5A-01

The LR-dispersion problem

Toshihiro Akagii* Tetsuya Araki† Shin-ichi Nakano*

1 Introduction

The facility location problem and many of its variants have been studied[7, 8]. A typical problem is to find a set of locations to place facilities with the designated cost minimized. By contrast, in this paper we consider the dispersion problem, which finds a set of locations with the designed cost maximized.

Given a set P of n points, and the distance d for each pair of points, and an integer k with $k \leq n$, we wish to find a subset $S \subset P$ with $|S| = k$ such that some designated cost is maximized[1, 5, 6, 10, 11, 12, 13].

In one of typical cases the cost to be maximized is the minimum distance between two points in S. If P is a set of points on the plane then the problem is NP-hard[12, 13], and if P is a set of points on the line then the problem can be solved in $O(\max\{n \log n, kn\})$ time[12, 13] by dynamic programming approach, and in $O(n \log \log n)$ time[1] by sorted matrix search method[4, 9].

In this paper we consider two variants of the dispersion problem on the line. Let P be a set of points on the horizontal line. We wish to find a subset $S \subset P$ with $|S| = k$ maximizing $cost(S)$ defined as follows.

Let the cost $cost(s)$ of $s \in S = \{s_1, s_2, \ldots, s_k\}$ be the sum of the distance to its left neighbor in S and the distance to its right neighbor in S. We assume $s_1, s_2, \ldots, s_k$ are sorted from left to right. Especially the leftmost point $s_1 \in S$ has no left neighbor, so we define the cost of s_1 is $d(s_1, s_2)$. Similarly the cost of the rightmost point s_k is $d(s_{k-1}, s_k)$. And the $cost(S)$ of S is the minimum cost among the costs $cost(s_1), cost(s_2), \ldots, cost(s_k)$. We call the problem above the *LR-dispersion problem.* An $O(kn^2 \log n)$ time algorithm based on dynamic programming is known[2].

In this paper we design an algorithm to solve the LR-dispersion problem. Our algorithm runs in $O(n \log n)$ time, and based on matrix search method[4, 9].

2 (λ, k)-LR-dispersion

In this section we give a linear time algorithm to solve a decision version of the LR-dispersion problem.

*Department of Computer Science, Gunma University
†National Institute of Informatics, Japan

Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of points on a horizontal line, and two numbers k and λ we wish to decide if there exists a subset $S \subset P$ with $|S| = k$ and $cost(S) \geq \lambda$. We call the problem as the (λ, k)-LR-dispersion problem. We have the following lemma.

Lemma 1. If (λ, k)-LR-dispersion problem has a solution $S = \{s_1, s_2, \ldots, s_k\} \subset P$, then $S' = \{p_1, s_2, s_3, \ldots, s_{k-1}, p_n\}$ is also a solution of the (λ, k)-LR-dispersion problem.

Proof. Since $cost(S) \leq cost(S')$, if S is a solution then S' is also a solution and $cost(S) = cost(S')$ holds. □

The algorithm below is a greedy algorithm to solve the (λ, k)-LR-dispersion problem. Note that $cost(s_i)$ for $i = 3, 4, \ldots, k-1$ is $d(s_{i-2}, s_i)$. By setting a dummy point $s_0 = s_1$, $cost(s_2)$ is also $d(s_{2-2}, s_2)$. Also note that $cost(k) = d(s_{k-1}, s_k)$.

Algorithm 1 find (λ, k)-LR-dispersion (P, k, λ)

```
/* P = {p_1, p_2, ..., p_n} and p_1, p_2, ..., p_n are sorted
from left to right */
/* Choose s_1 and s_k */
s_1 = p_1, s_k = p_n
s_0 = s_1                                   /* Dummy */
/* Choose s_2, s_3, ..., s_{k-1} */
c = 2
for i = 2 to k - 1 do
  while d(s_{i-2}, p_c) < λ and d(p_c, p_n) ≥ λ do
    c + +
  end while
  if d(p_c, p_n) < λ then
    /* no solution since d(p_c, p_n) < λ */
    return NO
  else
    /* d(s_{i-2}, p_c) ≥ λ holds */
    s_i = p_c                               /* s_i is found */
    c + +
  end if
end for
/* Output */
return S = {s_1, s_2, ..., s_k}
```

Now we prove the correctness of the algorithm. Assume for a contradiction that the algorithm output NO for a given problem but it has a solution.

Let $G = \{g_1, g_2, \ldots, g_{k'}\}$ with $k' < k$ be the points chosen by the algorithm, and $O = \{o_1, o_2, \ldots, o_k\}$ the

points of a solution. By Lemma 1 we can assume $o_1 = p_1$ and $o_k = p_n$. Note that $g_1 = o_1 = p_1$ and $g_{k'} = o_k = p_n$ hold. We have the following two cases.
Case 1 : For all i, $1 \leq i < k'$, $g_i \leq o_i$ holds.

Then our greedy algorithm can choose at least one more point $o_{k'}$ or more left point. A contradiction.
Case 2 : For some i, $1 \leq i < k'$, $g_i > o_i$ holds.

Since g_2 is chosen in a greedy manner, we can assume $g_2 \leq o_2$. Let j be the minimum such i. Since $g_{j-2} \leq o_{j-2}$ and $g_{j-1} \leq o_{j-1}$ hold, our greedy algorithm choose o_i or more left point as g_i. A contradiction.
Theorem 1. One can solve the decision version of the LR-dispersion problem in $O(n)$ time.

3 LR-dispersion

One can design an $O(n \log n)$ time algorithm to solve the LR-dispersion problem, based on the sorted matrix search method[4, 9]. See the long version[3] for detail.
Theorem 2. One can solve the LR-dispersion problem in $O(n \log n)$ time.

4 Generalization

In this section we consider one more variant of the dispersion problem and give an algorithm to solve the problem, which runs in $O(n \log n)$ time. In the original dispersion problem the cost is the minimum distance between two points s_i and s_{i+1}. We generalize this to the minimum distance between s_i and s_{i+h}, for given h.

Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of points on a horizontal line, and the distance d for each pair of points, and two integers k, h with $k, h \leq n$, we wish to find a subset $S = \{s_1, s_2, \ldots, s_k\} \subset P$ maximizing $cost(S)$ defined as follows.

$Lcost(S) = \min\{d(s_1, s_2), d(s_1, s_2), \ldots, d(s_1, s_h)\}$, $Rcost(S) = \min\{d(s_{k-h+1}, s_k), d(s_{k-h+2}, s_k), \ldots, d(s_{k-1}, s_k)\}$ and $Mcost(S) = \min\{d(s_1, s_{1+h}), d(s_2, s_{2+h}), \ldots, d(s_{k-h}, s_k)\}$, then $cost(S) = \min\{Lcost(S), Rcost(S), Mcost(S)\}$.

We call the problem above the *h-dispersion problem.* The original dispersion problem on the line is the h-dispersion problem with $h = 1$ and the LR-dispersion problem is the h-dispersion problem with $h = 2$.

See the long version[3] for detail.
Theorem 3. One can solve the h-dispersion problem in $O(n \log n)$ time.

5 Conclusion

In this paper we have presented two algorithms to solve the LR-dispersion problem and the h-dispersion problem. The running time of the algorithms are $O(n \log n)$.

An $O(n \log \log n)$ time algorithm to solve the original dispersion problem on the line is known[1]. Can we design an $O(n \log \log n)$ time algorithm to solve the h-dispersion problem ?

References

[1] T. Akagi and S. Nakano, Dispersion problem on the Line, Technical Report, 2016-AL-158-4, IPSJ (2016).

[2] T. Akagi, T. Araki and S. Nakano, Variants of the Dispersion Problem, Technical Report, 2017-AL-161-3, IPSJ (2017).

[3] T. Akagi and S. Nakano, The LR-dispersion problem, Technical Report, Gunma Univ. (2017). http://www.nakano-lab.cs.gunma-u.ac.jp/TP/akagi201611.pdf

[4] P. Agarwal and M. Sharir, Efficient Algorithms for Geometric Optimization, Computing Surveys, 30, pp.412-458 (1998).

[5] C. Baur and S. P. Feketee, Approximation of Geometric Dispersion Problems, Pro. of APPROX '98, Page 63-75 (1998).

[6] B. Chandra and M. M. Halldorsson, Approximation Algorithms for Dispersion Problems, J. of Algorithms, 38, pp.438-465 (2001).

[7] Z. Drezner, Facility Location: A Survey of Applications and Methods, Springer (1995).

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

[9] G. Frederickson, Optimal Algorithms for Tree Partitioning, Proc. of SODA'91 Pages 168-177 (1991).

[10] R. Hassin, S. Rubinstein and A. Tamir, Approximation Algorithms for Maximum Dispersion, Operation Research Letters, 21, pp.133-137 (1997).

[11] T. L. Lei and R. L. Church, On the unified dispersion problem: Efficient formulations and exact algorithms, European Journal of Operational Research, 241, pp.622-630 (2015).

[12] S. S. Ravi, D. J. Rosenkrantz and G. K. Tayi, Heuristic and Special Case Algorithms for Dispersion Problems, Operations Research, 42, pp.299-310 (1994).

[13] D. W. Wang and Y. S. Kou, A study on Two Geometric Location Problems, Information Processing Letters, 28, pp.281-286 (1988).