

# Efficient algorithms for the smallest enclosing ball problem

Guanglu Zhou <sup>\*</sup>, Kim-Chuan Toh <sup>†</sup>, Jie Sun <sup>‡</sup>

November 25, 2002

**Abstract.** Consider the problem of computing the smallest enclosing ball of a set of  $m$  balls in  $\mathfrak{R}^n$ . Existing algorithms are known to be inefficient when  $n > 30$ . In this paper we develop two algorithms that are particularly suitable for problems where  $n$  is large. The first algorithm is based on log-exponential aggregation of the maximum function and reduces the problem into an unconstrained convex program. The second algorithm is based on a second-order cone programming formulation, with special structures taken into consideration. Our computational experiments show that both methods are extremely efficient for large problems. The first algorithm is able to solve problems of dimension  $n$  up to 10,000 efficiently.

**Keywords:** Computational geometry, smoothing approximation, second order cone programming.

**AMS subject classification:** 90C30.

## 1 Introduction

A ball  $B_i$  in  $\mathfrak{R}^n$  with center  $c_i$  and radius  $r_i \geq 0$  is the closed set  $B_i = \{x \in \mathfrak{R}^n : \|x - c_i\| \leq r_i\}$ . Given a set of balls  $\mathcal{B} = \{B_1, B_2, \dots, B_m\}$  in  $\mathfrak{R}^n$ , the smallest enclosing ball  $mb(\mathcal{B})$  of the set  $\mathcal{B}$  is the ball with the smallest radius that encloses all balls in  $\mathcal{B}$ . Given a set of points  $\mathcal{P} = \{p_1, p_2, \dots, p_m\}$  in  $\mathfrak{R}^n$ , the smallest enclosing ball  $mp(\mathcal{P})$  of the points  $\mathcal{P}$  is the ball with the smallest radius that encloses all the points in  $\mathcal{P}$ . Clearly, the problem of smallest enclosing ball of points is a special case of the problem

---

<sup>\*</sup>Singapore-MIT Alliance, National University of Singapore, 4 Engineering Drive 3, Singapore 117576. E-mail: smazgl@nus.edu.sg.

<sup>†</sup>Department of Mathematics, National University of Singapore, 2 Science Drive 2, Singapore 117543. E-mail: mattohkc@nus.edu.sg. Research supported in part by the Singapore-MIT Alliance.

<sup>‡</sup>Department of Decision Sciences, National University of Singapore (NUS), Singapore 117592. E-mail: jsun@nus.edu.sg. Research supported in part by the Singapore-MIT Alliance and Grants RP314000-028/042-112 of NUS

of smallest enclosing ball of balls. In what follows, unless otherwise stated, we will term the problem of smallest enclosing ball of balls to be the smallest enclosing ball problem. This problem can be easily formulated as the following convex optimization problem:

$$\min_{x \in \mathbb{R}^n} \max_{1 \leq i \leq m} \{\|x - c_i\| + r_i\}. \quad (1)$$

Let

$$f_i(x) = \|x - c_i\| + r_i, \quad i = 1, 2, \dots, m, \quad (2)$$

and

$$f(x) = \max_{1 \leq i \leq m} \{f_i(x)\}. \quad (3)$$

Then problem (1) can be rewritten as

$$\min_{x \in \mathbb{R}^n} f(x). \quad (4)$$

It is readily seen that the solution to (1) exists since  $f(x)$  is coercive. Moreover, the solution is unique, for otherwise there would exist two different balls,  $C_1$  and  $C_2$ , of the same radius, with  $\cup_{j=1}^m B_j \subset C_i, i = 1, 2$ . Then one can construct a smaller ball containing  $C_1 \cap C_2$  and thus  $\mathcal{B}$  as well.

Problem (1) arises in applications such as location analysis and military operations and it is itself of interest as a problem in computational geometry; see [2, 3, 6, 9, 12, 13, 18, 20] for details. Many algorithms have been developed for (1). In particular, for the problem of the smallest enclosing ball of points, Megiddo [9] presented a deterministic  $O(m)$  algorithm for the case where  $n \leq 3$ . Welzl [18] developed a simple randomized algorithm with expected linear time in  $m$  for the case where  $n$  is small, and Gärtner described a C++ implementation thereof in [4]. For the problem of the smallest enclosing ball of balls, we are aware of only a C++ program developed by David White in [19], which is based on Welzl's algorithm [18] and Gärtner's implementation [4]. The existing software packages [4, 19], however, are not efficient for solving problems with  $n > 30$ . The goal of this paper is to present efficient algorithms that can be used to solve problems where  $n$  is large (say, in the order of thousands) and  $m$  is moderate (say, in the order of hundreds).

Problem (1) is a non-differentiable (nonsmooth) convex optimization problem. Due to the non-differentiability of the objective function, gradient-based algorithms cannot be used to solve this problem. To overcome this difficulty, in Section 2 we give a smooth approximation for the maximum function  $f$ . This approximation is based on the so-called log-exponential aggregation function studied, for instance, in [8]. Based on this smooth approximate function, in Section 3 we propose a globally convergent quasi-Newton algorithm for solving the problem. In Section 4, we reformulate (1) as a second order cone programming (SOCP) problem. Thus, we can also apply interior-point methods to solve (1). In Section 5, we report some numerical results, which show the algorithms proposed here are able to solve problems of dimension  $n$  up to 10,000 efficiently.

In this paper, unless otherwise stated, all vectors are column vectors. We let  $I_d$  denote the  $d \times d$  identity matrix. To represent a large matrix with several small matrices, we use semicolons “;” for column concatenation and commas “,” for row concatenation. These notations also apply to vectors. We denote the convex hull of a set  $S \subseteq \mathbb{R}^n$  by

$\text{co}(S)$ . For a convex function  $f : \mathfrak{R}^n \rightarrow \mathfrak{R}$ , we let  $\partial f(x)$  denote the subdifferential of  $f$  at  $x$ . Note that  $f$  is minimized at  $x$  over  $\mathfrak{R}^n$  if and only if  $0 \in \partial f(x)$ . For calculus rules on subdifferentials of convex functions, please refer to Chapter VI in [7].

## 2 Smooth approximation

For any  $p > 0$ , define the smoothing log-exponential aggregation function of  $f$  in (3) as

$$f(x; p) = p \ln \sum_{i=1}^m \exp(g_i(x; p)/p) \quad \text{where } g_i(x; p) = r_i + \sqrt{\|x - c_i\|^2 + p^2}. \quad (5)$$

It is easy to show that

$$\lim_{p \rightarrow 0} f(x; p) = f(x) \quad \forall x.$$

**Lemma 1** *The function  $f(x; p)$  has the following properties:*

(i) *For any  $x \in \mathfrak{R}^n$ , and  $p_1, p_2$  satisfying  $0 < p_1 < p_2$ , we have*

$$f(x; p_1) < f(x; p_2);$$

(ii) *For any  $x \in \mathfrak{R}^n$  and  $p > 0$ ,  $f(x) \leq f(x; p) \leq f(x) + p(1 + \ln m)$ ;*

(iii) *For any  $p > 0$ ,  $f(x; p)$  is continuously differentiable and strictly convex.*

**Proof.** (i) For any  $x \in \mathfrak{R}^n$  and  $p_1, p_2$  satisfying  $0 < p_1 < p_2$ , by Jensen's inequality [5],

$$\begin{aligned} \left[ \sum_{i=1}^m (\exp(g_i(x; p_2)))^{1/p_2} \right]^{p_2} &> \left[ \sum_{i=1}^m (\exp(g_i(x; p_2)))^{1/p_1} \right]^{p_1} \\ &> \left[ \sum_{i=1}^m (\exp(g_i(x; p_1)))^{1/p_1} \right]^{p_1}. \end{aligned}$$

Hence,  $f(x; p_1) < f(x; p_2)$ .

(ii) Fix  $p = p_2$  and let  $p_1 \rightarrow 0$  in (i), we have  $f(x) < f(x; p)$ . Let

$$g_\infty(x; p) = \max_{1 \leq i \leq m} \{g_i(x; p)\}. \quad (6)$$

It is readily proven that  $f(x) \leq g_\infty(x; p) \leq f(x) + p$ . Thus, from (5), we have

$$f(x; p) = g_\infty(x; p) + p \ln \sum_{i=1}^m \exp[(g_i(x; p) - g_\infty(x; p))/p] \leq g_\infty(x; p) + p \ln m.$$

Hence

$$f(x) \leq f(x; p) \leq f(x) + p(1 + \ln m).$$

(iii) For any  $p > 0$ , clearly,  $f(x; p)$  is continuously differentiable. Now we prove that  $f(x; p)$  is strictly convex. From (5),

$$\nabla f(x; p) = \sum_{i=1}^m \frac{\lambda_i(x; p)}{h_i(x; p)} (x - c_i), \quad (7)$$

where

$$h_i(x; p) = \sqrt{\|x - c_i\|^2 + p^2}, \quad \tau(x; p) = \sum_{i=1}^m \exp(g_i(x; p)/p), \quad (8)$$

$$\lambda_i(x; p) = \frac{\exp(g_i(x; p)/p)}{\tau(x; p)}. \quad (9)$$

Let

$$Q_{ij} = \frac{(x - c_i)(x - c_j)^T}{h_i(x; p)h_j(x; p)}, \quad i, j = 1, 2, \dots, m.$$

From (7), we get

$$\nabla^2 f(x; p) = \sum_{i=1}^m \left[ \frac{\lambda_i(x; p)}{h_i(x; p)} (I_n - Q_{ii}) + \frac{\lambda_i(x; p)}{p} Q_{ii} - \sum_{j=1}^m \frac{\lambda_i(x; p)\lambda_j(x; p)}{p} Q_{ij} \right].$$

For any  $z \in \mathfrak{R}^n$  with  $z \neq 0$ , by the Cauchy-Schwartz inequality,

$$\|z\|^2 - z^T Q_{ii} z \geq \|z\|^2 - \|z\|^2 \|(x - c_i)/h_i(x; p)\|^2 > 0, \quad \forall i = 1, \dots, m.$$

Thus,

$$\begin{aligned} z^T \nabla^2 f(x; p) z &= \sum_{i=1}^m \frac{\lambda_i(x; p)}{h_i(x; p)} (\|z\|^2 - z^T Q_{ii} z) \\ &\quad + \sum_{i=1}^m \left[ \frac{\lambda_i(x; p)}{p} z^T Q_{ii} z - \sum_{j=1}^m \frac{\lambda_i(x; p)\lambda_j(x; p)}{p} z^T Q_{ij} z \right] \\ &> \sum_{i=1}^m \left[ \frac{\lambda_i(x; p)}{p} z^T Q_{ii} z - \sum_{j=1}^m \frac{\lambda_i(x; p)\lambda_j(x; p)}{p} z^T Q_{ij} z \right] \\ &= \frac{1}{p} \sum_{i=1}^m \lambda_i(x; p) a_i^2 - \frac{1}{p} \left( \sum_{i=1}^m \lambda_i(x; p) a_i \right)^2 \\ &\geq 0, \end{aligned} \quad (10)$$

where  $a_i = z^T (x - c_i)/h_i(x; p)$ . This shows that  $\nabla^2 f(x; p)$  is positive definite. Therefore,  $f(x; p)$  is strictly convex. Note that the inequality (10) follows from the fact that

$$\left| \sum_{i=1}^m \lambda_i(x; p) a_i \right| \leq \sqrt{\sum_{i=1}^m \lambda_i(x; p)} \sqrt{\sum_{i=1}^m \lambda_i(x; p) a_i^2},$$

and  $\sum_{i=1}^m \lambda_i(x; p) = 1$ ,  $\lambda_i(x; p) \geq 0$  for  $i = 1, \dots, m$ .  $\square$

### 3 Log-exponential aggregation algorithm

We now give an algorithm for problem (1) based on Lemma 1, followed by a global convergence result.

**Algorithm 1**

Let  $\sigma \in (0, 1)$ ,  $x^0 \in \mathfrak{R}^n$  and  $p^0 > 0$  be given, and set  $k := 0$ .

**For**  $k = 0, 1, 2, \dots$ , **do**

1. Use an unconstrained minimization method to solve

$$\min_{x \in \mathfrak{R}^n} f(x; p^k). \quad (11)$$

Let the minimizer be  $x^k$ .

2. Set  $p^{k+1} = \sigma p^k$ , increment  $k$  by 1, and return to step 1.

**End**

**Lemma 2** *Let  $\{x^k\}_{k \geq 1}$  be the sequence of points produced by Algorithm 1. Then any limit points of  $\{x^k\}_{k \geq 1}$  are optimal solutions of (1).*

**Proof.** Let  $x^*$  be a limit point of  $\{x^k\}_{k \geq 1}$ . Without loss of generality, we suppose that  $x^k \rightarrow x^*$  as  $k$  tends to  $+\infty$ . From the fact that  $x^k$  is a solution of (11), we have

$$\nabla f(x^k; p^k) = \sum_{i=1}^m \frac{\lambda_i^k}{h_i(x^k; p^k)} (x^k - c_i) = 0,$$

where  $\lambda_i^k = \lambda_i(x^k; p^k)$ , and  $h_i(x; p^k)$  is defined as in (8). Let

$$I(x^*) = \{i : f_i(x^*) = f(x^*), i = 1, 2, \dots, m\}.$$

By noting that

$$\lambda_i^k = \frac{\exp[(g_i(x^k; p^k) - g_\infty(x^k; p^k))/p^k]}{\sum_{i=1}^m \exp[(g_i(x^k; p^k) - g_\infty(x^k; p^k))/p^k]},$$

we have

$$\lim_{k \rightarrow +\infty} \lambda_i^k = 0, \quad i \notin I(x^*). \quad (12)$$

By (12) and the fact that  $\sum_{i=1}^m \lambda_i^k = 1$  and  $\lambda_i^k > 0$ ,  $i = 1, 2, \dots, m$ , the sequence  $\{\lambda_i^k, i = 1, 2, \dots, m\}_{k \geq 1}$  has a convergent subsequence. Without loss of generality, we suppose that

$$\lim_{k \rightarrow +\infty} \lambda_i^k = \lambda_i^*, \quad i \in I(x^*).$$

Then  $\sum_{i \in I(x^*)} \lambda_i^* = 1$  and  $\lambda_i^* \geq 0$ ,  $i \in I(x^*)$ . For  $i \in I(x^*)$ ,

$$t_i^* = \lim_{k \rightarrow +\infty} \frac{x^k - c_i}{h_i(x^k; p^k)} \in \partial f_i(x^*).$$

Thus

$$\lim_{k \rightarrow +\infty} \nabla f(x^k; p^k) = \sum_{i \in I(x^*)} \lambda_i^* t_i^* = 0.$$

This shows that  $0 \in \text{co}(\partial f_i(x^*), i \in I(x^*)) = \partial f(x^*)$ . Therefore,  $x^*$  is an optimal solution of (1).  $\square$

**Theorem 3** *Let  $\{x^k\}_{k \geq 1}$  be the sequence of points produced by Algorithm 1 and  $x^*$  be the unique optimal solution of (1). Then*

$$\lim_{k \rightarrow +\infty} x^k = x^*.$$

**Proof.** For any  $k \geq 1$ , by Lemma 1,

$$f(x^1; p^1) > f(x^1; p^k) \geq f(x^k; p^k) \geq f(x^k). \quad (13)$$

Since  $f(x)$  is coercive, the level set

$$L = \{x \in \mathfrak{R}^n : f(x) \leq f(x^1; p^1)\}$$

is bounded. From (13), we have  $\{x^k\}_{k \geq 1} \subseteq L$ . Hence  $\{x^k\}_{k \geq 1}$  is bounded. As  $x^*$  is the unique solution of (1), it follows from Lemma 2 that

$$\lim_{k \rightarrow +\infty} x^k = x^*.$$

$\square$

In practice, we would stop Algorithm 1 once some stopping criteria are met. Moreover, we will use a first-order, or gradient based, unconstrained minimization algorithm for solving (11) in step 1. In the following algorithm, we will choose a limited-memory BFGS algorithm [1] to solve (11) as this algorithm can solve very large unconstrained optimization problems efficiently. The following is a more practical version of Algorithm 1.

### Algorithm 2

Let  $\sigma \in (0, 1)$ ,  $\epsilon_1, \epsilon_2 > 0$ ,  $x^0 \in \mathfrak{R}^n$  and  $p^0 > 0$  be given, and set  $k := 0$ .

**For**  $k = 0, 1, 2, \dots$ , **until**  $p^k \leq \epsilon_1$ , **do**

1. Use a version of the limited-memory BFGS algorithm to solve (11) approximately, and obtain an  $x^k$  such that  $\|\nabla f(x^k; p^k)\| \leq \epsilon_2$ .
2. Set  $p^{k+1} = \sigma p^k$ , increment  $k$  by 1, and return to step 1.

**End**

## 4 SOCP reformulation

Problem (1) is equivalent to the following problem.

$$\begin{aligned}
& \min && r \\
& \text{s.t.} && \|y - c_i\| + r_i \leq r, \quad i = 1, \dots, m, \\
& && y \in \mathfrak{R}^n.
\end{aligned} \tag{14}$$

It in turns can be reformulated as follows:

$$\begin{aligned}
& \max && -r \\
& \text{s.t.} && -t_i + r = r_i, \\
& && -s_i + y = c_i, \\
& && t_i \geq \|s_i\|, \quad i = 1, \dots, m.
\end{aligned} \tag{15}$$

Let

$$\begin{aligned}
N &= m(n+1), \quad K := \{(t; s) \in R^{n+1} : t \geq \|s\|\}, \quad \mathbf{y} = (r; y) \in R^{n+1}, \\
\mathbf{c} &= -(r_1; c_1; r_2; c_2; \dots; r_m; c_m) \in R^N, \quad \mathbf{A} = -(I_{n+1}, \dots, I_{n+1}) \in R^{(n+1) \times N}, \\
\mathbf{b} &= -(1; 0; \dots; 0) \in R^{n+1}, \quad \mathbf{s} = (t_1; s_1; t_2; s_2; \dots; t_m; s_m) \in R^N,
\end{aligned}$$

where  $I_{n+1}$  is an identity matrix. For later purpose, we use the notation  $\gamma(\alpha; \beta) = \sqrt{\alpha^2 - \|\beta\|^2}$  for  $(\alpha; \beta) \in K$ . The problem (15) can be written as a standard dual second order cone program as follows:

$$\begin{aligned}
(D) \quad & \max && \mathbf{b}^T \mathbf{y}, \\
& \text{s.t.} && \mathbf{s} = \mathbf{c} - \mathbf{A}^T \mathbf{y}, \\
& && \mathbf{s} \in \mathcal{K},
\end{aligned}$$

where  $\mathcal{K} = K \times K \times \dots \times K = K^m$ . Let

$$\mathbf{x} = (\tau_1; x_1; \tau_2; x_2; \dots; \tau_m; x_m) \in R^N.$$

The corresponding primal problem is

$$\begin{aligned}
(P) \quad & \min && \mathbf{c}^T \mathbf{x}, \\
& \text{s.t.} && \mathbf{A} \mathbf{x} = \mathbf{b}, \\
& && \mathbf{x} \in \mathcal{K}.
\end{aligned}$$

This pair of problems (P) and (D) are studied in Nesterov and Nemirovskii [10], Nesterov and Todd [11], and Xue and Ye [21]. We can solve these problems by using a primal-dual interior-point method (IPM) to compute an  $\epsilon$ -optimal solution in polynomial time.

There are IPM based general purpose softwares for solving SOCPs, examples are SeDuMi [14], SDPT3[16], and LOQO [17]. But the problems (P) and (D) have a very

special constraint matrix  $\mathbf{A}$ , and it is beneficial to analyze how such a structure can be exploited when computing the search direction in each IPM iteration. Suppose that at a particular IPM iteration, the current iterate is  $(\mathbf{x}, \mathbf{y}, \mathbf{s})$  and the target barrier parameter is  $\mu$ . To obtain the next iterate, the most expensive computation lies in assembling and solving a linear system of  $n + 1$  equations for the search direction  $\Delta \mathbf{y}$ . By adapting the results from Section 2.3 of [16] for the so-called Nesterov-Todd search direction, such a linear system takes the following form for our present SOCPs:

$$\underbrace{\left( \frac{1}{\sigma^2} J + \sum_{i=1}^m \frac{1}{w_i^2} \mathbf{g}_i \mathbf{g}_i^T \right)}_M \Delta \mathbf{y} = h, \quad (16)$$

where

$$J = \begin{bmatrix} -1 & \\ & I_n \end{bmatrix}, \quad w_i^2 = \frac{\gamma(t_i; s_i)}{\gamma(\tau_i; x_i)}, \quad \frac{1}{\sigma^2} = \sum_{i=1}^m \frac{1}{w_i^2},$$

$$\mathbf{g}_i = \frac{\sqrt{2}}{\gamma(\rho_i; \xi_i)} (-\rho_i; \xi_i), \quad (\rho_i; \xi_i) = \frac{1}{w_i} (t_i; s_i) - w_i (-\tau_i; x_i),$$

and

$$h = \mathbf{b} + \sum_{i=1}^m \frac{1}{w_i^2} \left( J + \mathbf{g}_i \mathbf{g}_i^T \right) ((r_i; c_i) + (t_i; s_i) - \mathbf{y}) - \sum_{i=1}^m \frac{\mu}{\gamma(t_i; s_i)^2} (-t_i; s_i).$$

Note that the matrix  $M$  is symmetric positive definite. In the above, we give a full description of the linear system (16) for the sake of completeness. But our main aim is to study the matrix  $M$ , and explore the possibility of solving (16) via a Krylov subspace method such as the preconditioned conjugate gradient (PCG) method when  $n$  is large.

Let

$$\mathcal{I} = \{i \in \{1, \dots, m\} : w_i \leq \sigma \|\mathbf{g}_i\|\}, \quad \mathcal{J} = \{i \in \{1, \dots, m\} : w_i > \sigma \|\mathbf{g}_i\|\}.$$

We can write the matrix  $M$  in (16) as follows:

$$M = \underbrace{\frac{1}{\sigma^2} \left( I + \sum_{i \in \mathcal{I}} \frac{\sigma^2}{w_i^2} \mathbf{g}_i \mathbf{g}_i^T \right)}_{M_0} + \underbrace{\frac{1}{\sigma^2} \sum_{j \in \mathcal{J}} \frac{\sigma^2}{w_j^2} \mathbf{g}_j \mathbf{g}_j^T}_{M_1} - \frac{2}{\sigma^2} e_1 e_1^T,$$

where  $M_0$  is positive definite,  $M_1$  is positive semidefinite, and  $e_1 = (1; 0; \dots; 0) \in \mathfrak{R}^{n+1}$ . Numerical experiments have shown that the cardinality of  $\mathcal{I}$  is usually much smaller than  $n$  when the barrier parameter  $\mu$  is small. Thus  $M_0$  is a diagonal matrix plus a low-rank matrix, and its inverse can be found readily via the Sherman-Morrison-Woodbury formula:

$$M_0^{-1} = \sigma^2 \left[ I - G(I + G^T G)^{-1} G^T \right], \quad G = \left( \frac{\sigma}{w_i} \mathbf{g}_i \mid i \in \mathcal{I} \right).$$

Suppose  $M_0$  is used as a preconditioner in (16), then the preconditioned matrix  $M_0^{-1}M$  has the form:

$$\begin{aligned} M_0^{-1}M &= I + \frac{1}{\sigma^2}M_0^{-1}M_1 - \frac{2}{\sigma^2}M_0^{-1}e_1e_1^T \\ &= I + \underbrace{\left[ I - G(I + G^T G)^{-1}G^T \right]}_B \left( \sum_{j \in \mathcal{J}} \frac{\sigma^2}{w_j^2} \mathbf{g}_j \mathbf{g}_j^T \right) - \frac{2}{\sigma^2}M_0^{-1}e_1e_1^T. \end{aligned} \quad (17)$$

To analyze the spectrum of  $M_0^{-1}M_1$ , we make use of the following lemma.

**Lemma 4** *Let  $p = |\mathcal{I}|$ . If  $p < n + 1$ , then the matrix  $B$  is positive semidefinite and*

$$\|B\| \leq 1$$

**Proof.** Let the singular value decomposition of  $G \in \mathfrak{R}^{(n+1) \times p}$  be

$$G = U\Sigma V^T,$$

where  $U \in \mathfrak{R}^{(n+1) \times p}$  has orthonormal columns,  $\Sigma \in R^{p \times p}$  is diagonal, and  $V \in \mathfrak{R}^{p \times p}$  is orthogonal. It is easily shown that

$$G(I + G^T G)^{-1}G^T = U\Sigma(I + \Sigma^2)^{-1}\Sigma U^T.$$

Thus

$$\begin{aligned} B &= I - U\Sigma(I + \Sigma^2)^{-1}\Sigma U^T \\ &= (I - UU^T) + U(I + \Sigma^2)^{-1}U^T, \end{aligned} \quad (18)$$

which is the sum of two positive semidefinite matrices, and hence  $B$  is also positive semidefinite. From (18), it is clear that  $x^T Bx \leq x^T x$  for any  $x \in \mathfrak{R}^{n+1}$ . Hence,  $\|B\| \leq 1$ .  $\square$

Notice that  $BM_1$  is similar to a positive semidefinite matrix and thus all its eigenvalues lie in the interval  $[0, \|BM_1\|] \subset [0, \|M_1\|]$ . For  $j \in \mathcal{J}$ ,  $\sigma\|\mathbf{g}_j\|/w_j \leq 1$ , thus we would expect  $\|M_1\|$  to be a moderate constant. Hence, we can expect the preconditioned matrix  $M_0^{-1}M$  to have a favorable spectral distribution (with all its eigenvalues clustered around 1 except possibly for an outlier due to the presence of a rank-1 perturbation in (17)) for the PCG to attain fast convergence.

So far, we have discussed the preconditioning strategy to solve (16) when the barrier parameter  $\mu$  is small. When  $\mu$  is not small, the cardinality of  $\mathcal{I}$  may be comparable to  $n$ , and it would be computationally expensive to use  $M_0$  as a preconditioner. Fortunately when  $\mu$  is not small, the matrix  $M$  itself is quite well-conditioned and preconditioning by the diagonal of  $M$  would be good enough for PCG to obtain fast convergence. In practice, we switch from the diagonal preconditioner to  $M_0$  when

$$\frac{\max(\|\mathbf{g}_i\|/w_i^2)}{\min(\|\mathbf{g}_i\|/w_i^2)} \geq 5 \times 10^2. \quad (19)$$

## 5 Computational results

We implemented Algorithm 2 described in Section 3 and the numerical experiments were done by using a Pentium II 450MHz personal computer. To solve (11), we choose a limited-memory BFGS algorithm with strong Wolfe-Powell line-search and 7 limited-memory vector updates [1]. We implemented our code in Fortran 77 and compared it with SeDuMi [14] and SDPT3 [16] (two high quality software packages with MATLAB interface for solving semidefinite-quadratic-linear programs). We also implemented an adaptation of SDPT3 to exploit the special structure present in our SOCPs in (P) and (D). The main change lies in replacing the direct solution method in computing the direction  $\Delta \mathbf{y}$  in SDPT3 by the PCG iterative solution method discussed in the last section. For convenience, we will call the modified algorithm SDPT3-PCG.

Throughout the computational experiments, we use the following parameters in Algorithm 2:

$$p^0 = 0.01, \quad \sigma = 0.1, \quad \epsilon_1 = 10^{-6}, \quad \text{and} \quad \epsilon_2 = 10^{-5}.$$

For SDPT3, we stop the interior-point algorithm when  $\phi$  is smaller than  $10^{-6}$ . Here

$$\phi = \max \left( \frac{\|\mathbf{b} - \mathbf{A}\mathbf{x}\|}{1 + \|\mathbf{b}\|}, \frac{\|\mathbf{c} - \mathbf{s} - \mathbf{A}^T \mathbf{y}\|}{1 + \|\mathbf{c}\|}, \frac{\mathbf{x}^T \mathbf{s}}{1 + \text{mean}(|\mathbf{b}^T \mathbf{y}|, |\mathbf{c}^T \mathbf{x}|)} \right).$$

We use the same stopping criterion for SDPT3-PCG. When solving the system (16), we stop the PCG iteration when the relative residual norm,  $\|h - M\Delta \mathbf{y}\|/\|h\|$ , is less than  $10^{-8}$ . For SeDuMi, we use all the default values except that the parameter `eps` is set to be  $10^{-6}$ .

The test problems are generated randomly. We use the following pseudo-random sequence:

$$\begin{aligned} \psi_0 &= 7, \quad \psi_{i+1} = (445\psi_i + 1) \bmod 4096, \quad i = 1, 2, \dots, \\ \bar{\psi}_i &= \frac{\psi_i}{40.96}, \quad i = 1, 2, \dots. \end{aligned}$$

The elements of  $c_i, i = 1, 2, \dots, m$ , are successively set to  $\bar{\psi}_1, \bar{\psi}_2, \dots$ , in the order:

$$r_1, c_1(1), c_1(2), \dots, c_1(n), r_2, c_2(1), \dots, c_2(n), \dots, r_m, c_m(1), \dots, c_m(n).$$

The starting point is  $x^0 = \mathbf{0}$ .

The numerical results we obtained are summarized in Tables 1 - 3. In these tables,  $\mathbf{n}$  and  $\mathbf{m}$  denote the dimension of the Euclidean space and the number of balls, respectively, **Obj Value** denotes the value of the objective function at the final iteration, **Iter** denotes the number of iterations, and **Time** denotes the CPU time in second for solving each problem.

The results reported in Tables 1 and 2 show that algorithms presented in this paper perform very well. We can see from Table 1 that all algorithms are able to obtain good accuracy for moderately large problems. Moreover, it appears from Table 2 that Algorithm 2 consistently takes more iterations than the SOCP algorithm and that for all tested problems, Algorithm 2 and SDPT3-PCG consistently use less CPU time than SeDuMi and SDPT3.

Problem		Obj Value			
n	m	SeDuMi	SDPT3	Alg2	SDPT3-PCG
100	1000	3.97018770e+02	3.97018768e+02	3.97018770e+02	3.97018769e+02
200	1000	5.15804093e+02	5.15804093e+02	5.15804093e+02	5.15804090e+02
300	1000	6.04048040e+02	6.04048034e+02	6.04048038e+02	6.04048056e+02
400	1000	6.79603197e+02	6.79603174e+02	6.79603176e+02	6.79603173e+02
500	1000	7.48589676e+02	7.48589651e+02	7.48589655e+02	7.48589651e+02
600	1000	8.09113512e+02	8.09113491e+02	8.09113496e+02	8.09113491e+02
700	1000	8.64883486e+02	8.64883445e+02	8.64883448e+02	8.64883545e+02
800	1000	9.16972254e+02	9.16972205e+02	9.16972207e+02	9.16972204e+02
900	1000	9.66019199e+02	9.66019181e+02	9.66019184e+02	9.66019182e+02
1000	1000	1.01466421e+03	1.01466424e+03	1.01466419e+03	1.01466419e+03

Table 1: Numerical results: Objective function value at the final iteration.

The results reported in Table 3 show Algorithm 2 can solve very large problems in a reasonable amount of CPU time. However, SeDuMi, SDPT3 and SDPT3-PCG fail for the problems in Table 3 due to excessive memory requirement.

We conclude that Algorithm 2 is better than SeDuMi, SDPT3 and SDPT3-PCG for very large problems, since it consumes less memory. For moderately large problems, Algorithm 2 and SDPT3-PCG are comparable.

## References

- [1] R.H. Byrd, P. Lu, J. Nocedal and C. Zhu, A limited memory algorithm for bound constrained optimization, *SIAM J. Sci. Comput.*, **16** (1995) 1190-1208.
- [2] J. Elzinga and D. Hearn, The minimum covering sphere problem, *Management Sci.*, **19** (1972) 96-104.
- [3] B. Gärtner, Fast and robust smallest enclosing balls, J. Neštril, editor, *Algorithms-ESA '99: 7th Annual European Symposium Proceedings*, Lecture Notes in Computer Science 1643, 325-338, Springer-Verlag.
- [4] B. Gärtner, Smallest enclosing ball - fast and robust in C++, <http://www.inf.ethz.ch/personal/gaertner/miniball.html>.
- [5] G.H. Hardy, J.E. Littlewood and G. Polya, *Inequalities*, Cambridge University Press, 1952.
- [6] D.W. Hearn and J. Vijan, Efficient algorithms for the minimum circle problem, *Oper. Res.*, **30** (1982) 777-795.
- [7] J.B. Hiriart-Urruty and C. Lemarechal, *Convex Analysis and Minimization Algorithm*, Springer-Verlag Berlin Heidelberg, 1993.

Problem		SeDuMi		SDPT3		Alg2		SDPT3-PCG			
n	m	Time	Iter	Time	Iter	Time	Iter	Time	Iter		
100	1000	69.4	17	50.2	16	14.5	608	16.4	14	31	13
200	1000	158.8	16	106.8	14	24.8	633	31.1	15	48	13
300	1000	286.2	16	204.6	15	52.0	797	41.8	14	64	14
400	1000	418.0	15	339.0	16	66.5	744	66.6	17	78	12
500	1000	598.8	15	470.3	16	70.2	608	77.5	16	75	16
600	1000	809.8	15	525.3	15	77.8	500	86.8	15	73	16
700	1000	1057.5	15	658.9	16	107.4	627	88.8	13	86	17
800	1000	1374.1	15	762.3	16	161.8	836	125.2	16	91	16
900	1000	1665.9	15	850.1	15	151.7	624	132.2	15	90	17
1000	1000	2033.5	15	883.8	13	138.2	537	137.7	14	94	18

Table 2: Performance comparison of SeDuMi, SDPT3, Algorithm 2 and SDPT3-PCG. The numbers in the last two columns correspond to  $\text{mean}(|\mathcal{I}|)$  and the mean number of PCG iterations in each interior-point iteration, respectively.

n	m	Obj Value	Time	Iter
2000	5000	1.39845777e+03	1814.5	710
3000	5000	1.68162526e+03	3165.0	827
4000	5000	1.92006896e+03	3091.4	558
5000	5000	2.13779784e+03	4564.0	683
6000	5000	2.33369211e+03	6321.4	792
7000	5000	2.51083844e+03	5190.4	681
8000	5000	2.67495154e+03	5761.2	693
9000	5000	2.83239358e+03	5563.9	600
10000	3000	2.97968593e+03	5461.1	664

Table 3: Performance of Algorithm 2 on very large problems.

- [8] X.S. Li, An aggregate function method for nonlinear programming, *Sci. China Ser. A*, **34** (1991) 1467-1473.
- [9] N. Megiddo, Linear-time algorithms for linear programming in  $\mathbb{R}^3$  and related problems, *SIAM J. Comput.*, **12** (1983) 759-776.
- [10] Yu. E. Nesterov and A. Nemirovskii, *Interior Polynomial Algorithms in Convex Programming*, SIAM, Philadelphia, 1994.
- [11] Yu. E. Nesterov and M. J. Todd, Self-scaled barriers and interior-point methods for convex programming, *Math. Oper. Res.*, **22** (1997) 1-42.
- [12] F.P. Preparata and M.I. Shamos, *Computational geometry: an introduction*, Texts and Monographs in Computer Science, Springer-Verlag, New York, 1985.
- [13] M.I. Shamos and D. Hoey, Closest-point problems, in 16th Annual Symposium on Foundations of Computer Science (Berkeley, Calif., 1975), 151–162. IEEE Computer Society, Long Beach, Calif., 1975.
- [14] J.F. Sturm, Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones, *Optim. Methods Softw.*, **11&12** (1999) 625–653.
- [15] P. Sun, and R.M. Freund, *Computation of minimum volume covering ellipsoids*, MIT Operations Research Center Working Paper OR 064-02, July, 2002.
- [16] R.H. Tütüncü, K.C. Toh, and M.J. Todd, Solving semidefinite-quadratic-linear programs using SDPT3, *Math. Programming*, to appear.
- [17] R.J. Vanderbei, LOQO user’s manual—version 3.10, *Optim. Methods Softw.*, **11&12** (1999) 485–514.
- [18] E. Welzl, Smallest enclosing disks (balls and ellipsoids), H. Maurer, editor, *New Results and New Trends in Computer Science*, 359-370, Springer-Verlag, 1991.
- [19] D. White, Enclosing ball software,  
<http://vision.ucsd.edu/~dwhite/ball.html>.
- [20] S. Xu, R. Freund and J. Sun, Solution methodologies for the smallest enclosing circle problem, *Computational Optimization and Applications*, forthcoming.
- [21] G.L. Xue and Y.Y. Ye, An efficient algorithm for minimizing a sum of Euclidean norms with applications, *SIAM J. Optim.*, **7** (1997) 1017-1036.