Scientific Research Publishing

Advances in Pure Mathematics, 2019, 9, 330-336
http://www.scirp.org/journal/apm
ISSN Online: 2160-0384
ISSN Print: 2160-0368

An Efficient Random Algorithm for Box Constrained Weighted Maximin Dispersion Problem

Jinjin Huang

Department of Mathematics, Shanghai University, Shanghai, China
Email: jiangnanhjj@i.shu.edu.cn

How to cite this paper: Huang, J.J. (2019) An Efficient Random Algorithm for Box Constrained Weighted Maximin Dispersion Problem. *Advances in Pure Mathematics*, **9**, 330-336.
https://doi.org/10.4236/apm.2019.94015

Received: March 13, 2019
Accepted: April 8, 2019
Published: April 11, 2019

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 reformulate the maximin dispersion problem as a non-convex quadratically constrained quadratic programming (QCQP) problem. We adopt the successive convex approximation (SCA) algorithm to solve the problem. Numerical results show that the proposed algorithm is efficient.

Keywords

Maximin Dispersion Problem, Successive Convex Approximation Algorithm, Quadratically Constrained Quadratic Programming (QCQP)

1. Introduction

The weighted maximin problem model with box constraints is as follows:

$$\max_{x\in\chi}\left\{f(x):=\min_{i=1,\cdots,m}\omega_i\left\|x-x^i\right\|^2\right\} \tag{1}$$

where $\chi=\left\{y\in R^n\left|\left(y_1^2,\cdots,y_n^2,1\right)^{\mathrm{T}}\in\kappa\right.\right\}$, κ is a convex cone; $x_1,\cdots,x_m\in R^n$ are m given points; these m points are equivalent to m locations; $\omega_i>0$ for $i=1,\cdots,m$ and $\|\cdot\|$ denotes the Euclidean norm. In our numerical calculation, ω_i is equal to 1. The goal is to find a point in a closed set $\chi=[-1,1]^n$ such that the minimum of the weighted Euclidean distance from given set of points $x_1,\cdots,x_m$ in R^n is maximized. The weighted maximin problem has been widely used in spatial management, facility location, and pattern recognition.

The weighted maximin dispersion problem with box constraints is known to be NP-hard in general [1]. For the low-dimensional cases ($n\le 3$ and χ being a polyhedral set), it is solvable in polynomial time [2] [3]. For $n > 4$, a heuristic algorithm [2] [4] is used to solve this problem.

In paper [5], they look for approximate solution through convex relaxation, and prove that the approximation bound is $\frac{1-O\left(\sqrt{\ln(m)\gamma^*}\right)}{2}$, where γ^* depends on χ, when $\chi = \{-1,1\}^n$ or $\chi = [-1,1]^n$, $\gamma^* = O\left(\frac{1}{n}\right)$. In paper [1], they use the linear programming relaxation method to give the approximate bounds of the ball problem, which is $\frac{1-O\left(\sqrt{\ln(m)/n}\right)}{2}$. In paper [5], they consider the problem of finding a point in a unit n-dimensional ℓ_p-ball $(p \geq 2)$ such that the minimum of the weighted Euclidean distance from given m points is maximized. They show in paper [6] that the SDP-relaxation-based approximation algorithm provides the first theoretical approximation bound of

$$\frac{1-O\left(\sqrt{\ln(m)/n}\right)}{2}.$$

In this paper, firstly, we model the maximin dispersion problem as a Quadratically constrained quadratic programming (QCQP), noting that (1) is a non-smooth, non-convex optimization problem, because the point-wise minimum of convex quadratics is non-differentiable and non-concave. We solve this problem with a general approximation framework, which is successive convex approximation (SCA), which can be summarized as follows: each quadratic component of (1) is locally linearized at the current iteration to construct its convex approximation function, so we obtain a convex subproblem. The solution of each subproblem is then used as the point about which we construct a convex surrogate function in the next iteration, repeat the steps, and then adopt the random block coordinate descent method (RBCDM) to obtain the solution of subproblem.

The remainder of the paper is organized as follows. In Section 2, we give technical preliminaries. In Section 3, we first reformulate maximin dispersion problem as a QCQP problem. Then, we describe the overall SCA approach and use the proposed methods (RBCDM) for solving each subproblem. In Section 4, we present some numerical results. Conclusions are made in Section 5.

2. Technical Preliminaries

The following concepts or definitions are adopted in our paper.

- We use R^n to denote the space of n dimensional real valued vectors, and $x \in R^n$, we denote the i^{th} component of x by x_i. Thus, each $x \in R^n$ is a column vector

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

- Let $y \in R^n$ and $\chi = [-1,1]^n$ be a set, then the distance of the point y from the set χ is defined as

$$d(y,\chi) = \inf_{x\in\chi} \|x-y\|_2$$

3. Algorithm of Generation

We now reformulation (1) into the following equivalent form,

$$\max_{x\in\chi} \min_{i=1,\cdots,m} \omega_i \left\|x-x^i\right\|^2 \Leftrightarrow -\min_{x\in\chi} \max_{i=1,\cdots,m} -\omega_i \left\|x-x^i\right\|^2, \tag{2}$$

and we finally obtain

$$\min_{x\in\chi} \max_{i=1,\cdots,m} -\omega_i \left\|x-x^i\right\|^2, \tag{3}$$

and we will work with this formulation, note that the problem still remains non-convex.

Our Algorithm

We first introduce our algorithm ideas. First, we construct a convex optimization function of the non-convex objective function (3) by locally linearizing each quadratic component of (3) about the iterate point $x^{(r)}$, we obtain a n-dimensional convex subproblem. Second, we adopt random block coordinate descent method (RBCDM) to transform the n-dimensional convex subproblem into one-dimensional convex subproblem to reduce the computational complexity, here, the optimization variables be decomposed into n independent blocks. At each iteration of this method, random one of the components of variable is optimized, while the remaining variables are held fixed, until all components of a variable are updated, remember as a round, repeat the above steps until we achieve the effect we want. Such block structure can lead to low-complexity algorithms. Finally, to solve the one-dimensional subproblem.

Defining $f(x) := \max_{i=1,\cdots,m} u_i(x)$, where $u_i(x) := -\omega_i \left\|x-x^i\right\|^2, i=1,\cdots,m$. Since $u_i(x)$ is concave for $i=1,\cdots,m$, on locally linearizing $u_i(x)$ about the current iterate point $x = x^{(r)}$, we can obtain a global upper-bound of original objective $f(x)$. At the point $x = x^{(r)}$, we construct a convex approximation function of $f(x)$ at $x = x^{(r)}$ as follows:

$$\begin{aligned} u_i(x) &\le u_i\left(x^{(r)}\right) + \nabla u_i\left(x^{(r)}\right)^{\mathrm{T}}\left(x-x^{(r)}\right) \\ &= 2\omega_i\left(x^i - x^{(r)}\right)^{\mathrm{T}} x + \omega_i\left(\left(x^{(r)}\right)^{\mathrm{T}} x^{(r)} - \left(x^i\right)^{\mathrm{T}} x^i\right) \\ &= \left(x^{(r)}\right)^{\mathrm{T}} x + d_i^{(r)}, \end{aligned}$$

where $c_i^{(r)} = 2\omega_i\left(x^i - x^{(r)}\right)$, $d_i^{(r)} = \omega_i\left(\left(x^{(r)}\right)^{\mathrm{T}} x^{(r)} - \left(x^i\right)^{\mathrm{T}} x^i\right)$, for $i=1,\cdots,m$.

Define $v\left(x,x^{(r)}\right) := \max_{i=1,\cdots,m} \left(c_i^{(r)}\right)^{\mathrm{T}} x + d_i^{(r)}$, the piecewise linear function $v\left(x,x^{(r)}\right)$ is an upper bound of the original function $f(x)$ at $x = x^{(r)}$, which is tight at $x = x^{(r)}$ [7]. We replace $f(x)$ with its piecewise linear approximation about $x^{(r)}$ to obtain the non-smooth, convex subproblem.

$$\min_{x\in\chi}\max_{i=1,\cdots,m}\left(c_i^{(r)}\right)^{\mathrm{T}}x+d_i^{(r)} \tag{4}$$

This subproblem is computationally expensive, so we transform the high-dimensional problem into one-dimensional problem to reduce the complexity.

The concrete steps are as follows: We random update the j^{th} component x_j of x at the current iterate point $x^{(r)}$ and keep the other components unchanged，it must be noted that the x_j is a component of random selection, let $x=\left(x_1^{(r)},\cdots,x_{j\text{-}1}^{(r)},x_j,x_{j+1}^{(r)},\cdots,x_n^{(r)}\right)^{\mathrm{T}}$, so we have

$$\begin{aligned}
&v_j\left(x_j,x^{(r)}\right)\\
&=\max_{i=1,\cdots,m}2\left(x^i-x^{(r)}\right)^{\mathrm{T}}x+\left(\left(x^{(r)}\right)^{\mathrm{T}}x^{(r)}-\left(x^i\right)^{\mathrm{T}}x^i\right)\\
&=\max_{i=1,\cdots,m}2\left(\left(x_1^i-x_1^{(r)}\right),\cdots,\left(x_j^i-x_j^{(r)}\right),\cdots,\left(x_n^i-x_n^{(r)}\right)\right)\left(x_1^{(r)},\cdots,x_{j-1}^{(r)},x_j,x_{j+1}^{(r)},\cdots,x_n^{(r)}\right)^{\mathrm{T}}\\
&\quad+\left(\left(x^{(r)}\right)^{\mathrm{T}}x^{(r)}-\left(x^i\right)^{\mathrm{T}}x^i\right)\\
&=\max_{i=1,\cdots,m}2\left(x_j^i-x_j^{(r)}\right)x_j+\left(\left(x^{(r)}\right)^{\mathrm{T}}x^{(r)}-\left(x^i\right)^{\mathrm{T}}x^i\right)\\
&\quad+\sum_{l=1}^{n}2\left(x_l^i-x_l^{(r)}\right)x_l^{(r)}-2\left(x_j^i-x_j^{(r)}\right)x_j^{(r)}\\
&=\max_{i=1,\cdots,m}a_i^{(r)}x_j+b_i^{(r)}
\end{aligned}$$

where $a_i^{(r)}=2\left(\left(x^i\right)_j-\left(x^{(r)}\right)_j\right)$

$$b_i^{(r)}=\left(\left(x^{(r)}\right)^{\mathrm{T}}x^{(r)}-\left(x^i\right)^{\mathrm{T}}x^i\right)+\sum_{i=1}^{n}\left(\left(x^i\right)_l-\left(x^{(r)}\right)_l\right)\left(x^{(r)}\right)_l-2\left(\left(x^i\right)_j-\left(x^{(r)}\right)_j\right)\left(x^{(r)}\right)_j.$$

obtain the one-dimensional convex subproblem

$$\min_{x_j\in\chi^*}\max_{i=1,\cdots,m}a_i^{(r)}x_j+b_i^{(r)}, \tag{5}$$

In order to solve the solution of one-dimensional piecewise linear function (5), we first arrange the $a_i^{(r)}$ of the m lines from small to large, *i.e.* $a_1^{(r)}\le a_2^{(r)}\le\cdots\le a_m^{(r)}$. For the convenience of description, we remember these m lines as $y_i=a_ix+b_i\left(i=1,2,\cdots,m\right)$, where $x=[-1,1]$. The following is the algorithmic frameworks for solving one-dimensional subproblem.

4. Numerical Results

In order to benchmark the performance of our proposed algorithms, we do some simple numerical comparisons. We do numerical experiments on 4 random instances when dimension n takes different values, respectively, such as n = 100, 500, 1000, 2000. The corresponding m we chose smaller than n, the same as n, and bigger than n. where all weights $\omega_1,\cdots,\omega_m$ are equal to 1, all the numerical tests are implemented in MATLAB R2016a and run on a laptop with 2.50 GHz processor and 4 GB RAM.

All the input points x^i orderly form an $n\times 45000$ matrix. We randomly generate this matrix using the following matlab scripts:

Table 1. Algorithmic frameworks of subproblem.

Algorithm 1: Piecewise-linear Algorithm

- The number of $a_i \ge 0(i = 1,2,\cdots,m)$ is *C*, let *t* = *C*/*m*.
- If $t \ge 0.5$, the minimax point is searched from the left $x = -1$, otherwise, from $x = 1$. We only describe the search process of start from the left, and the right is the same. Solving the intersection coordinates (X_i, Y_i) of the $x = -1$ and $y_i(i = 1,\cdots,m)$, let $Y = (Y_1,\cdots,Y_m)$ and $Y_p = \max(Y)$, if the corresponding $a_p \ge 0$, then terminate the search and $x = -1$ is the solution of the subproblem.
- If $a_p \le 0$, then solving the intersection coordinates $(\tilde{X}_j, \tilde{Y}_j)$ of the straight line with a slope bigger than a_p and the current line y_p.
- Here, the number of lines where the slope is larger than a_p is recorded as *w*, remember $\tilde{Y} = \tilde{Y}_1,\cdots,\tilde{Y}_W$, let $\tilde{Y}_h = \max(\tilde{Y})$, and judge whether the abscissa $\tilde{X}_h$ is out of bounds, if $\tilde{X}_h > 1$, then $x = 1$ is the solution of the subproblem, otherwise judge whether or not $a_h \ge 0$, if $a_h \ge 0$, then stop to research, and $\tilde{X}_h$ is the solution of the subproblem, otherwise repeat the step 3, until the minimax point is found.

Table 2. Numerical results.

n	m	CR	Wang Xia				Our Algorithm		
			max	min	ave	Time 1	f(x1)	f(x2)	Time 2
n = 100	50.00	243.46	209.39	148.99	179.27	0.62	197.41	210.51	0.70
	100.00	234.17	195.99	143.95	172.44	1.67	195.82	188.37	0.52
	150.00	231.58	193.74	143.98	171.21	3.24	195.01	199.84	1.08
	200.00	230.38	194.58	141.21	167.74	5.33	187.67	189.64	1.02
n = 500	250.00	1194.59	1065.03	934.69	1013.53	3.62	1060.51	1089.49	5.77
	500.00	1174.71	1048.90	940.91	999.97	10.73	1061.97	1078.25	10.74
	750.00	1160.43	1034.18	933.32	990.97	21.43	1042.69	1056.50	15.23
	1000.00	1151.97	1030.25	915.98	983.87	35.80	1008.60	1026.29	21.55
n = 1000	500.00	2381.02	2172.04	2008.44	2094.29	9.89	2166.56	2225.10	27.00
	750.00	2340.18	2127.92	1958.32	2070.12	28.07	2146.04	2203.33	42.47
	1000.00	2322.15	2122.13	1977.70	2064.92	55.10	2123.73	2162.60	54.63
	2000.00	2313.60	2114.86	1962.88	2061.42	91.23	2100.44	2123.65	107.83
n = 2000	1000.00	4724.03	4390.64	4159.86	4297.14	28.42	4356.34	4472.37	171.00
	2000.00	4674.71	4365.89	4159.95	4275.33	84.91	4346.58	4428.84	340.09
	3000.00	4653.52	4347.41	4138.40	4264.89	170.71	4336.45	4400.54	506.70
	4000.00	4637.10	4348.47	4101.46	4254.41	282.94	4306.64	4375.76	651.73

$$rand(state, 0); X = 4 * rand(n, 450) - 2;$$

We report the numerical results in Table 1. The columns $v(CR)'$ present the optimal objective function values of convex relaxation [1] of the 26 instances. In [1], they first reformulate (1) as an equivalent smooth optimization problem as following

$$\begin{aligned}&\max_{x,\zeta} \quad \zeta \\ &s.t. \ \ \omega_i\left(\|x\|^2 - 2\left(x^i\right)^{\mathrm{T}} x + \left\|x^i\right\|^2\right) \\ &\qquad x \in \chi\end{aligned}$$

product the following convex relaxation (CR) when $\chi = [-1,1]^n$:

$$\begin{aligned}&\max_{x,\zeta} \quad \zeta \\ &s.t. \ \ \omega_i\left(n - 2\left(x^i\right)^{\mathrm{T}} x + \left\|x^i\right\|^2\right) \\ &\qquad x \in \chi\end{aligned}$$

we solved it with CVX solver [8].

The next column present the statistical results over the 1000 runs of the general algorithm proposed in [1], the subcolumns "max", "min", "ave" and "time 1" give the best, the worst, the average objective function values and running time found among 1000 tests, respectively. The last column is the result of our algorithm, where we choose **0** vector as the initial point. Finally, add a rounding (*i.e.*, if $x_h^{(0)} \geq 0$, then $x_h^{(0)} = 1$, otherwise $x_h^{(0)} = -1$, for $h = 1, \cdots, n$ for the solution $x^{(0)}$ obtained by the iteration. The subcolumns "f(x1)", "f(x2)" and "time 2" represents the numerical result corresponding to no add rounding, add rounding and running time of our algorithm, respectively. Numerical results show that the effect of "f(x2)" is the best. Table 2 shows that the qualities of the solutions returned by our algorithm are generally higher than those obtained by the general algorithm in [1].

5. Conclusion

In this paper, we reformulate the maximin dispersion problem as QCQP problem and the original non-convex problem is approximated by a sequence of convex problems. Then, we adopt the random block coordinate descent method (RBCDM) to obtain the solution of subproblem. Numerical results show that the proposed algorithm is efficient.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Wang, S. and Xia, Y. (2016) On the Ball-Consterained Weighted Maximin Dispersion Problem. *SIAM Journal on Optimization*, **26**, 1565-1588.

[2] White, D.J. (1996) A Heuristic Approach to a Weighted Maxmin Disperation Problem. *IMA Journal of Management Mathematics*, **7**, 219-231. https://doi.org/10.1093/imaman/7.3.219

[3] Ravi, S.S., Rosenkrantz, D.J. and Tayi, G.K. (1994) Heuristic and Special Case Algo-

rithms for Dispersion Problems. *Operations Research*, **42**, 299-310. https://doi.org/10.1287/opre.42.2.299

[4] Dasarthy, B. and White, L.J. (1980) A Maximin Location Problem. *Operations Research*, **28**, 1385-1401. https://doi.org/10.1287/opre.28.6.1385

[5] Wu, Z.P., Xia, Y. and Wang, S. (2017) Approximating the Weighted Maximin Dispersion Problem over an $l_p - ball$: SDP Relaxation Is Misleading. *Optimization Letters*, **12**, 875-883.

[6] Haines, S., Loeppky, J., Tseng, P. and Wang, X. (2013) Convex Relaxations of the Weighted Maxmin Dispersion Problem. *SIAM Journal on Optimization*, **23**, 2264-2294. https://doi.org/10.1137/120888880

[7] Konar, A. and Sidiropoulos, N.D. (2017) Fast Approximation Algorithms for a Class of Nonconvex QCQP Problems Using First-Order Methods. *IEEE Transactions on Signal Processing*, **65**, 3494-3509.

[8] Grant, M. and Boyd, S. (2010) CVX User's Guide: For CVX Version 1.21. User's Guide, 24-75.