A REFORMULATION-LINEARIZATION BASED ALGORITHM FOR THE SMALLEST ENCLOSING CIRCLE PROBLEM

YI JIANG
School of Mathematical Sciences and VC/VR Key Lab, Sichuan Normal University
Chengdu, Sichuan 610066, China

YUAN CAI
School of Mathematical Sciences and VC/VR Key Lab, Sichuan Normal University
Chengdu, Sichuan 610066, China

(Communicated by Jie Sun)

Abstract. In this paper, an effective algorithm based on the reformulation-linearization technique (RLT) is developed to solve the smallest enclosing circle problem. Extensive computational experiments demonstrate that the algorithm based on the RLT outperforms the existing algorithms in terms of the solution time and quality in average.

1. Introduction. The smallest enclosing circle is defined as the circle with a minimum radius that encloses all the given circles in a 2D plane. It was initially proposed by Sylvester in 1857 [12]. This problem has many applications in modern manufacturing industry and a variety of location problems such as locating an industrial plant, identifying the location of a single facility, and setting up a radio transmitter to cover a particular region. The detailed descriptions of these applications can be found in [2, 10, 13].

We are given a finite set of circles \( \bar{o} = \{o_1, o_2, \ldots, o_m\} \) on the 2D plane. Let \( C = \{1, 2, \ldots, m\} \). Each circle \( o_i \) centers at \((a_i, b_i)\) with radius \( r_i \), for each \( i \in C \). Straightforwardly, the smallest enclosing circle problem can be formulated as follows:

\[
\begin{align*}
\min & \quad R \\
\text{s.t.} & \quad \sqrt{(x-a_i)^2 + (y-b_i)^2} + r_i \leq R, \quad i \in C,
\end{align*}
\]

where the center of the enclosing circle is denoted by \((x, y)\) with radius \( R \). In fact, problem (1) is second order cone reformulation which can be solved by the CPLEX Solver and SDPT3.

When \( r_i = 0 \) for all \( i \in C \), formulation (1) is reduced to the famous closed point problem that has been investigated extensively, see, e.g., [3, 4]. Then it extends...
to $\mathbb{R}^n$ dimensions and is called the smallest enclosing ball problem which is considered in [15]. When $r_i > 0$ for each $i \in \mathcal{C}$, there exist many approaches to solve problem (1). Xu, Freund, and Sun [14] propose a quadratic programming approach whose numerical performance outperforms those of the subgradient approach, the randomized incremental algorithm, and second order cone reformulation (1) solved by SeDuMi ([11]). More recently, Jiang, Luo, and Ling [5] develop an efficient cutting plane algorithm and their computational experiments demonstrate the cutting plane algorithm outperforms the quadratic programming approach given in [14] and second order cone reformulation (1) solved by the CPLEX Solver. Furthermore, efficient algorithms for computing the small enclosing ball of a set of $m$ balls in $\mathbb{R}^n$ are considered in [16, 9].

In this paper, we propose another algorithm to solve the smallest enclosing circle problem. The algorithm solves formulation (1) as a series of linear programs. The algorithm developed in Section 2 is inspired by the Reformulation-Linearization Technique (RLT) in [6, 7, 8] and the algorithm based on approximation of quadratic terms in [1]. Because the computational results show that the cutting plane algorithm in [5] performs better than the quadratic programming approach in [14], we compare the numerical performances of the algorithm proposed in this paper with those of the cutting plane algorithm in [5] and second order cone reformulation (1) solved by the CPLEX Solver and SDPT3. In Section 3, we demonstrate that the algorithm proposed in this paper outperforms the other two approaches through extensive numerical studies. Furthermore, as an extension, the smallest enclosing ball problem in the 3D space is implemented in Section 4. As confirmed by the computation results, the smallest enclosing ball problem can be efficiently solved by our algorithm as well.

2. The algorithm. In this section, we develop an algorithm to solve the smallest circle problem based on formulation (1). Firstly, one can observe the radius $R$ is bounded. For any fixed $(x_0, y_0)$,

- set $\overline{R} := \max_{i \in \mathcal{C}} \left\{ \sqrt{(x_0 - a_i)^2 + (y_0 - b_i)^2} + r_i \right\}$ and $\underline{R} := \max_{i \in \mathcal{C}} \{r_i\}$, then $\underline{R} \leq R \leq \overline{R}$.

Furthermore, one can see that the center of the enclosing circle $(x, y)$ is bounded as well.

- Set $\overline{a} := \max_{i \in \mathcal{C}} \{a_i\} + \max_{i \in \mathcal{C}} \{r_i\}$ and $\underline{a} := \min_{i \in \mathcal{C}} \{a_i\} - \max_{i \in \mathcal{C}} \{r_i\}$, then $\underline{a} \leq x \leq \overline{a}$.
- Set $\overline{b} := \max_{i \in \mathcal{C}} \{b_i\} + \max_{i \in \mathcal{C}} \{r_i\}$ and $\underline{b} := \min_{i \in \mathcal{C}} \{b_i\} - \max_{i \in \mathcal{C}} \{r_i\}$, then $\underline{b} \leq y \leq \overline{b}$.

By these, problem (1) can be reformulated as follows:

$$\begin{align*}
\min \quad & R \\
\text{s.t.} \quad & (x - a_i)^2 + (y - b_i)^2 \leq (R - r_i)^2, \quad i \in \mathcal{C}, \\
& \underline{a} \leq x \leq \overline{a}, \\
& \underline{b} \leq y \leq \overline{b}, \\
& \overline{R} \leq R \leq \overline{R}.
\end{align*}$$

(2)

In fact, problem (1) is equivalent to problem (2) as they have the same feasible region.

Inspired by the RLT in [6, 7, 8] and the algorithm based on the approximation of quadratic terms [1], we observe the decision variables should be bounded and then
introduce three additional variables \( v_1, v_2, \) and \( v_3 \) into (2). Hence, the following model \( M_{RLT} \) is obtained.

\[
\begin{align*}
\text{min} & \quad R \\
\text{s.t.} & \quad v_1 = x^2, v_2 = y^2, v_3 = R^2, \quad (3) \\
& \quad -2a_1x - 2b_1y + 2r_1R + v_1 + v_2 - v_3 \leq r_i^2 - a_i^2 - b_i^2, \quad i \in C, \quad (4) \\
& \quad v_1 - (\bar{a} + \bar{b})x \leq -\bar{a}x, v_2 - (\bar{b} + \bar{b})y \leq -\bar{b}y, v_3 - (\bar{R} + \bar{R})R \leq -\bar{R}R, \quad (5) \\
& \quad 2\bar{a}x - v_1 \leq \bar{a}^2, 2\bar{a}y - v_2 \leq \bar{b}^2, 2\bar{b}y - v_2 \leq \bar{b}^2, \quad (6) \\
& \quad 2\bar{R}R - v_3 \leq \bar{R}^2, 2\bar{R}R - v_3 \leq \bar{R}^2, \quad (7) \\
& \quad v_1 \geq 0, v_2 \geq 0, v_3 \geq 0. \quad (8)
\end{align*}
\]

**Theorem 2.1.** Model \( M_{RLT} \) is equivalent to problem (2).

**Proof.** Let \((\hat{x}, \hat{y}, \hat{R}, \hat{v}_1, \hat{v}_2, \hat{v}_3)\) be a feasible solution to \( M_{RLT} \). Then it satisfies the conditions (3)-(8). Since \( \hat{v}_1 = \hat{x}^2, \hat{v}_2 = \hat{y}^2 \) and \( \hat{v}_3 = \hat{R}^2 \), then

\[
-2a_1\hat{x} - 2b_1\hat{y} + 2r_1\hat{R} + \hat{v}_1 + \hat{v}_2 - \hat{v}_3 \leq r_i^2 - a_i^2 - b_i^2, \quad \forall i \in C,
\]

which is equivalent to

\[
(\hat{x} - a_i)^2 + (\hat{y} - b_i)^2 \leq (\hat{R} - r_i)^2, \quad \forall i \in C.
\]

Moreover, one can also verify

\[
\begin{align*}
\hat{v}_1 - (\bar{a} + \bar{b})\hat{x} & \leq -\bar{a}\hat{x} \iff \bar{a} \leq \hat{x} \leq \bar{a}, \\
2\bar{a}\hat{x} - \hat{v}_1 & \leq \bar{a}^2 \iff (\bar{a} - \hat{x})^2 \geq 0, \\
2\bar{a}\hat{x} - \hat{v}_1 & \leq \bar{a}^2 \iff (\bar{a} - \hat{x})^2 \geq 0, \\
\hat{v}_1 & \geq 0 \iff \hat{x}^2 \geq 0.
\end{align*}
\]

Similarly, we can also verify that both \( \hat{y} \) and \( \hat{R} \) satisfy the \( \bar{b} \leq \hat{y} \leq \bar{b}, \) and \( \bar{R} \leq \hat{R} \leq \bar{R} \). Hence, \((\hat{x}, \hat{y}, \hat{R})\) satisfies the following conditions:

- \((\hat{x} - a_i)^2 + (\hat{y} - b_i)^2 \leq (\hat{R} - r_i)^2, \quad \forall i \in C; \)
- \( \bar{a} \leq \hat{x} \leq \bar{a}, \bar{b} \leq \hat{y} \leq \bar{b}, \) and \( \bar{R} \leq \hat{R} \leq \bar{R} \).

which show that \((\hat{x}, \hat{y}, \hat{R})\) is a feasible solution of (2).

Conversely, let \((\hat{x}, \hat{y}, \hat{R})\) be a feasible solution of (2). Similarly, we can prove \((\hat{x}, \hat{y}, \hat{R}, \hat{v}_1, \hat{v}_2, \hat{v}_3)\) is a feasible solution to \( M_{RLT} \) in which \( \hat{v}_1 = \hat{x}^2, \hat{v}_2 = \hat{y}^2 \) and \( \hat{v}_3 = \hat{R}^2 \).

Without constrain condition (3), one can get the linear programming relaxation of problem \( M_{RLT} \) as follows:

\[
\begin{align*}
\text{min} & \quad R \\
\text{s.t.} & \quad (4) - (8).
\end{align*}
\]

**Theorem 2.2.** Let \((x^*, y^*, R^*, v_1^*, v_2^*, v_3^*)\) be the optimal solution to (10) in which \( v_1^* = x^*^2, v_2^* = y^*^2, \) and \( v_3^* = R^*^2 \). Then \((x^*, y^*, R^*)\) is the optimal solution to (2).

**Proof.** Because \((x^*, y^*, R^*, v_1^*, v_2^*, v_3^*)\) is optimal to (10), \((x^*, y^*, R^*)\) is feasible to (2). Suppose that \((x^*, y^*, R^*)\) is not optimal to problem (2). Then there exists an optimal solution \((\hat{x}, \hat{y}, \hat{R})\) to (2) such that \( \hat{R} < R^* \). Let \( \hat{v}_1 = \hat{x}^2, \hat{v}_2 = \hat{y}^2 \) and
\[ \hat{v}_3 = \hat{R}^2. \] Theorem 2.1 implies that \((\hat{x}, \hat{y}, \hat{R}, v_1, v_2, \hat{v}_3)\) is a feasible solution of (10). Since \(\hat{R} < \hat{R}^*\), we can see that it contradicts that \(\hat{R}^*\) is an optimal solution of (10). Therefore, the hypothesis is invalid, \((x^*, y^*, \hat{R}^*)\) is an optimal solution of (2). \[ \square \]

In fact, problem (10) is a linear programming problem, which can be solved efficiently. One can see that if \(v_1 = x^2, v_2 = y^2\) and \(v_3 = \hat{R}^2\), then \(\hat{R}\) is optimal for problem (1). Therefore, we reduce \(\hat{R}\) and add valid cutting planes to problem (10).

Based on this idea, we propose the following algorithm to solve the enclosing circle problem.

**Algorithm I**

Step 1. Start with \((x_0, y_0) = (0, 0)\) and compute

\[ \mathcal{R} := \max_{i \in C} \left\{ \sqrt{(x_0 - a_i)^2 + (y_0 - b_i)^2 + r_i} \right\}. \]

Step 2. Initialize: \(k = 0\). Use the CPLEX LP Solver to solve problem (10) and obtain the optimal solution \((x_k, y_k, R_k, v_{1k}, v_{2k}, v_{3k})\) to (10).

Step 3. Do while \(|x_k^2 - v_{1k}| > \varepsilon, |y_k^2 - v_{2k}| > \varepsilon\) and \(|R_k^2 - v_{3k}| > \varepsilon\) (where \(\varepsilon\) is a predefined tolerance). Re-compute

\[ \mathcal{R} := \max_{i \in C} \left\{ \sqrt{(x_k - a_i)^2 + (y_k - b_i)^2 + r_i} \right\}. \]

Add the following valid inequalities

\[ \begin{align*}
v_1 - 2\alpha_1 x + \alpha_1^2 &\geq 0, \\
v_2 - 2\alpha_2 y + \alpha_2^2 &\geq 0, \\
v_3 - 2\alpha_3 R + \alpha_3^2 &\geq 0,
\end{align*} \tag{11} \]

into problem (10). Let

\[ \alpha_1 := \begin{cases} (\sqrt{2} - 1)x_k + (2 - \sqrt{2})\sqrt{v_{1k}}, & \text{if } x_k \geq 0; \\
(\sqrt{2} - 1)x_k + (\sqrt{2} - 2)\sqrt{v_{1k}}, & \text{otherwise}, \end{cases} \]

\[ \alpha_2 := \begin{cases} (\sqrt{2} - 1)y_k + (2 - \sqrt{2})\sqrt{v_{2k}}, & \text{if } y_k \geq 0; \\
(\sqrt{2} - 1)y_k + (\sqrt{2} - 2)\sqrt{v_{2k}}, & \text{otherwise}, \end{cases} \]

and

\[ \alpha_3 := \begin{cases} (\sqrt{2} - 1)R_k + (2 - \sqrt{2})\sqrt{v_{3k}}, & \text{if } R_k \geq 0; \\
(\sqrt{2} - 1)R_k + (\sqrt{2} - 2)\sqrt{v_{3k}}, & \text{otherwise}. \end{cases} \]

Obtain the following linear program:

\[ \begin{align*}
\min & \quad R \\
\text{s.t.} & \quad (4) - (8), \\
& \quad v_1 - 2\alpha_1 x + \alpha_1^2 \geq 0, \\
& \quad v_2 - 2\alpha_2 y + \alpha_2^2 \geq 0, \\
& \quad v_3 - 2\alpha_3 R + \alpha_3^2 \geq 0. \tag{12}
\end{align*} \]

Solve problem (12) by the CPLEX LP Solver. Update \(k := k + 1\) and obtain the new optimal solution \((x_k, y_k, R_k, v_{1k}, v_{2k}, v_{3k})\) to problem (12).

Algorithm I proceeds by first considering linear programming relaxation (10). Then, valid cuts are added into (10) to approximate the feasible region of (2). To show the validity of Algorithm I, we should prove: (1) the cutting planes are valid
inequalities; (2) Algorithm I is convergent; and (3) the optimal solution returned by Algorithm I is also the optimal solution to problem (1).

**Theorem 2.3.** Cutting planes (11) are valid cuts. More precisely, if \((x_k, y_k, R_k, v_{1k}, v_{2k}, v_{3k})\) belongs to the feasible region of problem (10), then \((x_k, y_k, R_k, v_{1k}, v_{2k}, v_{3k})\) also belongs to the feasible region of problem (12).

**Proof.** Let \((x_k, y_k, R_k, v_{1k}, v_{2k}, v_{3k})\) is a feasible solution of (10).

If \(x_k^2 \neq v_{1k}, y_k^2 \neq v_{2k}, R_k^2 \neq v_{3k}\), then the convexity of quadratic functions implies

\[
v_{1k} - 2\alpha x_k + \alpha^2 \geq 0, \tag{13}
\]
\[
v_{2k} - 2\alpha y_k + \alpha^2 \geq 0, \tag{14}
\]
and
\[
v_{3k} - 2\alpha R_k + \alpha^2 \geq 0, \tag{15}
\]
where \((\alpha, \alpha^2)\) is on quadratic functions. Now, take \(\alpha_1\) as an example to show how to determine the value of \(\alpha\) in inequality (13). The method is inspired by \([1]\). In the following, we divide the discussion into four cases:

1. \(0 \leq x_{k-1} < \sqrt{v_{1k-1}}\),
2. \(\sqrt{v_{1k-1}} < x_{k-1} < 0\),
3. \(-\sqrt{v_{1k-1}} < x_{k-1} < 0\), and
4. \(x_{k-1} < -\sqrt{v_{1k-1}}\).

**Case (1).** When \(0 \leq x_{k-1} < \sqrt{v_{1k-1}}\), we propose to select the point \(\alpha\) that minimizes a potential error over the interval \([x_{k-1}, \sqrt{v_{1k-1}}]\). Let

\[
\epsilon_1 := v_{1k-1} - (2\alpha\sqrt{v_{1k-1}} - \alpha^2) \quad \text{and} \quad \alpha \epsilon_2 := \left(\frac{x_{k-1}^2}{2} + \frac{\alpha^2}{2}\right) - \alpha x_{k-1}.
\]

It is well illustrated in Figure 1. The least value of the maximum between \(\epsilon_1\) and \(\alpha \epsilon_2\) is attained when both terms are equal. One can verify that \(\epsilon_1 = \alpha \epsilon_2\) if and only if

\[
\alpha^2 + 2(x_{k-1} - 2\sqrt{v_{1k-1}})\alpha + 2v_{1k-1} - x_{k-1}^2 = 0.
\]

![Figure 1. The error for under-estimating a square.](image)
This quadratic function has two roots
\[ \alpha_1 = (\sqrt{2} - 1)x_{k-1} + (2 - \sqrt{2})\sqrt{v_{1k-1}} \quad \text{and} \quad \alpha_1^* = (2 + \sqrt{2})\sqrt{v_{1k-1}} - (\sqrt{2} + 1)x_{k-1}. \]
One can see that \( \alpha_1 \) is a convex combination of the endpoints in the interval \([x_{k-1}, \sqrt{v_{1k-1}}]\).

**Case (2).** If \( \sqrt{v_{1k-1}} < x_{k-1} \), then potential errors in the interval \([\sqrt{v_{1k-1}}, x_{k-1}]\) are defined as follows:
\[ \varepsilon_1 := v_{1k-1} - (2\alpha\sqrt{v_{1k-1}} - \alpha^2) \quad \text{and} \quad \alpha\varepsilon_2 := (\frac{x_{k-1}^2}{2} + \frac{\alpha^2}{2}) - \alpha x_{k-1}. \]
The least value of the maximum between \( \varepsilon_1 \) and \( \alpha\varepsilon_2 \) is attained when both terms are equal. One can verify that \( \varepsilon_1 = \alpha\varepsilon_2 \) if and only if
\[ \alpha^2 + 2(x_{k-1} - 2\sqrt{v_{1k-1}})\alpha + 2v_{1k-1} - x_{k-1}^2 = 0. \]
This quadratic function has two roots
\[ \alpha_1 = (\sqrt{2} - 1)x_{k-1} + (2 - \sqrt{2})\sqrt{v_{1k-1}} \quad \text{and} \quad \alpha_1^* = (2 + \sqrt{2})\sqrt{v_{1k-1}} - (\sqrt{2} + 1)x_{k-1}. \]
One can see that \( \alpha_1 \) is a convex combination of the endpoints in the interval \([\sqrt{v_{1k-1}}, x_{k-1}]\).

**Case (3).** If \( -\sqrt{v_{1k-1}} < x_{k-1} < 0 \), then potential errors over the interval \([-\sqrt{v_{1k-1}}, x_{k-1}]\) are defined as follows:
\[ \epsilon_1 := v_{1k-1} + 2\alpha\sqrt{v_{1k-1}} + \alpha^2 \quad \text{and} \quad -\alpha\epsilon_2 := (\frac{x_{k-1}^2}{2} + \frac{\alpha^2}{2}) - \alpha x_{k-1}. \]
The least value of the maximum between \( \epsilon_1 \) and \( -\alpha\epsilon_2 \) is attained when both terms are equal. One can verify that \( \epsilon_1 = -\alpha\epsilon_2 \) if and only if
\[ \alpha^2 + 2(x_{k-1} + 2\sqrt{v_{1k-1}})\alpha + 2v_{1k-1} - x_{k-1}^2 = 0. \]
This quadratic function has two roots
\[ \alpha_1 = (\sqrt{2} - 1)x_{k-1} + (\sqrt{2} - 2)\sqrt{v_{1k-1}} \quad \text{and} \quad \alpha_1^* = -((\sqrt{2} + 1)x_{k-1} - (\sqrt{2} + 2)\sqrt{v_{1k-1}}. \]
One can see that \( \alpha_1 \) is a convex combination of the endpoints in the interval \([-\sqrt{v_{1k-1}}, x_{k-1}]\).

**Case (4).** If \( x_{k-1} < -\sqrt{v_{1k-1}} \), then potential errors over the interval \([x_{k-1}, -\sqrt{v_{1k-1}}]\) are defined as follows:
\[ \epsilon_1 := v_{1k-1} + 2\alpha\sqrt{v_{1k-1}} + \alpha^2 \quad \text{and} \quad -\alpha\epsilon_2 := (\frac{x_{k-1}^2}{2} + \frac{\alpha^2}{2}) - \alpha x_{k-1}. \]
The least value of the maximum between \( \epsilon_1 \) and \( -\alpha\epsilon_2 \) is attained when both terms are equal. One can verify that \( \epsilon_1 = -\alpha\epsilon_2 \) if and only if
\[ \alpha^2 + 2(x_{k-1} + 2\sqrt{v_{1k-1}})\alpha + 2v_{1k-1} - x_{k-1}^2 = 0. \]
This quadratic function has two roots
\[ \alpha_1 = (\sqrt{2} - 1)x_{k-1} + (\sqrt{2} - 2)\sqrt{v_{1k-1}} \quad \text{and} \quad \alpha_1^* = -((\sqrt{2} + 1)x_{k-1} - (\sqrt{2} + 2)\sqrt{v_{1k-1}}. \]
One can see that \( \alpha_1 \) is a convex combination of the endpoints in the interval \([x_{k-1}, -\sqrt{v_{1k-1}}]\).
In conclusion, one can see that the value of $\alpha$ in (13) is determined by $\alpha_1$ and

$$\alpha_1 = \begin{cases} (\sqrt{2} - 1)x_{k-1} + (2 - \sqrt{2})\sqrt{v_{1k-1}}, & \text{if } x_{k-1} \geq 0, \\ (\sqrt{2} - 1)x_{k-1} + (\sqrt{2} - 2)\sqrt{v_{1k-1}}, & \text{if } x_{k-1} < 0. \end{cases}$$

By the same methods, one can obtain that the value of $\alpha$ in (14) is determined by $\alpha_2$ as follows:

$$\alpha_2 = \begin{cases} (\sqrt{2} - 1)y_{k-1} + (2 - \sqrt{2})\sqrt{v_{2k-1}}, & \text{if } y_{k-1} \geq 0, \\ (\sqrt{2} - 1)y_{k-1} + (\sqrt{2} - 2)\sqrt{v_{2k-1}}, & \text{if } y_{k-1} < 0, \end{cases}$$

and the value of $\alpha$ in (15) is determined by $\alpha_3$ as follows:

$$\alpha_3 = \begin{cases} (\sqrt{2} - 1)R_{k-1} + (2 - \sqrt{2})\sqrt{v_{3k-1}}, & \text{if } R_{k-1} \geq 0, \\ (\sqrt{2} - 1)R_{k-1} + (\sqrt{2} - 2)\sqrt{v_{3k-1}}, & \text{if } R_{k-1} < 0. \end{cases}$$

If $x_k^2 = v_{1k}, y_k^2 = v_{2k}, R_k^2 = v_{3k}$, then $(x_k - \alpha_1)^2 \geq 0, (y_k - \alpha_2)^2 \geq 0$, and $(R_k - \alpha_3)^2 \geq 0$ are natural set up, so expand those equations, we have

$$v_{1k} - 2\alpha_1 x_k + \alpha_1^2 \geq 0, \quad v_{2k} - 2\alpha_2 y_k + \alpha_2^2 \geq 0, \quad \text{and} \quad v_{3k} - 2\alpha_3 R_k + \alpha_3^2 \geq 0.$$ 

Therefore, $(x_k, y_k, R_k, v_{1k}, v_{2k}, v_{3k})$ is a feasible solution of (12).

\begin{proof}
Let $\{R_k\}_{k=1}^\infty$ and $\{(x_k, y_k)\}_{k=1}^\infty$ be the sequence of points produced by Algorithm I. Then they are convergent.

Proof. $\{R_k\}_{k=1}^\infty$ is convergent because it is non-decreasing monotone and bounded with $0 < R_k < R$. Moreover, from

$$0 \leq \sqrt{x_k^2 + y_k^2} \leq \overline{R}(x_k, y_k) := \max_{i \in \mathbb{C}} \{\sqrt{(x_k - a_i)^2 + (y_k - b_i)^2} + r_i\} \leq \overline{R}(x_0, y_0),$$

we can observe that $\{\overline{R}_k\}_{k=1}^\infty$ is bounded and decreasing monotone, where $\overline{R}_k := \overline{R}(x_k, y_k)$. It implies $\{(x_k, y_k)\}_{k=1}^\infty$ is decreasing monotone and bounded $\mathbb{R}^2$, then it is convergent.

\end{proof}

\begin{proof}
If $(x^*, y^*, R^*)$ is an optimal solution produced by Algorithm I, then $(x^*, y^*, R^*)$ is also an optimal solution to problem (2).

Proof. If $(x^*, y^*, R^*)$ is not an optimal solution of problem (2), then there exists an optimal solution $(x, y, R)$ and $R < R^*$. By Theorem 2.3, one can see $(x, y, R)$ is belonging to the feasible regions of problem (12). Since $R < R^*$, it contradicts with $(x^*, y^*, R^*)$ be an optimal solution produced by Algorithm I. Therefore, the hypothesis is invalid, $(x^*, y^*, R^*)$ is also an optimal solution to problem (2).

\end{proof}

\begin{remark}
From Theorem 2.2, Theorem 2.4 and Theorem 2.5, it is clear that the termination criterion of Algorithm I is well defined as

$$|x_k^2 - v_{1k}| \leq \varepsilon, \quad |y_k^2 - v_{2k}| \leq \varepsilon, \quad \text{and} \quad |R_k^2 - v_{3k}| \leq \varepsilon,$$

where $\varepsilon$ is the predefined tolerance.

\end{remark}
3. **Computational results in 2D.** In this section, through extensive numerical studies, we demonstrate that the Algorithm I proposed in Section 2 outperforms second order cone optimization (1) solved by the CPLEX Solver and SDPT3, and the cutting plane algorithm given in [5].

The numerical results are summarized in the following Tables 1-3 and Figure 2. In these tables, $m$ and $k$ denote the number of circles and the number of iterations, respectively. **Obj Value** represents the value of the objective function that is the radius of the smallest enclosing circle. **Time** denotes the average CPU time in millisecond for solving each problem. **SOCP** represents the method to solve second order cone optimization (1) by CPLEX Solver. **SDPT3** represents the method to solve second order cone optimization (1) by using SDPT3. **CP** denotes the best algorithm in [5]. **Algorithm I** denotes the method presented in this paper.

In Algorithm I, we set the value of starting point $(x_0, y_0) = (0, 0)$ and the tolerance $\varepsilon = 10^{-4}$ respectively. The test data is generated randomly. The center of each input circle $\{a_i, b_i\}$ ($i \in C$) are generated by an independent normal distribution $N(0,16)$, and the radii $r_i$ ($i \in C$) are generated according to a uniform distribution $U(0, 1)$. Furthermore, we make sure that no circle are contained in any other. Throughout the computational experiments, we use C SHARP and an Intel core 1.8 GHz personal computer with 4GB memory.

Taking the 30000 circles instance as an example, from Table 1, we can see that it takes four iterations to reach the final results.

**Table 1.** Computational results for 30000 circles by Algorithm I in 2D.

| k | R          | $(x, y)$               | Time  |
|---|------------|------------------------|-------|
| 1 | 22.39739458 | (-0.23034495, 1.54096380) | 1256  |
| 2 | 22.45136334 | (-0.22954489, 1.54016746) | 4437  |
| 3 | 22.45378444 | (-0.22950900, 1.54013173) | 4705  |
| 4 | 22.45382627 | (-0.22950838, 1.54013111) | 4817  |

In Table 2, the test problem is random instances of different scales, which ranges from 30 to 30000 circles, and 50 random instances are solved for each problem size. The results reported in Table 2 show that the number of iterations on Algorithm I for different instances. One can see that the maximum of iterations is six and the minimum of iterations is three. The average value of iterations is about four. Moreover, it is not hard to see there is no relation between the number of iterations and the number of circles.

**Table 2.** The number of iterations on Algorithm I in 2D.

| Problem | Iteration | Average | Maximum | Minimum |
|---------|-----------|---------|---------|---------|
| m       |           |         |         |         |
| 30      |           | 4.48    | 6       | 4       |
| 300     |           | 4.12    | 5       | 3       |
| 3000    |           | 3.78    | 4       | 3       |
| 30000   |           | 3.70    | 4       | 3       |
Now, we show the numerical performance comparison of the Algorithm I presented in this paper, SOCP, SDPT3 and CP given in [5].

Table 3. Objective function value and average CPU time of four methods in 2D.

| Problem | SOCP | CP | SDPT3 | Algorithm I |
|---------|------|----|-------|-------------|
| m       | Obj Value | Time | Obj Value | Time | Obj Value | Time | Obj Value | Time |
| 30      | 9.13411 | 128.98 | 9.13411 | 125.80 | 9.13411 | 1744.45 | 9.13411 | 66.50 |
| 300     | 15.05421 | 179.80 | 15.05421 | 165.26 | 15.05421 | 6848.30 | 15.05421 | 70.60 |
| 3000    | 18.31612 | 781.82 | 18.31612 | 918.92 | 18.31612 | 65688.71 | 18.31612 | 530.66 |
| 30000   | 21.21799 | 25641.96 | 21.21799 | 15716.18 | 21.21799 | 1548266.51 | 21.21799 | 5052.26 |

From Table 3, one can see that all algorithms are able to get accurate results for all test problem scales and the average CPU time of SDPT3 is much slower than SOCP, CP, and Algorithm I. To show the different average CPU time of SOCP, CP, and Algorithm I exactly, we give a figure.

Figure 2. Average CPU time of three methods in 2D.

In Figure 2, the test problem randomly ranges from 10 to 250 circles, and 50 random instances are solved for each problem size. The results reported in Figure 2 and Table 3 show that the four methods can get the optimal solution in reasonable time. One can observe that when the number of circle is small, SOCP and CP perform similarly. But, for the large problems, the average CPU time of CP is quicker than SOCP. Furthermore, for all the test problem scales, the average CPU time of Algorithm I is quicker than SOCP, SDPT3, and CP. It is interesting to note that as the number of circles increases, the time difference between Algorithm I and the other three methods increases gradually. Therefore, Algorithm I outperforms SOCP, SDPT3, and CP in terms of the solution time and quality in average.
4. *Computational results in 3D.* In this section, we consider the smallest enclosing ball problem in the three-dimensional as follows:

\[
\min R \\
\text{s.t. } \sqrt{(x - a_i)^2 + (y - b_i)^2 + (z - c_i)^2} + r_i \leq R, \quad i = 1, 2, \ldots, m.
\]

(17)

where the optimal center of the enclosing ball is \((x, y, z)\) with the radius \(R\), \(\bar{o} = \{o_1, \ldots, o_m\}\) is a set of given ball on \(\mathbb{R}^3\) plane with centers \((a_1, b_1, c_1), \ldots, (a_m, b_m, c_m)\) and radii \(\{r_1, \ldots, r_m\}\).

We implement Algorithm I in this paper, CPLEX Solver and SDPT3 to solve second order cone optimization (17), and CP given in [5]. The test data is generated randomly as that in Section 3. In Algorithm I, the starting point and the tolerance are set to be \((x_0, y_0) = (0, 0)\) and \(\varepsilon = 10^{-5}\) respectively. Now, we show the computational experiments.

In Table 4, we take the 20000 balls instance as an example. It only takes five iterations to get the optimal solution by Algorithm I.

| k  | R   | (x, y, z)                       | Time |
|----|-----|---------------------------------|------|
| 1  | 21.18238030 | (-0.41033399, 0.50362615, -1.77470296) | 2682 |
| 2  | 21.25471838 | (-0.40932025, 0.50524350, -1.77617447) | 4658 |
| 3  | 21.25805248 | (-0.40927352, 0.50531805, -1.77624230) | 6623 |
| 4  | 21.25808485 | (-0.40927307, 0.50531877, -1.77624296) | 10525 |
| 5  | 21.25808512 | (-0.40927306, 0.50531878, -1.77624296) | 14668 |

Table 5. The number of iterations on Algorithm I in 3D.

| Problem | Iteration |
|---------|-----------|
| m       | Average   | Maximum | Minimum |
| 20      | 8.96      | 13      | 5       |
| 200     | 6.54      | 11      | 4       |
| 2000    | 6.68      | 12      | 4       |
| 20000   | 5.84      | 10      | 4       |

In Table 5, the test problem is random instances of different scales, which ranges from 20 to 20000 balls, and 50 random instances are solved for each problem size. The results reported in Table 5 show that the number of iterations on Algorithm I for different instances. One can see that the maximum of iterations is thirteen and the minimum of iterations is three. The average value of iterations is about seven, which is bigger than that in the smallest enclosing circle problem. Moreover, one can also see that there is no relation between the number of iterations and the number of balls.

Now, we show the numerical performance comparison of the Algorithm I presented in this paper, SOCP, SDPT3, and CP given in [5].

From Table 6, one can see that all algorithms are able to get accurate results for all test problem scales and the average CPU time of SDPT3 is much slower than
SOCP, CP, and Algorithm I. To show the different average CPU time of SOCP, CP, and Algorithm I exactly, we give a figure.

In Figure 3, the test problem randomly ranges from 10 to 250 circles, and 50 random instances are solved for each problem size. The results reported in Figure 3 and Table 6 show that the four methods can get the optimal solution in reasonable time. One can observe that the average CPU time of SOCP is quicker than CP. It is different from the performance in the smallest enclosing circle problem. For all the test problem scales, the average CPU time of Algorithm I is quicker than SOCP, SDPT3, and CP as well. It is interesting to note that as the number of balls increases, the time difference between Algorithm I and the other three methods increases gradually. As confirmed by the computation results, the smallest enclosing ball problem can also be efficiently solved by Algorithm I.

| Problem | SOCP | CP | SDPT3 | Algorithm 1 |
|---------|------|----|-------|-------------|
| m       | Obj Value | Time | Obj Value | Time | Obj Value | Time | Obj Value | Time |
| 20      | 10.20133 | 131.26 | 10.20133 | 138.96 | 10.20133 | 1589.39 | 10.20133 | 120.92 |
| 200     | 13.90667 | 168.56 | 13.90667 | 251.66 | 13.90667 | 5157.42 | 13.90667 | 123.88 |
| 2000    | 16.47548 | 633.00 | 16.47548 | 1694.06 | 16.47548 | 43022.79 | 16.47548 | 161.02 |
| 20000   | 20.54257 | 16953.74 | 20.54257 | 18185.18 | 20.54257 | 907801.50 | 20.54257 | 1462.04 |

**Figure 3.** Average CPU time of three methods in 3D.

**REFERENCES**

[1] C. Audet, P. Hansen, B. Jaumard and G. Savard, *A branch and cut algorithm for nonconvex quadratically constrained quadratic programming*, *Math. Program.*, 87 (2000), 131–152.
[2] M. de Berg, O. Cheong, M. van Kreveld and M. Overmars. *Computational Geometry: Algorithms and Applications*, Springer, Berlin, Heidelberg, 2008.

[3] P. Chrystal. On the problem to construct the minimum circle enclosing n given points in a plane, *Proceedings of the Edinburgh Mathematical Society*, 3 (1884), 30–33.

[4] D. W. Hearn and J. Vijay. Efficient algorithms for the (weighted) minimum circle problem, *Oper. Res.*, 30 (1982), 777–795.

[5] Y. Jiang, C. Luo and S. G. Ling. An efficient cutting plane algorithm for the smallest enclosing circle problem, *J. Ind. Manag. Optim.*, 13 (2017), 147–153.

[6] H. D. Sherali and C. H. Tuncbilek. Global optimization algorithm for polynomial programming using a reformulation-linearization technique, *J. Global Optim.*, 2 (1992), 101–112.

[7] H. D. Sherali and C. H. Tuncbilek. Reformulation-convexification approach for solving non-convex quadratic programming problems, *J. Global Optim.*, 7 (1995), 1–31.

[8] H. D. Sherali and C. H. Tuncbilek. Comparison of two reformulation-linearization technique based linear programming relaxations for polynomial programming problems, *J. Global Optim.*, 10 (1997), 381–390.

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

[10] S. Skyum. A simple algorithm for computing the smallest enclosing circle, *Inform. Process. Lett.*, 37 (1991), 121–125.

[11] J. Sturm. Using SeDuMi 1.02, A MATLAB toolbox for optimization over symmetric cones. Interior point methods, *Optim. Methods Softw.*, 11/12 (1999), 625—653.

[12] J. J. Sylvester. A question in the geometry of situation, *Quart. J. Math.*, 1 (1857), 79.

[13] E. Welzl. Smallest enclosing disks (balls and ellipsoids), in *New Results and New Trends in Computer Science*, Springer, Berlin, 1991, 359–370.

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

[15] E. A. Yildirim. Two algorithms for the minimum enclosing ball problem, *SIAM J. Optim.*, 19 (2008), 1368–1391.

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

Received March 2020; revised June 2020.

E-mail address: yijiang103@163.com
E-mail address: yuan-cai1994@163.com