Comput Optim Appl (2011) 48: 325–340
DOI 10.1007/s10589-009-9248-3

Minimal containment under homothetics: a simple cutting plane approach

René Brandenberg · Lucia Roth

Received: 24 May 2007 / Published online: 5 May 2009

Abstract Minimal containment problems arise in a variety of applications, such as shape fitting and packing problems, data clustering, pattern recognition, or medical surgery. Typical examples are the smallest enclosing ball, cylinder, slab, box, or ellipsoid of a given set of points.

Here we focus on one of the most basic problems: minimal containment under homothetics, i.e., covering a point set by a minimally scaled translation of a given container. Besides direct applications this problem is often the base in solving much harder containment problems and therefore fast solution methods are needed, especially in moderate dimensions. While in theory the ellipsoid method suffices to show polynomiality in many cases, extensive studies of implementations exist only for Euclidean containers. Indeed, many applications require more complicated containers.

In Plastria (Eur. J. Oper. Res. 29:98–110, 1987) the problem is discussed in a more general setting from the facility location viewpoint and a cutting plane method is suggested. In contrast to Plastria (Eur. J. Oper. Res. 29:98–110, 1987), our approach relies on more and more accurate approximations of the container. For facet and vertex presented polytopal containers the problem can be formulated as an LP, and for many general containers as an SOCP. The experimental section of the paper compares those formulations to the cutting plane method, showing that it outperforms the LP formulations for vertex presented containers and the SOCP formulation for some problem instances.

L. Roth has been supported by the “Deutsche Forschungsgemeinschaft” through the graduate program “Angewandte Algorithmische Mathematik”, Technische Universität München.

R. Brandenberg · L. Roth (✉)
Zentrum Mathematik, Technische Universität München, Boltzmannstr. 3, 85747 Garching bei München, Germany
e-mail: roth@ma.tum.de

R. Brandenberg
e-mail: brandenb@ma.tum.de

Keywords Containment · Core sets · Cutting planes · Outer radius · 1-center · Convex optimization

1 Introduction

The notion of optimal containment encompasses a large number of geometric optimization problems, where the goal is to find an extremal representative C_{opt} of a given class of convex bodies, such that C_{opt} contains (or is contained in) another given body P. Optimal containment problems arise in different applications, such as shape fitting and packing problems, facility location, data clustering, pattern recognition, robotics, or surgery planning. Typical examples are computing the smallest enclosing ball, cylinder, slab, box, or ellipsoid of a given body; see [13] for a survey.

The problem of interest is the minimal containment problem under homothetics:

MINIMAL CONTAINMENT PROBLEM (**MCP**$^{\mathcal{V}}_{Hom}$)
Input: $n \in \mathbb{N}$, $P \in \mathcal{V}^n$, and $C \in \mathcal{C}^n$,
Task: $\min\limits_{c \in \mathbb{R}^n, \rho \geq 0} \rho$, such that $P \subset c + \rho C$,

where $\mathcal{V}^n$ is the family of vertex presented polytopes in $\mathbb{R}^n$ (abbreviated as $\mathcal{V}$-polytopes), that is $P = \operatorname{conv}\{v_1, \ldots, v_m\}$, or equivalently, the family of finite point sets and $\mathcal{C}^n$ a family of n-dimensional closed, convex sets with $0 \in \operatorname{int}(C)$ for all $C \in \mathcal{C}^n$. If, in addition, C is bounded and 0-symmetric (i.e. $C = -C$), MCP$^{\mathcal{V}}_{Hom}$ denotes the task of computing the outer radius of P due to the norm $\|x\|_C := \min\{\lambda \geq 0 : x \in \lambda C\}$. We will address ρ in the above definition as the radius of the containment problem, too, even when C is not symmetric. Other choices of the input set or the container are possible, yet less important in applications.

To the best of our knowledge, the first to consider minimal containment problems in a general setting are Eaves and Freund [7]. Besides this, mostly the special case where $\mathcal{C}^n$ contains only the Euclidean ball, the well known Euclidean 1-center problem (minimum enclosing ball, smallest enclosing ball), was studied and major progress has been achieved in the last years (see [5, 6, 8, 9, 17, 18, 24, 26], or [21] for an overview).

The complexity of MCP$^{\mathcal{V}}_{Hom}$ depends on the representation of C. In [12] it was shown that MCP$^{\mathcal{V}}_{Hom}$ can be solved by linear programming if C is restricted to vertex or facet presented polytopes, and furthermore that MCP$^{\mathcal{V}}_{Hom}$ can be approximated in polynomial time using the ellipsoid method if C is the unit ball of any l_p-space ($p \in \mathbb{N}$). The proof of the latter can be extended to show polynomial time approximability if C is presented by a strong separation oracle.

Definition 1.1 For any input $x \in \mathbb{R}^n$ a *strong separation oracle* outputs $x \in C$ or a halfspace $\{x : a^T x \leq 1\}$ supporting C in x/ρ, where $\rho > 1$ is chosen such that $x \in \operatorname{bd}(\rho C)$.

MCP$^{\mathcal{V}}_{Hom}$ appears directly in various applications (see [4, 19], and the Euclidean case literature) and indirectly as a subproblem in algorithms for other kinds of containment problems (for instance, k-center and its variants [3, 15], minimal enclosing slab or cylinder, or containment under similarities, i.e., homothetics plus rotations). As the latter are often $\mathbb{NP}$-hard, dimensions of solvable problem instances are quite small, and usually the $\mathrm{MCP}^{\mathcal{V}}_{Hom}$ subroutine has to be called extremely often. Hence the focus of our analysis is not polynomiality in the input dimension, but practicability of algorithms and generality of solvable instances.

While problem instances with facet presented polytopal containers C ($\mathcal{H}$-polytopes) can usually be solved in good running times with standard LP packages, the appropriate approach for general containers is less obvious, even for vertex presented containers. It is well known that the ellipsoid method performs very badly in practice [14]. So it is for sure not the algorithm to choose. A good overview on applied techniques to solve general convex programming problems is given in [10]. Depending on $\mathcal{C}^n$ some of these may be the appropriate choice for $\mathrm{MCP}^{\mathcal{V}}_{Hom}$ (see Sects. 2 and 3).

In Sect. 2.4, we present a simple geometric cutting plane algorithm which requires nothing but a separation oracle for C. It makes use of the special structure of $\mathrm{MCP}^{\mathcal{V}}_{Hom}$ and is easy to implement. Its worst-case running time could be exponential (as it is for the simplex method or the LP-type algorithms mentioned in Sect. 2.2, both widely accepted despite the fact that there exist polynomial solution methods for the underlying problems).

Cutting plane algorithms are a well-known technique used for many types of optimization problems. The one we suggest appears in [19] in the typical formulation where cutting planes are refinements of the epigraph of the objective function φ and hence needing an oracle for a subdifferential of φ. There, mostly minisum problems and other objective functions are considered. We observe that in the geometric setting of containment problems, cutting-planes are adaptive refinements of a polytopal approximation of the container.

Apart from that, cutting planes are a widely accepted super-polynomial approach to hard integer linear programming and sometimes also to solving very large LP problems. For non-smooth, non-linear optimization problems (in our case, for non-smooth and non-polytopal containers C) cutting plane methods are believed to be the ideal tool (see [10]). Our experiments approve the latter, showing the good performance of the cutting plane method. In typical applications, the number of data points exceeds the dimension of the data significantly, and often, the dimension is quite small. For these input sizes, the cutting plane method outperforms up to date SOCP solvers (see Sects. 2.3 and 3) and for $\mathcal{V}$-polytopal containers, it is even faster than a direct LP solution (see Sects. 2.1.2 and 3 for details).

2 Solution methods

$\mathrm{MCP}^{\mathcal{V}}_{Hom}$ is a convex programming problem, comprising (among others) linear and second-order cone programming problems. Thus depending on the class of possible containers $\mathcal{C}^n$, specialized methods should be applied to solve and/or approximate the containment problem.

2.1 $\mathcal{H}$- and $\mathcal{V}$-polytopes

As mentioned in the introduction, $\mathrm{MCP}^{\mathcal{V}}_{Hom}$ can be formulated as an LP if C is restricted to facet or vertex presented polytopes.

2.1.1 $\mathcal{H}$-polytopes

Since we have assumed that $0 \in \mathrm{int}(C)$, the offsets of the half-space defining inequalities can be normalized to 1. So let $P = \mathrm{conv}\{v_1, \ldots, v_m\} \subset \mathbb{R}^n$ and $C = \bigcap_{i=1}^k \{x : a_i^T x \le 1\} \subset \mathbb{R}^n$. Then

$$P \subset c + \rho C \quad \Longleftrightarrow \quad a_i^T(v_j - c) \le \rho \quad \text{for all } 1 \le i \le k,\ 1 \le j \le m.$$

Thus the smallest ρ with $P \subset c + \rho C$ for any $c \in \mathbb{R}^n$ is the solution of the following LP:

$$\begin{aligned} &\min \rho \\ \rho + a_i^T c &\ge \max\nolimits_{1 \le j \le m} a_i^T v_j \quad \text{for all } 1 \le i \le k, \\ \rho &\ge 0. \end{aligned} \tag{2.1}$$

The LP has $n + 1$ variables (ρ and c) and k inequalities (not counting non-negativity constraints). The number of vertices in P is of minor importance as it influences only the right hand side computations of the LP.

Mind that in the case where C is a parallelotope (and especially when $\|\cdot\|_C = \|\cdot\|_\infty$), one can solve $\mathrm{MCP}^{\mathcal{V}}_{Hom}$ simply by computing $\max_{j_1, j_2} a_i^T(v_{j_1} - v_{j_2})$ for all i.

2.1.2 $\mathcal{V}$-polytopes

If $C = \mathrm{conv}\{w_1, \ldots, w_k\}$ then

$$P \subset c + \rho C \quad \Longleftrightarrow \quad (1/\rho)(v_j - c) \in \mathrm{conv}\{w_1, \ldots, w_k\} \quad \text{for all } 1 \le j \le m.$$

By setting $\rho' = 1/\rho$ and $c' = c/\rho$ (assuming $\dim P > 0$) this can again be expressed as an LP:

$$\begin{aligned} &\max \rho' \\ \rho' v_j - c' - \textstyle\sum_{i=1}^k \lambda_{ij} w_i &= 0 \quad \text{for all } 1 \le j \le m, \\ \textstyle\sum_{i=1}^k \lambda_{ij} &= 1 \quad \text{for all } 1 \le j \le m, \\ \lambda_{ij} &\ge 0 \quad \text{for all } 1 \le i \le k,\ 1 \le j \le m. \end{aligned} \tag{2.2}$$

Thus the LP has $km + n + 1$ variables and $m(n + 1)$ constraints (not counting non-negativity constraints), which means especially that 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 depend linearly on m. Hence, the sizes of the LPs in formulation (2.2) are much worse than those in (2.1). Experiments show that even when C is a $\mathcal{V}$-polytope, the cutting plane method based on $\mathcal{H}$-approximations of C is usually much faster than solving the LP (2.2) (compare Sect. 3).

2.2 Euclidean containers

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

In [8] 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 resulting algorithm is combinatorial (LP-type) and computes the exact center and squared radius. It has exponential worst-case running time, yet the implementation is fast and still widely accepted as the best solution method in the Euclidean setting.

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

$$\min_{c\in\mathbb{R}^n,\rho\geq 0} \rho, \quad \text{s.t.} \quad \|v_j - c\|_2 \leq \rho, \quad j = 1,\ldots,m,$$

and tests show that implementations can compete with the LP-type algorithms (see e.g., [17, 26]).

In recent publications, core-sets for geometric optimization problems have been considered (see [1] for a survey). In the case of the Euclidean 1-center problem, a very easy incremental core-set algorithm has been stated in [6] and improved in [5]. It can be used as a speed-up for any minimum enclosing ball algorithm. (A similar analysis yields a basic subgradient method for the 1-center problem, too [5].)

Extensions of both the LP-type and the core-set method to more general containers may be possible (at least as super-polynomial methods), however both rely essentially on a “half-space lemma” [2], [6, Lemma 2.2] which cannot be transferred to non-Euclidean containers as for $n \geq 3$, in every non-Euclidean normed space, there exist point sets P such that the centers c of their minimal enclosing balls do not lie in $\text{conv}(P)$ [16].

Furthermore, if P is a regular simplex with center 0 and $C = -P$, we need a subset of P of size dependent on 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.

2.3 Combined containers

The following methods could theoretically be used to solve the $\text{MCP}^{\mathcal{V}}_{Hom}$ for complicated combinations of these containers. However, the number of variables and constraints would grow quite fast.

2.3.1 Intersections

Suppose C is the intersection of some Euclidean balls and polytopes, $C = \bigcap_{i=1}^{k}(c_i + r_i\mathbb{B}) \cap \bigcap_{i=1}^{l} Q_i$, where the Q_i are given in $\mathcal{H}$- or $\mathcal{V}$-presentation. Then $\text{MCP}^{\mathcal{V}}_{Hom}$ is equivalent to the following SOCP (LP if $k = 0$):

$$\begin{aligned} &\min \rho \\ \|v_j - c - \rho c_i\|_2 &\leq \rho r_i \qquad \text{for all } 1 \leq i \leq k,\ 1 \leq j \leq m, \\ v_j - c &\in \rho Q_i \qquad \text{for all } 1 \leq i \leq l,\ 1 \leq j \leq m, \end{aligned}$$

where the $v_j - c \in \rho Q_i$ conditions can be expressed as linear constraints by using the manipulations as shown in Sect. 2.1.

2.3.2 *Minkowski sums*

Suppose C is the Minkowski sum of some polytopes and maybe one Euclidean ball, $C = \sum_{i=1}^{k+l} Q_i$, where $Q_1, \dots, Q_k$ are again polytopes given in $\mathcal{H}$- or $\mathcal{V}$-presentation, $l \in \{0, 1\}$ and if $l = 1$ then $Q_{k+1} = \mathbb{B}$. Since

$$v_j \in c + \rho C \quad \Longleftrightarrow \quad v_j - c - \sum_{i=1}^{k+l} x_{ij} = 0, \quad x_{ij} \in \rho Q_i$$
$$\text{for all } 1 \le i \le k + l,$$

we can again transform the problem into an SOCP (LP if $l = 0$). In the special case when $C = \sum_{i=1}^{k} [\alpha_i, \beta_i] z_i$ is a zonotope we obtain the following LP with ρ' and c' as in Sect. 2.1.2:

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

If 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:[1]

$$\begin{aligned} &\min \rho \\ \|v_j - c - x_j\|_2 &\le \rho && \text{for all } 1 \le j \le m, \\ a_i^T x_j &\le \rho && \text{for all } 1 \le i \le k,\ 1 \le j \le m. \end{aligned}$$

2.3.3 *Examples*

(a) In [22] the *Minkowski symmetry* for a convex set P is defined as

$$\max\{\lambda : -\lambda P + t \subset P, t \in \mathbb{R}^n\}$$

a special containment problem under homothetics. Especially, when P is a $\mathcal{V}$-polytope computing its Minkowski symmetry means just solving an $\text{MCP}_{Hom}^{\mathcal{V}}$ with $C = -P$, $c = t/\lambda$, and $\rho = 1/\lambda$.

(b) Consider the situation where a robot arm should be able to grab a certain set of objects (modelled by P) in a plane. The arm may rotate freely in any direction, whereas the robot itself can move along a track above the plane, which can again move along an orthogonal track (see Fig. 1).

If we ask for the best robot position and size (assuming that the ratio between the arm size and the track sizes is fixed), we have to solve an $\text{MCP}_{Hom}^{\mathcal{V}}$ with a container C being the Minkowski sum of a circle and a square.

[1]The sizes of all the SOCPs are slightly reducible by using the fact that $v_j - c \in \rho(Q_1 + Q_2) \Leftrightarrow (v_j - c - \rho Q_1) \cap \rho Q_2 \neq \emptyset$.

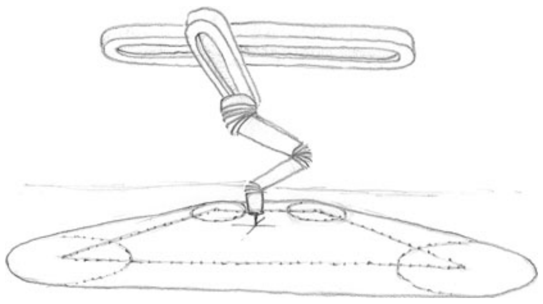

Fig. 1 A robot arm, hinged to two axis parallel tracks

(c) Let a certain source emit radiation which should reach a set of given targets P. Usually, we would be in the situation of a Euclidean container. However, the ray source is often bounded on its sides, limiting the directions of the emitted rays such that only a conical section (with the source as an apex) of the Euclidean ball remains. Let Q be the convex "window" permitting emissions and assume $0 \notin Q$. In standard cases, Q is a Euclidean ball or a polytope. Now covering the point set with a homothetic copy of $\mathrm{pos}(Q) \cap \mathbb{B}$ models the problem as $\mathrm{MCP}^{\mathcal{V}}_{Hom}$ where the optimal solution yields the minimal required reach and the optimal source location:

$$\begin{aligned} &\min \rho \\ &v_j - c \in \lambda_j Q \qquad \text{for all } 1 \le j \le m, \\ &\|v_j - c\|_2 \le \rho \qquad \text{for all } 1 \le j \le m, \\ &\lambda_j \ge 0 \qquad \text{for all } 1 \le j \le m. \end{aligned}$$

In an application, one may be interested in the optimal solution allowing rotation, which can be found by solving $\mathrm{MCP}^{\mathcal{V}}_{Hom}$ instances as subproblems.

2.4 A cutting plane algorithm

The most general polynomiality result can be obtained with help of the ellipsoid method. In [12] it was shown that any instance of $\mathrm{MCP}^{\mathcal{V}}_{Hom}$ can be approximated up to any given accuracy within polynomial time, if C is the unit ball of an l_p-space with $p \in \mathbb{N}$, and the variant of the ellipsoid method, stated in that paper only needs a bounding box or ball $B \supset C$ and a separation oracle to work. For the polynomiality proof the oracle must be strong.

Presuming the existence of a separation oracle helps to present the results as general as possible. It is easy to see that for a huge class of convex bodies (and their presentations) efficient strong separation oracles can be provided. For instance, the later is possible whenever a subgradient of the objective $\varphi(x) = \min\{\rho : P \subset x + \rho C\}$ (which is $\max_j \|v_j - x\|_C$ if C is 0-symmetric) can be computed.[2]

[2]One should recognize that the ability to provide a separation oracle for C strongly relies on the representation of C. For instance, if C should be the integer hull of some polytope a separation oracle could be hard to deduce.

Lemma 2.1 *If a is a subgradient of $\varphi(x) = \min\{\rho : P \subset x + \rho C\}$ at c then $-a$ is an outer normal of a hyperplane supporting $c + \rho C \supset P$ at $v_j \in P$.*

Proof Follows directly from the fact that the level sets of $\operatorname{epi}(\varphi)$ are $\bigcap_j v_j - \rho C$ and that a is a subgradient of φ in x if and only if $(a^T, -1)^T$ is an outer normal of a hyperplane supporting $\operatorname{epi}(\varphi)$ in $(x^T, \varphi(x))^T$. □

The cutting plane approach we present for $\mathrm{MCP}^{\mathcal{V}}_{Hom}$ is very intuitive as its approximation of the optimal value is based solely on polytopal approximations of the underlying container 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. The latter holds true for the cutting plane approach described in [19], and in case of the $\mathrm{MCP}^{\mathcal{V}}_{Hom}$ this method is also reducible to one directly operating with the container C.

The cutting plane approach can also be interpreted as a dual core-set method. Instead of finding a subset S of the points in P with almost the same radius, the cutting plane method looks for a small subset of the hyperplanes describing C which is essential for the containment.

Like the ellipsoid algorithm, our cutting plane method only requires an initial bounding box H_I and a separation oracle to work. For technical simplicity we assume that the oracle is strong. This is no real restriction for two reasons: Firstly, we already mentioned at the beginning of this section that for a huge class of convex bodies an efficient oracle can be given in its strong version, and secondly, one can easily obtain an approximatively strong version of every separation oracle by a one dimensional search for the appropriate dilatation factor (by calling the separation oracle successively until the required approximation is obtained).

Let $P = \operatorname{conv}\{v_1, \dots, v_m\}$ and $\varepsilon > 0$ the desired accuracy. At first, H is assigned an $\mathcal{H}$-polytope containing C, e.g., $H = H_I$. 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 $\mathrm{MCP}^{\mathcal{V}}_{Hom}$ with input P and H obtained via linear programming (see Sect. 2.1.1). As $C \subset H$, the value of ρ is a lower bound for the minimal radius ρ^* of the $\mathrm{MCP}^{\mathcal{V}}_{Hom}$ with input P and C (see Fig. 2).

Obviously, even if φ is not given in an explicit form, $\varphi(c)$ for any fixed c can be computed by using the oracle. Hence, we can easily check for every vertex v_j of P if $v_j \in c + \rho C$, and if not, compute $\bar{\rho} = \varphi(c)$, which is an upper bound for ρ^*.

If $\bar{\rho}/\rho \le 1 + \varepsilon$ we are done. Otherwise, we replace H by $H \cap \{x : a_{k+1}^T x \le 1\}$ where $\{x : a_{k+1}^T x \le \bar{\rho}\}$ is a hyperplane supporting $\bar{\rho} C$ at $v_j - c \in P - c$, and the next step of the iteration begins.

Finally, as each H is a subset of the preceding one, the sequence of lower bounds ρ is increasing. This is not true for the upper bounds $\bar{\rho}$. Hence, it makes sense to store the best obtained upper bound and the corresponding center $\bar{c}$.

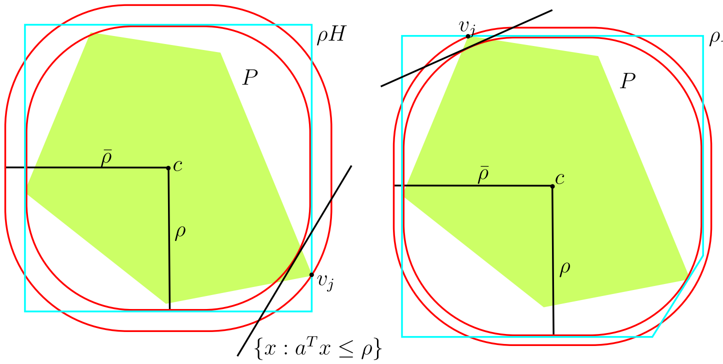

Fig. 2 Two iterations of the cutting plane method: Here C is the Minkowski sum of a square and a disc and P the green polygon. In the first step (*left picture*) H is the square circumscribing C. Obviously, c and ρ are an optimal solution for $\text{MCP}^{\mathcal{V}}_{Hom}$ with P and H as input. Since v_j is the vertex of P with maximal distance to c (with respect to φ) the cut $\{x : a^T x \leq 1\}$ supporting C in $(v_j - c)/\bar{\rho}$ 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

```
CUTTING-PLANE ALGORITHM

INPUT: $P = \text{conv}\{v_1, \ldots, v_m\}$, $C$ via separation oracle,
    $H = \bigcap_{i=1}^{k} \{x : a_i^T x \leq 1\}$ a bounding polytope of $C$, and $\varepsilon > 0$.

Set $\bar{\rho} = \infty$, loop = TRUE.
WHILE(loop)
    solve the LP: $\rho_* := \min\{\rho : \rho + a_i^T c \geq \max_j a_i^T v_j \ \forall i\}$
    IF $v_j - c \in \rho_* C \ \forall j$, set $\bar{\rho} = \rho_*$, $\bar{c} = c$, loop = FALSE.
    ELSE compute $\bar{\rho} = \min\{\bar{\rho}, \varphi(c)\}$,
        set $\bar{c}$ to the corresponding $c$.
        IF $\bar{\rho}/\rho_* \leq 1 + \varepsilon$, set loop = FALSE.
        ELSE get $a_{k+1}$ from the strong separation oracle with input $(v_j - c)$,
            set $H = H \cap \{x : a_{k+1}^T x \leq 1\}$.
END

OUTPUT: $\varepsilon$-approximation $\bar{\rho}$ of $\rho^*$ and center $\bar{c}$.
```

Obviously, the running time of the algorithm is linear in the number of points 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.

Essentially, the number of iterations of the algorithm 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. Hence, we may have an exponential number of iterations. Indeed, 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 upper bound therefore primarily shows the convergence of the algorithm. In theory, our algorithm cannot compete with the ellipsoid method. However, our aim is also the experimental behaviour which is addressed in the next section.

3 Experimental behaviour of the cutting plane method

The theoretical analysis mentioned above does not pay credit to the adaptive principle of the cutting plane algorithm. Consequently, our a-priori upper bound is much too pessimistic. 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 [25], 9216 hyperplanes only suffice for $\varepsilon = 0.45$. The following test results for different data sets P and C show a substantially better general performance of the cutting plane algorithm.

Another issue is the handling of the linear programs. Despite the exponential worst-case running time of the simplex method, using it as an LP solver in our experiments yields better running times than, e.g., (polynomial) interior point methods. This is a typical observation in almost all cutting-plane algorithms, arguably relying on the good average case behavior of the simplex method. The reason is that the dual simplex permits an easy warm start strategy, that is, using the optimal solution of the preceding iteration as a starting point. Therefore, though we add a row to the linear program in each iteration, the running time per iteration is almost constant.

We have already addressed several possibilities to solve 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 $\mathrm{MCP}^{\mathcal{V}}_{Hom}$. In [10] such a framework for unifying convex programming (cvx) is proposed and has been implemented. Some instances of $\mathrm{MCP}^{\mathcal{V}}_{Hom}$ can easily be formulated in cvx, others would require separate "graph implementations" (see [10]). 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 [11, p. 6].

One of our main goals is to find fast solution methods for (maybe thousands of) $\mathrm{MCP}^{\mathcal{V}}_{Hom}$ instances occurring as subproblems when solving harder containment problems, e.g., k-center problems or containment under similarities. Hence, this usually demands fast solutions in small dimensions, whereas high dimensions are out of reach anyway.

A small test on different data sets (see Table 1) showed that cvx is not the appropriate choice for this kind of problems as even the smallest example took more than five seconds.

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 Sect. 2.3), the SOCP formulation yields $O(kmn)$ constraints and variables

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

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 mem-

Table 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

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 2 Running times of the cutting plane method compared to directly solving the linear program in Sect. 2.1.2. The container C is the cross polytope and P are samples of $(0,1)$-normally distributed data points. In both cases, the problem is solved up to an accuracy of $\varepsilon < 10^{-14}$

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 (Sect. 2.1.2)	Time (s)	0.37	15.20	2143.60	2.11	107.42	5594.63

ory" error (indicated by a '*' in Table 1) in cvx.[3] The performance of the cutting plane algorithm deteriorates in higher dimensions, 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 hardly comparable (Matlab versus C++), an extensive comparison of the cutting plane approach to cvx (or similar CP solvers) would be of minor value. Instead, we present some examples to illustrate and analyze the course of the cutting plane algorithm.

In a first experiment (see Table 2), we compared the cutting plane algorithm with the direct approach via linear programming given in Sect. 2.1.2 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 $\|\cdot\|_1$-norm unit ball, the cross polytope.

Considering for instance the 30-dimensional cross polytope, it seems obvious that solving the containment problem via the facet representation is foolish and one has to use the vertex representation as shown in Sect. 2.1.2. However, when $m = 10000$, the vertex representation of the cross polytope yields an LP (2.2) with a constraint matrix of about 300000 rows, 600000 columns, and 1500000 non-zero entries, whereas,

[3] A direct solution via SeDuMi (the Matlab solver used by cvx, see [20, 23]) does not cause the "out of memory" error but needs 1 h 45 min to solve the instance with 10000 points in $\mathbb{R}^3$.

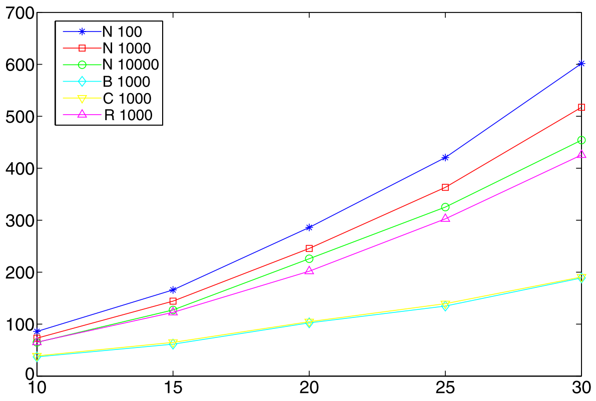

Fig. 3 Averaged number of iterations for different data sets P in dimensions $n = 10, 15, 20, 25,$ and 30. Here, C is the Euclidean unit ball and $\varepsilon = 0.0001$. N—(0, 1) normally distributed data, B—points on the surface of the unit ball, C—vertices of the unit cube chosen at random, R—equally distributed data in $[0, 1]^n$. 100, 1000, and 10000 are the numbers of points in the input polytope P

applying the cutting plane algorithm, one can see that about 200 of the 2^{30} hyperplanes in the hyperplane representation are enough to solve the containment problem for P.[4] The algorithm computes a suitable approximation (depending on P) of the cross polytope in facet representation, and as we can see from formulation (2.1), 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 2). Clearly, we suggest to use the direct approach when the number of points m is very small.

In a second experiment (see Fig. 3) we considered six differently distributed data sets P of different sizes in several dimensions $n \leq 30$ with the Euclidean unit ball as container C and $\varepsilon = 0.0001$.

We sampled 100 instances for each data point in Fig. 3. All instances were solved within 1 second in dimension 10 and those with at most 1000 points within 6 seconds in dimension 30. The computations for the largest input sets (10000 points, dimension 30) took about 15 seconds on average. For sure, the cutting plane method cannot compete with specialized algorithms for the Euclidean 1-center problem that are making use of specific properties of the Euclidean ball. In [8], the implementations by Fischer, Gärtner, and Kutz [8], by Kumar, Mitchell, and Yıldırım [17] as well as by Zhou, Toh, and Sun [26] are compared. Their tests go up to dimension 2000 which is out of reach for our algorithm. Yet, our program seems to be competitive for moderate dimensions which is sufficient for many applications.

The results illustrated in Fig. 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. 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. These observations may be natural, as the shapes where the algorithm performs less iterations are more symmetric or ball-like. We have also observed the decrease of the accuracy (the

[4] An accuracy of 10^{-14}, about the tolerance of the LP solver, is reached after less than 200 iterations.

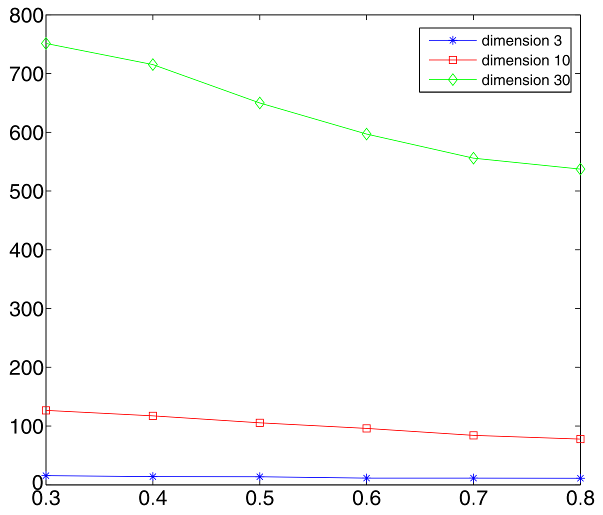

Fig. 4 Number of iterations for different values of q and different dimensions

difference between upper and lower bound) with the number of iterations. The tests suggest that it is exponential with a rate depending on the dimension.

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

In the first of these examples (see Fig. 4), 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 5 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 Figs. 4 and 5 is again 100.

For the initial bounding polytope H_I of C, boxes as well as regular simplices have been considered. Experiments have also included random normal vectors for the initial bounding polytope. A more precise initial approximation can reduce the number of iterations. However, though the box is usually a much finer 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 involved as box-shaped containers 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 will add many unnecessary constraints to the LP, making the algorithm perform slower especially in higher dimensions.

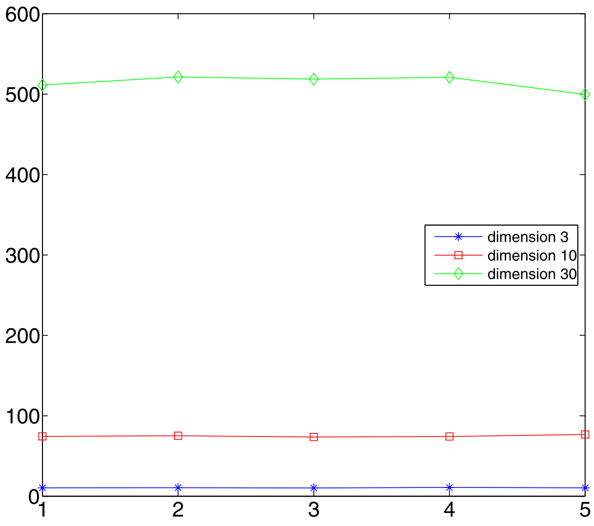

Fig. 5 Number of iterations for different dimensions, where C is the intersection of 1 to 5 balls

Furthermore, experiments for 0-symmetric C have included not only adding 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, it has been tried to minimize the distance (measured in the distance function induced by H) between the new center and its predecessor. This involves solving a second linear program in each iteration. The number of iterations reduces noticeably for normally distributed input data, but the observed running time deteriorates due to the effort spent solving the additional linear programs.[5]

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 it is possible only in the case of linear constraints to consider just one point (where the maximum is attained) in order to ensure that the whole set P satisfies the constraint. More specialized subroutines than the general SOCP solvers might be helpful to improve the performance of the alternate approach. Yet, we are aware

[5] Another idea to improve the performance of cutting-plane algorithms is reported in [19]. The computing times are improved using an interpolation technique, so we have experimented with this approach, too. Though we can confirm that the interpolation step reduces the number of iterations a little, we conclude 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 (the mentioned examples involve other objective functions) or simply that the problem settings are hardly comparable since in [19], only point sets of up to 50 points are considered.

of the existence of such methods only for the Euclidean case [8, 17, 26] (relying fundamentally on properties of the Euclidean ball), a lack motivating this paper. Note that the incremental core-set approach [6, 17] can in principal be used for any kind of problem involving the containment of a point set, but polynomiality is only proven in the Euclidean case.

Experiments have been 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 XpressMP as LP-Solver have been used.

References

1. Agarwal, P., Har-Peled, S., Varadarajan, K.R.: Geometric approximation via coresets. In: Goodman J.E., Pach, J., Welzl, E. (eds.) Combinatorial and Computational Geometry. MSRI Publications, vol. 52. Cambridge University Press, Cambridge (2005)
2. Bonnesen, T., Fenchel, W.: Theorie der konvexen Körper. Springer, Berlin (1974). Translation: Theory of Convex Bodies. BCS Associates, Moscow, Idaho (USA) (1987)
3. Brandenberg, R., Roth, L.: New algorithms for k-center and extensions. J. Comb. Optim. (2009, to appear)
4. Brandenberg, R., Gerken, T., Gritzmann, P., Roth, L.: Modeling and optimization of correction measures for human extremities. In: Jäger, W., Krebs, H.-J. (eds.) Mathematics—Key Technology for the Future. Joint Projects between Universities and Industry 2004–2007, pp. 131–148. Springer, Berlin (2008)
5. Bădoiu, M., Clarkson, K.L.: Smaller coresets for balls. In: Proc. 14th ACM-SIAM Sympos. Discrete Algorithms, pp. 801–802 (2003)
6. Bădoiu, M., Har-Peled, S., Indyk, P.: Approximate clustering via core-sets. In: Annual ACM Symposium on Theory of Computing archive. Proc. 34th Annu. ACM Sympos. Theory of Computing, pp. 250–257. ACM, New York (2002)
7. Eaves, B.C., Freund, R.M.: Optimal scaling of balls and polyhedra. Math. Program. **23**, 138–147 (1982)
8. Fischer, K., Gärtner, B., Kutz, M.: Fast smallest-enclosing-ball computation in high dimensions. In: Proc. 11th Annual European Symposium on Algorithms (ESA), pp. 630–641 (2003)
9. Gärtner, B.: Fast and robust smallest enclosing balls. In: Proc. 7th Annu. European Symposium on Algorithms (ESA). Lecture Notes in Computer Science, vol. 1643, pp. 325–338. Springer, Berlin (1999)
10. Grant, M., Boyd, S., Ye, Y.: Disciplined convex programming. In: Liberti, L., Maculan, N. (eds.) Global Optimization: From Theory to Implementation, Nonconvex Optimization and Its Applications. Kluwer, Dordrecht (2005)
11. Grant, M., Boyd, S., Ye, Y.: cvx Users' guide, version 1.0. http://www.stanford.edu/~boyd/cvx/cvx_usrguide.pdf, June 2006
12. Gritzmann, P., Klee, V.: Computational complexity of inner and outer j-radii of polytopes in finite-dimensional normed spaces. Math. Program. **59**, 163–213 (1993)
13. Gritzmann, P., Klee, V.: On the complexity of some basic problems in computational convexity I: Containment problems. Discrete Math. **136**, 129–174 (1994)
14. Grötschel, M., Lovasz, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization, Algorithms and Combinatorics, vol. 2. Springer, Berlin (1993)
15. Halperin, D., Sharir, M., Goldberg, K.: The 2-center problem with obstacles. J. Algorithms **42**, 109–134 (2002)
16. Klee, V.: Circumspheres and inner products. Math. Scand. **8**, 363–370 (1960)
17. Kumar, P., Mitchell, J.S.B., Yıldırım, E.A.: Approximate minimum enclosing balls in high dimensions using core-sets. J. Exp. Algorithmics **8**, 1.1 (2003)
18. Nielsen, F., Nock, R.: Approximating smallest enclosing balls. In: Nielsen, F., Nock, R. (eds.) Proceedings of International Conference on Computational Science and Its Applications (ICCSA). Lecture Notes in Computer Science, vol. 3045. Springer, Berlin (2004)
19. Plastria, F.: Solving general continous single facility location problems by cutting planes. Eur. J. Oper. Res. **29**, 98–110 (1987)

20. Pólik, I.: Addendum to the SeDuMi user guide version 1.1. Technical report, Advanced Optimization Laboratory, McMaster University (2005)
21. Roth, L.: Exakte und ϵ-approximative Algorithmen zur Umkugelberechnung. Diploma thesis, Zentrum Mathematik, TU München, October 2005
22. Schneider, R.: Convex Bodies: The Brunn-Minkowski Theory. Encyclopedia of Mathematics and Its Applications, vol. 44. Cambridge University Press, Cambridge (1993)
23. Sturm, J.F.: Using SEDUMI 1.02, a MATLAB toolbox for optimization over symmetric cones. Optim. Methods Softw. **11-12**, 625–653 (1999)
24. Welzl, E.: Smallest enclosing disks (balls and ellipsoids). In: Maurer, H. (ed.) New Results and New Trends in Computer Science. Lecture Notes in Computer Science, vol. 555, pp. 359–370. Springer, Berlin (1991)
25. Yao, A.C.: On constructing minimum spanning trees in k-dimensional spaces and related problems. SIAM J. Comput. **11**, 721–736 (1982)
26. Zhou, G.L., Toh, K.C., Sun, J.: Efficient algorithms for the smallest enclosing ball problem. Comput. Optim. Appl. **30**(2), 147–160 (2005)