IDENTIFICATION AND ELIMINATION OF INTERIOR POINTS FOR THE MINIMUM ENCLOSING BALL PROBLEM

S. DAMLA AHIPAŞAOĞLU[*] AND E. ALPER Yıldırım[†]

Abstract. Given $\mathcal{A} := \{a^1, \ldots, a^m\} \subset \mathbb{R}^n$, we consider the problem of reducing the input set for the computation of the minimum enclosing ball of $\mathcal{A}$. In this note, given an approximate solution to the minimum enclosing ball problem, we propose a simple procedure to identify and eliminate points in $\mathcal{A}$ that are guaranteed to lie in the interior of the minimum-radius ball enclosing $\mathcal{A}$. Our computational results reveal that incorporating this procedure into the two recent algorithms proposed by Yıldırım leads to significant speed-ups in running times especially for randomly generated large-scale problems. We also illustrate that the extra overhead due to the elimination procedure remains at an acceptable level for spherical or almost spherical input sets.

Key words. Minimum enclosing balls, input set reduction, approximation algorithms.

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

1. Introduction. Given $\mathcal{A} := \{a^1, \ldots, a^m\} \subset \mathbb{R}^n$, we denote the unique minimum enclosing ball of $\mathcal{A}$ by MEB($\mathcal{A}$), i.e.,

$$\text{MEB}(\mathcal{A}) = \mathcal{B}_{c^*,\rho^*} := \{x \in \mathbb{R}^n : \|x - c^*\| \leq \rho^*\},$$

where $c^* \in \mathbb{R}^n$ is the optimal center, $\rho^* \in \mathbb{R}$ is the optimal radius, and $\|\cdot\|$ denotes the Euclidean norm. Given $\epsilon > 0$, a ball $\mathcal{B}_{c,\rho}$ is said to be a $(1+\epsilon)$-approximate solution to MEB($\mathcal{A}$) if

$$\rho \leq \rho^*, \quad \mathcal{A} \subset \mathcal{B}_{c,(1+\epsilon)\rho} \;. \tag{1.1}$$

In this note, given a $(1+\epsilon)$-approximate solution $\mathcal{B}_{c,\rho}$ to MEB($\mathcal{A}$), we propose a simple condition that should be satisfied by each point in $\mathcal{A}$ that lies on the boundary of MEB($\mathcal{A}$). Furthermore, we derive an upper bound on the Euclidean distance between c and c^*.

2. Main Result. LEMMA 2.1. *Given $\mathcal{A} := \{a^1, \ldots, a^m\} \subset \mathbb{R}^n$ and $\epsilon > 0$, let $\mathcal{B}_{c,\rho}$ be a $(1+\epsilon)$-approximate solution to MEB($\mathcal{A}$). Then,*

$$\|c - c^*\| \leq (2\epsilon + \epsilon^2)^{1/2}\rho. \tag{2.1}$$

Furthermore, each point $a^i \in \mathcal{A}$ on the boundary of MEB($\mathcal{A}$) satisfies

$$\|a^i - c\| \geq (1 - (2\epsilon + \epsilon^2)^{1/2})\rho. \tag{2.2}$$

Proof. Suppose that $c \neq c^*$. Consider the hyperplane $\mathcal{H}$ passing through c^* perpendicular to $c^* - c$. Let $\mathcal{H}_+$ denote the closed halfspace bounded by $\mathcal{H}$ and not

[*]School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14853. e-mail: `dse8@cornell.edu`. This author was supported in part by NSF through grant DMS-0513337 and ONR through grant N00014-08-1-0036.

[†]Department of Industrial Engineering, Bilkent University, 06800 Bilkent, Ankara, Turkey. e-mail: `yildirim@bilkent.edu.tr`. This author was supported in part by TÜBİTAK (Turkish Scientific and Technological Research Council) Grant 107M411.

containing c. Then, by [2, Lemma 2.2], there exists a point $a^j \in \mathcal{H}_+ \cap \mathcal{A}$ such that $\|a^j - c^*\| = \rho^*$. Therefore, $\|c - a^j\|^2 \geq \|c - c^*\|^2 + \|c^* - a^j\|^2$, which implies that

$$\begin{aligned} \|c - c^*\|^2 &\leq \|c - a^j\|^2 - \|c^* - a^j\|^2, \\ &\leq (1+\epsilon)^2\rho^2 - (\rho^*)^2, \\ &\leq (1+\epsilon)^2\rho^2 - \rho^2, \\ &= (2\epsilon + \epsilon^2)\rho^2, \end{aligned}$$

where we used (1.1) to derive the second and third inequalities. This establishes (2.1).

Let a^i be any point on the boundary of MEB($\mathcal{A}$). Then, $\|a^i - c^*\| \leq \|a^i - c\| + \|c - c^*\|$, which implies that

$$\begin{aligned} \|a^i - c\| &\geq \rho^* - \|c - c^*\|, \\ &\geq \rho - (2\epsilon + \epsilon^2)^{1/2}\rho, \\ &= (1 - (2\epsilon + \epsilon^2)^{1/2})\rho, \end{aligned}$$

where we used (1.1) and (2.1) to derive the second inequality. This completes the proof. $\square$

3. Computational Results. Recently, Yıldırım [2] proposed two first-order algorithms that can compute a $(1+\epsilon)$-approximate solution to the minimum enclosing ball of a finite input set $\mathcal{A}$ of points for any given $\epsilon > 0$. Each algorithm generates a sequence of approximate minimum enclosing balls $\mathcal{B}_{c^k,\rho^k}$ which converge to MEB($\mathcal{A}$) in the limit. Each such ball is a $(1+\epsilon^k)$-approximate solution to MEB($\mathcal{A}$) for a certain $\epsilon^k > 0$ and the algorithm terminates when $\epsilon^k \leq \epsilon$. Both of these algorithms extract a small core set $\mathcal{X} \subseteq \mathcal{A}$ and can be extended to much more general input sets without sacrificing the small core set result.

Lemma 2.1 can be easily incorporated into both of the algorithms in [2] in an attempt to eliminate interior points in $\mathcal{A}$ (with respect to MEB($\mathcal{A}$)) thereby reducing the size of the input set. This elimination procedure does not affect the minimum enclosing ball and may decrease the computational cost of each iteration due to the reduction in the input size.

In order to assess the implications of Lemma 2.1 in practice, we have performed computational tests in which the simple elimination procedure proposed in this note was incorporated into each of the two algorithms in [2]. In our experiments, we checked the boundary condition (2.2) at an approximate minimum enclosing ball generated throughout either algorithm only if the right-hand side of (2.2) is sufficiently bounded away from zero. This strategy eliminates the computational cost of checking the boundary condition at an iterate where it would be unlikely to remove a large subset of input points. At iterate k, (2.2) is checked in our computational experiments only if $1 - (2\epsilon^k + (\epsilon^k)^2)^{1/2} > 0.55$, where 0.55 is a threshold value that was found to work well empirically.

The computational experiments were carried out on a 3.40 GHz Pentium IV processor with 1.0 GB RAM using MATLAB version R2006b on four different data sets. The first two data sets were randomly generated using different procedures outlined below. The last two sets consist of spherical or almost spherical input sets.

3.1. Random Input Sets. The first data set was randomly generated as in [2] with sizes (n, m) varying from $(10, 500)$ to $(100, 100000)$, while the second one was generated using the standard normal distribution with the same sizes (n, m). We used

		CPU Time						Reduced Input Size	
n	m	A1	A1E	Speed-up	A2	A2E	Speed-up	A1E	A2E
10	500	0.0594	0.0541	1.10	0.0219	0.0156	1.40	124.2	99.8
10	1000	0.0694	0.0469	1.48	0.0297	0.0203	1.46	202.4	200.7
20	5000	2.2016	0.5078	4.34	0.3594	0.2172	1.65	420.4	330.3
20	10000	3.9844	0.5484	7.27	0.5641	0.1484	3.80	147.8	158.2
30	30000	14.1031	0.8516	16.56	2.8281	0.5562	5.08	121.1	107.3
50	50000	48.9359	5.3875	9.08	12.0109	4.1469	2.90	695.8	400.9
100	100000	141.6518	35.0223	4.04	62.692	30.5357	2.05	1626.2	1650.1

Table 3.1
Computational Results for the First Data Set ($\epsilon = 10^{-3}$)

$\epsilon = 10^{-3}$ for both data sets. For each fixed (n, m), ten different problem instances were generated for each data set. The computational results are reported in terms of averages over these instances in Table 3.1 and Table 3.2, each of which is divided into three sets of columns. The first set of columns reports the size (n, m). The second set of columns presents the results regarding the CPU time and is further divided into two parts, the first of which is devoted to the computational results related to [2, Algorithm 3.1] (adaptation of the Frank-Wolfe algorithm to the minimum enclosing ball problem) while the second one displays those results using [2, Algorithm 4.1] (adaptation of the Frank-Wolfe algorithm with *away steps* to the minimum enclosing ball problem). In the first part, A1 and A1E denote the CPU times in seconds using [2, Algorithm 3.1] without and with the elimination procedure, respectively, and speed-up denotes the resulting speed-up factor in running time due to the elimination procedure measured in terms of the ratio of A1 to A1E. Similarly, A2 and A2E denote the CPU times in seconds using [2, Algorithm 4.1] without and with the elimination procedure, respectively, and speed-up denotes the resulting speed-up factor in running time measured in terms of the ratio of A2 to A2E. The last set of columns reports the number of remaining input points upon termination using each algorithm with the elimination procedure.

		CPU Time						Reduced Input Size	
n	m	A1	A1E	Speed-up	A2	A2E	Speed-up	A1E	A2E
10	500	0.2016	0.1953	1.03	0.0250	0.0094	2.66	12.7	12.2
10	1000	0.2018	0.1469	1.37	0.0484	0.025	1.94	15.4	15
20	5000	3.0062	0.3281	9.16	0.475	0.1109	4.28	38.4	37
20	10000	5.0328	0.3312	15.20	0.9188	0.1812	5.07	42	40.9
30	30000	24.5359	1.2594	19.48	3.9656	0.9094	4.36	85.5	79.7
50	50000	52.8751	4.1865	12.63	13.0204	3.8463	3.39	202.2	213.4
100	100000	267.05	27.9984	9.54	56.1344	20.7188	2.71	430.9	423.8

Table 3.2
Computational Results for the Second Data Set ($\epsilon = 10^{-3}$)

As illustrated by Table 3.1 and Table 3.2, the incorporation of the elimination procedure into each of the two algorithms results in significant savings in running times especially for large instances where $m \gg n$. The procedure described in Lemma 2.1 identifies and eliminates 75% to 99% of the data points in our experiments and the running times may improve by more than a factor of 19 on some instances. It is also worth noticing that the speed-up factors obtained from Algorithm 3.1 are generally considerably larger than those obtained with Algorithm 4.1. This may be due to the reason that the asymptotical linear convergence property of Algorithm 4.1 [2] already

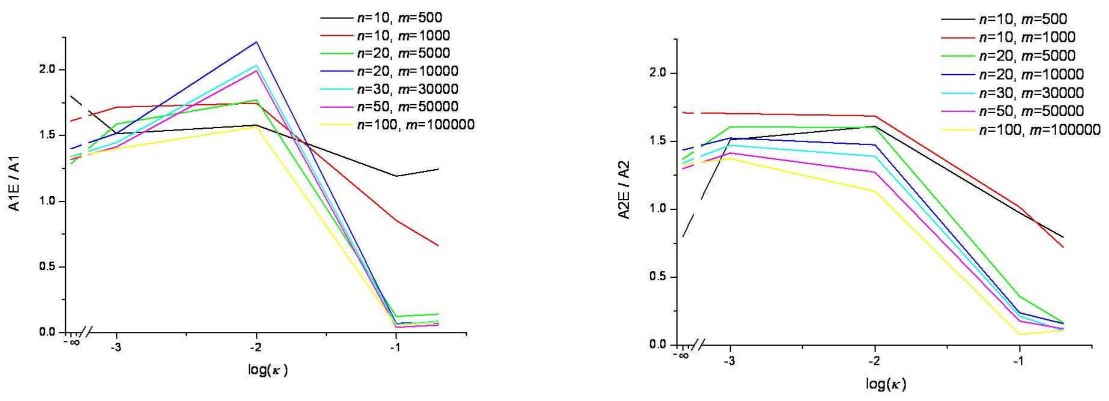

FIG. 3.1. *Experimental Results for Almost Spherical Input Sets*

results in significantly better performance compared to that of Algorithm 3.1, which may not leave much room for further improvement. Finally, we remark that the elimination procedure does not seem to have a noticeable effect on the core set sizes and on the number of iterations for either of the two algorithms.

3.2. Spherical and Almost Spherical Input Sets. In an attempt to assess the extent of extra overhead due to the elimination procedure, we considered data sets where all points lie on (or almost on) the unit sphere centered at the origin. An input set $\mathcal{A}$ is said to lie on a κ-approximate unit sphere centered at the origin, denoted by $\mathcal{S}_\kappa$, if $\mathcal{A} \subset \mathcal{S}_\kappa := \{x \in \mathbb{R}^n : 1 - \kappa \leq \|x\| \leq 1 + \kappa\}$. For an input set $\mathcal{A} \subset \mathcal{S}_\kappa$ where $\kappa \geq 0$ is small, the elimination procedure will keep testing input points for removal at each iteration but will be unable to remove a substantial subset of the input set. In the extreme case where $\kappa = 0$, none of the input points can be removed since there would be no interior point. This extra overhead will necessarily result in an increase in the running time of an algorithm that uses the elimination procedure. We generated random input sets $\mathcal{A} \subset \mathcal{S}_\kappa$, where $\kappa \in \{0, 0.001, 0.01, 0.1, 0.2\}$, with sizes (n, m) varying from $(10, 500)$ to $(100, 100000)$ as in our experiments with the first two data sets. For each choice of experimental parameters, the computational results averaged over ten data sets are illustrated in Figure 3.1. The horizontal axis in each graph corresponds to κ using the logarithmic scale while the vertical axis in the graph on the left (on the right) corresponds to the "slow-down" factor measured in terms of the ratio of the running time of Algorithm 3.1 (Algorithm 4.1) with the elimination procedure to the running time of the same algorithm without the elimination. A close examination of these two graphs reveals that the slow-down factors usually remain at an acceptable level especially for the faster Algorithm 4.1. Note that the elimination procedure leads to an extra overhead of at most 70% on all instances for Algorithm 4.1. A comparison of the slow-down and speed-up factors stemming from our experiments seems to justify the use of the elimination procedure especially since spherical input sets would not likely be encountered in practical applications.

We also tested the two algorithms on data sets which consist of the vertices of the unit simplex where $n \in \{1000, 2500, 5000\}$. Note that each point in such an input set lies on the boundary of the minimum enclosing ball and it is known that each point should be in the core set if $\epsilon \leq 1/n$ [1]. We tested each of the two algorithms with and without the elimination procedure using $\epsilon = 1/n$. The computational results are reported in Table 3.3, which is organized in a similar manner to that of Table 3.1. Note that the increase in the running time of each algorithm due to the inclusion of the elimination procedure is only around 35% for large spherical instances.

4. Concluding Remarks. In this paper, we have described a procedure that identifies and eliminates data points that cannot lie on the boundary of the minimum enclosing ball of a finite set of points. This procedure can be easily incorporated into any iterative algorithm that generates a sequence of approximate minimum enclosing balls converging to the minimum enclosing ball of a given input set. Our computational results demonstrate the resulting significant improvements in the practical performance of the two algorithms proposed in [2] especially for randomly generated input sets. The extra overhead of the elimination procedure remains at an acceptable level for spherical or almost spherical input sets.

Furthermore, the same elimination procedure can also be incorporated into algorithms that can compute an approximate minimum enclosing ball of more general input sets such as a set of balls or ellipsoids for which the algorithms in [2] can still be applied. Such input sets can be viewed as an infinite set of points and condition (2.2) essentially means that all input points that lie in the interior of a ball of a certain radius centered at the current approximate center c can be safely removed without affecting the optimal solution. In this case, an element of a more general input set (such as a ball or ellipsoid) can be completely removed if the furthest point on that element from the current approximate center c already lies in the interior of the aforementioned ball centered at c, which readily implies that every point on that element should necessarily violate (2.2). This may lead to considerable savings in the computation of minimum enclosing balls of more general input sets arising from practical applications.

Acknowledgments. We thank Mike Todd for encouraging us to prepare this manuscript. We gratefully acknowledge the thoughtful comments of two anonymous referees and the Associate Editor.

REFERENCES

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

[2] E. A. Yildirim, *Two algorithms for the minimum enclosing ball problem*, SIAM Journal on Optimization, to appear (2008).

		CPU Time					
n	m	A1	A1E	A1E/A1	A2	A2E	A2E/A2
1000	1000	39.5312	53.625	1.357	40.078	54.219	1.352
2500	2500	251.75	336.7188	1.338	252.234	339.391	1.346
5000	5000	988.2812	1301.5312	1.317	983.25	1289.547	1.311

Table 3.3
Computational Results for the Vertices of the Unit Simplex ($\epsilon = 1/n$)