentrum Mathematik, Technische Universität München, 2008.

[101] P. Kumar. *Clustering and reconstructing large data sets.* PhD thesis, Department of Computer Science, Stony Brook University, 2004.

[102] P. Kumar, J.S.B. Mitchell, and E.A. Yıldırım. Approximate minimum enclosing balls in high dimensions using core-sets. *Journal of Experimental Algorithmics*, 8, 2003.

[103] P. Kumar, J.S.B. Mitchell, and E.A. Yıldırım. Minimum enclosing balls: Matlab code. `http://www.compgeom.com/~piyush/`, 2003.

[104] P. Kumar and E.A. Yıldırım. Minimum volume enclosing ellipsoids and core sets. *Journal of Optimization Theory and Applications*, 126(1):1–21, 2005.

[105] C.L. Lawson. The smallest covering cone or sphere (C. Groenewod and L. Eusanio). *SIREV*, 7(3):415 – 416, 1965.

[106] M. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications. Special Issue on Linear Algebra in Control, Signals and Image Processing*, 284:193–228, 1998.

[107] L. Lovász. *Recent Advances in Algorithms and Combinatorics*, chapter Semidefinite programs and combinatorial optimization, pages 137–194. CMS Books in Mathematics. Springer, 2003.

[108] O.L. Mangasarian, R. Setiono, and W.H. Wolberg. Pattern recognition via linear programming: theory and application to medical diagnosis. In T.F. Coleman and Y. Li, editors, *Large-Scale Numerical Optimization*, pages 22–31. SIAM, 1990.

[109] J. Matoušek and B. Gärtner. *Understanding and Using Linear Programming.* Springer, 2007.

[110] N. Megiddo. On the complexity of some geometric problems in unbounded dimension. *Journal of Symbolic Computation*, 10(3/4):327–334, 1990.

[111] V.J. Milenkovic. Rotational polygon containment and minimum enclosure. In *Proceedings of the 14th Annual Symposium on Computational Geometry*, pages 1–8, 1998.

[112] A. Nemirovski. Advances in convex optimization: Conic programming. In M. Sanz-Sol, J. Soria, J.L. Varona, and J. Verdera, editors, *Proceedings of International Congress of Mathematicians*, volume 1, pages 413–444, Madrid, 2007. EMS - European Mathematical Society Publishing House.

[113] F. Nielsen and R. Nock. Approximating smallest enclosing balls. In F. Nielsen and R. Nock, editors, *Computational Science and Its Applications - ICCSA 2004*, volume 3045 of *LNCS*, 2004.

[114] A. Packer. NP-hardness of largest contained and smallest containing simplices for V- and H-polytopes. *Discrete and Computational Geometry*, 28(3):349–377, 2002.

[115] F. Plastria. Solving general continuous single facility location problems by cutting planes. *European Journal of Operational Research*, 29:98–110, 1987.

[116] I. Pólik. Addendum to the sedumi user guide version 1.1. Technical report, Advanced Optimization Laboratory, McMaster University, 2005.

[117] C.M. Procopiuc. Clustering problems and their applications: A survey. Department of Computer Science, Duke University, 1997.

[118] C.V. Robinson. Spherical theorems of Helly type and congruence indices of spherical caps. *American Journal of Mathematics*, 64(1):260–272, 1942.

[119] L. Roth. Exakte und ϵ-approximative Algorithmen zur Umkugelberechnung. Diplomarbeit, Zentrum Mathematik, Technische Universität München, 2005.

[120] A. Schrijver. *Theory of linear and integer programming.* John Wiley & Sons, Inc., New York, NY, USA, 1986.

[121] M. Sharir. Algorithmic motion planning in robotics. *Computer*, 22(3):9–20, 1989.

[122] M. Sharir. A near-linear algorithm for the planar 2-center problem. *Discrete and Computational Geometry*, 18:125–134, 1997.

[123] M. Sharir and E. Welzl. Rectilinear and polygonal p-piercing and p-center problems. In *Proceedings of the 12th Annual ACM Symposium on Computational Geometry*, pages 122–132, 1996.

[124] C.-S. Shin, J.-H. Kim, S.K. Kim, and K.-Y. Chwa. Two-center problems for a convex polygon (extended abstract). In *Proceedings of the 6th Annual European Symposium on Algorithms*, volume 1461 of *LNCS*, pages 199–210, 1998.

[125] R. Shioda and L. Tunçel. Clustering via minimum volume ellipsoids. *Computational Optimization and Applications*, 37(3):247–295, 2007.

[126] J.F. Sturm. Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11-12:625–653, 1999.

[127] P. Sun and R. Freund. Computation of minimum volume covering ellipsoids. *Operations Research*, 52(5):690 – 706, 2004.

[128] A. Suzuki and Z. Drezner. The p-center location problem in an area. *Location Science*, 4(1/2):69–82, 1996.

[129] J.J. Sylvester. A question in the geometry of situation. *The Quarterly Journal of Mathematics*, 1:79, 1857.

[130] R.C. Thompson. Singular values, diagonal elements, and convexity. *SIAM Journal on Applied Mathematics*, 32(1):39–63, 1977.

[131] A. del Val. On 2-SAT and renamable Horn. In *Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence*, pages 279–284, 2000.

[132] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Reviews*, 38(1):49–95, 1996.

[133] K.R. Varadarajan, S. Venkatesh, Y. Ye, and J. Zhang. Approximating the radii of point sets. *SIAM Journal on Computing*, 36(6):1764–1776, 2007. Preliminary versions 2002, 2003.

[134] V.V. Vazirani. *Approximation Algorithms.* Springer-Verlag, corrected second edition, 2003.

[135] H. Wei, A.T. Murray, and N. Xiao. Solving the continuous space p-centre problem: planning application issues. *IMA Journal of Management Mathematics*, 17:413–425, 2006.

[136] E. Welzl. Smallest enclosing disks (balls and ellipsoids). In H. Maurer, editor, *New Results and New Trends in Computer Science*, number 555 in LNCS, pages 359–370. Springer-Verlag, 1991.

[137] R. Wenger. Progress in geometric transversal theory. In B. Chazelle, J.E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, pages 375–393. AMS, Providence, 1998.

[138] G. Xu and J. Xu. An efficient k-center clustering algorithm for geometric objects. 14th Annual Fall Workshop on Computational Geometry, MIT, Cambridge, MA, 2004.

[139] A.C. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11:721–736, 1982.

[140] E.A. Yıldırım. Two algorithms for the minimum enclosing ball problem. *SIAM Journal on Optimization*, 19(3):1368–1391, 2008.

[141] H. Yu, P.K. Agarwal, R. Poreddy, and K.R. Varadarajan. Practical methods for shape fitting and kinetic data structures. In *Proceedings of the 20th Annual Symposium on Computational Geometry*, pages 263 –272, 2004.

[142] G.L. Zhou, K.C. Toh, and J. Sun. Efficient algorithms for the smallest enclosing ball problem. *Computational Optimization and Applications*, 30(2):147–160, 2005.

Index