Active Set Strategies for the Computation of Minimum-Volume Enclosing Ellipsoids

Linus Källberg* Daniel Andrén*

Abstract

We describe and evaluate several variants of an active set algorithm for the problem of computing a $(1+\epsilon)$-approximation to the minimum-volume ellipsoid enclosing a given point set. The general approach is to run an existing algorithm repeatedly on smaller subsets of the points, and thereby achieve improved solution times compared to solving the whole problem directly. As the underlying algorithm, we use that of Todd and Yıldırım, which belongs to a group of algorithms based on the first-order Frank–Wolfe method. We propose multiple strategies to choose a new active set in each iteration, including an improved version of an existing strategy by Sun and Freund. In addition, we develop a variation of the elimination heuristic by Harman and Pronzato, that eliminates input points more aggressively in each iteration and then checks correctness of the solution before returning it. When used to $(1+10^{-6})$-approximate the minimum-volume ellipsoid enclosing sets of 10^6 points in 2 to 25 dimensions, the proposed techniques generate speedups up to 70× compared to our baseline.

1 Introduction

Let $\mathcal{P} \coloneqq \{p_1, \dots, p_n\} \subset \mathbb{R}^d$ be a given set of points with affine hull $\mathbb{R}^d$. An enclosing ellipsoid of $\mathcal{P}$ is defined by a center $c \in \mathbb{R}^d$ and a symmetric positive-definite matrix $Q \in \mathbb{R}^{d\times d}$ as

$$\mathcal{E}_{Q,c} \coloneqq \{x \in \mathbb{R}^d : (x-c)^{\mathrm{T}} Q(x-c) \leq 1\}$$

such that $\mathcal{E}_{Q,c} \supset \mathcal{P}$. The volume of $\mathcal{E}_{Q,c}$ is given by

$$\operatorname{vol} \mathcal{E}_{Q,c} = \zeta \det Q^{-1/2},$$

where ζ is the volume of the unit ball in $\mathbb{R}^d$. We are concerned with the problem of finding the minimum-volume enclosing ellipsoid (MVEE) of the given point set $\mathcal{P}$, $\operatorname{MVEE}(\mathcal{P})$. This ellipsoid, also known as the Löwner–John ellipsoid, has utility in diverse applications such as collision detection [8], classification [22], convex optimization [20], and optimal experimental design [23]. Specifically, we are concerned with computing a $(1+\epsilon)$-approximation to the MVEE, which is an enclosing ellipsoid $\mathcal{E}$ satisfying

$$\operatorname{vol} \mathcal{E} \leq (1+\epsilon) \operatorname{vol} \operatorname{MVEE}(\mathcal{P}).$$

A closely related problem is that of computing an $(1+\eta)d$-rounding of the convex hull of $\mathcal{P}$, $\operatorname{conv}(\mathcal{P})$, which is an ellipsoid $\mathcal{E}$ satisfying

$$((1+\eta)d)^{-1}\mathcal{E} \subset \operatorname{conv}(\mathcal{P}) \subset \mathcal{E},$$

where the left-hand side denotes the ellipsoid $\mathcal{E}$ scaled about its center by $((1+\eta)d)^{-1}$. The exact MVEE of $\mathcal{P}$ provides a d-rounding of $\operatorname{conv}(\mathcal{P})$ in the general case, and a $\sqrt{d}$-rounding if $\mathcal{P}$ is centrally symmetric [14].

The main focus of this paper is on a family of first-order algorithms for the MVEE problem that can be derived from the more general Frank–Wolfe method [11]. This includes some early algorithms for the equivalent D-optimal design problem [23], namely, the ones proposed by Wynn [27], Fedorov [10], Atwood [3], and Böhning [7]. It further includes the algorithms by Khachiyan [16], Kumar and Yıldırım [17], Todd and Yıldırım [25], and Cong et al. [9], which compute a $(1+\epsilon)$-approximation of the MVEE, that is at the same time a $(1+\eta)d$-rounding with $\eta = (1+\epsilon)^{2/(d+1)} - 1$.

We study the combining of these algorithms with an *active set* strategy to improve their speed in practice. The basic idea is to invoke the underlying algorithm multiple times on smaller subsets of the input data as opposed to running it once on the full problem. The returned solution from each step informs the selection of the next active set, until the solution to the current subproblem provides a $((1+\epsilon)$-approximate) solution to the original problem. The rationale of such an approach is that only a limited

*Mälardalen University, Sweden, linus.kallberg@mdh.se, daniel.andren@mdh.se

number of points in $\mathcal{P}$ are essential to compute or approximate MVEE($\mathcal{P}$). Indeed, it was shown by John [14] that MVEE($\mathcal{P}$) is determined by at most $d(d+3)/2$ points on its boundary. The concept of *core-sets* also builds on the idea of finding a small subset of relevant points that can be used to approximate the solution to within a guaranteed accuracy. In the context of the MVEE problem, $\mathcal{K} \subseteq \mathcal{P}$ is called an *ϵ-core-set* if there exists a scale factor s such that $s\,\mathrm{MVEE}(\mathcal{K}) \supset \mathcal{P}$ and $\mathrm{vol}\, s\,\mathrm{MVEE}(\mathcal{K}) \leq (1+\epsilon)\,\mathrm{MVEE}(\mathcal{P})$.

A similar active set approach to the MVEE problem was previously developed by Sun and Freund as a complement to their path-following Newton method [21]. Since each Newton step requires forming and factorizing an $n \times n$ matrix, such an active set scheme is necessary when n is large to reduce the number of points fed to the algorithm. In the computational study included in [2], Ahipaşaoğlu et al. found this algorithm to be up to 12 times faster than the aforementioned first-order algorithm by Todd and Yıldırım when $n \gg d$. The latter algorithm, on the other hand, is able to handle problem instances in higher dimension d due to its lower memory requirements. Combining an active set strategy with this type of first-order algorithm, as is the goal of this paper, therefore appears promising.

In this paper, we evaluate different strategies for selecting the points to be added to the active set in each step. This includes the straightforward approach of selecting the C farthest points from the current working ellipsoid, for some constant C, as well as an orthant scanning strategy previously developed for the related minimum enclosing ball (MEB) problem [18]. We further propose a modification of the strategy by Sun and Freund, which attempts to increase the spread of the selected points around the ellipsoid. Finally, we introduce a new elimination heuristic, inspired by the elimination heuristic by Harman and Pronzato [13] as well as similar heuristics for the MEB problem [1, 15], that removes points more aggressively from the input and then checks correctness before returning the final solution. We remark that all the techniques discussed in this paper are heuristics, in the sense that they come without theoretical guarantees of improved running times. They might, in fact, increase the theoretical worst-case time complexity compared to the underlying algorithm. However, it should be emphasized that all of the discussed methods are guaranteed to return a $(1+\epsilon)$-approximation of the MVEE, and the final active set can easily be shown to be an ϵ-core-set.

In the next section, the MVEE problem is stated in more detail, and the considered algorithms are introduced. In Section 3, the various active strategies included in our evaluation are described. Then in Section 4 our results are presented. Section 5 concludes the paper with a discussion.

2 Minimum-Volume Enclosing Ellipsoids

If the point set $\mathcal{P}$ is centrally symmetric, i.e., $\mathcal{P} = -\mathcal{P}$, then the problem simplifies to that of finding the minimum-volume ellipsoid *centered at the origin* that contains $\mathcal{P}$, because this ellipsoid is known to also be the true MVEE. To reap the same benefit also in the general case, $\mathcal{P}$ can be "lifted" to the $(d+1)$-dimensional centrally symmetric set

$$\widehat{\mathcal{P}} \coloneqq \{\pm\widehat{p}_i : i = 1, \ldots, n\}, \text{ where } \widehat{p}_i \coloneqq (p_i^{\mathrm{T}}, 1)^{\mathrm{T}}.$$

Although this trick appears to double the number of points entered into the algorithm, the mirror copy $-\widehat{p}_i$ of each $\widehat{p}_i$ does not need to be represented explicitly, since the centered MVEE of the points $\widehat{p}_i$ is guaranteed to enclose also these points. If an ellipsoid $\widehat{\mathcal{E}}$ is the $(d+1)$-dimensional MVEE of $\widehat{\mathcal{P}}$, or a $(1+\epsilon)$-approximation of it, then the d-dimensional MVEE of $\mathcal{P}$ (or a $(1+\epsilon)$-approximation of it) is given by the intersection of $\widehat{\mathcal{E}}$ and the hyperplane $\{(x^{\mathrm{T}}, 1)^{\mathrm{T}} : x \in \mathbb{R}^d\}$ [23].

The algorithms considered here compute a sequence of feasible solutions to the following formulation of the centered MVEE problem, which is equivalent to the D-optimal design problem:

$$\begin{aligned} \text{(D)} \quad \text{maximize}_u \quad & \ln\det M(u)^{-1} \\ \text{subject to} \quad & \sum_{i=1}^{n} u_i = 1, \\ & u_i \geq 0 \quad \text{for } i = 1, \ldots, n, \end{aligned}$$

where $u \coloneqq (u_1, \ldots, u_n)^{\mathrm{T}} \in \mathbb{R}^n$ and

$$M \coloneqq M(u) \coloneqq \Big((d+1)\sum_{i=1}^{n} u_i \widehat{p}_i \widehat{p}_i^{\mathrm{T}}\Big)^{-1}.$$

Any feasible but suboptimal solution to problem (D) translates to a trial ellipsoid $\mathcal{E}_{M,0}$ that is an underapproximation of MVEE($\widehat{\mathcal{P}}$), which means that

$\mathcal{E}_{M,0} \not\supseteq \widehat{\mathcal{P}}$ and $\operatorname{vol}\mathcal{E}_{M,0} < \operatorname{vol}\mathrm{MVEE}(\widehat{\mathcal{P}})$. An optimal solution $u = u^*$ satisfies, for $i = 1, \ldots, n$,

$$\widehat{p}_i^{\mathrm{T}} M(u^*)\widehat{p}_i \leq 1, \tag{1}$$

$$u_i^* > 0 \text{ implies } \widehat{p}_i^{\mathrm{T}} M(u^*)\widehat{p}_i = 1. \tag{2}$$

Geometrically, (1) states that the distance from the center of the MVEE to any point in $\widehat{\mathcal{P}}$ is at most 1 in the ellipsoidal norm, and (2) states that if $u_i^* > 0$ then $\widehat{p}_i$ lies on the boundary of the MVEE.

When the goal is to approximate the MVEE, it is of interest to find a feasible solution $u = \tilde{u}$ that satisfies the following *approximate* versions of these conditions:

$$\widehat{p}_i^{\mathrm{T}} M(\tilde{u})\widehat{p}_i \leq 1 + \eta, \tag{3}$$

$$\tilde{u}_i > 0 \text{ implies } \widehat{p}_i^{\mathrm{T}} M(\tilde{u})\widehat{p}_i \geq 1 - \eta, \tag{4}$$

for $i = 1, \ldots, n$. The resulting ellipsoid $\mathcal{E}_{M,0}$ then satisfies $\sqrt{1+\eta}\mathcal{E}_{M,0} \supset \widehat{\mathcal{P}}$. Furthermore, if $\eta = (1+\epsilon)^{2/(d+1)} - 1$ is used then $\operatorname{vol}\sqrt{1+\eta}\mathcal{E}_{M,0} \leq (1+\epsilon)\operatorname{vol}\mathrm{MVEE}(\widehat{\mathcal{P}})$.

The d-dimensional counterpart $\mathcal{E}_{Q,c}$ of $\mathcal{E}_{M,0}$ is given by

$$Q \coloneqq Q(u) \coloneqq \frac{d+1}{d} M(u)_{1:d,1:d},$$

$$c \coloneqq c(u) \coloneqq \sum_{i=1}^{n} u_i p_i,$$

where $1{:}d$ denotes the matrix indices 1 through d inclusive. It can be shown that $\mathcal{E}_{Q,c}$ corresponding to $u = \tilde{u}$ satisfies $\sqrt{1+\eta'}\mathcal{E}_{Q,c} \supset \mathcal{P}$ and $\operatorname{vol}\sqrt{1+\eta'}\mathcal{E}_{Q,c} \leq (1+\epsilon)\operatorname{vol}\mathrm{MVEE}(\mathcal{P})$, where $\eta' \coloneqq \eta(1+d^{-1})$. Letting $\widehat{x} \coloneqq (x^{\mathrm{T}}, 1)^{\mathrm{T}}$ for any $x \in \mathbb{R}^d$, we point out the following additional relation between $\mathcal{E}_{M,0}$ and $\mathcal{E}_{Q,c}$:

$$(x-c)^{\mathrm{T}} Q (x-c) = d^{-1}\big((d+1)\widehat{x}^{\mathrm{T}} M \widehat{x} - 1\big).$$

For derivations and proofs of the statements made above, we refer to the original publications discussing the algorithms. For a more exhaustive treatment, we refer to Todd's book [24].

2.1 First-Order Algorithms

At the heart of the algorithms considered in this paper lies the Frank–Wolfe strategy [11] applied to problem (D). In each step, a linearization of the problem is formed at the current iterate u, then the next iterate is found along the line segment going from u to the solution of this linear subproblem. Since the feasible region of problem (D) is the unit simplex, each subproblem is maximized at a vertex of the simplex. Specifically, if e_i is defined as the vector with 1 in position i and 0 in all other positions, then each subproblem is maximized at the vertex e_j, where j is the index of the largest component of the gradient of $\ln\det M(u)^{-1}$. Formally, j is given by $\arg\max_i \widehat{p}_i^{\mathrm{T}} M(u)\widehat{p}_i$. Geometrically, the point $\widehat{p}_j$ is the farthest point from $\mathcal{E}_{M,0}$ and p_j is the farthest point from $\mathcal{E}_{Q,c}$, measured in the respective ellipsoidal norms. This approach can be equivalently viewed as the barycentric coordinate descent method.

The algorithm by Wynn [27] moves the iterate u toward e_j using a step size that shrinks with each iteration. Fedorov [10] instead uses a line search to maximize the growth of the objective function, and shows that this search has a closed-form solution. Atwood [3] further augments the algorithm with *away steps*, which are equivalent to what was independently proposed by Wolfe [26]. In the context of the MVEE problem, an away step is a move of the current iterate *away* from the corner e_{j_-} of the unit simplex, where $j_- \coloneqq \arg\min_{i:u_i>0} \widehat{p}_i^{\mathrm{T}} M(u)\widehat{p}_i$; that is, j_- is the index of the smallest component of the gradient of $\ln\det M(u)^{-1}$, subject to the condition $u_i > 0$. Equivalently, the points $\widehat{p}_j$ and p_j are the farthest points from the boundary on the inside of the current trial ellipsoids $\mathcal{E}_{M,0}$ and $\mathcal{E}_{Q,c}$, respectively, among points with positive weights. Again, there is a closed-form expression for the step length that maximizes the growth of the objective function. The choice between a regular iteration and an away step is made dynamically.

Khachiyan [16] applies the algorithm by Fedorov [10] to the MVEE problem, and shows that if the initial solution u^0 is set to $u_i^0 = 1/n$ for $i = 1, \ldots, n$, then a solution satisfying (3) is computed in $O(nd^2(\eta^{-1} + \ln d + \ln\ln n))$ time. This bound is improved by the analysis of Todd [24] to $O(nd^2(\eta^{-1} + \ln\ln n))$. Kumar and Yıldırım [17] achieve a further improvement of the asymptotic time complexity by replacing the initialization by a procedure that identifies $\max(2d, n)$ points in $\mathcal{P}$ using the volume approximation algorithm by Betke and Henk [6]. By setting $u_i^0 = 1/\max(2d, n)$ for the points p_i located by this method, the algorithm finishes in $O(nd^2(\eta^{-1} + \ln\ln d))$ time [24]. Note that this algorithm has the potential to return a considerably more sparse solution u. Indeed, a core-set given

by $\{p_i : u_i > 0\}$ has cardinality $O(d(\eta^{-1} + \ln \ln d))$.

As a basis to motivate and exemplify our active set strategies, we use the algorithm by Todd and Yıldırım [25]. This algorithm, which is outlined in Algorithm 1, is the Kumar–Yıldırım variant combined with Atwood's away steps. The away steps enable the algorithm to use a refined termination test (Line 6) that ensures that the returned u satisfies (4) in addition to (3). Since away steps can reduce positive weights to zero, this algorithm can return even more sparse solutions (smaller core-sets) in practice. However, it retains the same theoretical bounds as the Kumar–Yıldırım algorithm on the time complexity and the size of the core-set.

Algorithm 1 Todd and Yıldırım's MVEE algorithm.

Input: $\mathcal{P} \coloneqq \{p_1, \ldots, p_n\} \subset \mathbb{R}^d$, $\eta > 0$
Output: $u \in \mathbb{R}^n$ satisfying (3) and (4)

1: Initialize u using the Kumar–Yıldırım scheme
2: **loop**
3: $\quad g_i \leftarrow \widehat{p}_i^{\mathrm{T}} M(u) \widehat{p}_i \quad$ for $i = 1, \ldots, n$
4: $\quad j_+ \leftarrow \arg\max_i g_i; \quad j_- \leftarrow \arg\min_{i:u_i>0} g_i$
5: $\quad \kappa_+ \leftarrow g_{j_+}; \quad \kappa_- \leftarrow g_{j_-}$
6: $\quad$ **if** $1 - \eta \leq \kappa_-$ **and** $\kappa_+ \leq 1 + \eta$ **then**
7: $\quad\quad$ Break the loop
8: $\quad$ **end if**
9: $\quad$ **if** $\kappa_+ - 1 > 1 - \kappa_-$ **then**
10: $\quad\quad \beta \leftarrow \frac{\kappa_+ - 1}{(d+1)\kappa_+ - 1}$
11: $\quad\quad u \leftarrow (1 - \beta)u + \beta e_{j_+}$
12: $\quad$ **else**
13: $\quad\quad \beta \leftarrow \min\left(\frac{1 - \kappa_-}{(d+1)\kappa_- - 1}, \frac{u_{j_-}}{1 - u_{j_-}}\right)$
14: $\quad\quad u \leftarrow (1 + \beta)u - \beta e_{j_-}$
15: $\quad$ **end if**
16: **end loop**
17: **return** u

The updates of u on Lines 11 and 14 both correspond to rank-1 updates of $M(u)^{-1}$. This means that from the second iteration onward, g can be computed on Line 3 from the previous value of g in $\Theta(dn)$ time per iteration, as opposed to the $\Theta(d^2 n)$ time it would take to evaluate the full quadratic form for $i = 1, \ldots, n$ [25]. Despite this, however, Line 3 is still the main bottleneck of the algorithm. As a motivation for the techniques discussed in this paper, we point out that in practice, the index j_+ computed on Line 4 already satisfies $u_{j_+} > 0$ in the vast majority of the iterations. In these iterations, it would therefore suffice to compute and scan only the g_i where $u_i > 0$, of which there are typically far fewer than n. Although it cannot be determined beforehand whether j_+ will refer to a point already in the solution or a new point that should be included, it makes sense to put more effort into improving the solution over a small subset $\mathcal{A}$ of points before considering the full data set again. The idea is to have the set $\mathcal{A}$, which we call the *active set*, contain all the points p_i where currently $u_i > 0$, as well as a subset of promising points that are likely to be included in the solution in a future iteration.

3 Active Set Strategies

A generic active set strategy is outlined in Algorithm 2. A complete such strategy has the following three components: (i) computing an initial active set, (ii) adding new points to the active set before each invocation of the underlying algorithm, and (iii) removing superfluous points from the active set after each such invocation. Sun and Freund [21] provide ideas for how to realize each of these components. In this paper, we rely on the initialization scheme by Kumar and Yıldırım [17] for the first part, since it is known to yield a good initial candidate solution (Line 1). For the third part, we take advantage of the fact that the discussed first-order algorithms will tend to increase the weights only of promising points, and that the away steps used in Algorithm 1 can even reduce some weights to zero. Thus, we leave it up to the inner algorithm to increase the weights of newly added points from zero, and we deem any points with zero weights in the returned solution to be superfluous and remove them afterwards (Line 5). For the second part (Line 11), we consider a number of methods below.

Before moving on, we point out that it is assumed on Line 4 that Algorithm 1 can be called with an additional optional parameter u^0, which it then uses as the initial solution instead of running the Kumar–Yıldırım procedure. Furthermore, to be sure of the correctness of Algorithm 2, note that $\kappa_{\mathcal{A}}$ always equals the final value of κ_+ in Algorithm 1, which means that $\kappa_{\mathcal{A}} \leq 1 + \eta$. Thus, if $\kappa_{\mathcal{P}} = \kappa_{\mathcal{A}}$, then the $(1 + \eta)$-approximate optimality condition (3) must hold also for $\mathcal{P}$. Otherwise, since $\mathcal{A} \subseteq \mathcal{P}$, it must be that $\kappa_{\mathcal{P}} > \kappa_{\mathcal{A}}$, which means that there is at least one point p_i outside of the current ellipsoid with $g_i > \kappa_{\mathcal{A}}$, which should be added to $\mathcal{A}$.

A straightforward and intuitive way to compute $\Delta\mathcal{A}$ is to include the C points p_i with the largest values of g_i, where C is some constant, or simply all points with $g_i > 1$ if there are C or fewer such

Algorithm 2 Active set algorithm.

Input: $\mathcal{P} \coloneqq \{p_1, \ldots, p_n\} \subset \mathbb{R}^d$, $\eta > 0$
Output: $u \in \mathbb{R}^n$ satisfying (3) and (4)

```
Initialize u using the Kumar–Yıldırım scheme
A ← {p_i : u_i > 0}
loop
    Update u_i for p_i ∈ A by calling Algorithm 1 on A with the current values of u_i as the initial solution
    A ← {p_i : u_i > 0}
    g_i ← p̂_i^T M(u) p̂_i   for i = 1, ..., n
    κ_P ← max_i g_i;   κ_A ← max_{i:p_i∈A} g_i
    if κ_P = κ_A then
        Break the loop
    end if
    Compute ΔA
    A ← A ∪ ΔA
end loop
return u
```

points. Indeed, this strategy has been used successfully with $C = 1$ for the minimum enclosing ball problem [12, 4, 19]. We include this "C-farthest" strategy in this evaluation with $C = 1, d, d^2$, and d^3. As noted by Sun and Freund [21], however, it might be a mistake to judge potential active points only based on their distance from the ellipsoid, since there might be clusters present in the data that make the farthest point dominate the next farthest point, for example. This could occur, for instance, for data sets containing many copies (or near identical copies) of each point.

For this reason, they present the following scheme to ensure that the points in $\Delta\mathcal{A}$ are reasonably spread out around the ellipsoid: If fewer than 30 points p_i satisfy $g_i > 1$, then they are all included in $\Delta\mathcal{A}$. Otherwise, $\Delta\mathcal{A}$ is first set to contain only the farthest point, and then the remaining points are considered in falling order of their values of g_i. Each p_i is added to $\Delta\mathcal{A}$ if $\sum_{p_\ell \in \Delta\mathcal{A}} (p_\ell - c)^{\mathrm{T}} Q (p_i - c) < 0$. Note that each term in this sum represents the dot product between the vectors p_ℓ and p_i in the local coordinate system of the ellipsoid $\mathcal{E}_{Q,c}$. Henceforth we refer to this strategy as the "SF" strategy.

With the same goal of achieving a large spread among the points added to the active set, Larsson et al. [18] propose the following *orthant scanning* procedure for the computation of the minimum enclosing ball: First $\mathcal{P}$ is partitioned based on which orthants the vectors $p_i - c$ fall into. Note that two vectors $v^1, v^2 \in \mathbb{R}^d$ belong to the same orthant iff $N(v_i^1) = N(v_i^2)$ holds for $i = 1, \ldots, d$, where N is a map defined as $N(x) = 1$ if $x < 0$ and $N(x) = 0$ otherwise. Then, from each group in which at least one point p_i satisfies $g_i > 1$, the farthest point in the group is included in $\Delta\mathcal{A}$. In this paper we include an additional version of this strategy that transforms the vectors $p_i - c$ to the local coordinate system of the ellipsoid before doing the orthant mapping.

We also introduce a new strategy that can be seen as a variation of the SF method above. We refer to this strategy as the "dot" strategy, and it works as follows: First the d farthest points are added to $\Delta\mathcal{A}$. Then, in a similar manner as before, the remaining points p_i with $g_i > 1$ are considered in order of decreasing values of g_i, but each p_i is added to $\Delta\mathcal{A}$ if $(p_\ell - c)^{\mathrm{T}} Q (p_i - c) \geq 0$ holds for fewer than d of the points $p_\ell \in \Delta\mathcal{A}$.

Note that these strategies only consider points for addition to the active set that are outside of the ellipsoid $\mathcal{E}_{Q,c}$. While it makes sense that these points are more likely to be active in the final solution, we remark that this is merely a heuristic to avoid having to consider every point in $\mathcal{P}$ in every iteration. There might indeed be points inside of said ellipsoid that wind up as active points in the final $(1 + \epsilon)$-approximation, and these will eventually be included in $\mathcal{A}$ in a later iteration.

3.1 Elimination Strategies

Harman and Pronzato [13] show that every support point p_* of MVEE($\mathcal{P}$) must satisfy $\widehat{p}_*^{\mathrm{T}} M(u) \widehat{p}_* \geq H$, where

$$H \coloneqq 1 + \frac{\delta}{2} - \frac{\sqrt{\delta(4 + \delta - 4/(d+1))}}{2},$$
$$\delta \coloneqq (d+1)(\max_i \widehat{p}_i^{\mathrm{T}} M(u) \widehat{p}_i - 1).$$

Conversely, any point p_i satisfying

$$\widehat{p}_i^{\mathrm{T}} M(u) \widehat{p}_i < H \tag{5}$$

cannot be a support point. Thus, eliminating all such points from $\mathcal{P}$ in each iteration of Algorithm 1 has no effect on the final result, but can make the subsequent iterations go faster as $\mathcal{P}$ gets continuously reduced. This technique can be viewed as a type of active set strategy that does not rely on an outer loop like that of Algorithm 2. Instead, the current active set is simply considered to be the points of $\mathcal{P}$ that have not yet been eliminated.

We now introduce an "aggressive" variant of this strategy. Instead of using the test (5), we remove from $\mathcal{P}$ any point p_i that satisfies $u_i = 0$ and

$$\widehat{p}_i^{\mathrm{T}} M(u) \widehat{p}_i < 1,$$

i.e., any non-active point that falls in the interior of the current trial ellipsoid. Since this might remove support points of MVEE($\mathcal{P}$), some additional measures must taken to ensure correctness. Initially, a full copy of $\mathcal{P}$ is saved. Then, once the termination criteria have been fulfilled, $\mathcal{P}$ is restored to its original state and the approximate optimality conditions (3) and (4) are verified against the original n points. If they do hold, the solution is returned, otherwise the algorithm is continued with $\mathcal{P}$ restored and the current solution as the starting point, and the process repeats.

3.2 Distance Filtering

In contrast to the minor modifications made to u in each iteration of Algorithm 1, which correspond to rank-1 corrections of $M(u)^{-1}$, the changes to u across the iterations of Algorithm 2 are extensive enough that the n values g_i must be recomputed from scratch in $\Theta(d^2 n)$ time in every iteration. In high dimensions, this computational cost might outweigh the savings from using the active set method. A similar issue occurs in the aggressive elimination method above, because all the g_i need to be recomputed every time $\mathcal{P}$ is restored.

To remedy these problems, we use the *distance filtering* technique that was introduced by Källberg and Larsson [15] to accelerate the types of algorithms discussed in this paper. The idea is that instead of maintaining the exact values $g_i = \widehat{p}_i^{\mathrm{T}} M(u) \widehat{p}_i$, the algorithm maintains upper bounds $h_i \geq \widehat{p}_i^{\mathrm{T}} M(u) \widehat{p}_i$ that are cheaper to compute. In the first iteration, each h_i is initialized to the exact value of $\widehat{p}_i^{\mathrm{T}} M(u) \widehat{p}_i$. Then in the subsequent iterations, each h_i first undergoes a constant-time update that ensures $h_i \geq \widehat{p}_i^{\mathrm{T}} M(u) \widehat{p}_i$ for the current u. Then $h_i \leftarrow \widehat{p}_i^{\mathrm{T}} M(u) \widehat{p}_i$ only if $h_i \geq 1$ after this initial update, which is typically the case only for a few of the values h_i.

Let u^{k+1} and u^k be the current and previous values of u, respectively. For each p_i define the generalized Rayleigh quotient $r_i := (\widehat{p}_i^{\mathrm{T}} M(u^{k+1}) \widehat{p}_i) / (\widehat{p}_i^{\mathrm{T}} M(u^k) \widehat{p}_i)$. Then the following holds for $i = 1, \ldots, n$:

$$\begin{aligned} \widehat{p}_i^{\mathrm{T}} M(u^{k+1}) \widehat{p}_i &= r_i (\widehat{p}_i^{\mathrm{T}} M(u^k) \widehat{p}_i) \\ &\leq \lambda_* (\widehat{p}_i^{\mathrm{T}} M(u^k) \widehat{p}_i) \\ &\leq \lambda_* h_i, \end{aligned}$$

where $\lambda = \lambda_*$ is the largest solution to the generalized eigenvalue problem [5]

$$M(u^{k+1}) x = \lambda M(u^k) x.$$

Thus, setting $h_i \leftarrow \lambda_* h_i$ ensures that $h_i \geq \widehat{p}_i^{\mathrm{T}} M(u^{k+1}) \widehat{p}_i$ for $i = 1, \ldots, n$, and takes only constant time per h_i.

4 Experiments

For the evaluation, we used random data sets generated in the same manner as in [21] and [2], that is, as the unions of one to four clusters drawn from independent multivariate normal distributions. All code was written for MATLAB 9.5.0 (R2018b) and executed on a PC with an Intel Core i5-8250U processor and 16 GB of main memory. Our results are summarized in Table 1.

The conservative elimination method of Harman and Pronzato does not reach the same levels of acceleration as the other methods do. Furthermore, its effects diminish as the number of dimensions rises beyond $d = 10$, which can be explained as being an example of the "curse of dimensionality": The elimination test (5) is equivalent to testing whether a point falls inside a shrunken copy of the current ellipsoid. As the dimensions increase, the ratio of the volume of this ellipsoid to the volume of the portion of space occupied by $\mathcal{P}$ decreases rapidly. Thus, it becomes increasingly unlikely that any points in $\mathcal{P}$ fall inside the smaller ellipsoid.

The aggressive elimination method consistently outperforms the conservative method, and contrary to the latter, its effect appears to *increase* with the number of dimensions. For $d \geq 20$, it is the fastest strategy overall.

As can be expected, the C-farthest strategies show increasing reductions in the number of iterations with larger values of C. In the lower-dimensional instances, this also tends to translate into improved solution times. For $d \geq 20$, however, the d^3-farthest strategy begins to exhibit worse execution times than the d- and d^2-farthest strategies. This is caused by too many points being included in $\Delta\mathcal{A}$ in each iteration, which slows down the inner solver. Note that

strategy	iter.	speed	iter.	speed	iter.	speed	iter.	speed
	$d = 2$		$d = 3$		$d = 4$		$d = 6$	
baseline	1.0	0.6 sec	1.0	1.9 sec	1.0	5.2 sec	1.0	11.9 sec
elim.	1.0	3.2×	1.0	4.4×	1.0	7.2×	1.0	10.7×
elim.-agg.	1.0	5.1	1.0	10.8	1.0	22.8	1.0	40.2
1-farthest	1.5	8.6	4.4	13.7	7.0	22.3	13.5	25.9
d-farthest	1.5	8.7	3.6	14.0	4.6	28.3	6.0	39.4
d^2-farthest	1.5	8.1	3.0	16.1	3.7	31.0	4.3	47.8
d^3-farthest	1.5	8.7	2.9	16.4	3.3	33.2	3.9	50.2
SF	1.5	8.7	3.1	14.9	3.6	31.4	4.8	43.8
orth.	1.5	8.0	3.3	11.6	4.0	30.2	5.1	41.8
orth.-local	1.5	8.1	3.1	11.6	3.8	29.8	4.2	46.5
dot	1.5	8.2	3.1	14.4	3.5	30.9	4.1	48.0
	$d = 10$		$d = 15$		$d = 20$		$d = 25$	
baseline	1.0	22.8 sec	1.0	52.9 sec	1.0	85.7 sec	1.0	127.4 sec
elim.	1.0	5.1×	1.0	5.2×	1.0	3.7×	1.0	4.0×
elim.-agg.	1.0	41.0	1.0	61.1	1.0	67.4	1.0	73.6
1-farthest	37.5	14.7	73.5	14.3	122.7	10.0	185.7	9.1
d-farthest	8.4	33.9	9.8	46.3	11.2	43.6	12.0	50.3
d^2-farthest	5.2	41.9	5.7	60.9	6.0	63.9	6.0	65.1
d^3-farthest	4.6	46.9	4.7	64.5	4.8	57.5	4.7	47.6
SF	6.7	34.1	8.5	41.4	10.5	28.8	12.5	18.6
orth.	7.9	32.6	8.0	45.6	8.4	34.7	5.7	19.8
orth.-local	4.9	44.9	4.8	51.1	3.7	30.2	3.1	15.9
dot	5.8	38.9	6.9	54.5	7.9	53.4	8.0	61.7

Table 1: Computational results. For each shown value of d, ten point sets with $n = 10^6$ were generated and a $(1+10^{-6})$-approximate MVEE of each point set was computed with all of the strategies. The reported figures are the geometric means from the ten point sets. The iteration counts refer to the number of invocations of Algorithm 1, which is always 1 for the baseline and the elimination strategies. The "speed" columns show the actual execution time only for the baseline, and the relative *speedup factors* for the remaining versions.

as a particular strategy generates sets $\Delta\mathcal{A}$ of larger and larger sizes, its behaviour becomes increasingly similar to the baseline.

Clearly, the data sets used here do not warrant an active set strategy that specifically attempts to increase the spread of the points in the active set. This is evident from the fact that none of the SF, orthant, or dot strategies provide significant further improvements over simply including the C farthest points in each iteration. Nevertheless, some interesting effects can be found in these results.

Firstly, although SF is comparable to d-farthest w.r.t. the number of iterations, it is considerably slower when d is large. This can be explained by the fact that selecting $\Delta\mathcal{A}$ using the SF strategy has a time complexity that is quadratic in $|\Delta\mathcal{A}|$, and although it does not show in Table 1, our more detailed results confirm that $|\Delta\mathcal{A}|$ tends to be very large when d is large. This is particularly the case for the $\Delta\mathcal{A}$ computed in the first iteration, when there are still many points outside of the candidate ellipsoid.

Although this drawback could possibly be remedied by enforcing a cap of, say, d^2 on the cardinality of $\Delta\mathcal{A}$, this could negatively affect the number of iterations of the overall algorithm. Furthermore, the speedup figures suggest that the dot strategy—which has the same quadratic time complexity as SF—does not experience this problem. Our detailed results show that, usually, $|\Delta\mathcal{A}| \leq 4d$ with this strategy. In addition, from Table 1 it appears that this method incurs fewer iterations in general than SF.

Finally, we note that the orthant scanning strategies are at best comparable to the other strategies. It is nevertheless interesting that for the larger values of d, the local variant reduces the number of iterations considerably compared to the original version, which is due to larger numbers of points being added to the active set in every iteration (which, again, is

also the reason that the reduction in iterations does not result in improved execution times). As only orthants with at least one point outside of the current ellipsoid provide a point to $\Delta\mathcal{A}$, this suggests that the local orthants partition the points not enclosed by the ellipsoid more evenly than the global orthants do, which is an interesting result in and of itself.

5 Discussion

It is evident that the techniques developed in this paper can significantly improve performance in practice. However, as mentioned in Section 1, it is not clear that they are guaranteed to do so in theory. The main issue is that Algorithm 1 only "sees" part of the point set in each invocation: Whenever the value of κ_+ is smaller than the global largest distance measured over the full point set, the update on Line 11 is suboptimal; an excessive number of such updates can negatively affect performance.

While a detailed analysis of the worst-case performance of the algorithms has to be left as future work, fairly conservative bounds on the number of iterations can be derived with limited effort. Todd's analysis of Algorithm 1 [24] rests on the facts that the initial solution u computed using the Kumar–Yıldırım method has an optimality gap at most $4(d+1)\ln(d+1)$ (Corollary 3.4 in [24]), and that in every iteration that does not reduce a weight to zero, the gap shrinks by at least $\ln(1+\delta)-\delta/(1+\delta)$, where $\delta \coloneqq \max(\kappa_+ - 1, 1-\kappa_-)$ (Lemmas 3.5 and 3.10 in [24]). It is clear that the bound of $O(d(\eta^{-1}+\ln\ln d))$ iterations must hold in each individual invocation of Algorithm 1 as a subroutine from Algorithm 2, since each starting solution u^0 is at least as good as the one used in the original algorithm. Furthermore, it is easily shown that the same bound holds for the number of iterations in Algorithm 2: The initial optimality gap is clearly the same. Furthermore, all the discussed strategies for computing $\Delta\mathcal{A}$ ensure that each $\mathcal{A}$ contains the farthest point p_j, $j \coloneqq \arg\max_i g_i$. Thus, at least in the first iteration of every invocation of Algorithm 1, the value of κ_+ matches the global largest distance. Consequently, each such invocation must shrink the optimality gap at least as much as a regular iteration of the original algorithm. Analogous arguments can be applied to the elimination scheme of Section 3.1 to bound the number of times the algorithm is paused and then resumed with the restored point set, as well as the number of iterations between these restarts.

Although promising effects were seen on the artificial data sets used in this study, in the future, it would be interesting to evaluate the discussed techniques in real-world settings involving large data sets, for example, in clustering and classification applications. Such case studies could perhaps better reveal the suitability of the different variants of the algorithm for different situations.

Acknowledgments

We thank Selin Damla Ahipaşaoğlu for sharing the MATLAB code to generate the data sets used in the evaluation.

References

[1] S. Damla Ahipaşaoğlu and E. Alper Yıldırım. Identification and elimination of interior points for the minimum enclosing ball problem. *SIAM J. Optim.*, 19(3):1392–1396, 2008.

[2] Selin Damla Ahipaşaoğlu, Peng Sun, and Michael J. Todd. Linear convergence of a modified Frank–Wolfe algorithm for computing minimum-volume enclosing ellipsoids. *Optimization Methods and Software*, 23(1):5–19, feb 2008.

[3] Corwin L. Atwood. Sequences converging to D-optimal designs of experiments. *The Annals of Statistics*, 1(2):342–352, 1973.

[4] Mihai Bădoiu and Kenneth L. Clarkson. Smaller core-sets for balls. In *Proceedings of the Fourteenth Annual ACM–SIAM Symposium on Discrete Algorithm*, pages 801–802, 2003.

[5] Generalized Hermitian eigenvalue problems. In Zhaojun Bai, James Demmel, Jack Dongarra, Axel Ruhe, and Henk van der Vorst, editors, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, 2000.

[6] U. Betke and M. Henk. Approximating the volume of convex bodies. *Discrete & Computational Geometry*, 10(1):15–21, 1993.

[7] D. Böhning. A vertex-exchange-method in D-optimal design theory. *Metrika*, 33(1):337–347, 1986.

[8] Yu Jen Chen, Ming Yi Ju, and Kao Shing Hwang. A virtual torque-based approach to kinematic control of redundant manipulators.

IEEE Transactions on Industrial Electronics, 64(2):1728–1736, 2017.

[9] Wei-Jie Cong, Hong-Wei Liu, Feng Ye, and Shui-Sheng Zhou. Rank-two update algorithms for the minimum volume enclosing ellipsoid problem. *Computational Optimization and Applications*, 51(1):241–257, 2012.

[10] Valerii Vadimovich Fedorov. *Theory of Optimal Experiments*. Elsevier, 1972.

[11] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

[12] Bernd Gärtner. Fast and robust smallest enclosing balls. In *Algorithms – ESA '99*, volume 1643 of *Lecture Notes in Computer Science*, pages 325–338. Springer Berlin Heidelberg, 1999.

[13] Radoslav Harman and Luc Pronzato. Improvements on removing nonoptimal support points in D-optimum design algorithms. *Statistics & Probability Letters*, 77(1):90–94, 2007.

[14] Fritz John. Extremum problems with inequalities as subsidiary conditions. In *Studies and Essays, Presented to R. Courant on His 60th Birthday*, pages 187–204. Wiley Interscience, 1984.

[15] Linus Källberg and Thomas Larsson. A filtering heuristic for the computation of minimum-volume enclosing ellipsoids. In *Combinatorial Optimization and Applications (COCOA)*, pages 744–753. Springer International Publishing, 2016.

[16] Leonid G. Khachiyan. Rounding of polytopes in the real number model of computation. *Mathematics of Operations Research*, 21(2):307–320, 1996.

[17] P. Kumar and E. A. Yıldırım. Minimum-volume enclosing ellipsoids and core sets. *Journal of Optimization Theory and Applications*, 126(1):1–21, 2005.

[18] Thomas Larsson, Gabriele Capannini, and Linus Källberg. Parallel computation of optimal enclosing balls by iterative orthant scan. *Computers & Graphics*, 56:1–10, 2016.

[19] Thomas Larsson and Linus Källberg. Fast and robust approximation of smallest enclosing balls in arbitrary dimensions. *Computer Graphics Forum*, 32(5):93–101, 2013.

[20] Yu Nesterov. Rounding of convex sets and efficient gradient methods for linear programming problems. *Optimization Methods and Software*, 23(1):109–128, 2008.

[21] Peng Sun and Robert M. Freund. Computation of minimum-volume covering ellipsoids. *Operations Research*, 52(5):690–706, 2004.

[22] Kasemsit Teeyapan, Nipon Theera-Umpon, and Sansanee Auephanwiriyakul. Ellipsoidal support vector data description. *Neural Computing and Applications*, 28(Suppl. 1):337–347, 2017.

[23] D. M. Titterington. Optimal design: Some geometrical aspects of D-optimality. *Biometrika*, 62(2):313, 1975.

[24] Michael J. Todd. *Minimum-Volume Ellipsoids*. Society for Industrial and Applied Mathematics, 2016.

[25] Michael J. Todd and E. Alper Yıldırım. On Khachiyan's algorithm for the computation of minimum-volume enclosing ellipsoids. *Discrete Applied Mathematics*, 155(13):1731–1744, 2007.

[26] Philip Wolfe. Convergence theory in nonlinear programming. *Integer and Nonlinear Programming*, pages 1–36, 1970.

[27] Henry P. Wynn. The sequential generation of D-optimum experimental designs. *Ann. Math. Statist.*, 41(5):1655–1664, 10 1970.