We need the first algorithm in this paper.

SIAM J. OPTIM.
Vol. 19, No. 3, pp. 1368–1391

TWO ALGORITHMS FOR THE MINIMUM ENCLOSING BALL PROBLEM[*]

E. ALPER YILDIRIM[†]

m n-dimensional points

Abstract. Given $\mathcal{A} := \{a^1, \ldots, a^m\} \subset \mathbb{R}^n$ and $\epsilon > 0$, we propose and analyze two algorithms for the problem of computing a $(1+\epsilon)$-approximation to the radius of the minimum enclosing ball of $\mathcal{A}$. The first algorithm is closely related to the Frank–Wolfe algorithm with a proper initialization applied to the dual formulation of the minimum enclosing ball problem. We establish that this algorithm converges in $O(1/\epsilon)$ iterations with an overall complexity bound of $O(mn/\epsilon)$ arithmetic operations. In addition, the algorithm returns a "core set" of size $O(1/\epsilon)$, which is independent of both m and n. The latter algorithm is obtained by incorporating "away" steps into the former one at each iteration and achieves the same asymptotic complexity bound as the first one. While the asymptotic bound on the size of the core set returned by the second algorithm also remains the same as the first one, the latter algorithm has the potential to compute even smaller core sets in practice, since, in contrast to the former one, it allows "dropping" points from the working core set at each iteration. Our analysis reveals that the leading terms in the asymptotic complexity analysis are reasonably small. In contrast to the first algorithm, we also establish that the second algorithm asymptotically exhibits linear convergence, which provides further insight into our computational results, indicating that the latter algorithm indeed terminates faster with smaller core sets in comparison with the first one. We also discuss how our algorithms can be extended to compute an approximation to the minimum enclosing ball of more general input sets without sacrificing the iteration complexity and the bound on the core set size. In particular, we establish the existence of a core set of size $O(1/\epsilon)$ for a much wider class of input sets. We adopt the real number model of computation in our analysis.

Key words. minimum enclosing balls, core sets, approximation algorithms

AMS subject classifications. 90C25, 90C46, 65K05

DOI. 10.1137/070690419

1. Introduction.

Given a finite set of points $\mathcal{A} := \{a^1, \ldots, a^m\} \subset \mathbb{R}^n$, we are concerned with the problem of computing an approximation to the minimum enclosing ball of $\mathcal{A}$, which we shall denote by $\mathrm{MEB}(\mathcal{A})$.

For $c \in \mathbb{R}^n$ and a nonnegative $\rho \in \mathbb{R}$, let $\mathcal{B}_{c,\rho} \subset \mathbb{R}^n$ denote the ball centered at c with radius ρ, i.e., center point

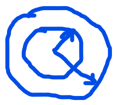
$$\mathcal{B}_{c,\rho} := \{x \in \mathbb{R}^n : \|x - c\| \le \rho\},$$

where $\|\cdot\|$ denotes the Euclidean norm.

Given $\epsilon > 0$, a ball $\mathcal{B}_{c,\rho}$ is said to be a $(1+\epsilon)$-approximation to $\mathrm{MEB}(\mathcal{A})$ if

$$\mathcal{A} \subset \mathcal{B}_{c,\rho}, \quad \underset{\text{computed radius}}{\rho} \le (1+\epsilon)\underset{\text{true radius}}{\rho_{\mathcal{A}}}, \tag{1}$$

maximum error = epsilon



epsilon = 0.1
ro_A = 10 (smaller circle)
=> ro <= 11 (bigger circle)

where $\mathcal{B}_{c_{\mathcal{A}},\rho_{\mathcal{A}}} := \mathrm{MEB}(\mathcal{A})$.

A subset $\mathcal{X} \subseteq \mathcal{A}$ is said to be an ϵ-core set (or a core set) of $\mathcal{A}$ if

$$\rho_{\mathcal{X}} \le \rho_{\mathcal{A}} \le (1+\epsilon)\rho_{\mathcal{X}}, \tag{2}$$

where $\mathcal{B}_{c_{\mathcal{X}},\rho_{\mathcal{X}}} := \mathrm{MEB}(\mathcal{X})$. Small core sets play an important role in designing efficient algorithms for large-scale problems, since they provide a compact representation of

[*]Received by the editors May 3, 2007; accepted for publication (in revised form) May 22, 2008; published electronically November 21, 2008.
http://www.siam.org/journals/siopt/19-3/69041.html

[†]Department of Industrial Engineering, Bilkent University, 06800 Bilkent, Ankara, Turkey (yildirim@bilkent.edu.tr). This research was partially supported by TÜBİTAK (Turkish Scientific and Technological Research Council) Grant 107M411.

Convex Combination = w1 * v1 + w2 * v2 + ... + wn * vn

In a convex combination, the weights act as proportions or probabilities of how much each vector contributes to the final combination. The constraint that the weights sum up to 1 ensures that the combination lies within the convex hull of the set of vectors, meaning it doesn't go outside the bounds defined by the vectors' positions.

Carathéodory's Theorem: Any point within the convex hull of a set of points in a vector space can be written as a convex combination of at most n+1 points (where n is the dimension of the vector space) from that set.

the input set $\mathcal{A}$. If a small ϵ-core set $\mathcal{X}$ is available, then solving the problem on $\mathcal{X}$ already yields a good approximation to MEB($\mathcal{A}$). Since the center $c_{\mathcal{A}}$ of MEB($\mathcal{A}$) lies in the convex hull of $\mathcal{A}$ (cf. section 2), it follows from Carathéodory's theorem that there always exists a 0-core set of size at most $n + 1$.

nothing useful in [24]

Minimum enclosing balls have numerous important applications in clustering, nearest neighbor search, data classification, support vector machines, machine learning, facility location, collision detection, computer graphics, and military operations. We refer the reader to [24] and the references therein. In particular, many of these applications give rise to large-scale instances of the MEB problem, and a reasonably small accuracy suffices for such applications.

(*) Importance of MEB!

The minimum enclosing ball problem has a fairly rich literature dating back to at least the 19th century [37]. One of the earliest known solution methods is given by Sylvester [38], which is attributed to Peirce, and later rediscovered by Chrystal [9]. The reader is referred to [6] for a detailed account of the earlier history of this problem. More recent references include [25, 14, 28, 11, 8, 35, 7, 23, 22, 27, 31, 40, 16, 17, 4, 2, 24, 42, 13, 12, 29, 30, 21, 44, 32].

(*) History of MEB!

main problem with greedy methods

The earliest known algorithm due to Chrystal and Peirce [38, 9] computes the exact minimum enclosing ball of m points in the plane in $O(m^2)$ operations in the worst case. For a fixed dimension n, the minimum enclosing ball of m points can be computed in $O(m)$ operations [27, 40]. However, the dependence on the dimension n is exponential. Bădoiu, Har-Peled, and Indyk [4] established the existence of an ϵ-core set of size $O(1/\epsilon^2)$. Note that the size of the core set is independent of m and n. Based on this result, their algorithm can compute a $(1+\epsilon)$-approximation to MEB($\mathcal{A}$) in $O\left(mn/\epsilon^2 + (1/\epsilon^{10})\log(1/\epsilon)\right)$ operations. Bădoiu and Clarkson [2] and Kumar, Mitchell, and Yıldırım [24] independently discovered the existence of an ϵ-core set of size $O(1/\epsilon)$. As noted in [2], this improved core set result can be combined with the algorithm of [4] to obtain an improved running time of $O\left(mn/\epsilon + 1/\epsilon^5\right)$. The algorithm of [24] achieves a slightly improved complexity bound of $O\left(mn/\epsilon + (1/\epsilon^{4.5})\log(1/\epsilon)\right)$ using second-order cone programming combined with column generation. In addition, Bădoiu and Clarkson [2] proposed another simple algorithm that computes a $(1+\epsilon)$-approximation in $O(mn/\epsilon^2)$ operations. In another paper, the same authors established a tight upper bound of $\lceil 1/\epsilon \rceil$ on the size of an ϵ-core set [3]. However, their construction is based on the assumption that $n \geq \lfloor 1/\epsilon \rfloor$. The algorithm of Panigrahy [33] computes a $(1+\epsilon)$-approximation in $O(mn/\epsilon)$ operations. Note that this algorithm has the best known dependence on ϵ, and each of these algorithms is polynomial for fixed ϵ. If ϵ is viewed as part of the input data, the minimum enclosing ball can be formulated as an instance of convex programming problem and can be solved using the ellipsoid method in $O\left(n^3 m \log(1/\epsilon)\right)$ operations [19]. Alternatively, interior-point methods yield an overall complexity bound of $O\left(n^2 m^{3/2}\log(1/\epsilon)\right)$ operations if the problem is formulated as an instance of second-order cone programming [24].

In this paper, we focus on large-scale instances of the minimum enclosing ball problem for which a reasonably small value of ϵ is satisfactory. Throughout this paper, we adopt the real number model of computation [5], i.e., we assume that arithmetic operations with real numbers and comparisons can be done at unit cost. We propose and analyze two algorithms that compute a $(1+\epsilon)$-approximation to MEB($\mathcal{A}$) for a given $\epsilon > 0$. Our first algorithm is closely related to the Frank–Wolfe algorithm [15] applied to the dual formulation of the problem. At each iteration, the algorithm can only add points to the working core set. The second algorithm is obtained by incorporating "away" steps into each iteration of the first one (see, e.g., [41, 20]). As such, the latter algorithm has the potential to compute a smaller core set than the former one, since it allows "dropping" points from the working core set at each iteration. A similar algorithm has recently been proposed for the minimum-volume enclosing ellipsoid problem [39]. Both of our algorithms compute a $(1+\epsilon)$-approximation to MEB($\mathcal{A}$) in $O(mn/\epsilon)$ operations, which matches the currently best known dependence on ϵ. In addition, each algorithm explicitly computes an ϵ-core set of size $O(1/\epsilon)$. Our analysis reveals that the leading terms in the complexity analysis are reasonably small, which further contributes to the efficiency of our algorithms. Furthermore, we establish that the second algorithm asymptotically exhibits linear convergence. Our computational results indicate that the sizes of the core sets returned by our algorithms are generally much smaller than the corresponding worst-case estimates. Furthermore, as expected, the latter algorithm almost always outperforms the former one both in terms of the running time and the core set size.

radius closer to the
optimal MEB radius

We also discuss how our algorithms can be extended to compute an approximate minimum enclosing ball of more general input sets. In particular, we establish that the asymptotic core set size of $O(1/\epsilon)$ extends to a much larger class of input sets.

We first compare our algorithms with the one proposed by Panigrahy [33], which computes a $(1+\epsilon)$-approximation to the minimum enclosing ball of a finite set of points in $O(mn/\epsilon)$ arithmetic operations. Panigrahy's algorithm starts with a ball whose radius is known to be smaller than that of the minimum enclosing ball and maintains an upper bound ζ on the difference between these two radii. At each iteration, the algorithm moves the current ball toward the furthest point from the center until the ball touches that particular point without changing the radius of the ball. After repeating such iterations $O(1/\zeta)$ times, the algorithm either provides a certificate that an approximate solution has been computed or decides that either the radius can be increased or the error bound ζ can be decreased. The whole procedure is then repeated using the new parameters for the radius and the error bound. Similarly to Panigrahy's algorithm, each of our algorithms also constructs a sequence of balls, and our first algorithm moves the center toward the furthest point from the center of the current ball at each iteration. However, the center moves by only a fraction of this distance. Furthermore, the second algorithm also allows us to move the current center away from the closest point in the working core set. Unlike Panigrahy's algorithm, our algorithms construct balls of strictly increasing radii in each iteration, and the radius and the error bound are updated at each iteration. While Panigrahy's algorithm checks the termination criterion after each set of $O(1/\zeta)$ iterations, our algorithms employ a simpler termination criterion in each iteration. This strategy has the potential advantage of earlier termination than that predicted by the theoretical worst-case estimate. Finally, while Panigrahy exclusively works with an input set of finite points with more general enclosing shapes, our algorithms can easily be modified to compute an approximation of the minimum enclosing ball of a much wider class of input sets without sacrificing the core set bound of $O(1/\epsilon)$.

$\rightarrow$ move the center towards
the farthest point

Algorithm 1 constructs a
sequence of increasing radii.

After the first version of this manuscript had been submitted, Clarkson [10] announced several results concerning the convergence properties of the Frank–Wolfe algorithm, which is the main ingredient in both of our algorithms. He studied the problem of maximizing a general concave function over the unit simplex, of which the dual formulation of the minimum enclosing problem is a special case. By giving a general definition of an *additive* ϵ-core set, he established core set results for several variants of the Frank–Wolfe algorithm in a more general setting. Due to the special structure of the objective function in the dual formulation of the minimum enclosing ball problem, his additive core set definition almost matches with our multiplicative core set definition given by (2). He presented an improved complexity bound of $O(mn/\epsilon)$ for a slightly modified version of the algorithm of [2], which matches the complexity bounds of our algorithms.

skip this paragraph

On the other hand, he establishes an ϵ-core set size of $O(1/\epsilon^2)$ for the general problem using his Algorithm 1.1, which, apart from the choice of the initial solutions, coincides with our first algorithm that computes a core set of size $O(1/\epsilon)$. He proposes a more sophisticated algorithm (cf. Algorithm 4.2 in [10]), which requires the computation of the optimal solution of a sequence of subproblems restricted to the smaller faces of the unit simplex, to establish the improved core set result of $O(1/\epsilon)$. In addition, he also studies a variant of the Frank–Wolfe algorithm that uses "away" steps (cf. Algorithm 5.1), for which he establishes a core set size of $O(1/\epsilon)$. However, this algorithm differs from our second algorithm, since it again requires the computation of the optimal solution of a sequence of subproblems. In contrast, we establish a core set size of $O(1/\epsilon)$ using much simpler (and lazier) algorithms. We derive explicit small constants inside the asymptotic bounds. Finally, we extend each of our algorithms to much more general input sets and establish the existence of core sets of size $O(1/\epsilon)$. Therefore, while Clarkson's results apply to a more general class of problems, we achieve the same or stronger results using simpler algorithms for the special case of the minimum enclosing ball problem, but we allow more general input sets.

This paper is organized as follows. In the remainder of this section, we define our notation. In section 2, we discuss optimization formulations for the minimum enclosing ball problem. Section 3 presents our first algorithm. The second algorithm is the topic of section 4. We discuss the extensions of our algorithms in section 5. The computational results are presented in section 6. Finally, we conclude the paper with some future research directions in section 7.

1.1. Notation.

Vectors are denoted by lowercase Roman letters. For a vector p, p_i denotes its ith component. Inequalities on vectors apply to each component. We reserve e^j for the jth unit vector. Uppercase Roman letters are reserved for matrices. We use $\log(\cdot)$ to denote the natural logarithm. Functions and operators are denoted by uppercase Greek letters. Scalars except for m and n are represented by lowercase Greek letters unless they represent components of a vector or elements of a sequence of scalars, vectors, or matrices. We reserve i, j, and k for such indexing purposes. Uppercase script letters are used for all other objects such as sets, balls, and ellipsoids.

2. Optimization formulations.

In this section, we review the optimization formulations of the minimum enclosing ball problem. We remark that most of the material of this section already appears in the earlier literature (see, e.g., [11, 26]). Some results and proofs are included for the sake of completeness.

Let $\mathcal{A} := \{a^1, \dots, a^m\} \subset \mathbb{R}^n$. The minimum enclosing ball of $\mathcal{A}$ can be computed by solving the following optimization problem:

$$(\mathcal{P}_1) \quad \begin{array}{ll} \min_{c,\rho} & \rho \\ \text{subject to} & \\ & \|a^i - c\| \le \rho, \quad i = 1, \dots, m, \end{array}$$

center, radius

where $c \in \mathbb{R}^n$ and $\rho \in \mathbb{R}$ are the decision variables. By squaring the constraints and defining $\gamma := \rho^2$, $(\mathcal{P}_1)$ can be converted into the following optimization problem with smooth, convex quadratic constraints:

$$(\mathcal{P}_2) \quad \min_{c,\gamma} \quad \gamma \quad \text{subject to} \quad (a^i)^T a^i - 2(a^i)^T c + c^T c \le \gamma, \quad i = 1, \ldots, m.$$

sqrt () <= ro / ^2 both sides

The Lagrangian dual of $(\mathcal{P}_2)$ is given by

$$(\mathcal{D}) \quad \max_u \quad \Phi(u) := \sum_{i=1}^m u_i (a^i)^T a^i - \left(\sum_{i=1}^m u_i a^i\right)^T \left(\sum_{i=1}^m u_i a^i\right)$$
$$\text{subject to} \quad \sum_{i=1}^m u_i = 1, \quad u_i \ge 0, \quad i = 1, \ldots, m,$$

https://www.researchgate.net/profile/Christian-Bauckhage/publication/331152293_Lecture_Notes_on_Machine_Learning_Minimum_Enclosing_Balls/links/5c756435299bf1268d282875/Lecture-Notes-on-Machine-Learning-Minimum-Enclosing-Balls.pdf

where $u \in \mathbb{R}^m$ is the decision variable.

Since $(\mathcal{P}_2)$ is a concave maximization problem with linear constraints, it follows from the Karush–Kuhn–Tucker optimality conditions that $(c_\mathcal{A}, \gamma_\mathcal{A}) \in \mathbb{R}^n \times \mathbb{R}$ is an optimal solution of $(\mathcal{P}_2)$ if and only if there exists $u^* \in \mathbb{R}^m$ such that

(3a) $$\sum_{i=1}^m u_i^* = 1,$$ from derivative of the Lagrangian with respect to ro

(3b) $$c_\mathcal{A} = \sum_{i=1}^m u_i^* a^i,$$ from derivative of the Lagrangian with respect to c

(3c) $$(a^i)^T a^i - 2(a^i)^T c_\mathcal{A} + (c_\mathcal{A})^T c_\mathcal{A} \le \gamma_\mathcal{A}, \quad i = 1, \ldots, m,$$

(3d) $$u_i^* \left((a^i)^T a^i - 2(a^i)^T c_\mathcal{A} + (c_\mathcal{A})^T c_\mathcal{A} - \gamma_\mathcal{A}\right) = 0, \quad i = 1, \ldots, m,$$

(3e) $$u^* \ge 0.$$

A simple manipulation of the optimality conditions reveals that

(4) $$\text{ro^2} = \gamma_\mathcal{A} = \Phi(u^*),$$

which implies that $u^* \in \mathbb{R}^m$ is an optimal solution of $(\mathcal{D})$ and that strong duality holds between $(\mathcal{P}_2)$ and $(\mathcal{D})$. Note that the center $c_\mathcal{A}$ of the minimum enclosing ball of $\mathcal{A}$ is given by a convex combination of the elements of $\mathcal{A}$ by (3b). In addition, it follows from (3d) that only the components of u^* corresponding to the points on the boundary of MEB$(\mathcal{A})$ can have a positive value. the others will be 0, since each ui >= 0

Strong duality = the optimal values of both the primal and dual problems are equal.

=> the center is a convex combination of points on the boundary of MEB

LEMMA 2.1. *Let $\mathcal{A} = \{a^1, \ldots, a^m\}$. The minimum enclosing ball of $\mathcal{A}$ exists and is unique. Let $u^* \in \mathbb{R}^m$ denote the optimal solution of $(\mathcal{D})$. Then, MEB$(\mathcal{A}) = \mathcal{B}_{c_\mathcal{A}, \rho_\mathcal{A}}$, where*

(5) $$\left[c_\mathcal{A} = \sum_{i=1}^m u_i^* a^i, \quad \rho_\mathcal{A} = \sqrt{\Phi(u^*)}. \right]$$

Proof. Note that $\mathcal{A} \subset \mathcal{B}_{0,\rho^u}$, where $\rho^u := \max_{i=1,\ldots,m} \|a^i\|$. By adding the redundant constraint $\gamma \le (\rho^u)^2$ to $(\mathcal{P}_2)$, the feasible region becomes a closed and bounded set and the objective function is continuous, which establishes the existence of MEB$(\mathcal{A})$. If there were two different minimum enclosing balls, one can then construct a ball of smaller radius that encloses the intersection of the two balls and hence also $\mathcal{A}$, which is a contradiction. The relationships (5) directly follow from the discussions preceding the lemma. ☐

By Lemma 2.1, MEB($\mathcal{A}$) can be computed by solving the dual problem ($\mathcal{D}$), which will be the basis of both of our algorithms in this paper. We close this section by the following technical result, which will play an important role in finding a good initial feasible solution in our algorithms. The reader is referred to [18, 4, 12] for the proof of this result.

(*) IMPORTANT!

LEMMA 2.2. *Let $\mathcal{A} = \{a^1, \ldots, a^m\}$, and let MEB($\mathcal{A}$) $= \mathcal{B}_{c_\mathcal{A}, \rho_\mathcal{A}}$. Then, any closed half-space that contains $c_\mathcal{A}$ also contains at least one point $a^j \in \mathcal{A}$ such that $\|a^j - c_\mathcal{A}\| = \rho_\mathcal{A}$.*

3. The first algorithm.

Given $\mathcal{A} := \{a^1, \ldots, a^m\} \subset \mathbb{R}^n$ and $\epsilon > 0$, we present our first algorithm that computes a $(1+\epsilon)$-approximation to MEB($\mathcal{A}$) in this section.

Algorithm 3.1 The first algorithm that computes a $(1+\epsilon)$-approximation to MEB($\mathcal{A}$).

```
Require: Input set of points $\mathcal{A} = \{a^1, \ldots, a^m\} \subset \mathbb{R}^n, \epsilon > 0$.
 1: $\alpha \leftarrow \arg\max_{i=1,\ldots,m} \|a^i - a^1\|^2$,   $\beta \leftarrow \arg\max_{i=1,\ldots,m} \|a^i - a^\alpha\|^2$;
 2: $u_i^0 \leftarrow 0$,   $i = 1, \ldots, m$; u0 is a feasible solution of the dual problem (D)
 3: $u_\alpha^0 \leftarrow 1/2$,   $u_\beta^0 \leftarrow 1/2$; set the vector u0 to 0, and the indices alpha and beta to 1/2 => sum of u = 1
 4: $\mathcal{X}_0 \leftarrow \{a^\alpha, a^\beta\}$; core set
 5: $c^0 \leftarrow \sum_{i=1}^m u_i^0 a^i$; center
 6: $\gamma^0 \leftarrow \Phi(u^0)$; radius^2 = objective function value
 7: $\kappa \leftarrow \arg\max_{i=1,\ldots,m} \|a^i - c^0\|^2$; index of the point furthest away from the center c0
 8: $\delta_0 \leftarrow \left(\|a^\kappa - c^0\|^2 / \gamma^0\right) - 1$; distance of farthest point ^2 / radius^2 - 1 >= 0
                                           => this is a measure of the error
 9: $k \leftarrow 0$;                        stopping condition
10: While $\delta_k > (1+\epsilon)^2 - 1$, do
11: loop
12:     $\lambda^k \leftarrow \delta_k / [2(1+\delta_k)]$; step size -> it will be diminishing
13:     $k \leftarrow k + 1$; iteration counter
14:     $u^k \leftarrow (1 - \lambda^{k-1}) u^{k-1} + \lambda^{k-1} e^\kappa$; update u_k to ensure dual feasibility
15:     $c^k \leftarrow (1 - \lambda^{k-1}) c^{k-1} + \lambda^{k-1} a^\kappa$; center: c_k+1 is computed by shifting ck toward the furthest point a_k
16:     $\mathcal{X}_k \leftarrow \mathcal{X}_{k-1} \cup \{a^\kappa\}$; add a_k to the working core set
17:     $\gamma^k \leftarrow \Phi(u^k)$; radius^2
18:     $\kappa \leftarrow \arg\max_{i=1,\ldots,m} \|a^i - c^k\|^2$; compute k and delta same as above (before the loop)
19:     $\delta_k \leftarrow \left(\|a^\kappa - c^k\|^2 / \gamma^k\right) - 1$;   ak is the furthest point from ck
20: end loop
21: Output $c^k, \mathcal{X}_k, u^k, \sqrt{(1+\delta_k)\gamma^k}$.
```

Find the index alpha of the furthest point from a1 and then find the index beta of the furthest point from a_alpha.

e.g.: delta0 = 4
=> lambda0 = 4 / 2*5 = 2/5 = 0.4
Next, delta1 = 2
=> lambda1 = 2 / 2*3 = 1/3 = 0.33
=> diminishing step size

We now describe Algorithm 3.1 in more detail. In step 1, the algorithm computes the furthest point $a^\alpha \in \mathcal{A}$ from $a^1 \in \mathcal{A}$ and then computes the furthest point $a^\beta \in \mathcal{A}$ from a^α. Steps 2 and 3 initialize the vector $u^0 \in \mathbb{R}^m$. Note that u^0 is a feasible solution of the dual problem ($\mathcal{D}$). The core set $\mathcal{X}_0$ is initialized at step 4. At each iteration, the algorithm implicitly constructs a "trial" ball with center c^k and radius $(\gamma^k)^{1/2}$. By Lemma 2.1, this ball coincides with MEB($\mathcal{A}$) if and only if u^k is an optimal solution of ($\mathcal{D}$). Otherwise, at least one point in $\mathcal{A}$ lies outside of this ball. Note that δ_k satisfies $\|a^\kappa - c^k\|^2 = (1+\delta_k)\gamma^k$, where $a^\kappa \in \mathcal{A}$ is the furthest point from c^k. It follows that the trial ball encloses $\mathcal{A}$ if its radius is expanded by a factor of $(1+\delta_k)^{1/2}$, i.e., $\Phi(u^k) \leq \Phi(u^*) \leq (1+\delta_k)\Phi(u^k)$. Unless the termination criterion is satisfied, the new center c^{k+1} is computed by shifting c^k toward the furthest point a^κ, which is added to the working core set $\mathcal{X}_{k+1}$, and u^{k+1} is updated accordingly to ensure that dual feasibility is maintained. The algorithm continues in an iterative manner by computing a new trial ball corresponding to u^{k+1}.

Algorithm 3.1 is the adaptation of the Frank–Wolfe algorithm to the dual problem $(\mathcal{D})$. At each iteration, the quadratic objective function $\Phi(u)$ of $(\mathcal{D})$ is linearized at the current feasible solution u^k. Since the feasible region of $(\mathcal{D})$ is the unit simplex, the unit vector e^κ, where κ is the index of the furthest point in $\mathcal{A}$ from c^k, solves the linearized subproblem. It is easy to verify that

<-- unit simplex

$$\lambda^k = \arg \max_{\lambda\in[0,1]} \Phi\left((1-\lambda)u^k + \lambda e^\kappa\right).$$

We remark that Algorithm 3.1 uses only the first-order approximation to the objective function Φ. As such, each iteration is fairly cheap, but the number of iterations is usually significantly higher than other algorithms that use second-order information such as interior-point methods. However, such general-purpose algorithms become computationally infeasible for larger problems, since each iteration is usually much more expensive. This observation provides one of our motivations to develop a specialized algorithm for this problem.

3.1. Analysis of the first algorithm. This subsection is devoted to the analysis of Algorithm 3.1.

In the first pass, skipped this section, but revisited it later...

LEMMA 3.1. *$u^0 \in \mathbb{R}^m$ satisfies $\gamma^0 = \Phi(u^0) \geq (1/3)\Phi(u^*) = (1/3)\gamma_{\mathcal{A}}$, where $u^* \in \mathbb{R}^m$ and $\gamma_{\mathcal{A}}$ are the optimal solution and the optimal value of $(\mathcal{D})$, respectively. Furthermore, $\delta_0 \leq 8$.*

5^2 >= 1/3 * 6^2
25 >= 12

Proof. For any vectors $y, z \in \mathbb{R}^n$ and any $\varphi \in \mathbb{R}$, it is easy to verify that

fi(1-fi) (y^2 - 2yz + z^2)

(1-fi)*y^2*(1 - fi) = (1-fi)^2 * y^2
fi*z^2*(1-(1-fi)) = fi*z^2*fi = fi^2*z^2
-fi(1-fi)*(-2yz) = 2*(1-fi)*fi*y*z

$$\|(1-\varphi)y + \varphi z\|^2 = (1-\varphi)\|y\|^2 + \varphi\|z\|^2 - \varphi(1-\varphi)\|y-z\|^2. \tag{6}$$

$\rightarrow$ (1-fi)^2y^2 + 2*(1-fi)*fi*y*z + fi^2*z^2 =

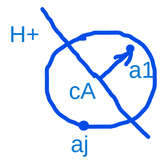
Note that

$$\Phi(u^0) = (1/2)\|a^\alpha\|^2 + (1/2)\left\|a^\beta\right\|^2 - \left\|(1/2)\left(a^\alpha + a^\beta\right)\right\|^2 = (1/4)\left\|a^\alpha - a^\beta\right\|^2, \tag{7}$$

where we used (6) to derive the second equality. The proof is based on establishing that at least one of a^α and a^β is sufficiently away from the center $c_{\mathcal{A}}$ of MEB$(\mathcal{A})$.

First, suppose that $\left\|a^1 - c_{\mathcal{A}}\right\| \geq (1/\sqrt{3})\rho_{\mathcal{A}}$, where $\rho_{\mathcal{A}}$ is the radius of MEB$(\mathcal{A})$. Let $\mathcal{H}$ be the hyperplane passing through $c_{\mathcal{A}}$ that is perpendicular to $a^1 - c_{\mathcal{A}}$. Let $\mathcal{H}_+$ denote the closed half-space whose boundary is $\mathcal{H}$ and which does not contain a^1. By Lemma 2.2, $\mathcal{H}_+$ contains a point $a^j \in \mathcal{A}$ such that $\left\|a^j - c_{\mathcal{A}}\right\| = \rho_{\mathcal{A}}$. Therefore, $\left\|a^\alpha - a^1\right\|^2 \geq \left\|a^1 - c_{\mathcal{A}}\right\|^2 + (\rho_{\mathcal{A}})^2 \geq (4/3)\gamma_{\mathcal{A}}$, where $\gamma_{\mathcal{A}} = \Phi(u^*) = (\rho_{\mathcal{A}})^2$ is the optimal value of $(\mathcal{D})$. It follows from (7) that

The distance from point a1 to the optimal center is greater than 0.57 * the optimal center.



$$\Phi(u^0) = (1/4)\left\|a^\beta - a^\alpha\right\|^2 \geq (1/4)\left\|a^1 - a^\alpha\right\|^2 \geq (1/3)\Phi(u^*).$$

triangle inequality

Suppose now that $\left\|a^1 - c_{\mathcal{A}}\right\| = \theta\rho_{\mathcal{A}}$, where $\theta < 1/\sqrt{3}$. In this case, $\left\|a^1 - a^\alpha\right\| \leq \left\|a^1 - c_{\mathcal{A}}\right\| + \|c_{\mathcal{A}} - a^\alpha\|$, which implies that

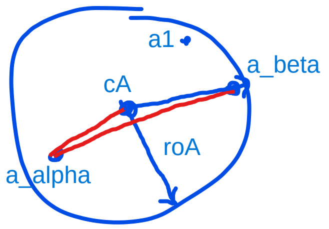
$$\|c_{\mathcal{A}} - a^\alpha\| \geq \left\|a^1 - a^\alpha\right\| - \left\|a^1 - c_{\mathcal{A}}\right\| \geq (1+\theta^2)^{1/2}\rho_{\mathcal{A}} - \theta\rho_{\mathcal{A}} = [(1+\theta^2)^{1/2} - \theta]\rho_{\mathcal{A}},$$

sqrt((theta * roA)^2 + roA^2) = sqrt(roA^2(1 + theta^2)) = roA * sqrt(1 + theta^2) where we again invoked Lemma 2.2 to obtain a lower bound on $\|a^1 - a^\alpha\|$. Therefore, one more application of Lemma 2.2 yields

alpha, beta: Find the index alpha of the furthest point from a1 and then find the index beta of the furthest point from a_alpha.

$$\begin{aligned}\Phi(u^0) &= (1/4)\left\|a^\beta - a^\alpha\right\|^2 \quad \text{roA >= ||cA - a\_beta|| and just use triangle inequality}\\ &\geq (1/4)\left(\left\|a^\alpha - c_{\mathcal{A}}\right\|^2 + (\rho_{\mathcal{A}})^2\right)\\ &\geq (1/4)\left(1+\theta^2+\theta^2-2\theta(1+\theta^2)^{1/2}+1\right)\gamma_{\mathcal{A}}\\ &= (1/2)\left(1+\theta^2-\theta(1+\theta^2)^{1/2}\right)\gamma_{\mathcal{A}}.\end{aligned}$$

||a_alpha - cA||^2 >= (1 + delta^2) + delta^2 - 2*delta*sqrt(1+delta^2)



It is easy to verify that $(1/2)\left(1+\theta^2-\theta(1+\theta^2)^{1/2}\right)$ is a decreasing function of θ. Since $\theta < 1/\sqrt{3}$, it follows that

$$\text{gamma\_0 =} \quad \Phi(u^0) \geq (1/2)(1 + 1/3 - 2/3)\gamma_{\mathcal{A}} = (1/3)\Phi(u^*),$$

which completes the first part of the proof.

Let a^κ be the furthest point in $\mathcal{A}$ from $c^0 = (1/2)(a^\alpha + a^\beta)$. Then,

$$\begin{aligned}\left\|a^\kappa - c^0\right\| &\leq \left\|a^\kappa - a^\alpha\right\| + \left\|a^\alpha - c^0\right\|, \quad \text{triangle inequality}\\ &\leq \left\|a^\beta - a^\alpha\right\| + (1/2)\left\|a^\beta - a^\alpha\right\| = (3/2)\left\|a^\beta - a^\alpha\right\|,\end{aligned}$$

where we used the definition of c^0 and the fact that a^β is the furthest point in $\mathcal{A}$ from a^α to derive the second inequality. Therefore, $\delta_0 = (\|a^\kappa - c^0\|^2/\gamma^0) - 1 \leq [4(9/4)(\gamma^0/\gamma^0)] - 1 = 8$, where we used (7). The second part of the assertion follows. ☐

Lemma 3.1 establishes several properties of the initial feasible solution $u^0 \in \mathbb{R}^m$. The next lemma relates the dual objective function values evaluated at the successive iterates generated by Algorithm 3.1.

LEMMA 3.2. *For each $k = 0, 1, \ldots,$ the following relationship is satisfied:*

$$\gamma^{k+1} = \gamma^k\left(1 + \frac{\delta_k^2}{4(1+\delta_k)}\right). \quad (8)$$

bound for the error between iterations

delta0 <= 8
gamma_1/gamma0 <=
<= (1+ 64/4*9) = 2.77

Proof. Let a^κ denote the furthest point from c^k. Then, $u^{k+1} = (1-\lambda^k)u^k + \lambda^k e^\kappa$. Therefore,

$$\begin{aligned}\gamma^{k+1} &= \Phi\left((1-\lambda^k)u^k + \lambda^k e^\kappa\right)\\ &\overset{(D)}{=} (1-\lambda^k)\sum_{i=1}^m u_i^k (a^i)^T(a^i) + \lambda^k (a^\kappa)^T(a^\kappa) - \left\|(1-\lambda^k)\left(\sum_{i=1}^m u_i^k a^i\right) + \lambda^k a^\kappa\right\|^2\\ &\overset{(6)}{=} (1-\lambda^k)\left(\sum_{i=1}^m u_i^k (a^i)^T(a^i) - \left\|\sum_{i=1}^m u_i^k a^i\right\|^2\right) + \lambda^k(1-\lambda^k)\left\|\sum_{i=1}^m u_i^k a^i - a^\kappa\right\|^2\\ &= (1-\lambda^k)\gamma^k + \lambda^k(1-\lambda^k)\underline{\left\|a^\kappa - c^k\right\|}^2 \quad \text{delta\_k = ||a\_k - c\_k||\^{}2 / gamma\_k - 1} \\ &\qquad \text{=> ||a\_k - c\_k||\^{}2 = (delta\_k + 1) * gamma\_k}\\ &= (1-\underline{\lambda}^k)\gamma^k + \underline{\lambda}^k(1-\underline{\lambda}^k)(1+\delta_k)\gamma^k \quad \text{lambda\_k = delta\_k / 2* (1+delta\_k)}\\ &= \gamma^k\left(1 + \frac{\delta_k^2}{4(1+\delta_k)}\right),\end{aligned}$$

where we used (6) in the third equality, the definitions of c^k and δ_k in the fourth and fifth equalities, respectively, and the definition of λ^k in the last equality. ☐

delta_k: distance of farthest point from the current center ^2 / radius^2 - 1 >= 0
=> delta_k is a measure of the current error

We now focus on establishing an upper bound on the number of iterations required to have an iterate u^k with δ_k sufficiently small. To that end, let us define

$$\tau_\nu := \min\left\{k : \delta_k \leq \frac{1}{2^\nu}\right\}, \quad \nu = 0, 1, \ldots . \tag{9}$$

LEMMA 3.3. *τ_ν satisfies the following relationships:*

$$\tau_0 \leq 9, \tag{10a}$$

$$\tau_\nu - \tau_{\nu-1} \leq 12.5(2^\nu) \quad \nu = 1, 2, \ldots . \tag{10b}$$

Proof. Let us first consider τ_0. At each iteration $k < \tau_0$, we have $\delta_k > 1$. By Lemma 3.2,

$$\begin{aligned}\gamma^{k+1} &= \gamma^k\left(1 + \frac{\delta_k^2}{4(1+\delta_k)}\right), \\ &\geq \gamma^k(1 + 1/8),\end{aligned}$$

where we used the fact that $1 + (1/4)(x^2/(1+x))$ is an increasing function of x. Iterating this inequality, we obtain $\gamma^{k+1} \geq (9/8)^{k+1}\gamma^0$. By Lemma 3.1 and the feasibility of u^{k+1}, we have

Lemma 3.1

$$\gamma_\mathcal{A} \geq \gamma^{k+1} \geq (9/8)^{k+1}\gamma^0 \geq (9/8)^{k+1}(\gamma_\mathcal{A}/3),$$

which implies that $\tau_0 \leq k + 1 \leq \log(3)/\log(9/8)$ or, equivalently, that $\tau_0 \leq 9$.

Let us now consider $\tau_\nu - \tau_{\nu-1}$ for $\nu = 1, 2, \ldots$. Let $\mu := \tau_{\nu-1}$. At each iteration k with $\delta_k > 1/2^\nu$, we similarly have

$$\gamma^{k+1} = \gamma^k\left(1 + \frac{\delta_k^2}{4(1+\delta_k)}\right) \geq \gamma^k\left(1 + \frac{1}{2^{2+\nu}(2^\nu+1)}\right).$$

At iteration μ, we have $\delta_\mu \leq 1/2^{\nu-1}$. Since the ball centered at c^μ with radius $[(1+\delta_\mu)\gamma^\mu]^{1/2}$ encloses $\mathcal{A}$, it follows that $\gamma^\mu \leq \gamma_\mathcal{A} \leq (1+\delta_\mu)\gamma^\mu \leq (1 + (1/2^{\nu-1}))\gamma^\mu$. Together with the repeated application of the inequality above, we have

the radii we produce
are increasing

$$\gamma_\mathcal{A} \geq \gamma^{\mu+k} \geq \gamma^\mu\left(1 + \frac{1}{2^{2+\nu}(2^\nu+1)}\right)^k \geq \frac{\gamma_\mathcal{A}}{1 + (1/2^{\nu-1})}\left(1 + \frac{1}{2^{2+\nu}(2^\nu+1)}\right)^k,$$

which implies that

$$\begin{aligned}\tau_\nu - \tau_{\nu-1} &\leq \frac{\log\left(1 + \frac{1}{2^{\nu-1}}\right)}{\log\left(1 + \frac{1}{2^{2+\nu}(2^\nu+1)}\right)} \\ &\leq \frac{1}{2^{\nu-1}}\,\frac{\frac{1}{2^{2+\nu}(2^\nu+1)} + 1}{\frac{1}{2^{2+\nu}(2^\nu+1)}} = \frac{2}{2^\nu} + 8(2^\nu+1) \\ &\leq 9 + 8(2^\nu) \leq (12.5)2^\nu,\end{aligned}$$

where we used the inequalities $\log(1+x) \leq x$ for $x > -1$ and $\log(1+x) \geq x/(x+1)$ for $x > -1$. $\square$

The following lemma establishes an upper bound on the number of iterations to obtain an iterate with $\delta_k \leq \delta$.

LEMMA 3.4. *Let $\delta \in (0,1)$. Algorithm 3.1 computes an iterate k satisfying $\delta_k \leq \delta$ in at most $9 + 50/\delta$ iterations.*

Proof. Let σ be an integer such that $1/2^\sigma \leq \delta \leq 2/2^\sigma$. Therefore, after at most τ_σ iterations, Algorithm 3.1 computes an iterate k satisfying $\delta_k \leq \delta$. By Lemma 3.3,

$$\tau_\sigma = \tau_0 + \sum_{\nu=1}^{\sigma}(\tau_\nu - \tau_{\nu-1}) \leq 9 + 12.5\sum_{\nu=1}^{\sigma} 2^\nu \leq 9 + 25(2^\sigma) \leq 9 + 50/\delta. \qquad \square$$

We now have all of the ingredients to establish the iteration complexity of Algorithm 3.1.

m n-dimensional points

THEOREM 3.1. *Given $\mathcal{A} := \{a^1, \ldots, a^m\} \subset \mathbb{R}^n$ and $\epsilon \in (0,1)$, Algorithm 3.1 computes a $(1+\epsilon)$-approximation to MEB$(\mathcal{A})$ in at most $9 + 25/\epsilon$ iterations.*

Proof. Let u^η denote the final iterate computed by Algorithm 3.1, and let $\gamma^\eta = \Phi(u^\eta)$. Then, the trial ball centered at c^η with radius $[(1+\delta_\eta)\gamma^\eta]^{1/2}$ encloses $\mathcal{A}$. Note that u^η is a feasible solution of $(\mathcal{D})$, and $\delta_\eta \leq (1+\epsilon)^2 - 1$ by the termination criterion. stopping criteria satisfied

the ball we return is <= optimal ball

Therefore, $(\gamma^\eta)^{1/2} \leq \rho_\mathcal{A} \leq [(1+\delta_\eta)\gamma^\eta]^{1/2} \leq (1+\epsilon)(\gamma^\eta)^{1/2}$, which implies that the ball centered at c^η with radius $[(1+\delta_\eta)\gamma^\eta]^{1/2}$ is a $(1+\epsilon)$-approximation to MEB$(\mathcal{A})$.

By Lemma 3.4, Algorithm 3.1 computes such an iterate with $\delta \leq (1+\epsilon)^2 - 1$ in at most $9 + 50/(2\epsilon + \epsilon^2) \leq 9 + 25/\epsilon$ iterations. $\square$

Theorem 3.1 establishes that Algorithm 3.1 converges in $O(1/\epsilon)$ iterations. The next result presents the overall complexity.

(*) Iterations convergence in O(1/epsilon) proven.

THEOREM 3.2. *Given $\mathcal{A} := \{a^1, \ldots, a^m\} \subset \mathbb{R}^n$ and $\epsilon \in (0,1)$, Algorithm 3.1 computes a $(1+\epsilon)$-approximation to MEB$(\mathcal{A})$ in at most $O(mn/\epsilon)$ arithmetic operations.*

Proof. The computation of the initial feasible solution u^0 requires two furthest point computations, which can be performed in $O(mn)$ operations. At each iteration, the dominating work is the computation of the furthest point from the center of the current trial ball, which also requires $O(mn)$ operations (note that γ^k can be updated using (8) in $O(1)$ operations). The result follows from Theorem 3.1. $\square$

For a given point, finding the furthest point requires scanning (m-1) d-dimensional points.
=> O(mn) arithmetic operations
The coreset is of size O(1/epsilon).
=> O(mn/epsilon)

We remark that the overall complexity of Algorithm 3.1 is linear in the number of points m and also linear in the dimension n. As such, the worst-case running time asymptotically matches the currently best known bound due to [33]. In particular, Theorem 3.2 suggests that Algorithm 3.1 is especially well-suited for large instances of the minimum enclosing ball problem where a moderately small value of ϵ (such as 10^{-3}) would be satisfactory.

We close this section by establishing that Algorithm 3.1 explicitly computes a core set of size $O(1/\epsilon)$, which also asymptotically matches the currently best known bound.

THEOREM 3.3. *Given $\mathcal{A} := \{a^1, \ldots, a^m\} \subset \mathbb{R}^n$ and $\epsilon \in (0,1)$, let η denote the index of the final iterate computed by Algorithm 3.1. Then, $\mathcal{X}_\eta \subseteq \mathcal{A}$ is an ϵ-core set of $\mathcal{A}$. Furthermore, $|\mathcal{X}_\eta| = O(1/\epsilon)$.*

Proof. Let u^η denote the final iterate returned by Algorithm 3.1, and let $\gamma^\eta = \Phi(u^\eta)$. Clearly, the restriction of u^η to its positive entries is a feasible solution of the dual formulation of the minimum enclosing ball problem for $\mathcal{X}_\eta$. Therefore, $\gamma^\eta \leq (\rho_{\mathcal{X}_\eta})^2 \leq (\rho_\mathcal{A})^2$, where $\rho_{\mathcal{X}_\eta}$ is the radius of MEB$(\mathcal{X}_\eta)$. However, $\gamma_\mathcal{A} = (\rho_\mathcal{A})^2 \leq (1+\delta_\eta)\gamma^\eta \leq (1+\epsilon)^2\gamma^\eta$ by Theorem 3.1. Combining these inequalities, we obtain $\rho_{\mathcal{X}_\eta} \leq \rho_\mathcal{A} \leq (1+\epsilon)(\gamma^\eta)^{1/2} \leq (1+\epsilon)\rho_{\mathcal{X}_\eta}$ as desired.

Note that $|\mathcal{X}_\eta|$ is precisely equal to the number of positive components of u^η. However, the initial solution u^0 has only two positive components. Each iteration can add at most one positive component to u^k. Therefore, $|\mathcal{X}_\eta| \leq 11 + 25/\epsilon = O(1/\epsilon)$ by Theorem 3.1. $\square$

4. The second algorithm. In this section, we describe our second algorithm, which is a modification of Algorithm 3.1.

Algorithm 4.1 The second algorithm that computes a $(1+\epsilon)$-approximation to MEB($\mathcal{A}$).

```
Require: Input set of points $\mathcal{A} = \{a^1, \ldots, a^m\} \subset \mathbb{R}^n$, $\epsilon > 0$.
 1: $\alpha \leftarrow \arg\max_{i=1,\ldots,m} \left\|a^i - a^1\right\|^2, \quad \beta \leftarrow \arg\max_{i=1,\ldots,m} \left\|a^i - a^\alpha\right\|^2$;
 2: $u_i^0 \leftarrow 0, \quad i = 1, \ldots, m$;
 3: $u_\alpha^0 \leftarrow 1/2, \quad u_\beta^0 \leftarrow 1/2$;
 4: $\mathcal{X}_0 \leftarrow \{a^\alpha, a^\beta\}$;
 5: $c^0 \leftarrow \sum_{i=1}^m u_i^0 a^i$;
 6: $\gamma^0 \leftarrow \Phi(u^0)$;
 7: $\kappa \leftarrow \arg\max_{i=1,\ldots,m} \left\|a^i - c^0\right\|^2, \quad \xi \leftarrow \arg\min_{i:a^i \in \mathcal{X}_0} \left\|a^i - c^0\right\|^2$;
 8: $\delta_0^+ \leftarrow \left(\left\|a^\kappa - c^0\right\|^2 / \gamma^0\right) - 1, \quad \delta_0^- \leftarrow 1 - \left(\left\|a^\xi - c^0\right\|^2 / \gamma^0\right)$;
 9: $\delta_0 \leftarrow \max\{\delta_0^+, \delta_0^-\}$;
10: $k \leftarrow 0$;
11: While $\delta_k > (1+\epsilon)^2 - 1$, do
12: loop
13:   if $\delta_k > \delta_k^-$, then
14:     $\lambda^k \leftarrow \delta_k / [2(1+\delta_k)]$;
15:     $k \leftarrow k+1$;
16:     $u^k \leftarrow (1-\lambda^{k-1}) u^{k-1} + \lambda^{k-1} e^\kappa$;
17:     $c^k \leftarrow (1-\lambda^{k-1}) c^{k-1} + \lambda^{k-1} a^\kappa$;
18:     $\mathcal{X}_k \leftarrow \mathcal{X}_{k-1} \cup \{a^\kappa\}$;
19:   else
20:     $\lambda^k \leftarrow \min\left\{\frac{\delta_k^-}{2(1-\delta_k^-)}, \frac{u_\xi^k}{1-u_\xi^k}\right\}$;
21:     if $\lambda^k = u_\xi^k/(1-u_\xi^k)$, then
22:       $\mathcal{X}_{k+1} \leftarrow \mathcal{X}_k \backslash \{a^\xi\}$;
23:     else
24:       $\mathcal{X}_{k+1} \leftarrow \mathcal{X}_k$;
25:     end if
26:     $k \leftarrow k+1$;
27:     $u^k \leftarrow (1+\lambda^{k-1}) u^{k-1} - \lambda^{k-1} e^\xi$;
28:     $c^k \leftarrow (1+\lambda^{k-1}) c^{k-1} - \lambda^{k-1} a^\xi$;
29:   end if
30:   $\gamma^k \leftarrow \Phi(u^k)$;
31:   $\kappa \leftarrow \arg\max_{i=1,\ldots,m} \left\|a^i - c^k\right\|^2, \quad \xi \leftarrow \arg\min_{i:a^i \in \mathcal{X}_k} \left\|a^i - c^k\right\|^2$;
32:   $\delta_k^+ \leftarrow \left(\left\|a^\kappa - c^k\right\|^2 / \gamma^k\right) - 1, \quad \delta_k^- \leftarrow 1 - \left(\left\|a^\xi - c^k\right\|^2 / \gamma^k\right)$;
33:   $\delta_k \leftarrow \max\{\delta_k^+, \delta_k^-\}$;
34: end loop
35: Output $c^k, \mathcal{X}_k, u^k, \sqrt{(1+\delta_k)\gamma^k}$.
```

Algorithm 4.1 starts off with the same initial solution u^0 as the one computed by Algorithm 3.1. At each iteration, the furthest point in $\mathcal{A}$ from the center c^k of the trial ball is computed as in Algorithm 3.1. In contrast, each iteration of Algorithm 4.1 also includes the computation of the *closest* point to c^k among all points in $\mathcal{X}_k \subseteq \mathcal{A}$.

Geometrically, the parameter δ_k^- is the largest number such that the current ball shrunken by a factor of $(1-\delta)^{1/2}$ does not contain any points in $\mathcal{X}_k$ for any $\delta > \delta_k^-$. Algebraically, the step performed by Algorithm 4.1 in this case corresponds to moving away from the vertex of the unit simplex that minimizes the linear approximation to $\Phi(u)$ at u^k, where the minimization is over the vertices $\{e^j : u_j^k > 0\}$. The feasible solution u^k is updated in different ways based on these two computations. If $\delta_k = \delta_k^+$, then Algorithm 4.1 uses the exact same update as in Algorithm 3.1. Otherwise, the new center c^{k+1} is obtained by moving the current center c^k *away* from the closest point $a^\xi \in \mathcal{X}_k$. Therefore, Algorithm 4.1 is obtained by incorporating "away" steps into Algorithm 3.1. For "away" steps, it is easy to verify that

$$\lambda^k = \arg \max_{\lambda \in \left[0, u_\xi^k/\left(1-u_\xi^k\right)\right]} \Phi\left((1+\lambda)u^k - \lambda e^\xi\right). \tag{11}$$

Note that the range of λ is chosen to ensure that the dual feasibility constraint $u^{k+1} \geq 0$ is satisfied.

4.1. Analysis of the second algorithm.

The analysis of Algorithm 4.1 is very similar to that of Algorithm 3.1. As in [39], we call iteration k a *plus*-iteration if $\delta_k = \delta_k^+$. If $\delta_k = \delta_k^-$ and $\lambda^k = (\delta_k^-)/[2(1-\delta_k^-)]$, then we call it a *minus*-iteration. The working core set remains unchanged at a minus-iteration. Finally, if $\delta_k = \delta_k^-$ and $\lambda^k = u_\xi^k/(1-u_\xi^k)$, we then call it a *drop*-iteration, since the ξth component of u^k drops to 0 and a^ξ is removed from the working core set.

Our analysis mimics the analysis of [39] for a similar algorithm that computes an approximation to the minimum-volume enclosing ellipsoid of a finite set of points. The next lemma establishes a lower bound on the improvement at each plus- or minus-iteration.

LEMMA 4.1. *At each plus- or minus-iteration,*

$$\gamma^{k+1} \geq \gamma^k \left(1 + \frac{\delta_k^2}{4(1+\delta_k)}\right), \quad k = 0, 1, \ldots. \tag{12}$$

Proof. At a plus-iteration, the result directly follows from Lemma 3.2. At a minus-iteration, a similar application of (6) reveals that

$$\gamma^{k+1} = \Phi\left((1+\lambda^k)u^k - \lambda^k e^\xi\right) = \gamma^k \left(1 + \frac{(\delta_k^-)^2}{4(1-\delta_k^-)}\right).$$

The result easily follows from the observation that

$$\frac{(\delta_k^-)^2}{4(1-\delta_k^-)} \geq \frac{(\delta_k^-)^2}{4(1+\delta_k^-)}$$

and that $\delta_k^- = \delta_k$ at a minus-iteration. ▯

Lemma 4.1 establishes that Algorithm 4.1 makes at least as much improvement as Algorithm 3.1 at each plus- or minus-iteration. At a drop-iteration, it is easy to show that $\gamma^{k+1} \geq \gamma^k$. However, we can no longer find a positive lower bound on $\gamma^{k+1} - \gamma^k \geq 0$. Using similar reasoning as in [39], each drop-iteration can be paired with the most recent plus-iteration k at which u_ξ^k was increased from 0, except for the αth and βth components, which were positive at the initial solution and may be decreased to zero for the first time. Therefore, we can double the iteration count (and add two iterations to account for the initial positive components of u^0) in the analysis of Algorithm 3.1 to establish that Algorithm 4.1 can compute a $(1+\epsilon)$-approximation to MEB$(\mathcal{A})$ in at most twice as many iterations as that required by Algorithm 3.1. Note that this does not affect the asymptotic iteration bound of Algorithm 3.1. Furthermore, each iteration still requires $O(mn)$ operations, which implies that the asymptotic overall complexity of Algorithm 4.1 also remains the same as that of Algorithm 3.1. Finally, the asymptotic bound on the size of the core set is also unaffected. However, we remark that Algorithm 4.1 has the potential to compute even smaller core sets than those returned by Algorithm 3.1 due to the possible inclusion of minus- and drop-iterations. We summarize these results in the following theorem.

THEOREM 4.1. *Given $\mathcal{A} := \{a^1, \ldots, a^m\} \subset \mathbb{R}^n$ and $\epsilon \in (0,1)$, Algorithm* 4.1 *computes a $(1+\epsilon)$-approximation to MEB$(\mathcal{A})$ in $O(mn/\epsilon)$ operations. Furthermore, upon termination, $\mathcal{X}_\eta \subseteq \mathcal{A}$ is an ϵ-core set and $|\mathcal{X}_\eta| = O(1/\epsilon)$, where η is the index of the final iterate computed by Algorithm* 4.1.

4.2. Linear convergence of the second algorithm. Despite the fact that Algorithm 4.1 appears to be a simple modification of Algorithm 3.1, it turns out that these two algorithms actually exhibit different characteristics. In particular, we establish that Algorithm 4.1 enjoys linear convergence, while a similar rate of convergence cannot, in general, be expected from Algorithm 3.1.

As observed in [41, 20], the search directions of Algorithm 3.1 always point toward the extreme points of the unit simplex. Therefore, the angle between these directions and the gradient of the objective function gets increasingly closer to the right angle in the situation when the optimal solution lies on the boundary of the unit simplex and is not an extreme point. For the minimum enclosing ball problem, an optimal solution of the dual problem will almost always lie in a lower-dimensional face of the unit simplex, except for the trivial cases such as a single input point or an input set sampled from the surface of a ball. It follows that Algorithm 3.1 is, in general, expected to exhibit a sublinear rate of convergence. In fact, this result has been formalized in [41] (see also [20, Theorem 3]) for the Frank–Wolfe algorithm under even stronger assumptions than those satisfied by the dual formulation of the minimum enclosing ball problem.

In an attempt to circumvent this drawback of Algorithm 3.1, Algorithm 4.1 works with an enlarged set of search directions by including those directions pointing away from the extreme points of the unit simplex. Such a general algorithm that incorporates "away" steps into the Frank–Wolfe algorithm was first proposed by Wolfe [41], and its convergence properties have been investigated by several authors. For the general problem of maximizing a concave function over a polytope, Wolfe [41] sketched and Guélat and Marcotte [20] detailed the proof of linear convergence under the assumptions of Lipschitz continuity of the gradient of the objective function, strong concavity of the objective function, and strict complementarity. More recently, Ahipaşaoğlu, Sun, and Todd [1] established the linear convergence of such an algorithm for the problem of maximizing a concave function over the unit simplex under a slightly different set of assumptions. Unfortunately, none of these previous results is directly applicable to our case, since either set of these assumptions implies the uniqueness of the optimal solution, which is not, in general, satisfied by the dual formulation of the minimum enclosing ball problem.

We therefore use an argument similar to that of [1] to establish the linear convergence of Algorithm 4.1. We work with a perturbation of the primal formulation $(\mathcal{P}_2)$ and show that the distance from an optimal primal-dual solution of the perturbed problem to the set of optimal primal-dual solutions of $(\mathcal{P}_2)$ and $(\mathcal{D})$ satisfies a Lipschitz condition using the stability results of Robinson [34] for general nonlinear programming problems.

Let us define the following perturbation of $(\mathcal{P}_2)$:

$$(\mathcal{P}(z(u,\delta))) \quad \begin{array}{ll} \min_{c,\gamma} & \gamma \\ \text{subject to} & \\ & (a^i)^T a^i - 2(a^i)^T c + c^T c \le \gamma + z_i(u,\delta), \quad i = 1,\dots,m, \end{array}$$

where $u \in \mathbb{R}^m$ lies on the unit simplex, $\delta \ge 0$, and $z(u,\delta)$ is given by

$$z_i(u,\delta) := \left\{ \begin{array}{ll} \delta\Phi(u) & \text{if } u_i = 0, \\ (a^i)^T a^i - 2(a^i)^T c(u) + c(u)^T c(u) - \Phi(u) & \text{else}, \end{array} \right. ; \quad i = 1,\dots,m,$$

where

$$c(u) := \sum_{i=1}^m u_i a^i.$$

Let $z^k := z(u^k, \delta_k),\ k = 0,1,\dots,$ where $u^k \in \mathbb{R}^m$ denotes the kth iterate and δ_k is the corresponding measure as computed by Algorithm 4.1. By a definition of δ_k,

$$(a^i)^T a^i - 2(a^i)^T c^k + (c^k)^T c^k - \Phi(u^k) \le \delta_k \Phi(u^k), \quad i = 1,\dots,m,$$

and

$$(a^i)^T a^i - 2(a^i)^T c^k + (c^k)^T c^k - \Phi(u^k) \ge -\delta_k \Phi(u^k), \quad \text{if } u_i^k > 0,$$

which implies that $|z_i^k| \le \delta_k \Phi(u^k)$ for $i = 1,\dots,m$. We remark that the latter inequality above is not necessarily satisfied by the kth iterate computed by Algorithm 3.1. Furthermore,

$$(13) \qquad (u^k)^T z^k = \sum_{i:u_i^k>0} u_i^k (a^i)^T a^i - 2(c^k)^T(c^k) + (c^k)^T c^k - \Phi(u^k) = 0,$$

where we used the definitions of c^k and $\Phi(u)$ together with the fact that u^k lies on the unit simplex. Using the fact that $c(u^k) = c^k$, it follows that $(c^k, \Phi(u^k))$ is a feasible solution of $(\mathcal{P}(z^k))$. The next lemma establishes that $(c^k, \Phi(u^k))$ is actually an optimal solution.

LEMMA 4.2. *For all* $k = 0,1,\dots,$ $(c^k, \Phi(u^k))$ *is an optimal solution of* $(\mathcal{P}(z^k))$.

Proof. The feasibility of $(c^k, \Phi(u^k))$ follows from the discussions preceding the lemma. Since $(\mathcal{P}(z^k))$ is a convex optimization problem and $(c^k, \Phi(u^k))$ satisfies the optimality conditions along with u^k as the Lagrange multipliers, the result follows. ▯

Let $\Xi(z(u,\delta))$ denote the optimal value of $(\mathcal{P}(z(u,\delta)))$. Note that Ξ is a convex function of $z(u,\delta)$, and if u^* is any Lagrange multiplier corresponding to the optimal solution of $\mathcal{P}(0)$ (equivalently, of $(\mathcal{P}_2)$), then u^* is a subgradient of Ξ at 0. Therefore, for all $k = 0,1,\dots,$

$$(14) \qquad \begin{array}{rcl} \Phi(u^k) = \Xi(z^k) & \ge & \Xi(0) + (u^*)^T z^k \\ & = & \Phi(u^*) + (u^* - u^k)^T z^k \\ & \ge & \Phi(u^*) - \|u^k - u^*\| \|z^k\|, \end{array}$$

where we used Lemma 4.2 and (13).

Let Δ denote the diameter of the input set $\mathcal{A}$, i.e., the maximum distance between any pair of points in $\mathcal{A}$. Since $(1/4)\Delta^2 \leq \Phi(u^*) \leq \Delta^2$, we have, for all k,

$$|z_i^k| \leq \delta_k \Phi(u^k) \leq \delta_k \Phi(u^*) \leq \delta_k \Delta^2,$$

which implies that $\|z^k\| \leq \sqrt{m}\Delta^2\delta_k$.

We will next use the stability results of Robinson [34] to establish an upper bound on $\|u^k - u^*\|$. We need to verify that all of the assumptions are satisfied for the unperturbed problem $(\mathcal{P}(0))$. Since the problem is convex and Slater's constraint qualification is satisfied, the constraints are regular at any feasible solution. Furthermore, let (c^*, γ^*) be the unique optimal solution of $(\mathcal{P}(0))$, and let u^* be any corresponding Lagrange multiplier (i.e., any optimal solution of $(\mathcal{D})$). Then, the Lagrangian function $\mathcal{L} : \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}$ for the problem $(\mathcal{P}(0))$ is given by

$$\mathcal{L}((c,\gamma),u) = \gamma + \sum_{i=1}^m u_i \left((a^i)^T a^i - 2(a^i)^T c + c^T c - \gamma\right).$$

By taking derivatives with respect to the primal variables $(c,\gamma) \in \mathbb{R}^n \times \mathbb{R}$, we obtain

$$\nabla_{(c,\gamma)}\mathcal{L}((c,\gamma),u) = \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \sum_{i=1}^m u_i \begin{bmatrix} -2a^i + 2c \\ -1 \end{bmatrix},$$
$$\nabla^2_{(c,\gamma)}\mathcal{L}((c,\gamma),u) = \sum_{i=1}^m u_i \begin{bmatrix} 2I & 0 \\ 0 & 0 \end{bmatrix},$$

where $I \in \mathbb{R}^{n\times n}$ denotes the identity matrix. Note that any direction $d \in \mathbb{R}^{n+1}$ orthogonal to the gradient of the objective function of $(\mathcal{P}(0))$ is of the form $d = [(d')^T, 0]^T$, where $d' \in \mathbb{R}^n$. Therefore, for any such direction d,

$$d^T \nabla^2_{(c,\gamma)}\mathcal{L}(c^*,\gamma^*,u^*)d = 2(d')^T d' = 2\|d\|^2,$$

since u^* lies on the unit simplex, which implies that Robinson's second-order sufficient condition is satisfied (see Definition 2.1 in [34]) by the optimal solution (c^*,γ^*) of $(\mathcal{P}(0))$ along with any dual optimal solution u^*. Therefore, by Theorem 4.2 in [34], there exists a dual optimal solution u^* and a positive constant ℓ such that

$$\|u^k - u^*\| \leq \ell\|z^k\| \leq \ell\sqrt{m}\Delta^2\delta_k \tag{15}$$

for all sufficiently small δ_k. Combining this inequality with (14), we obtain

$$\Phi(u^*) - \Phi(u^k) \leq m\ell\Delta^4(\delta_k)^2 \tag{16}$$

for all sufficiently small δ_k.

Suppose now that $\delta_k \leq 1/2$. Since $\Phi(u^k) \leq \Phi(u^*) \leq (1+\delta_k)\Phi(u^k) \leq (3/2)\Phi(u^k)$, it follows that

$$\Phi(u^k) \geq (2/3)\Phi(u^*) \geq (1/6)\Delta^2.$$

At each plus- or minus-iteration, by Lemma 4.1, we obtain

$$\Phi(u^{k+1}) \geq \Phi(u^k)\left(1 + \frac{\delta_k^2}{4(1+\delta_k)}\right) \geq \Phi(u^k) + \frac{\delta_k^2\Delta^2}{36}. \tag{17}$$

Combining (16) and (17), at each plus- or minus-iteration, we obtain

$$\Phi(u^*) - \Phi(u^{k+1}) \leq \Phi(u^*) - \Phi(u^k) - \frac{\delta_k^2 \Delta^2}{36} \leq \left(1 - \frac{1}{36m\ell\Delta^2}\right)(\Phi(u^*) - \Phi(u^k)) \tag{18}$$

for all sufficiently small δ_k. This establishes the linear convergence of Algorithm 4.1.

THEOREM 4.2. *Given $\mathcal{A} := \{a^1, \ldots, a^m\} \subset \mathbb{R}^n$, Algorithm* 4.1 *computes iterates u^k such that $\Phi(u^*) - \Phi(u^k)$ is nonincreasing. Asymptotically, this gap is decreased by at least a factor of* $(1 - 1/(36m\ell\Delta^2))$ *at each plus- or minus-iteration. There exist constants $\bar{\tau}$ and ϑ that depend on the input data such that Algorithm* 4.1 *computes a $(1+\epsilon)$-approximation to MEB$(\mathcal{A})$ in* $\bar{\tau} + \vartheta \log(1/\epsilon)$ *operations for* $\epsilon \in (0, 1)$.

Proof. Lemma 4.1 and the following discussions imply that $\Phi(u^*) - \Phi(u^k)$ is a nonincreasing sequence. The asymptotic linear convergence follows from (18). Therefore, we need only to establish the last statement.

Let $\tau := \max\{\tau_1, \tau^*\}$, where τ_1 is defined as in (9) and τ^* is the smallest value of k such that the inequality (15) is satisfied. After iteration τ, the sequence $\Phi(u^*) - \Phi(u^k)$ satisfies the relationship (18). By the termination criterion of Algorithm 4.1, it suffices to compute an iterate k_* such that $\Phi(u^{k_*}) \leq \Phi(u^*) \leq (1+\delta_{k_*})\Phi(u^{k_*}) \leq (1+\epsilon)^2\Phi(u^{k_*})$. This implies that the final iterate satisfies $\Phi(u^*) - \Phi(u^{k_*}) \leq [(1+\epsilon)^2 - 1]\Phi(u^{k_*})$. Since $\Phi(u^k) \geq (1/6)\Delta^2$ for all $k \geq \tau$, it follows that the termination criterion is satisfied if $\Phi(u^*) - \Phi(u^{k_*}) \leq (1/6)[(1+\epsilon)^2 - 1]\Delta^2$. By (18), $\Phi(u^*) - \Phi(u^{k+1}) \leq (1 - (1/\bar{\mu}))(\Phi(u^*) - \Phi(u^k)) \leq (1 - (1/\bar{\mu}))(\Delta^2 - (1/6)\Delta^2) = (5/6)(1 - (1/\bar{\mu}))\Delta^2$ at each plus- or minus-iteration for all $k \geq \tau$, where $\bar{\mu} := 36m\ell\Delta^2$. Therefore, once Algorithm 4.1 computes iterate τ, we have

$$\Phi(u^*) - \Phi(u^{\tau+\hat{k}}) \leq \frac{5}{6}\left(1 - \frac{1}{\bar{\mu}}\right)^{\hat{k}} \Delta^2$$

after $\hat{k}$ plus- or minus-iterations. Therefore, if

$$\frac{5}{6}\left(1 - \frac{1}{\bar{\mu}}\right)^{\hat{k}} \Delta^2 \leq \frac{1}{6}(2\epsilon + \epsilon^2)\Delta^2,$$

then the termination criterion is satisfied after $\hat{k}$ plus- or minus-iterations. It follows that $\hat{k}$ satisfies

$$\log 5 + \hat{k} \log\left(1 - \frac{1}{\bar{\mu}}\right) \leq \log \epsilon + \log(\epsilon + 2).$$

Using the inequality $\log(1+x) \leq x$ for all $x > -1$, a sufficient condition in order for the above inequality to be satisfied is given by

$$\log 5 - \frac{\hat{k}}{\bar{\mu}} \leq \log 2 + \log \epsilon,$$

which implies that $\tau' + \bar{\mu}\log(1/\epsilon)$ plus- or minus-iterations will suffice, where $\tau' = \bar{\mu}\log(5/2)$. By the argument following Lemma 4.1, we can double the iteration count and add two iterations to account for the drop-iterations, which completes the proof. ∎

We remark that Theorem 4.2 establishes a polynomial convergence result for Algorithm 4.1 even if ϵ is part of the input data. In addition, it implies that the convergence is "fast" once inequality (15) is satisfied. However, the bound on the number of iterations depends on the data as it is not known a priori when the linear convergence will kick in. As such, it does not provide a better global complexity bound than that of Theorem 4.1. Nevertheless, the results of this section will shed some light into the usually better practical performance of Algorithm 4.1 in section 6.

5. Extensions.

In this section, we establish that the algorithmic frameworks of sections 3 and 4 can be used to compute an approximation to the minimum enclosing ball of more general input sets. While the cost of each iteration of the corresponding algorithms may depend on the input set, the iteration complexity and the asymptotic size of the core set remain unchanged. Therefore, the existence of an ϵ-core set of size $O(1/\epsilon)$ extends to more general sets including those with uncountably many points.

In the first pass, skipped this section, but revisited it later...

We remark that the analysis of both of the algorithms heavily relies on the structure of the dual optimization formulation $(\mathcal{D})$ of the minimum enclosing ball problem of a finite set of points. In this section, we argue that the same algorithmic framework can be applied to much more general input sets with minor modifications. We employ similar arguments as in [43], where a Frank–Wolfe-type algorithm for the problem of computing the minimum-volume enclosing ellipsoid of a finite set of ellipsoids is studied. Given a possibly infinite set of points, the primal optimization formulation $(\mathcal{P}_2)$ can be extended to a semi-infinite optimization problem with a linear objective function and infinitely many convex quadratic constraints. The main idea is to approximate the given input set using only a carefully selected finite subset of points and then to refine this approximation by adding more points if necessary. This leads to an approximation of the primal formulation with only a finite number of constraints, and this approximation is refined by adding more constraints. In the dual formulation, we therefore start with a finite number of variables and add more variables if necessary.

Let $\mathcal{A} \subset \mathbb{R}^n$ be an arbitrary compact input set, and let us first consider Lemma 3.1, which establishes the quality of the initial feasible solution computed by each of the two algorithms. The initial working core set $\mathcal{X}_0$ provides the first approximation to the given input set with only two points. Let $\Phi_0(\cdot)$ denote the objective function of the dual formulation of the minimum enclosing ball problem for $\mathcal{X}_0$, and let $\gamma_{\mathcal{A}}$ denote the optimal value of the aforementioned semi-infinite primal formulation. The result of Lemma 3.1 continues to hold, since the proof relies on Lemma 2.2, which remains true for arbitrary compact input sets. The proof of Lemma 2.2 is based on the argument that an enclosing ball of smaller radius can be constructed by moving the center away from the half-space in the direction of the normal vector of the bounding hyperplane if the hypothesis of Lemma 2.2 is not satisfied by that half-space. Therefore, we still have $\Phi_0(u^0) \geq (1/3)\gamma_{\mathcal{A}}$, which implies that the quality of the initial solution is independent of the input set. initial radius ^2 >= (1/3) * optimal radius^2

(*) Use these in the report as an explanation of the Algorithm 3.1.

Similarly, let $\Phi_k(\cdot)$ denote the objective function of the dual formulation of the minimum enclosing ball problem for $\mathcal{X}_k \subset \mathcal{A}$. At iteration k in each algorithm, $\mathcal{X}_k$ provides the current finite approximation to $\mathcal{A}$. Let $c^k \in \mathbb{R}^n$ denote the current center. Each algorithm computes the furthest point in $\mathcal{A}$ from c^k. In Algorithm 3.1, $\mathcal{X}_{k+1}$ is obtained by adding this point to $\mathcal{X}_k$. Unless the furthest point in $\mathcal{A}$ already belongs to $\mathcal{X}_k$, the dual formulation for $\mathcal{X}_{k+1}$ differs from that for $\mathcal{X}_k$ in only one variable. Therefore, $[(u^k)^T, 0]^T$ is a feasible solution for the new dual formulation that satisfies $\Phi_{k+1}([(u^k)^T, 0]^T) = \Phi_k(u^k)$, which implies that the improvement in each iteration still obeys the relation given by Lemma 3.2, with γ^{k+1} replaced by $\Phi_{k+1}(u^{k+1})$ and γ^k by $\Phi_k(u^k)$. Note that the dimension of u^{k+1} is one more than that of u^k in this case. It follows that the upper bound on the number of iterations required by Algorithm 3.1 to achieve a prescribed accuracy as well as the bound on the core set remain unchanged for a general compact input set $\mathcal{A} \subset \mathbb{R}^n$.

skip this paragraph (Algorithm 4.1)

The preceding argument establishes the same improvement result at a plus-iteration of Algorithm 4.1 for a general input set $\mathcal{A}$. Since $\mathcal{X}_k$ is finite, the computation of the closest point in $\mathcal{X}_k$ is straightforward independently of the input set. At a minus-iteration, the dimension of the dual formulation remains the same. Therefore, Lemma 4.1 still applies. At a drop-iteration, we can reverse the argument employed at a plus-iteration, since the number of dual variables actually decreases in this case. We conclude that the iteration complexity of Algorithm 4.1 and the upper bound on the size of the core set also remain unchanged for a general compact input set $\mathcal{A} \subset \mathbb{R}^n$. On the other hand, our analysis that leads to the linear convergence of Algorithm 4.1 is not likely to be extended to more general input sets, since it explicitly relies on the stability results for nonlinear programming problems with a finite number of constraints.

skip also this

We give another perspective on the extension of the two algorithms to more general input sets. Let $\mathcal{X}_\eta \subset \mathcal{A}$ denote the finite set computed by either one of the two algorithms upon termination on a general input set $\mathcal{A}$. Then, each algorithm would geometrically behave exactly the same way on the input set $\mathcal{X}_\eta$ as it would on the original input set $\mathcal{A}$. However, the termination criterion is satisfied for the whole set $\mathcal{A}$. Clearly, the set $\mathcal{X}_\eta \subset \mathcal{A}$ is not known a priori and is sequentially generated by each algorithm. Furthermore, the cost of each iteration is likely to be higher for a general input set $\mathcal{A}$ in comparison with that for $\mathcal{X}_\eta$. Therefore, the main work involved in each algorithm is the extraction of the finite set $\mathcal{X}_\eta$ from $\mathcal{A}$.

In order to transform this conceptual algorithmic framework into a practical algorithm, we need to ensure that each operation required by either algorithm can be carried out efficiently for a given input set. Note that both of the algorithms in this paper compute the initial feasible solution in a similar fashion. This computation entails finding the furthest point in the input set from a fixed point. In addition, similar furthest point computations are performed at each iteration of both of the algorithms. Therefore, the extent of input sets which are amenable to these algorithms highly depends on the efficiency with which such computations can be performed.

We now specify several input sets for which similar algorithmic frameworks can be applied.

and skip this section as well :)

5.1. Set of balls. Let $\mathcal{A} = \{\mathcal{B}_1, \ldots, \mathcal{B}_m\} \subset \mathbb{R}^n$ be a set of m balls. Given $\mathcal{B}_{c,\rho}$ and $x \in \mathbb{R}^n$, the furthest point in $\mathcal{B}_{c,\rho}$ from x is given by $x^* = c + \rho(c - x)/\|c - x\|$, which can be computed in $O(n)$ operations. Therefore, each iteration of Algorithm 3.1 still requires $O(mn)$ operations, which implies that Algorithm 3.1 computes a $(1+\epsilon)$-approximation to MEB($\mathcal{A}$) in $O(mn/\epsilon)$ operations and returns an ϵ-core set of size $O(1/\epsilon)$. In addition to computing the furthest point at each iteration, Algorithm 4.1 also requires the computation of the closest point in a finite set. The size of this set is bounded above by $O(1/\epsilon)$, which implies that each iteration can be performed in $O(mn + n/\epsilon)$ operations. Therefore, Algorithm 4.1 can compute a $(1+\epsilon)$-approximation to MEB($\mathcal{A}$) in $O(mn/\epsilon + n/\epsilon^2)$ operations and returns an ϵ-core set of size $O(1/\epsilon)$.

5.2. Set of ellipsoids. Let $\mathcal{A} = \{\mathcal{E}_1, \ldots, \mathcal{E}_m\} \subset \mathbb{R}^n$ be a set of m ellipsoids given by $\mathcal{E}_i := \{x \in \mathbb{R}^n : (x - c^i)^T Q^i (x - c^i) \leq 1\}$, where $c^i \in \mathbb{R}^n$ and $Q^i \in \mathbb{R}^{n\times n}$ is symmetric and positive definite for $i = 1, \ldots, m$. The furthest point in an ellipsoid from a given point can be computed using a tight semidefinite programming relaxation with a fixed number of constraints in $O(n^{O(1)})$ operations in the real number model of computation [43], where $O(1)$ denotes a universal constant greater than three. Therefore, Algorithm 3.1 computes a $(1+\epsilon)$-approximation to MEB($\mathcal{A}$) in $O(mn^{O(1)}/\epsilon)$ operations and returns an ϵ-core set of size $O(1/\epsilon)$. Similarly, Algorithm 4.1 can compute a $(1+\epsilon)$-approximation to MEB($\mathcal{A}$) in $O(mn^{O(1)}/\epsilon + n^{O(1)}/\epsilon^2)$ operations and returns an ϵ-core set of size $O(1/\epsilon)$.

5.3. Set of half ellipsoids. $\mathcal{H}$ is said to be a half ellipsoid if it is given by the intersection of an ellipsoid with a half-space. Let $\mathcal{A} = \{\mathcal{H}_1, \ldots, \mathcal{H}_m\}$, where $\mathcal{H}_i := \{x \in \mathbb{R}^n : (x-c^i)^T Q^i (x-c^i) \leq 1, \quad (f^i)^T x \leq \omega^i\}$, where $c^i \in \mathbb{R}^n, f^i \in \mathbb{R}^n, \omega^i \in \mathbb{R}$, and $Q^i \in \mathbb{R}^{n\times n}$ is symmetric and positive definite for $i = 1, \ldots, m$. Sturm and Zhang [36] established that the maximization of any quadratic function over a half ellipsoid can be cast as a semidefinite programming problem with a fixed number of constraints similarly to quadratic optimization over an ellipsoid. Therefore, the asymptotic overall complexity bounds of Algorithms 3.1 and 4.1 are identical to those for the case of a set of ellipsoids. In particular, both algorithms return an ϵ-core set of size $O(1/\epsilon)$.

5.4. Set of intersections of a pair of similar ellipsoids. Two n-dimensional ellipsoids $\mathcal{E}_1$ and $\mathcal{E}_2$ are said to be similar if they both admit a representation using the same semidefinite matrix. This implies that the length and the alignment of the corresponding axes are the same. Let $\mathcal{A} = \{\mathcal{T}_1, \ldots, \mathcal{T}_m\}$, where $\mathcal{T}_i := \{x \in \mathbb{R}^n : (x-c^i)^T Q^i (x-c^i) \leq 1, \quad (x-h^i)^T Q^i (x-h^i) \leq 1\}$, where $c^i \in \mathbb{R}^n, h^i \in \mathbb{R}^n$, and $Q^i \in \mathbb{R}^{n\times n}$ is symmetric and positive definite for $i = 1, \ldots, m$. It follows from the results of [36] that any quadratic optimization problem over the intersection of a pair of similar ellipsoids can be decomposed into two quadratic optimization problems over two half ellipsoids. Therefore, the asymptotic complexity bounds of Algorithms 3.1 and 4.1 are identical to those for the case of a set of half ellipsoids. Similarly, both algorithms return an ϵ-core set of size $O(1/\epsilon)$.

skip until here

5.5. Further extensions. We have described several classes of more general input sets for which an approximate minimum enclosing ball can be computed in polynomial time (for fixed ϵ) using the appropriate extensions of Algorithms 3.1 and 4.1. Obviously, the results can be extended to input sets that are composed of a combination of elements from each of the above classes. In particular, it is remarkable that the existence of an ϵ-core set of size $O(1/\epsilon)$ extends to much more general classes of input sets including those with uncountably many points.

Similarly to the discussion in [43], the extent of input sets to which similar algorithmic frameworks can be applied largely depends on the efficiency of the furthest point computation required at each iteration of each of the two algorithms. It is well known that the maximization of a convex quadratic function over certain sets (such as polytopes defined by inequalities) is computationally intractable. Therefore, our algorithmic framework does not yield a polynomial-time algorithm for an input set of polytopes. In summary, the discovery of polynomial-time routines for quadratic optimization over other classes of input sets may lead to further efficient generalizations of our algorithms.

However, for us the constraints on the objective function form a polytope. The input is not a polytope.

6. Computational results. In this section, we report the results of our computational experiments. We implemented Algorithms 3.1 and 4.1 in MATLAB. For the purposes of comparison, we also implemented the first-order algorithm of Bădoiu and Clarkson [2] (henceforth the BC algorithm). Their simple algorithm starts by setting any arbitrary point $a^i \in \mathcal{A}$ as the initial center c^1. At iteration k, let a^{j_k} denote the furthest point from c^k, $k = 1, 2, \ldots$. The center is updated according to the following

TABLE 1
Computational results on instances with $m \gg n$ ($\epsilon = 10^{-3}$).

n	#points m	Time A1	Time A2	Time BC	Core set size A1	Core set size A2	Core set size BC	Iterations A1	Iterations A2	Iterations BC
10	500	0.06	0.03	0.12	4.2	3.9	5.2	168.7	44.5	435.5
10	1000	0.15	0.03	0.14	4.6	3.8	5.4	330.7	41.6	344.4
20	5000	1.7	0.36	3.11	5.9	5.2	7	246.8	46	464.2
20	10000	4.46	0.58	4.65	4.9	4.1	5.8	319.2	36.3	334.4
30	30000	27	6.45	24.59	8.6	6.8	9.1	446.4	103.6	409
50	50000	71.62	16.87	68.78	10.5	9.5	11.8	429.8	98.4	415.1
100	100000	287.99	77.74	268.11	15.9	14.5	16.6	451.7	119	422.6

(*) This is a good table to have in our report.

core set size << number of points

relation:

$$c^{k+1} = [1 - 1/(k+1)]c^k + [1/(k+1)]a^{j_k}, \quad k = 1, 2, \ldots .$$

Bădoiu and Clarkson establish that $1/\epsilon^2$ such updates suffice in order to obtain a $(1+\epsilon)$-approximation to MEB($\mathcal{A}$). Note that each iteration requires $O(mn)$ operations, which yields an overall complexity bound of $O(mn/\epsilon^2)$.

Similarly to Algorithms 3.1 and 4.1, it is easy to verify that the BC algorithm also generates a sequence of feasible solutions for the dual formulation of the minimum enclosing ball problem. Therefore, in order to have a fair and meaningful comparison, we employed the same termination criterion that we used for Algorithms 3.1 and 4.1 rather than running the BC update for $1/\epsilon^2$ times.

In contrast with Algorithms 3.1 and 4.1, the objective functions evaluated at the iterates generated by the BC algorithm are not monotonically increasing in general. Therefore, the analysis of the BC algorithm uses entirely different tools [2].

The computational experiments were carried out on a Pentium IV processor with a clock speed of 2.80 GHz and 512 MB RAM running under Linux. We used MATLAB version 7.3.0.298 (R2006b) in our experiments.

Configuration used for testing.

We used three data sets in our experiments. The first data set is restricted to instances with $m \gg n$ and was randomly generated as in [1], with sizes (n, m) varying from $(10, 500)$ to $(100, 100000)$. For each fixed (n, m), ten different data sets were generated, and the results are reported in terms of the averages over these data sets in Table 1, which is divided into four sets of columns. The first set of columns reports the size (n, m). The next three sets of columns present the CPU time, core set size, and the number of iterations, respectively. Each one of these three sets is further divided into three columns labeled A1, A2, and BC corresponding to Algorithm 3.1, Algorithm 4.1, and the BC algorithm, respectively. In all of our experiments, we set $\epsilon = 10^{-3}$.

points >> dimension of points

As illustrated by Table 1, each of the three algorithms is capable of quickly computing an approximation to the minimum enclosing ball of the given input set. In particular, all three algorithms terminated under eight minutes even on the largest instances. In terms of CPU time, Algorithm 4.1 has significantly better performance than Algorithm 3.1 and the BC algorithm, both of which have similar running times. All three algorithms computed very small core sets of similar sizes. Algorithm 4.1 always returned the smallest core sets for each input set. The core sets computed by Algorithm 3.1 and the BC algorithm have similar sizes with the former being slightly better than the latter. In terms of the number of iterations, Algorithm 4.1 once again significantly outperforms the other two algorithms. Unlike Algorithm 3.1, the number

TABLE 2
Computational results on instances with $n \gg m$ ($\epsilon = 10^{-3}$).

	#points	Time			Core set size			Iterations		
n	m	A1	A2	BC	A1	A2	BC	A1	A2	BC
10000	100	7.56	7.62	29.9	90.6	90.4	90.8	117.4	118.2	476.6
10000	1000	149.39	148.16	321.25	198.4	197	202	241	238.8	524.2
25000	1000	541.47	539.06	957.02	266.6	265.6	272.4	303.2	301.8	541.2

The algorithms are much slower now.

core set size < #points
all algorithms produce similar sized coresets

TABLE 3
Vertices of the unit simplex ($m = n = 1000$).

varying epsilon

	Time			Core set size			Iterations		
ϵ	A1	A2	BC	A1	A2	BC	A1	A2	BC
1	.24	.25	.19	2	2	2	0	0	1
.1	.83	.83	.82	11	11	11	9	9	10
.01	6.57	6.58	7.27	101	101	101	99	99	100
.001	63.89	64.07	71.36	1000	1000	1000	998	998	999

core set size <= # points

of iterations of the BC algorithm seems to be independent of the dimensions of the input set.

A close examination of Table 1 reveals that Algorithm 4.1 resulted in reductions of 73% to 88% in terms of running time and of 74% to 90% in terms of the number of iterations in comparison with the other two algorithms. These results seem to indicate that the linear convergence of Algorithm 4.1 may be responsible for the improved performance. Furthermore, due to allowing points to be dropped from the working core set, the sizes of the core sets computed by Algorithm 4.1 are about 10% to 30% smaller than those returned by the other two algorithms.

The second data set consists of instances with $n \gg m$. In particular, we generated random instances with (n, m) varying from $(10000, 100)$ to $(25000, 1000)$. The averaged results are presented in Table 2, which is organized similarly to Table 1. The results indicate that all three algorithms compute core sets of similar sizes. Algorithm 3.1 and Algorithm 4.1 exhibit similar performances in terms of running time and the number of iterations due to the fact that "away" steps are performed relatively infrequently on such instances. On the other hand, the running time and the number of iterations of the BC algorithm are considerably larger than either of our two algorithms. Once again, note that the number of iterations of the BC algorithm seems to be relatively insensitive to m and n, which suggests a stronger relationship with $1/\epsilon$ in comparison with our algorithms.

The final data set we considered is the vertices of the unit simplex. Bădoiu and Clarkson [3] establish a tight upper bound of $\lceil 1/\epsilon \rceil$ on the size of the core set for such an input set under the assumption that $n \geq \lfloor 1/\epsilon \rfloor$. In an attempt to assess the performances of the three algorithms on such a data set, we considered the vertices of the unit simplex with $n = 1000$ using $\epsilon \in \{1, .1, .01, .001\}$. The results of this experiment are presented in Table 3, which is organized similarly to Table 1.

As illustrated by Table 3, all three algorithms have similar performances on the vertices of the unit simplex in $\mathbb{R}^n$, with $n = 1000$. Note that both the size of the core set and the number of iterations grow proportionally to $1/\epsilon$. These results are in agreement with the tight core set bound of [3]. This example illustrates that the asymptotic bounds on the core set size and the number of iterations for Algorithms 3.1 and 4.1, in general, cannot be improved. However, all three algorithms computed the exact minimum enclosing ball for $\epsilon = 10^{-3}$ (and for any $\epsilon \geq 10^{-3}$). Therefore, this example illustrates that the upper bound of $\lceil 1/\epsilon \rceil$ on the size of the core set is no longer tight for $n \leq \lfloor 1/\epsilon \rfloor$.

We do not compare our algorithms with other exact or approximate algorithms, since such computational studies have been performed in earlier literature. For instance, it is well known that the minimum enclosing ball problem can be formulated as an instance of second-order cone programming and interior-point methods can achieve very high accuracy (e.g., 10^{-8}) in small- and medium-scale instances. However, each iteration requires the computation and factorization of an $(n+1) \times (n+1)$ matrix, which can be performed in $O(n^3)$ and $O(mn^2)$ operations, respectively [24]. Therefore, such an approach is not computationally feasible for large instances as illustrated by the results of [42, 44]. Similarly, exact algorithms [16] perform well on small- and medium-scale instances, but the performance degrades significantly for large-scale instances [24]. Since our focus is on applications with large-scale instances in which a moderate accuracy suffices, our computational results indicate that our algorithms are capable of solving such instances in a reasonable amount of time.

7. Concluding remarks.

In this paper, we proposed and analyzed two algorithms that compute an approximation to the minimum enclosing ball of a given finite set of points. Both algorithms exploit the special structure of the dual formulation of the problem and can geometrically be viewed as generating a sequence of trial balls until a ball with desired properties is computed. Each of the two algorithms is especially well-suited for the large-scale instances of the minimum enclosing ball problem for which a moderate approximation suffices. Both algorithms can compute a small core set whose size depends only on the approximation parameter. The second algorithm asymptotically exhibits linear convergence, which further contributes to its efficiency. We have discussed how our algorithms can be extended to more general input sets without sacrificing the iteration complexity and hence the size of the core set. In particular, we established that several more general classes of input sets admit small and finite core sets. Our computational experiments reveal that both of our algorithms are capable of quickly computing a good approximation to the minimum enclosing ball of a finite set of points. Algorithm 4.1, which is obtained by incorporating "away" steps into Algorithm 3.1, seems to exhibit a significantly better performance than other first-order algorithms. The sizes of the core sets computed by our algorithms are usually fairly small. The example that consists of the vertices of the unit simplex illustrates that our analysis, in general, cannot be improved.

While the discovery of efficient algorithms such as interior-point methods revolutionized convex optimization, the computational cost of each iteration of such algorithms quickly becomes prohibitive as the size of the problems increases. Therefore, it seems desirable to design specialized algorithms for large-scale problems that exploit the underlying special structure of the problem. We have developed two such algorithms for the minimum enclosing ball problem in this paper. We intend to continue our work on developing specialized algorithms for other classes of large-scale structured optimization problems in the near future.

Acknowledgments. I gratefully acknowledge the insightful comments and suggestions by the Associate Editor and two anonymous referees, which contributed significantly to the improvement of the manuscript. In particular, section 4.2 was added based on the comments of an anonymous referee, and the alternative perspective on the extension of the algorithms to more general input sets in section 5 was suggested by another anonymous referee.

REFERENCES

[1] D. Ahipaşaoğlu, P. Sun, and M. J. Todd, *Linear convergence of a modified Frank-Wolfe algorithm for computing minimum-volume enclosing ellipsoids*, Optim. Methods Softw., 23 (2008), pp. 5–19.

[2] M. Bădoiu and K. L. Clarkson, *Smaller core-sets for balls*, in Proceedings of the 14th Annual Symposium on Discrete Algorithms, 2003, pp. 801–802.

[3] M. Bădoiu and K. L. Clarkson, *Optimal core-sets for balls*, Comput. Geom. Theory Appl., 40 (2008), pp. 14–22.

[4] M. Bădoiu, S. Har-Peled, and P. Indyk, *Approximate clustering via core-sets*, in Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 2002, pp. 250–257.

[5] L. Blum, M. Shub, and S. Smale, *On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions, and universal machines*, Bull. Amer. Math. Society (N.S.), 21 (1989), pp. 1–46.

[6] L. M. Blumenthal and G. E. Wahlin, *On the spherical surface of smallest radius enclosing a bounded subset of n-dimensional Euclidean space*, Bull. Amer. Math. Soc. (N.S.), 47 (1941), pp. 771–777.

[7] R. K. Chakraborty and P. K. Chaudhuri, *Note on geometrical solutions for some minimax location problems*, Transportation Sci., 15 (1981), pp. 164–166.

[8] J. A. Chatelon, D. W. Hearn, and T. J. Lowe, *A subgradient algorithm for certain minimax and minisum location problems*, Math. Program., 15 (1978), pp. 130–145.

[9] G. Chrystal, *On the problem to construct the minimum circle enclosing n given points in the plane*, in Proc. Edinb. Math. Soc., 3 (1885), pp. 30–33.

[10] K. L. Clarkson, *Coresets, sparse greedy approximation and the Frank-Wolfe algorithm*, in Proceedings of the 19th Annual Symposium on Discrete Algorithms, 2008, pp. 922–931.

[11] D. J. Elzinga and D. W. Hearn, *The minimum covering sphere problem*, Management Sci., 19 (1972), pp. 96–104.

[12] K. Fischer and B. Gärtner, *The smallest enclosing ball of balls: Combinatorial structure and algorithms*, Internat. J. Comput. Geom. Appl., 14 (2004), pp. 341–378.

[13] K. Fischer, B. Gärtner, and M. Kutz, *Fast smallest-enclosing-ball computation in high dimensions*, in Algorithms–ESA, Lect. Notes in Comput. Sci. 2832, G. Di Battista and U. Zwick. eds., Springer, Berlin/Heidelberg, 2003, pp. 630–641.

[14] R. L. Francis, *Some aspects of a minimax location problem*, Oper. Res., 15 (1967), pp. 1163–1169.

[15] M. Frank and P. Wolfe, *An algorithm for quadratic programming*, Naval Res. Logist., 3 (1956), pp. 95–110.

[16] B. Gärtner, *Fast and robust smallest enclosing balls*, in Proceedings of the 7th Annual European Symposium on Algorithms (ESA), Lect. Notes in Comput. Sci. 1643, J. Nesetril, ed., Springer, New York, 1999, pp. 325–338.

[17] 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, 2000, pp. 110–118.

[18] A. Goel, P. Indyk, and K. R. Varadarajan, *Reductions among high-dimensional proximity problems*, in Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms, 2001, pp. 769–778.

[19] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Algorithms Combin. 2, Springer, New York, 1988.

[20] J. Guélat and P. Marcotte, *Some comments on Wolfe's away steps*, Math. Program., 35 (1986), pp. 110–119.

[21] C. H. Guo, M. Y. Lu, J. T. Sun, and Y. C. Lu, *A new algorithm for computing the minimal enclosing sphere in feature space*, in Fuzzy Systems and Knowledge Discovery, Lect. Notes in Comput. Sci. 3614, L. Wang and Y. Jin, eds., Springer, Berlin/Heidelberg, 2005, pp. 196–204.

[22] D. W. Hearn and J. Vijay, *Efficient algorithms for the (weighted) minimum circle problem*, Oper. Res., 30 (1981), pp. 777–795.

[23] S. K. Jacobsen, *An algorithm for the minimax Weber problem*, European J. Oper. Res., 6 (1981), pp. 144–148.

[24] P. Kumar, J. S. B. Mitchell, and E. A. Yildirim, *Approximate minimum enclosing balls in high dimensions using core-sets*, ACM J. Exp. Algorithmics, 8 (2003), article no. 1.1.

[25] C. L. Lawson, *The smallest covering cone or sphere*, SIAM Rev., 7 (1965), pp. 415–416.

[26] J. Matoušek and B. Gärtner, *Understanding and Using Linear Programming*, Universitext, Springer, New York, 2006.

[27] N. Megiddo, *Linear time algorithms for linear programming in $\mathbb{R}^3$ and related problems*, SIAM J. Comput., 12 (1983), pp. 759–776.

[28] K. P. K. Nair and R. Chandrasekaran, *Optimal location of a single service center of certain types*, Naval Res. Logist., 18 (1971), pp. 503–510.

[29] F. Nielsen and R. Nock, *Approximating smallest enclosing balls*, in Computational Science and Its Applications, Lect. Notes in Comput. Sci. 3045, A. Laganá, M. Gavrilov, V. Kumar, Y. Mun, C. Tan, and O. Gervasi, eds., Springer, Berlin/Heidelberg, 2004, pp. 147–157.

[30] F. Nielsen and R. Nock, *A fast deterministic smallest enclosing disk approximation algorithm*, Inform. Process. Lett., 93 (2005), pp. 263–268.

[31] B. J. Oommen, *An efficient geometric solution to the minimum spanning circle problem*, Oper. Res., 35 (1987), pp. 80–86.

[32] S. H. Pan and X. S. Li, *An efficient algorithm for the smallest enclosing ball problem in high dimensions*, Appl. Math. Comput., 172 (2006), pp. 49–61.

[33] R. Panigrahy, *Minimum Enclosing Polytope in High Dimensions*, manuscript, 2006.

[34] S. M. Robinson, *Generalized equations and their solutions, part* ii: *Applications to nonlinear programming*, Math. Program. Study, 19 (1982), pp. 200–221.

[35] M. I. Shamos, *Computational Geometry*, Ph.D. thesis, Yale University, New Haven, CT, 1978.

[36] J. F. Sturm and S. Z. Zhang, *On cones of nonnegative quadratic functions*, Math. Oper. Res., 28 (2003), pp. 246–267.

[37] J. J. Sylvester, *A question in the geometry of situation*, Q. J. Pure Appl. Math., 1 (1857).

[38] J. J. Sylvester, *On Poncelet's approximate linear valuation of Surd forms*, Philos. Mag., 20 (1860), pp. 203–222. Fourth Series.

[39] M. J. Todd and E. A. Yildirim, *On Khachiyan's algorithm for the computation of minimum volume enclosing ellipsoids*, Discrete Appl. Math., 155 (2007), pp. 1731–1744.

[40] E. Welzl, *Smallest enclosing disks (balls and ellipsoids)*, in New Results and New Trends in Computer Science, Lect. Notes in Comput. Sci. 555, H. Maurer, ed., Springer-Verlag, 1991, pp. 359–370.

[41] P. Wolfe, *Convergence theory in nonlinear programming*, in Integer and Nonlinear Programming, J. Abadie, ed., North-Holland, Amsterdam, 1970, pp. 1–36.

[42] S. Xu, R. M. Freund, and J. Sun, *Solution methodologies for the smallest enclosing circle problem*, Comput. Optim. Appl., 25 (2003), pp. 283–292.

[43] E. A. Yildirim, *On the minimum volume covering ellipsoid of ellipsoids*, SIAM J. Optim., 17 (2006), pp. 621–641.

[44] G. Zhou, K. C. Toh, and B. Sun, *Efficient algorithms for the smallest enclosing ball problem*, Comput. Optim. Appl., 30 (2005), pp. 147–160.