Vol.2021-AL-183 No.7
2021/5/7

Max-Min 3-dispersion on a Convex Polygon

Yasuaki Kobayashi[1,a)] Shin-ichi Nakano[2,b)] Kei Uchizawa[3,c)] Takeaki Uno[4,d)]
Yutaro Yamaguchi[5,e)] Katsuhisa Yamanaka[6,f)]

Abstract: Given a set P of n points and an integer k, we wish to place k facilities on points in P so that the minimum distance between facilities is maximized. The problem is called the k-dispersion problem, and the set of such k points is called a k-dispersion of P. Note that the 2-dispersion problem corresponds to the computation of the diameter of P. Thus, the k-dispersion problem is a natural generalization of the diameter problem. In this paper, we consider the case of $k = 3$, which is the 3-dispersion problem, when P is in convex position. We present an $O(n^2)$-time algorithm to compute a 3-dispersion of P.

1. Introduction

The facility location problem and many of its variants have been studied [11], [12]. Typically, given a set P of points in the Euclidean plane and an integer k, we wish to place k facilities on points in P so that a designated function on distance is minimized. In contrast, in the *dispersion problem*, we wish to place facilities so that a designated function on distance is maximized.

The intuition of the problem is as follows. Assume that we are planning to open several coffee shops in a city. We wish to locate the shops mutually far away from each other to avoid self-competition. In other words, we wish to find k points so that the minimum distance between the shops is maximized. See more applications, including *result diversification*, in [9], [20], [21].

Now, we define the *max-min k-dispersion problem*. Given a set P of n points in the Euclidean plane and an integer k with $k < n$, we wish to find a subset $S \subset P$ with $|S| = k$ in which $\min_{u,v \in S} d(u, v)$ is maximized, where $d(u, v)$ is the distance between u and v in P. Such a set S is called a k-dispersion of P. This is the max-min version of the k-dispersion problem [20], [24]. Several heuristics to solve the problem are compared [14]. The max-sum version [6], [7], [8], [9], [10], [15], [17], [20] and a variety of related problems [4], [6], [10] are studied.

The max-min k-dispersion problem is NP-hard even when the triangle inequality is satisfied [13], [24]. An exponential-time exact algorithm for the problem is known [2]. The running time is $O(n^{\omega k/3} \log n)$, where $\omega < 2.373$ is the matrix multiplication exponent.

The problem in the D-dimensional Euclidean space can be solved in $O(kn)$ time for $D = 1$ if a set P of points are given in the order on the line and is NP-hard for $D = 2$ [24]. One can also solve the case $D = 1$ in $O(n \log \log n)$ time [3] by the sorted matrix search method [16] (see a good survey for the sorted matrix search method in [1], Section 3.3), and in $O(n)$ time [2] by a reduction to the path partitioning problem [16]. Even if a set P of points are not given in the order on the line the running time for $D = 1$ is $O((2k^2)^k n)$ [5]. Thus, if k is a constant, we can solve the problem in $O(n)$ time. If P is a set of points on a circle, the points in P are given in the order on the circle, and the distance between them is the distance along the circle, then one can solve the k-dispersion problem in $O(n)$ time [23].

For approximation, the following results are known. Ravi et al. [20] proved that, unless P = NP, the max-min k-dispersion problem cannot be approximated within any constant factor in polynomial time, and cannot be approximated with a factor less than two in polynomial time when the distance satisfies the triangle inequality. They also gave a polynomial-time algorithm with approximation ratio two when the triangle inequality is satisfied.

When k is restricted, the following results for the D-dimensional Euclidean space are known. For the case $k = 3$, one can solve the max-min k-dispersion problem in $O(n^2 \log n)$ time [18]. For $k = 2$, the max-min k-dispersion of P corresponds to the computation of the diameter of P, and one can compute it in $O(n \log n)$ time [19].

In this paper, we consider the case where P is a set of points in convex position and d is the Euclidean distance. See an example of a 3-dispersion of P in Fig. 1. By the brute force algorithm and the algorithm in [18] one can compute a 3-dispersion of P in $O(n^3)$ and $O(n^2 \log n)$ time, respectively, for a set of points on the plane. In this paper, we present an algorithm to compute a

1 Kyoto University, Japan
2 Gunma University, Japan
3 Yamagata University, Japan
4 National Institute of Informatics, Japan
5 Kyushu University, Japan
6 Iwate University, Japan
a) kobayashi@iip.ist.i.kyoto-u.ac.jp
b) nakano@cs.gunma-u.ac.jp
c) uchizawa@yz.yamagata-u.ac.jp
d) uno@nii.jp
e) yutaro_yamaguchi@inf.kyushu-u.ac.jp
f) yamanaka@cis.iwate-u.ac.jp

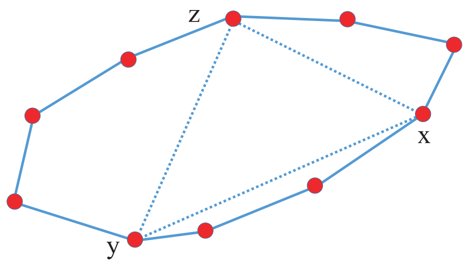

Fig. 1 An example of 3-dispersion. $\{x, y, z\}$ is a 3-dispersion.

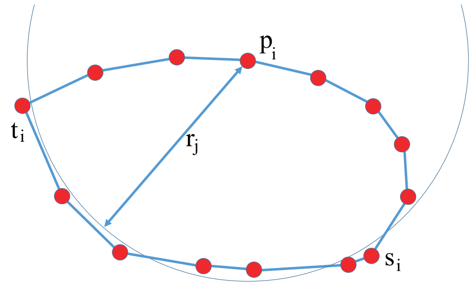

Fig. 2 An example of s_i and t_i for p_i. The drawn circle is a circle with the center of p_i the radius of length r_j.

3-dispersion of P in $O(n^2)$ time using the property that P is a set of points in convex position.

2. Preliminaries

Let P be a set of n points in convex position on the plane. In this paper, we assume $n \geq 3$. We denote the Euclidean distance between two points u, v by $d(u, v)$. The cost of a set $S \subset P$ is defined as $\mathrm{cost}(S) = \min_{u,v \in S} d(u, v)$. Let $\mathcal{S}_3$ be the set of all possible three points in P. We say $S \in \mathcal{S}_3$ is a 3-dispersion of P if $\mathrm{cost}(S) = \max_{S' \in \mathcal{S}_3} \mathrm{cost}(S')$.

We have the following two lemmas, which can be checked easily.

Lemma 1 If a triangle with corner points p_i, p_r, p_ℓ satisfies $d(p_i, p_r) \geq L$, $d(p_i, p_\ell) \geq L$ and $d(p_\ell, p_r) < L$ for some L, then $\angle p_\ell p_i p_r < 60°$.

Lemma 2 If a triangle with corner points p_i, p_r, p_ℓ satisfies $d(p_i, p_r) < L$, $d(p_i, p_\ell) < L$ and $d(p_\ell, p_r) \geq L$ for some L, then $\angle p_\ell p_i p_r > 60°$.

3. Algorithm

Let $P = \langle p_1, p_2, \ldots, p_n \rangle$ be the set of points in convex position and assume that they appear clockwise in this order. Note that the successor of p_n is p_1. Let D be the distance matrix of the points in P, that is, the element at row y and column x is $d(p_x, p_y)$. Let $C_1 = \{d(p_i, p_j) \mid 1 \leq i < j \leq n\}$. The cost of a 3-dispersion in P is the distance between some pair of points in P, so it is in C_1.

The outline of our algorithm is as follows. Our algorithm is a binary search and proceeds in at most $\lceil 2 \log n \rceil$ stages. For each stage $j = 1, 2, \ldots, k$, where k is at most $\lceil 2 \log n \rceil$, we (1) compute the median r_j of C_j, where C_j is a subset of C_{j-1}, which is computed in the $(j-1)$st stage (except the case of $j = 1$), (2) compute n square submatrices of D defined by r_j along the main diagonal in D, and (3) check if at least one square submatrix among them has an element greater than or equal to r_j, or not. We prove later that at least one square submatrix above has an element greater than or equal to r_j if and only if P has a 3-dispersion with cost r_j or more. If the answer of (3) is YES then we set C_{j+1} as the subset of C_j consisting of the values greater than or equal to r_j, otherwise we set C_{j+1} as the subset of C_j consisting of the values less than r_j. Note that in either case the cost of a 3-dispersion of P is in C_{j+1} and $|C_{j+1}| \leq \lceil |C_j|/2 \rceil$ holds. Since the size of C_{j+1} is at most half of C_j and $|C_1| \leq n^2$, the number of stages is at most $\lceil \log n^2 \rceil = \lceil 2 \log n \rceil$.

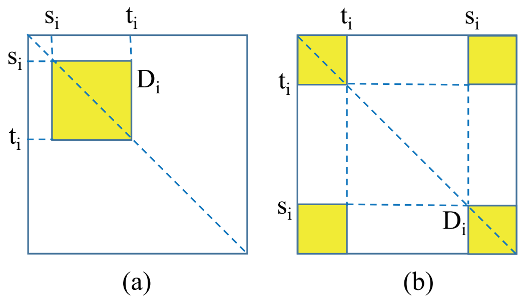

Fig. 3 Illustrations for the square submatrix D_i of D for p_i.

Now, we explain the detail of each stage. For the computation of the median in (1), we simply use a linear-time median-finding algorithm [22].

Next, we explain the detail of (2) for each stage j. Given r_j, for each $p_i \in P$, we compute the first point, say $s_i \in P$, in P with $d(p_i, s_i) \geq r_j$ when we check the points clockwise from p_i. Similarly, we compute the first point, say $t_i \in P$, in P with $d(p_i, t_i) \geq r_j$ when we check the points counterclockwise from p_i. See such an example in Fig. 2. Note that, when we check the points clockwise from s_i to t_i, a point p_c between them may satisfy $d(p_i, p_c) < r_j$. See Fig. 2. For each p_i we define a square submatrix D_i of D induced by the rows $s_i, \ldots, t_i$ and the columns $s_i, \ldots, t_i$. See Fig. 3(a). Note that D_i is located in D along the main diagonal. The square submatrix D_i may appear in D as four separated squares if it contains p_1 on the clockwise contour from s_i to t_i. See Fig. 3(b).

Now, we explain how to compute s_i and t_i of p_i. The method for compute t_i can be done in the similar way for finding s_i. Hence, we focus on how to find s_i. If we search each s_i independently by scanning then the total running time for the search of $s_1, s_2, \ldots, s_n$ is $O(n^2)$ in each stage, and $O(n^2 \log n)$ in the whole algorithm. We are going to improve this. Since s_{i+1} may appear before s_i on the clockwise contour (See Fig. 4) the search is not so simple.

We first explain how to compute s_i of p_i for each $i = 1, 2, \ldots, n$ in stage 1. Given r_1, we check each point clockwise starting at p_i, and s_i is the first point from p_i which has the distance r_1 or more. It can be observed that the total number of checks for the distance in stage 1 is at most $n + |C_1|/2 \leq n + n^2/2$. In this estimation, n checks are required for the pairs of (s_i, p_i) for every $i = 1, 2, \ldots, n$ and $|C_1|/2$ checks are required for the pairs (p, p_i) which satisfies that p appears between p_i and s_i clockwise and $d(p, p_i) < r_1$, for every $i = 1, 2, \ldots, n$. Remember that r_1 is the

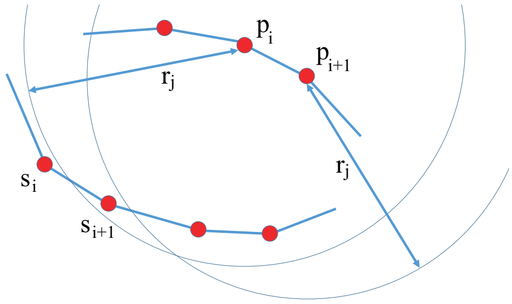

Fig. 4 The point s_{i+1} may appear before s_i on the clockwise contour.

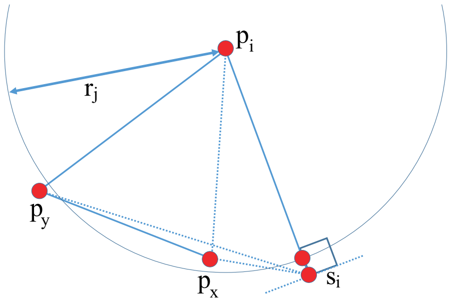

Fig. 5 An illustration for Lemma 3.

median of distances in C_1. Then, in each stage $j = 2, 3, \ldots, k$ ($k \le \lceil 2 \log n \rceil$), given r_j, if the answer to (3) of the preceding stage $j-1$ is YES then we check each point clockwise starting at s_i of the preceding stage $j-1$ (since $r_j > r_{j-1}$ holds, all points before s_i of the preceding stage are within distance r_j from p_i), otherwise we check each point clockwise starting again at the starting point of the preceding stage $j-1$. In either case, we check at most $jn + n^2/2 + n^2/2^2 + \cdots + n^2/2^j$ points in total for the search for $s_1, s_2, \ldots, s_n$ in every stage ℓ for $\ell = 1, 2, \ldots, j$. In the estimation, jn is the total number of checks for $s_1, s_2, \ldots, s_n$ and $n^2/2 + n^2/2^2 + \cdots + n^2/2^j$ is the total number of checks for the points with distance less than r_ℓ from its p_i. When $j = n$, we have the estimation $O(n^2)$ for the total number of checks for computing $s_1, s_2, \ldots, s_n$ in all the stages. By the symmetric way, we can compute $t_1, t_2, \ldots, t_n$ in each stage and the total number of checks for computing $t_1, t_2, \ldots, t_n$ in all the stages is estimated in the same way.

Now, we present a lemma mentioned in (3). Assume that we are at stage j, and s_i and t_i of p_i are given. If there is a set of three points in P containing p_i with cost r_j or more, then the square submatrix D_i has an element greater than or equal to r_j. The reverse may be wrong. If the submatrix D_i for some p_i has an element greater than or equal to r_j at row y and column x, it only ensures $d(p_x, p_y) \ge r_j$. That is, $d(p_i, p_x) < r_j$ and/or $d(p_i, p_y) < r_j$ may hold. We show that this situation cannot occur in the following lemma.

Lemma 3 The square submatrix D_i of stage j has an element greater than or equal to r_j if and only if there is a set of three points $S \subset P$ including p_i with $\mathrm{cost}(S) \ge r_j$.

Proof. If there is a set of three points $S \subset P$ including p_i with $\mathrm{cost}(S) \ge r_j$ then clearly the square submatrix D_i of stage j has an element greater than or equal to r_j.

We only prove the other direction, that is, if the square submatrix D_i of stage j has an element greater than or equal to r_j, then there is a set of three points $S \subset P$ including p_i with $\mathrm{cost}(S) \ge r_j$. Assume that D_i has an element greater than or equal to r_j at row y and column x, that is $d(p_x, p_y) \ge r_j$. We have the following four cases and in each case we show that there exists a set S of three points such that $\mathrm{cost}(S) \ge r_j$.

Case 1: $d(p_i, p_x) \ge r_j$ and $d(p_i, p_y) \ge r_j$.
The set $S = \{p_i, p_x, p_y\}$ has $\mathrm{cost}(S) \ge r_j$.

Case 2: $d(p_i, p_x) < r_j$ and $d(p_i, p_y) < r_j$.
We show that, for $S = \{p_i, s_i, t_i\}$, $\mathrm{cost}(S) \ge r_j$ holds. We assume for a contradiction that $d(s_i, t_i) < r_j$ holds. Then, we have $\angle s_i p_i t_i < 60°$ by Lemma 1 and $\angle p_x p_i p_y > 60°$ by Lemma 2. This is a contradiction to the convexity of P.

Case 3: $d(p_i, p_x) < r_j$ and $d(p_i, p_y) \ge r_j$.
In this case, we show that the set $\{p_i, s_i, p_y\}$ attains $\mathrm{cost}(S) \ge r_j$. Since $d(p_i, p_y) \ge r_j$ and $d(p_i, s_i) \ge r_j$, we have to prove $d(s_i, p_y) \ge r_j$.

Assume for a contradiction that $d(s_i, p_y) < r_j$ holds. See Fig. 5. Now, we first show that $\{s_i, p_x, p_y\}$ forms an obtuse triangle with the obtuse angle p_x, below. We focus on the rectangle consisting of p_i, s_i, p_x, and p_y. Since $d(p_i, p_y) \ge r_j$ and $d(p_i, s_i) \ge r_j$, and $d(s_i, p_y) < r_j$, we have $\angle s_i p_i p_y < 60°$ by Lemma 1. Let p' be the point on the line segment between p_i and s_i with $d(p_i, p') = r_j$. Since $\angle p_i p' p_x < 90°$ holds, we can observe that $\angle p_i s_i p_x < 90°$ holds. Since $d(p_i, p_y) \ge r_j$, $d(p_x, p_y) \ge r_j$, and $d(p_i, p_x) < r_j$, we have $\angle p_i p_y p_x < 60°$ by Lemma 1. Now, the sum of the internal angles of the quadrangle consisting of p_i, s_i, p_x, and p_y implies that $\angle s_i p_x p_y \ge 150°$, and $\{s_i, p_x, p_y\}$ are the points of an obtuse triangle with obtuse angle at p_x. However $d(p_x, p_y) \ge r_j$ and $d(s_i, p_y) < r_j$, which is a contradiction.

Case 4: $d(p_i, p_x) \ge r_j$ and $d(p_i, p_y) < r_j$.
Symmetry to **Case 3**. Omitted. □

Now, we are ready to describe our algorithm and the estimation of the running time. First, as a preprocessing, we construct the set $C_1 = \{d(p_i, p_j) \mid 1 \le i < j \le n\}$ and $n \times n$ distance matrix D. Next, we repeat the following stage for each $j = 1, 2, \ldots, k$, where $k \le \lceil 2 \log n \rceil$. (1) we compute the median r_j of C_j, (2) compute s_i and t_i of p_i for $i = 1, 2, \ldots, n$, and (3) check whether there exists an index i, $(1 \le i \le n)$, such that the maximum value of D_i is greater than or equal to r_j. Then, if such i exists, we set $C_{j+1} = \{d(p_i, p_j) \in C_j \mid d(p_i, p_j) \ge r_j\}$, otherwise, we set $C_{j+1} = \{d(p_i, p_j) \in C_j \mid d(p_i, p_j) < r_j\}$.

The analysis of the running time is as follows. The preprocessing can be done in $O(n^2)$ time. For (1), we can compute the median r_j of stage j in $O(n^2/2^{j-1})$ time by using a linear-time median-finding algorithm [22], and hence $O(n^2)$ time for the whole algorithm. The computation for (2) can be done in $O(n^2)$ time in the whole algorithm, as described above. For (3), after $O(n^2)$-time preprocessing for D, we can compute the maximum element in the given submatrix in D in $O(1)$ time for each query by using the range-query algorithm [25], so we need $O(n)$ time as preprocessing. (For a separated square as shown in Fig. 3(b), we need four queries but total time is still a constant.)

Now, we have our main theorem.

Theorem 1 Let P be a set of n points in convex position. After $O(n^2)$-time preprocessing, one can compute a 3-dispersion of P in $O(n^2)$ time.

Acknolwedgement. This work was supported by JSPS KAKENHI Grant Numbers JP18H04091, JP19K11812, JP20H05793, JP20H05962, JP20K19742. The fourth author is also supported by JST CREST Grant Number JPMJCR1401.

References

[1] P. Agarwal and M. Sharir. Efficient algorithms for geometric optimization. *ACM Comput. Surv.*, 30:412–458, 1998.

[2] T. Akagi, T. Araki, T. Horiyama, S. Nakano, Y. Okamoto, Y. Otachi, T. Saitoh, R. Uehara, T. Uno, and K. Wasa. Exact algorithms for the max-min dispersion problem. *Proc. of FAW 2018*, LNCS 10823:263–272, 2018.

[3] T. Akagi and S. Nakano. Dispersion on the line. *IPSJ SIG Technical Reports*, 2016-AL-158-3, 2016.

[4] K. Amano and S. Nakano. An approximation algorithm for the 2-dispersion problem. *IEICE TRANS. INF.SYST.*, E103-D:506–508, 2020.

[5] T. Araki and S. Nakano. The max-min dispersion on a line. *Proc. of COCOA 2018*, LNCS 11346:672–678, 2018.

[6] C. Baur and S. P. Fekete. Approximation of geometric dispersion problems. *Proc. of APPROX 1998*, pages 63–75, 1998.

[7] B. Birnbaum and K. J. Goldman. An improved analysis for a greedy remote-clique algorithm using factor-revealing LPs. *Algorithmica*, 50:42–59, 2009.

[8] A. Cevallos, F. Eisenbrand, and R. Zenklusen. Max-sum diversity via convex programming. *Proc. of SoCG 2016*, pages 26:1–26:14, 2016.

[9] A. Cevallos, F. Eisenbrand, and R. Zenklusen. Local search for max-sum diversification. *Proc. of SODA 2017*, pages 130–142, 2017.

[10] B. Chandra and M. M. Halldorsson. Approximation algorithms for dispersion problems. *J. of Algorithms*, 38:438–465, 2001.

[11] Z. Drezner. *Facility location: A Survey of Applications and Methods*. Springer, 1995.

[12] Z. Drezner and H. Hamacher. *Facility Location: Applications and Theory*. Springer, 2004.

[13] E. Erkut. The discrete p-dispersion problem. *European Journal of Operational Research*, 46:48–60, 1990.

[14] E. Erkut, Y. Ulkusal, and O. Yenicerioglu. A comparison of p-dispersion heuristics. *Computers & Operational Research*, 21:1103–1113, 1994.

[15] S. P. Fekete and H. Meijer. Maximum dispersion and geometric maximum weight cliques. *Algorithmica*, 38:501–511, 2004.

[16] G. Frederickson. Optimal algorithms for tree partitioning. *Proc. of SODA 1991*, pages 168–177, 1991.

[17] R. Hassin, S. Rubinstein, and A. Tamir. Approximation algorithms for maximum dispersion. *Operation Research Letters*, 21:133–137, 1997.

[18] T. Horiyama, S. Nakano, T. Saitoh, K. Suetsugu, A. Suzuki, R. Uehara, T. Uno, and K. Wasa. Max-min 3-dispersion problems. *Proc. of COCOON 2019*, LNCS 11653:291–300, 2019.

[19] F. P. Preparata and M. I. Shamos. *Computational geometry: an introduction*. Springer-Verlag, 1985.

[20] S. S. Ravi, D. J. Rosenkrantz, and G. K. Tayi. Heuristic and special case algorithms for dispersion problems. *Operations Research*, 42:299–310, 1994.

[21] M. Sydow. Approximation guarantees for max sum and max min facility dispersion with parameterised triangle inequality and applications in result diversification. *Mathematica Applicanda*, 42:241–257, 2014.

[22] R. L. R. T. H. Cormen, C. E. Leiserson and C. Stein. *Introduction to algorithms, Third Edition*. MIT Press, 2000.

[23] K. H. Tsai and D. W. Wang. Optimal algorithms for circle partitioning. *Proc. of COCOON 1997*, LNCS 1276:304–310, 1997.

[24] D. W. Wang and Y.-S. Kuo. A study on two geometric location problems. *Information Processing Letters*, 28:281–286, 1988.

[25] H. Yuan and M. J. Atallah. Data structures for range minimum queries in multidimensional arrays. *Proc. of SODA 2010*, pages 150–160, 2010.

