PAPER Special Section on Foundations of Computer Science - New Trends of Theory of Computation and Algorithm -

An $O(n^2)$-Time Algorithm for Computing a Max-Min 3-Dispersion on a Point Set in Convex Position*

Yasuaki KOBAYASHI[†a)], ***Nonmember***, **Shin-ichi NAKANO**[††b)], **Kei UCHIZAWA**[†††c)], **Takeaki UNO**[††††d)], ***Members***, **Yutaro YAMAGUCHI**[†††††e)], ***Nonmember***, ***and*** **Katsuhisa YAMANAKA**[††††††f)], ***Member***

SUMMARY 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.

key words: *dispersion problem, facility location*

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], [22], [23].

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 [22], [26]. Several heuristics to solve the problem are compared [14]. The max-sum version [6]–[10], [15], [18], [22] 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], [26]. 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 [17].

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 $D = 2$ [26]. 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, Sect. 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 [25].

For approximation, the following results are known. Ravi et al. [22] 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 [19]. 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 [21].

In this paper, we focus on the k-dispersion problem for

Manuscript received March 26, 2021.
Manuscript revised July 9, 2021.
Manuscript publicized November 1, 2021.

[†]The author is with Kyoto University, Kyoto-shi, 606–8501 Japan.
[††]The author is with Gunma University, Kiryu-shi, 376–8515 Japan.
[†††]The author is with Yamagata University, Yonezawa-shi, 992–8510 Japan.
[††††]The author is with National Institute of Informatics, Tokyo, 101–8430 Japan.
[†††††]The author is with Osaka University, Suita-shi, 565–0871 Japan.
[††††††]The author is with Iwate University, Morioka-shi, 020–8551 Japan.

[*]A preliminary version will appear in the Proceedings of the 37th The European Workshop on Computational Geometry (EuroCG2021), 2021 [20].

a) E-mail: kobayashi@iip.ist.i.kyoto-u.ac.jp
b) E-mail: nakano@cs.gunma-u.ac.jp
c) E-mail: uchizawa@yz.yamagata-u.ac.jp
d) E-mail: uno@nii.jp
e) E-mail: yutaro.yamaguchi@ist.osaka-u.ac.jp
f) E-mail: yamanaka@cis.iwate-u.ac.jp

DOI: 10.1587/transinf.2021FCP0013

Copyright © 2022 The Institute of Electronics, Information and Communication Engineers

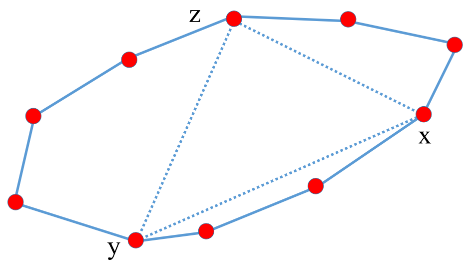


Fig. 1 An example of 3-dispersion. $\{x, y, z\}$ is a 3-dispersion.

$k = 3$. For this case, can we improve the running time $O(n^2 \log n)$? We show that the problem can be solved in $O(n^2)$ time when inputs have some restrictions. 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 [19] 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 3-dispersion of P in $O(n^2)$ time using the property that P is a set of points in convex position.

As mentioned above, if input points are on a circle, the problem can be solved efficiently [25]. On the other hand, we investigate that one can use properties of the convex position, which is a restriction to input point set looser than a circle, to design an efficient algorithm.

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 a 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 $\lceil \log n^2 \rceil = \lceil 2 \log n \rceil$.

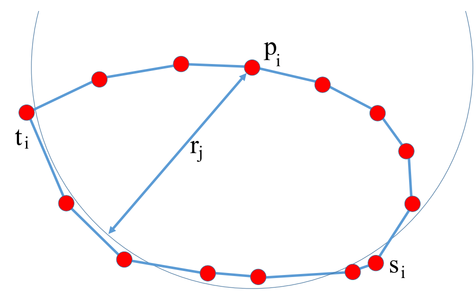


Fig. 2 An example of s_i and t_i for p_i. The circle is centered at p_i and of radius r_j.

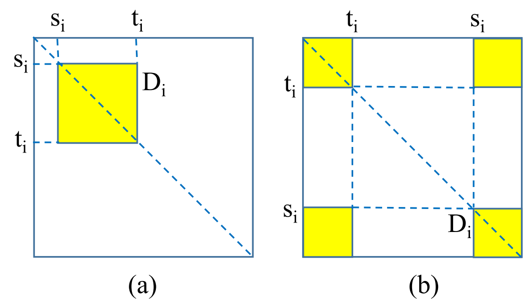


Fig. 3 Illustrations for the square submatrix D_i of D for p_i.

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 [24].

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

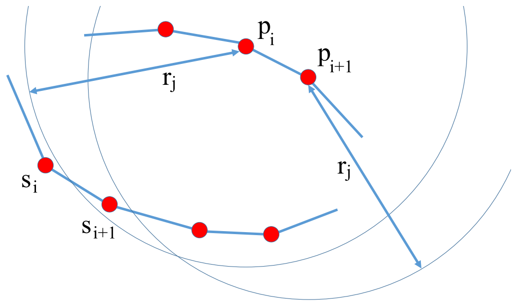


Fig. 4 The point s_{i+1} may appear before s_i on the clockwise contour.

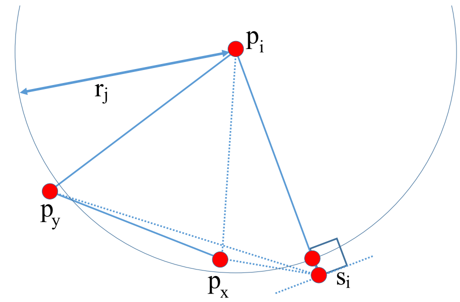


Fig. 5 An illustration for Lemma 3.

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. Since t_i can be computed in a similar way for finding s_i, 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 median of distances in C_1. Then, in each stage $j = 2, 3, \ldots, k$ ($k \leq \lceil 2 \log n \rceil$), 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 $d(p_x, p_y) \geq r_j$. 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* $\mathrm{cost}(S) \geq r_j$.

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

Case 1: $d(p_i, p_x) \geq r_j$ and $d(p_i, p_y) \geq r_j$.

The set $S = \{p_i, p_x, p_y\}$ has $\mathrm{cost}(S) \geq r_j$.

Case 2: $d(p_i, p_x) < r_j$ and $d(p_i, p_y) < r_j$.

We show that, for $S = \{p_i, s_i, t_i\}$, $\mathrm{cost}(S) \geq r_j$ 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 $d(p_i, p_y) \geq r_j$.

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

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 $d(p_i, p_y) \geq r_j$ and $d(p_i, s_i) \geq r_j$, 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 $d(p_i, p_y) \geq r_j$, $d(p_x, p_y) \geq r_j$, 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 $\angle s_i p_x p_y \geq 150°$, and $\{s_i, p_x, p_y\}$ are the points of an obtuse triangle with obtuse angle at p_x. However $d(p_x, p_y) \geq r_j$ and $d(s_i, p_y) < r_j$, which is a contradic-

Algorithm 1 Binary Search for the Dispersion Problem

```
Let C = {d(p_i, p_j) | 1 ≤ i < j ≤ n}.
while |C| ≥ 2 do
  Let r be the median in C.
  flag = NO
  for i = 1 to n do
    Let s_i ∈ P be the closest point satisfying d(p_i, s_i) ≥ r from p_i
    in the clockwise order. /* The search starts at s_i of the preceding
    stage if the flag of the preceding stage is YES, and starts at the
    starting point of the preceding stage otherwise. */
    Let t_i ∈ P be the closest point satisfying d(p_i, t_i) ≥ r from p_i in
    the counterclockwise order.
    if the submatrix defined by s_i . . . t_i is not empty then
      Find the maximum value x of the submatrix
      if x ≥ r then
        flag = YES
      end if
    end if
  end for
  if flag = YES then
    Remove all elements less than r from C.
  else
    Remove all elements greater than or equal to r from C.
  end if
end while
Output the element in C.
```

tion.

Case 4: $d(p_i, p_x) \geq r_j$ 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. Our algorithm is shown in Algorithm 1. First, as a preprocessing, we construct the set $C_1 = \{d(p_i, p_j) \mid 1 \leq i < j \leq n\}$ and $n \times n$ distance matrix D. Next, we repeat the following stage for each $j = 1, 2, \ldots, k$, where $k \leq \lceil 2 \log n \rceil$. (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, $(1 \leq i \leq n)$, such that the maximum value of D_i is greater than or equal to r_j. Then, if such i exists, we set $C_{j+1} = \{d(p_i, p_j) \in C_j \mid d(p_i, p_j) \geq r_j\}$, 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 [24], 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 [27], 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. One can compute a 3-dispersion of P in $O(n^2)$ time.*

4. Conclusion

In this paper, we have designed an algorithm to solve the 3-dispersion problem for a set of n points in convex position. We presented an $O(n^2)$-time algorithm to compute the 3-dispersion of P.

Acknowledgements

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.K. Agarwal and M. Sharir, "Efficient algorithms for geometric optimization," ACM Comput. Surv., vol.30, no.4, pp.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. FAW 2018, LNCS 10823, pp.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., vol.E103-D, no.3, pp.506–508, 2020.

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

[6] C. Baur and S.P. Fekete, "Approximation of geometric dispersion problems," Proc. APPROX 1998, vol.1444, pp.63–75, 1998.

[7] B. Birnbaum and K.J. Goldman, "An improved analysis for a greedy remote-clique algorithm using factor-revealing LPs," Algorithmica, vol.55, no.1, pp.42–59, 2009.

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

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

[10] B. Chandra and M.M. Halldórsson, "Approximation algorithms for dispersion problems," J. Algorithms, vol.38, no.2, pp.438–465, 2001.

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

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

[13] E. Erkut, "The discrete p-dispersion problem," European Journal of Operational Research, vol.46, no.1, pp.48–60, 1990.

[14] E. Erkut, Y. Ülküsal, and O. Yeniçerioğlu, "A comparison of p-dispersion heuristics," Computers & Operational Research, vol.21, no.10, pp.1103–1113, 1994.

[15] S.P. Fekete and H. Meijer, "Maximum dispersion and geometric maximum weight cliques," Algorithmica, vol.38, no.3, pp.501–511, 2004.

[16] G. Frederickson, "Optimal algorithms for tree partitioning," Proc. SODA 1991, pp.168–177, 1991.

[17] F.L. Gall, "Powers of tensors and fast matrix multiplication," Proc. ISSAC 2014, pp.296–303, 2014.

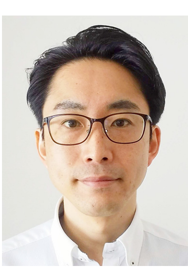
[18] R. Hassin, S. Rubinstein, and A. Tamir, "Approximation algorithms for maximum dispersion," Operation Research Letters, vol.21, no.3, pp.133–137, 1997.

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

[20] Y. Kobayashi, S. Nakano, K. Uchizawa, T. Uno, Y. Yamaguchi, and K. Yamanaka, "Max-min 3-dispersionon a convex polygon," Proc. EuroCG 2021, accepted, 2021.
[21] F.P. Preparata and M.I. Shamos, Computational geometry: an introduction, Springer-Verlag, 1985.
[22] S.S. Ravi, D.J. Rosenkrantz, and G.K. Tayi, "Heuristic and special case algorithms for dispersion problems," Operations Research, vol.42, no.2, pp.299–310, 1994.
[23] M. Sydow, "Approximation guarantees for max sum and max min facility dispersion with parameterised triangle inequality and applications in result diversification," Mathematica Applicanda, vol.42, no.2, pp.241–257, 2014.
[24] R.L. Rivest T.H. Cormen, C.E. Leiserson, and C. Stein, Introduction to algorithms, Third Edition, MIT Press, 2000.
[25] K.-H. Tsai and D.-W. Wang, "Optimal algorithms for circle partitioning," Proc. COCOON 1997, LNCS 1276, pp.304–310, 1997.
[26] D.W. Wang and Y.-S. Kuo, "A study on two geometric location problems," Information Processing Letters, vol.28, no.6, pp.281–286, 1988.
[27] H. Yuan and M.J. Atallah, "Data structures for range minimum queries in multidimensional arrays," Proc. SODA 2010, pp.150–160, 2010.

Yasuaki Kobayashi is an assistant professor of Graduate School of Informatics, Kyoto University. He received B.E., M.E. and Ph.D. degrees from Meiji University in 2009, 2011 and 2014, respectively. His research interests lie in algorithms and complexity, particularly in parameterized algorithms and complexity of hard problems.

Shin-ichi Nakano received his B.E. and M.E. degrees from Tohoku University, Sendai, Japan, in 1985 and 1987, respectively. In 1987 he joined Seiko Epson Corp. and in 1990 he joined Tohoku University. In 1992, he received Dr. Eng. degree from Tohoku University. Since 1999 he has been a faculty member of Department of Computer Science, Faculty of Engineering, Gunma University. His research interests are graph algorithms and graph theory. He is a member of IPSJ and ACM.



Kei Uchizawa received his B.E., M.S. and Ph.D. degrees from Tohoku University in 2003, 2005 and 2008, respectively. He was an assistant professor of Graduate School of Information Sciences at Tohoku University from 2008 to 2012. He is an associate professor of Graduate School of Science and Engineering at Yamagata University. His research interests include computational complexity and neural networks.

Takeaki Uno received the Ph.D. degree (Doctor of Science) from Department of Systems Science, Tokyo Institute of Technology Japan, 1998. He was an assistant professor in Department of Industrial and Management Science in Tokyo Institute of Technology from 1998 to 2001, and was an associate professor of National Institute of Informatics Japan, from 2001 to 2013. He is currently a professor of National Institute of Informatics Japan, from 2014. His research topic is discrete algorithms, especially enumeration algorithms, algorithms on graph classes, and data mining algorithms. On the theoretical part, he studies low degree polynomial time algorithms, and hardness proofs. In the application area, he works on the paradigm of constructing practically efficient algorithms for large scale data that are data oriented and theoretically supported. In an international frequent pattern mining competition in 2004 he won the best implementation award. He got Young Scientists' Prize of The Commendation for Science and Technology by the Minister of Education, Culture, Sports, Science and Technology in Japan, 2010.

Yutaro Yamaguchi is an associate professor of Graduate School and Faculty of Information Science and Electrical Engineering, Kyushu University. He received M.Sc. degree from Kyoto University in 2013 and Ph.D. degree from University of Tokyo in 2016, respectively. His research interests include combinatorial optimization, algorithms, and discrete mathematics.

Katsuhisa Yamanaka is a professor of Faculty of Science and Engineering, Iwate University. He received B.E., M.E. and Dr. Eng. degrees from Gunma University in 2003, 2005 and 2007, respectively. His research interests include combinatorial algorithms and graph algorithms.