AN EFFICIENT LOW COMPLEXITY ALGORITHM FOR BOX-CONSTRAINED WEIGHTED MAXIMIN DISPERSION PROBLEM

Zi Xu*, Siwen Wang and Jinjin Huang
Department of Mathematics, College of Sciences, Shanghai University
200444 Shanghai, China

(Communicated by Jinyan Fan)

ABSTRACT. The box-constrained weighted maximin dispersion problem is to find a point in an \( n \)-dimensional box such that the minimum of the weighted Euclidean distance from given \( m \) points is maximized. In this paper, we first propose a two-phase method to solve it. In the first phase, we adopt a block successive upper bound minimization (BSUM) algorithm framework and choose a special piecewise linear upper bound function for the weighted maximin dispersion problem. The per-iteration complexity of our algorithm is very low, since the subproblem is a one-dimensional piecewise linear minimax problem over the box constraints, or equivalently, a two-dimensional linear programming problem which can be solved in at most \( O(m) \) time by existing algorithms. In the second phase, a useful rounding is employed to enhance the solution. Moreover, we propose another strengthened two-phase algorithm, which employs a maximum improvement successive upper-bound minimization (MISUM) algorithm instead of BSUM algorithm in the first phase. At each step, only the block that provides the maximum improvement of the upper bound function is updated. Then, it can be proved that every limit point of the iterate generated by this strengthened algorithm is a stationary point. Numerical results show that the proposed algorithms are efficient.

1. Introduction. Consider the following weighted maximin problem:

\[
\max_{x \in \chi} \{f(x) := \min_{i=1, \ldots, m} \omega_i \|x - x^i\|^2\}, \tag{1}
\]

where \( x^1, \ldots, x^m \in \mathbb{R}^n \) are \( m \) given points in \( n \) dimensional real vector space, \( \chi = \{y \in \mathbb{R}^n \mid (y_1^2, \ldots, y_n^2, 1)^T \in \mathcal{K}\} \) with \( \mathcal{K} \) being a closed convex cone, \( \omega_i > 0 \) for \( i = 1, \ldots, m \) and \( \| \cdot \| \) denotes the Euclidean norm. This problem aims to find a point in an closed set \( \chi \) such that the minimum of the weighted Euclidean distance from the \( m \) given points is maximized. It has wide applications in spatial management, facility location, pattern recognition and many other fields (see [12, 2, 5] and references therein).

2010 Mathematics Subject Classification. Primary: 90C06, 90C26; Secondary: 93E10.
Key words and phrases. Maximin dispersion problem, block successive upper bound minimization, maximum improvement successive upper-bound minimization, successive convex approximation.

The first author is supported by National Natural Science Foundation of China under the grant 11571221 and 11771208.

* Corresponding author: Zi Xu.

1
The weighted maximin dispersion problem is known to be NP-hard in general, even in the case of equal weights (i.e., \( \omega_1 = \cdots = \omega_m \)) and \( \chi = [-1,1]^n \) (\([4]\)) or \( \chi = \{ x : ||x|| \leq 1 \} \) (\([11]\)). We denote the two special cases by \( P_{\text{box}} \) and \( P_{\text{ball}} \), which correspond to setting \( K = \{ y \in \mathbb{R}^{n+1} | y_j \leq y_{n+1}, j = 1, \ldots, n \} \) and \( K = \{ y \in \mathbb{R}^{n+1} | y_1 + \cdots + y_n \leq y_{n+1} \} \) respectively. Throughout this paper, we focus on the \( P_{\text{box}} \) case, i.e., box-constrained weighted maximin dispersion problem.

When \( n \leq 3 \) and \( \chi \) being a polyhedral set, (1) is solvable in polynomial time \([12, 9]\). For \( n > 4 \), heuristic approaches have been proposed \([12, 2]\). Haines et al. \([4]\) designs approximation algorithms based on semidefinite programming (SDP) relaxation and second order cone programming (SOCP) relaxation for (1) and obtains approximation bounds for both \( P_{\text{box}} \) and \( P_{\text{ball}} \) cases. Wang and Xia \([11]\) then focus on the study of \( P_{\text{ball}} \) and proposes a better approximation algorithm based on a linear programming relaxation. Furthermore, Wu et. al \([13]\) propose a much simpler approximation algorithm which owns the same approximation bound as in \([11]\) when \( \chi = \{ x : ||x||_p \leq 1 \} \) for any \( 2 \leq p \leq +\infty \).

In this paper, we focus on developing efficient low complexity algorithms for (1). We first rewrite the maximin dispersion problem as an equivalent minimax problem. Then, we propose a two-phase method to solve it. In the first phase, we employ a Block Successive Upper bound Minimization (BSUM) algorithm, which allows only one component (say, \( x_i \)) of the variable \( x \) to be updated at each iteration. Instead of the original function of \( x_i \), a non-smooth, convex surrogate function is constructed as the upper bound function to be minimized by locally linearizing each quadratic component. Since then, the subproblem is a one-dimensional piecewise linear minimax problem over the box constraints, or equivalently, a two-dimensional linear programming problem which can be solved very efficiently by existing algorithms. In the second phase, we rounding the first phase’s solution to the boundary of \( \chi = [-1,1]^n \). This heuristic step can improve our algorithm’s performance in practice. However, we could not prove the global convergence of this algorithm. To overcome this shortcoming, we propose another strengthened two-phase algorithm, which employs a maximum improvement successive upper-bound minimization (MISUM) algorithm instead of the BSUM algorithm in the first phase. At each step, only the block that provides the maximum improvement of the upper bound function is updated. Then, it can be proved that every limit point of the iterate generated in the first phase of this strengthened algorithm is a stationary point of (1). Numerical results show the efficiency of both algorithms.

The remainder of the paper is organized as follows. In Section 2, we introduce a low complexity two-phase algorithm for (1). A strengthened two-phase algorithm is proposed in Section 3, which employs a MISUM algorithm instead of the BSUM algorithm in the first phase. We present some numerical comparisons in Section 4. Some conclusions are shown in the last section.

**Preliminaries.** Throughout the paper, we adopt the following notation and definitions. We use \( \mathbb{R}^n \) to denote the space of \( n \) dimensional real valued vectors. The distance of the point \( x \) from the set \( \chi \) is defined as \( d(x, \chi) = \inf_{y \in \chi} ||x - y||_2 \), where \( || \cdot ||_2 \) denotes the Euclidean 2-norm. Let \( f : D \to \mathbb{R} \) be a function where \( D \subseteq \mathbb{R}^n \) is a convex set. The directional derivative of \( f \) at point \( x \) in direction \( d \) is defined by

\[
f'(x; d) = \lim_{\lambda \downarrow 0} \inf_{\lambda} \frac{f(x + \lambda d) - f(x)}{\lambda}
\]
The point $x$ is said to be a stationary point of $f(\cdot)$ if $f'(x; d) \geq 0$ for all $d$ such that $x + d \in D$. The function $f$ is quasi-convex if

$$f(\theta x + (1 - \theta)y) \leq \max(f(x), f(y)), \forall \theta \in (0, 1), \forall x, y \in \text{dom}(f).$$

2. A low complexity algorithm.

2.1. Model reformulation. Note that (1) is a non-smooth, non-convex optimization problem since the point-wise minimum of convex quadratics is non-differentiable and non-concave. For convenience, we rewrite (1) into the following equivalent form,

$$\max_{x \in \chi} \min_{i=1, \ldots, m} \omega_i \|x - x^i\|^2 = -\min_{x \in \chi} -\max_{i=1, \ldots, m} -\omega_i \|x - x^i\|^2. \quad (2)$$

In the rest of this paper, we focus on the following model:

$$\min_{x \in \chi} \max_{i=1, \ldots, m} -\omega_i \|x - x^i\|^2. \quad (3)$$

Note that the problem still remains non-convex and non-smooth. Before introducing our algorithm, we briefly introduce the block successive upper bound minimization algorithm, which is closely related to our algorithm.

2.2. Block successive upper bound minimization algorithm (BSUM). Consider the general optimization problem:

$$\min_{x \in \chi} f(x) \quad \text{s.t.} \quad x \in \chi, \quad (4)$$

where $\chi$ is a closed convex set. When the objective function $f(x)$ is nonconvex and/or nonsmooth, solving (4) directly may not be easy. The successive upper bound minimization (SUM) is one of the efficient algorithms to solve it by optimizing a sequence of approximate objective functions instead. More specifically, starting from a feasible point $x^0$, the algorithm generates a sequence $x^r$ according to the following update rule

$$x^r \in \arg\min_{x \in \chi} u(x, x^{r-1}), \quad (5)$$

where $x^{r-1}$ is the point generated by the algorithm at the $(r-1)$th iteration and $u(x, x^{r-1})$ is an approximation of $f(x)$ at the $r$th iteration. Moreover, under some mild assumption on $u(\cdot, \cdot)$, it can be proved that every limit point of the iterates generated by the SUM algorithm is a stationary point of the problem (4) (see [10]).

In many practical applications, the optimization variables can be decomposed into independent blocks. Assume that the feasible set $\chi$ is the Cartesian product of $m$ closed convex sets: $\chi = \chi_1 \times \chi_2 \times \cdots \times \chi_m$ with $\chi_i \in \mathbb{R}^{m_i}$ and $\sum_{i=1}^m m_i = n$. Accordingly, the optimization variable $x \in \mathbb{R}^n$ can be decomposed as $x = (x_1, x_2, \cdots, x_m)$ with $x_i \in \chi_i$. In the following, we introduce a block successive upper bound minimization (BSUM) algorithm, which effectively takes such block structure into consideration. Different from the SUM algorithm, the BSUM algorithm updates
only a single block of variables in each iteration. More precisely, at iteration \( r \), the selected block (say, block \( i \)) is computed by solving the subproblem

\[
\min_{x_i} u_i(x_i, x^{r-1})
\]

s.t. \( x_i \in \chi_i, \) 

where \( u_i(\cdot, x^{r-1}) \) is again an approximation of \( f(\cdot) \) at the point \( x^{r-1} \). Algorithm 1 summarizes the main steps of the BSUM algorithm.

Algorithm 1: Pseudocode of the BSUM algorithm

1. Find a feasible point \( x_0 \in \chi \) and set \( r = 0 \)
2. repeat
   3. \( r = r + 1 \), \( i = (r \mod m) + 1 \)
   4. Let \( x^*_r \in \arg \min_{x_i \in \chi_i} u_i(x_i, x^{r-1}) \)
   5. Set \( x^*_r = x^{r-1} \) (\( \forall k \neq i \))
   6. until some convergence criterion is met.

Under some regularity assumptions, the main convergence result of Algorithm 1 is shown as follows without proof.

Assumption 2.1. \( u_i(\cdot, \cdot) \) satisfies the following regularity conditions:

\[
\begin{align}
\text{(A1)} & & u_i(y_i, y) = f(y), \forall y \in \chi, \forall i, \\
\text{(A2)} & & u_i(x_i, y) \geq f(y_1, y_2, \cdots, y_{i-1}, x_i, y_{i+1}, \cdots, y_m), \forall x_i \in \chi_i, \forall y \in \chi, \forall i, \\
\text{(A3)} & & u'_i(x_i, y; d_i)_{|x_i = y_i} = f'(y; d), \forall d = (0, \cdots, d_i, \cdots, 0) \text{ s.t. } x_i + d_i \in \chi_i \forall i \\
\text{(A4)} & & u_i(x_i, y) \text{ is continuous in } (x_i, y), \forall i.
\end{align}
\]

Theorem 2.2 ([10]). (a) Suppose that the function \( u_i(x_i, y) \) is quasi-convex in \( x_i \) for \( i = 1, \cdots, m \) and Assumption 2.1 holds. Furthermore, assume that the subproblem (6) has a unique solution for any point \( x^{r-1} \in \chi \). Then, every limit point \( z \) of the iterates generated by Algorithm 1 is a coordinatewise minimum of (4) (i.e., \( f(z + d_k^m) \geq f(z) \) for \( k = 1, \cdots, m \), with \( d_k^m = (0, \cdots, d_k, \cdots, 0) \), \( \forall d_k \in \mathbb{R}^m \), which satisfies \( z + d_k \in \text{dom}(f) \)). In addition, if \( f(\cdot) \) is regular at \( z \), then \( z \) is a stationary point of (4).

(b) Suppose the level set \( \chi^0 = \{ x \mid f(x) \leq f(x^0) \} \) is compact and Assumption 2.1 holds. Furthermore, assume that \( f(\cdot) \) is regular at any point in \( \chi^0 \) and the subproblem (6) has a unique solution for any point \( x^{r-1} \in \chi^0 \) for at least \( m - 1 \) blocks. Then, the iterates generated by Algorithm 1 converge to the set of stationary points, i.e.,

\[
\lim_{r \to \infty} d(x^r, \chi^*) = 0,
\]

where \( \chi^* \) is the set of stationary points of (4).

2.3. A low complexity algorithm. We propose a low complexity two-phase algorithm for solving (3) by adopting the framework of the BSUM algorithm.

In Phase I, we adopt Algorithm 1 to solve (3). Note that, a key step in Algorithm 1 is in Step 4 to construct a surrogate function \( u_i(x_i, x^{r-1}) \) which satisfies Assumption 2.1. By exploiting the structure of (3), we construct a convex upper bound function by linearizing every quadratic component function of (3) at each iteration in

\[^1\text{f(\cdot) is regular at } z, \text{ if } f'(z; d) \geq 0 \text{ for all } d = (d_1, d_2, \cdots, d_n) \text{ with } f'(z; d_k^m) \geq 0 \text{ where } d_k^m = (0, \cdots, d_k, \cdots, 0) \text{ and } d_k \in \mathbb{R}^m \text{ for all } k.\]
our algorithm. We can show that this surrogate function satisfies Assumption 2.1 in Section 2.2. Denote \( f(x) := \max_{i=1,\ldots,m} f_i(x) \) with \( f_i(x) := -\omega_i \| f - x \|^2, i = 1, \ldots, m \). Since \( f_i(x) \) is concave for \( i = 1, \ldots, m \), only locally linearizing \( f_i(x) \) at the current iterate point \( x = x^{(r)} \), we can obtain a global upper bound of original objective \( f(x) \). More detailedly, at each iteration, we update only one component of the current iterate point \( x^{(r)} \), e.g., the \( j \)-th component \( x_j \) by solving the following subproblem:

\[
x_j^{(r)} = \arg \min_{x_j \in [-1, 1]} \max_{i=1,\ldots,m} a_i^{(r-1,j)} x_j + b_i^{(r-1,j)},
\]

where for \( i = 1, \ldots, m \),

\[
a_i^{(r-1,j)} = 2 \omega_i ((x^i)_j - x_j^{(r-1)}),
\]

\[
b_i^{(r-1,j)} = -\omega_i \| (x^{(r-1)} - x^i) \|^2 + 2 \omega_i x_j^{(r-1)} \left( x_j^{(r-1)} - (x^i)_j \right).
\]

In Phase II, we do a simple rounding for the solution obtained in phase I, to obtain another feasible solution with each component being \(-1\) or \(1\). Algorithm 2 shows the main steps of the proposed algorithm.

**Algorithm 2** Pseudocode of our two-phase algorithm

0 Input \( \omega_i > 0 \) and \( x^i \in \mathbb{R}^{n \times 1}, i = 1, \ldots, m \). Find a feasible point \( x^{(0)} \in \chi \) and set \( r = 0 \)

1 **Phase I:**
2 repeat
3 \( r = r + 1, j = (r \mod n) + 1 \)
4 Let \( x_j^{(r)} \in \arg \min_{x_j \in [-1, 1]} \max_{-1 \leq x_j \leq 1, i=1,\ldots,m} a_i^{(r-1,j)} x_j + b_i^{(r-1,j)} \), where \( a_i^{(r-1,j)} \) and \( b_i^{(r-1,j)} \) are defined in (8) and (9) respectively.
5 Set \( x_k^{(r)} = x_k^{(r-1)} \) (\( k \neq j \))
6 until some convergence criterion is met. Record \( \bar{x} = x^{(r)} \). Compute \( f(\bar{x}) \).

7 **Phase II:**
8 Generate \( \hat{x} \) as follows: for \( j = 1, \ldots, n \), if \( \bar{x} \geq 0 \), set \( \hat{x}_j = 1 \); otherwise set \( \hat{x}_j = -1 \). Compute \( f(\hat{x}) \).
9 Return \( \bar{x}, \hat{x}, f(\bar{x}), f(\hat{x}) \).

Note that the objective function in (7) is a one-dimensional piecewise linear function. Therefore, the subproblem is equivalent to a two-dimensional linear programming problem, i.e., \( n = 2 \), there exists a fast algorithm to solve it, which takes at most \( O(m) \) (in number of constraints) number of iterations. We refer to [7] for more details. Due to the special structure of (1), we add Phase II in our algorithm to find \( \hat{x} \), which is the nearest point from \( \bar{x} \) to the set \( \{ x \in \mathbb{R}^n | x_i \in \{ 1, -1 \}, i = 1, \ldots, n \} \). Actually, since that the box constraints are often active for this problem, after we obtain a good local solution of this problem in Phase I, we would like to see whether the nearest extreme point on the boundary is better or not, and Phase II is aim to do this. We choose the better one between \( \hat{x} \) and \( \bar{x} \) as the output of our algorithm. The computation cost of Phase II needs only \( n \) comparasion operations, and can be ignored compared to Phase I. Our algorithm possess the desirable property of low per-iteration complexity and an comparable iteration complexity over existing methods. This is also the reason that we call it a low complexity algorithm. However, we can not guarantee the global convergence of this algorithm since that the
optimal solution of (7) is not unique. In order to overcome this shortcoming, we propose a strengthened algorithm in Section 3.

3. A strengthened efficient algorithm and its convergence result.

3.1. A strengthened algorithm. Consider the general optimization problem (4), Chen et al. [1] have proposed a related maximum block improvement (MBI) algorithm, which differs from the conventional block coordinate descent algorithm only by its update schedule. More specifically, only the block that provides the maximum improvement is updated at each step. Remarkably, by utilizing such a modified updating rule, the per-block subproblems are allowed to have multiple solutions. Inspired by this development, Razaviyayn et al. [10] propose modifying the BSUM algorithm similarly by simply updating the block that gives the maximum improvement. They name the resulting algorithm the maximum improvement successive upper-bound minimization (MISUM) algorithm. We list its main steps in Algorithm 3 for completeness. Under some regularity assumptions, the main convergence result of Algorithm 3 is shown in Theorem 3.1 without proof.

Algorithm 3 Pseudocode of the MISUM algorithm

1. Find a feasible point \( x_0 \in \chi \) and set \( r = 0 \)
2. repeat
3. \( r = r + 1 \)
4. Let \( k = \arg \min_i \min_{x_i} u_i(x_i, x_{r-1}) \)
5. Let \( x_k^* \in \arg \min_{x_k} \chi_k u_k(x_k, x_{r-1}) \)
6. Set \( x_i^r = x_{i}^{r-1} (\forall i \neq k) \)
7. until some convergence criterion is met.

Theorem 3.1 ([10]). Suppose that Assumption 2.1 is satisfied. Then, every limit point \( z \) of the iterates generated by the MISUM algorithm is a coordinatewise minimum of (4). In addition, if \( f(\cdot) \) is regular at \( z \), then \( z \) is a stationary point of (4).

We propose a low complexity two-phase algorithm for solving (3) by adopting the framework of the MISUM algorithm. In Phase I, we adopt Algorithm 3 to solve (3). Similarly to Algorithm 2, we construct a convex upper bound function by linearizing every quadratic component function of (3) at each iteration in our algorithm. We can show that this surrogate function satisfies Assumption 2.1 in Section 2.2. In Phase II, we do a simple rounding for the solution obtained in phase I, to obtain another feasible solution with each component being \(-1\) or \(1\). Algorithm 4 shows the main steps of the proposed algorithm.

Note that to the subproblem in Step 4 in Algorithm 4 is equivalent to \( n \) two-dimensional linear programming problems, each can be solved in at most \( O(m) \) (\( m \) is number of constraints) number of iterations. Although Algorithm 4 will take more computational cost to solve its subproblem than that of Algorithm 2, numerical results in Section 4 show that its number of iterate to reach a stationary point is much less than that of Algorithm 2. Moreover, Algorithm 4 is more suitable when parallel processing units are available, since the minimizations with respect to all the blocks can be carried out simultaneously. We do a simple rounding for the solution in Phase II in Algorithm 4, which is similar to that of Algorithm 2.
Algorithm 4 Pseudocode of the strengthened two-phase algorithm

0 Input $\omega_i > 0$ and $x^i \in R^{n \times 1}, i = 1, \ldots, m$. Find a feasible point $x^{(0)} \in \chi$ and set $r = 0$

1 **Phase I:**
2 repeat
3 $r = r + 1$,
4 Let $k = \arg \min_{j=1,\ldots,n} \left\{ \min_{-1 \leq x_j \leq 1} \max_{i=1,\ldots,m} a_i^{(r-1,j)} x_j + b_i^{(r-1,j)} \right\}$.
5 Let $x^{(r)}_k \in \arg \min_{-1 \leq x_k \leq 1} \max_{i=1,\ldots,m} a_i^{(r-1,k)} x_k + b_i^{(r-1,k)}$, where $a_i^{(r-1,k)}$ and $b_i^{(r-1,k)}$
6 are defined in (8) and (9) respectively.
7 Set $x^{(r)}_j = x^{(r-1)}_j$ $(\forall j \neq k)$
8 until some convergence criterion is met. Record $\bar{x} = x^{(r)}$. Compute $f(\bar{x})$.

8 **Phase II:**
9 Generate $\hat{x}$ as follows: for $j = 1, \ldots, n$, if $\bar{x} \geq 0$, set $\hat{x}_j = 1$; otherwise set $\hat{x}_j = -1$. Compute $f(\hat{x})$.
10 **Return** $\bar{x}, \hat{x}, f(\bar{x}), f(\hat{x})$.

3.2. **Convergence analysis.** The following theorem shows the convergence result of Algorithm 4.

**Theorem 3.2.** Every limit point $z$ of the iterates $x^{(r)}$ generated by Phase I in Algorithm 4 is a stationary point of (1).

**Proof.** Firstly, note that $\chi = [-1, 1]^n$ is a compact set. By Claim 3 in [6], we know that $f(\cdot)$ is regular at any point in $\chi$. Secondly, we need to check that Assumption 2.1 holds for our choice of upper bound function, i.e., the piecewise linear function. It can be easily verified that (A1), (A2), (A4) are all satisfied. In addition, by the proof of Proposition 1 in Appendix A in [6], we can prove that (A3) is also satisfied. The proof is almost the same except that in our algorithm, the objective function of each subproblem is a one-dimensional piecewise linear function, while that in [6] is an $n$-dimensional one. Furthermore, by Theorem 3.1, the proof is completed. □

4. **Numerical results.** In this section, we do some numerical comparisons. All the numerical tests are implemented in MATLAB R2016a and run on a laptop with 2.50 GHz processor and 4 GB RAM.

We present the numerical comparison between Algorithm 2, Algorithm 4 and the state-of-the-art randomized approximation algorithm proposed in [13] for solving $P_{box}$. We set all the weights $\omega_i$ to be 1. We do numerical tests on 28 random instances, where the dimension $n$ varies from 10 to 2000, and the number of input point $m$ varies from $n/2$ to $2n$. All the input points $x^i$ orderly form an $n \times 45000$ matrix. We randomly generate this matrix using the following matlab scripts:

```
rand('state', 0); X = 4 * rand(n, 45000) - 2;
```

We choose ten initial points with each components being $-1$ or 1 randomly for Algorithm 2 and Algorithm 4 respectively, and the best results and the total time over ten independent runs are reported. For Algorithm 2, the number of iteration in Phase I is set to be $5n$ for convenience for all the test problems. Whereas, for Algorithm 4, we stop Phase I when the points remain the same during several consequent iterate.
Table 1. Numerical results for Algorithm 2, Algorithm 4 and a randomized algorithm in [13].

| n  | m     | Algorithm 2 | Algorithm 3 | Algorithm 4 |
|----|-------|-------------|-------------|-------------|
|    |       | CR          | f(\bar{x})  | f(\tilde{x}) | time(s)     |
| 10 | 5     | 25.26       | 24.46      | 15.41       | 0.17        |
|    | 10    | 25.06       | 24.98      | 15.76       | 0.19        |
|    | 15    | 15.23       | 14.98      | 15.76       | 0.19        |
|    | 20    | 15.23       | 14.98      | 15.76       | 0.19        |
| 50 | 25    | 15.23       | 14.98      | 15.76       | 0.19        |
|    | 50    | 15.23       | 14.98      | 15.76       | 0.19        |
| 100| 100   | 15.23       | 14.98      | 15.76       | 0.19        |
| 500| 500   | 15.23       | 14.98      | 15.76       | 0.19        |
| 1000|1000 | 15.23       | 14.98      | 15.76       | 0.19        |

Table 1 shows the detailed numerical results. The column “CR” present the optimal objective function values of a linear programming relaxation proposed in [11], which can be regarded as an upper bound of the optimal value. The next column present the statistical results over the 1000 successful runs of the randomized approximation algorithm proposed in [11] respectively, while the subcolumns max, min and ave give the best, the worst and the average objective function values among 1000 successful runs respectively, and the last subcolumn time records the time cost (in seconds). The last two columns are the results of Algorithm 2 and Algorithm 4 respectively, where those two subcolumns “f(\bar{x})” and “f(\tilde{x})” corresponds to the two outputs in Phase I and Phase II respectively, and the last subcolumn time records the time cost. We highlight in bold each statistical result if it is the best of all algorithms.

Although lack of convergence, Algorithm 2 is comparable with the randomized approximation algorithm proposed in [11] in about the same time. Note that for Algorithm 2, the number of iteration in Phase I is set to be a fixed number, i.e., 5n, for all test problems. For convenience if some other stopping rules are used, it may take less time for some test problems. Algorithm 4 finds the best solutions for 19 of the 28 instances. Especially when n becomes large, it performs much better than the randomized approximation algorithm, whereas it takes more time. However, the time cost can be deeply reduced if we use parallel computation.
5. Conclusions. In this paper, we propose two new algorithms to solve an NP-hard problem, i.e., the weighted maximin dispersion problem. In the first phase, we adopt a BSUM framework or a MISUM framework and choose a special piecewise linear upper bound function for the weighted maximin dispersion problem. The per-iteration complexity of these two algorithms is very low. In the second phase, a useful rounding is employed to enhance the solution. Numerical results show that the proposed Algorithm 2 is comparable to the state-of-the-art algorithm in almost the same time, although lack of convergence. Algorithm 4 outperforms the state-of-the-art algorithm, especially when the dimension is large, whereas it takes more time. Fortunately, it is more suitable when parallel processing units are available, since the minimizations with respect to all the blocks can be carried out simultaneously. This endows the algorithms with the ability to scale well to problems in high dimensions with a large number of constraints.

Acknowledgments. We would like to thank two anonymous reviewers for their helpful comments and suggestions to improve this paper.

REFERENCES

[1] B. Chen, S. He, Z. Li and S. Zhang, Maximum block improvement and polynomial optimization, *SIAM J. Optim.*, 22 (2012), 87–107.
[2] B. Dasarthy and L. White, A maximin location problem, *Oper. Res.*, 28 (1980), 1385–1401.
[3] M. Grant and S. Boyd, CVX User’s guide: For CVX version 1.21, *User’s Guide*, (2010), 24–75.
[4] S. Haines, J. Loeppky, P. Tseng and X. Wang, Convex relaxations of the weighted maximin dispersion problem, *SIAM J. Optim.*, 23 (2013), 2264–2294.
[5] M. Johbson, L. Moore and D. Ylvisaker, Maximin Distance Designs, *Statist. Plann. J.*, 26 (1990), 131–148.
[6] A. Konar and N. Sidiropoulos, Fast approximation algorithms for a class of nonconvex QCQP problems using first-order methods, *IEEE Trans. Signal Process.*, 65 (2017), 3494–3509.
[7] N. Megiddo, Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems, *SIAM J. Comput.*, 12 (1983), 759–776.
[8] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, NJ. 1970.
[9] S. Ravi, D. Rosenkrantz and G. Tayi, Heuristic and special case algorithms for dispersion problems, *Oper. Res.*, 42 (1994), 299–310.
[10] M. Razaviyayn, M. Hong and Z.-Q. Luo, A unified convergence analysis of block successive minimization methods for nonsmooth optimization, *SIAM J. Optim.*, 23 (2013), 1126–1153.
[11] S. Wang and Y. Xia, On the ball-constrained weighted maximin dispersion problem, *SIAM J. Optim.*, 26 (2016), 1565–1588.
[12] D. J. White, A heuristic approach to a weighted maximin dispersion problem, *IMA J. Math. Appl. Bus. Indust.*, 7 (1996), 219–231.
[13] Z. Wu, Y. Xia and S. Wang, Approximating the weighted maximin dispersion problem over an $\ell_p$-ball: SDP relaxation is misleading, *Optim. Lett.*, 12 (2018), 875–883.

Received September 2018; revised August 2019.

E-mail address: xuzi@i.shu.edu.cn
E-mail address: 2417341097@qq.com
E-mail address: jiangnanhjj@i.shu.edu.cn