Vol.2017-AL-165 No.9
2017/11/16

Efficient Algorithms for the Partial Sum Dispersion Problem

Toshihiro Akagi[1,a)] Tetsuya Araki[2,b)] Hiroshi Ishikawa[2,c)]
Shin-ichi Nakano[1,d)]

Abstract:
The dispersion problem is a variant of the facility location problem. Given a set P of n points and an integer k, we intend to find a subset S of P with $|S| = k$ such that the cost $\min_{x \in S}\{cost(x)\}$ is maximized, where $cost(x)$ is the sum of the distances from x to the nearest c points in S. The main focus is the dispersion problem with partial c sum cost, referred to as the PcS-dispersion problem. In this paper we present two algorithms to solve the P2S-dispersion problem if all the points of P are on a line. The run time of the algorithms are $O(kn^2 \log n)$ and $O(n \log n)$, respectively. We also present an algorithm to solve the PcS-dispersion problem if all points of P are on a line. The run time of the algorithm is $O(kn^{c+1})$.

1. Introduction

The facility location problem and many of its variants have been studied [6], [7]. A typical problem is to find a set of locations to place facilities with the designated cost minimized. In this paper we consider the dispersion problem (or obnoxious facility location problem), which seeks to find a set of locations with a certain objective function based on distance maximized.

Given a set P of n possible locations, the distance d for each pair of locations, and an integer k with $k \leq n$, we wish to find a subset $S \subset P$ with $|S| = k$ such that the designated objective function based on distance is maximized [1], [3], [4], [5], [9], [10], [11], [12], [13].

The intuition of the problem is as follows. Assume that we plan to open several chain stores in a city. We wish to position the stores mutually far away from each other to avoid self-competition. We wish to find k locations so that the objective function based on the distance is maximized. Additional applications, including *result diversification*, are outlined in [10], [11], [12].

In one of the basic cases, the objective function to be maximized is the minimum distance between two points in S. Papers [11], [13] show if P is a set of points on the plane then the problem is NP-hard, 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 [11] by the dynamic programming method, and in $O(n)$ time by the sorted matrix search method [8].

In this paper we consider the following problem [10]. Given a set P of n points, the distance d for each pair of points, and an integer k, we intend to find a subset S of P with $|S| = k$ such that the $cost(S) = \min_{p \in S}\{cost(p)\}$ is maximized, where $cost(p)$ is the sum of the distances from p to the nearest c points in S. Fig. 1 depicts an example of S with $c = 2$ and $cost(S) = 4$. We refer to this as the dispersion problem with partial c sum cost [10] (PcS-dispersion problem). Intuitively, this cost models self-competition to the nearest c stores. A number of experimental results (for more general problems) are known. See [10]. The basic dispersion problem is P1S-dispersion problem.

In this paper we designed two algorithms to solve the P2S-dispersion problem if all the points of P are on a line. The run time of the algorithms are $O(kn^2 \log n)$ and $O(n \log n)$, respectively. Similarly, we design an algorithm to solve the PcS-dispersion problem for any constant c if all points of P are on a line. The run time of the algorithm is $O(kn^{c+1})$.

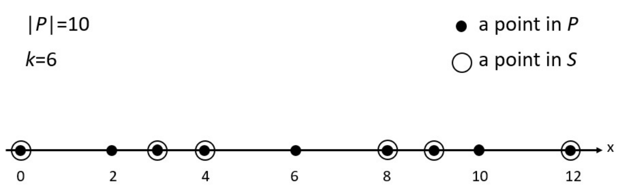

Fig. 1 An example of S with $cost(S) = 4$

1 Department of Computer Science, Gunma University, Kiryu, 376–8515, Japan
2 Faculty of System Design, Tokyo Metropolitan University, Hino, Tokyo 191–0065, Japan
a) akagi@nakano-lab.cs.gunma-u.ac.jp
b) araki@tmu.ac.jp
c) ishikawa-hiroshi@tmu.ac.jp
d) nakano@cs.gunma-u.ac.jp

2. P2S-dispersion problem on a line

2.1 Dynamic programming method

In this section, we designed an algorithm to solve the P2S-dispersion problem, which is based on the dynamic programming method, if all points of P are on a horizontal line. We define the subproblem $P2S(h,i;k)$ for our dynamic programming as follows.

Let P_i be the subset of the points in P located on the left of $p_i \in P$ including p_i, where p_i is the i-th point from the left in P. Given $p_h \in P_i$ and an integer $k \geq 3$, we intend to find a subset $S \subset P_i$ such that $|S| = k$ and the rightmost two points in S are p_h and p_i, with $h < i$. As a result, $cost(S)$ is maximized. This is the subproblem $P2S(h,i;k)$. We denote $cost(h,i;k)$ as the cost of a $P2S(h,i;k)$ solution. This is the P2S-dispersion problem in which the rightmost two points in S are designated. We can observe that $P2S(h,i;k)$ has a solution S containing the leftmost and rightmost points in P_i. Thus we can assume $p_1, p_i \in S$.

We have the following lemma.

Lemma 1. *If $k = 3$ then $cost(h,i;k) = d(p_1,p_i)$.*

Proof. The solution of $P2S(h,i;3)$ is $\{p_1,p_h,p_i\}$. Then $cost(p_h) = d(p_h,p_i) + d(p_h,p_1) = d(p_1,p_i)$, $cost(p_1) = d(p_1,p_h) + d(p_1,p_i) > cost(p_h)$, and $cost(p_i) = d(p_h,p_i) + d(p_1,p_i) > cost(p_h)$ hold. Thus $cost(h,i;3) = d(p_1,p_i)$. □

Thus when we compute $cost(h,i;k)$ which is the minimum over $cost(p)$ for $p \in S$, we can ignore $cost(p_i)$ since $cost(p_i) > cost(p_h)$ always holds.

Lemma 2. *If $k \geq 4$ then $cost(h,i;k) = max_{h'=k-2,k-1,\cdots,h-1} \min\{cost(h',h;k-1), d(p_{h'},p_i)\}$*

Proof. Assume S be the solution of $P2S(h,i;k)$, and $p_{h'}$ the third rightmost point in S. Now $h' \geq k-2$ holds, since $|S| = k$. Assume to $cost(h,i;k) = cost(p_x)$ for a number of $p_x \in S$. We have the following three cases.

Case 1: $x < h$.
$cost(p_x) = cost(h',h;k-1)$, and $cost(p_x) \leq cost(p_h) \leq d(p_{h'},p_i)$. Thus $cost(p_x) = \min\{cost(h',h;k-1), d(p_{h'},p_i)\}$ holds.

Case 2: $x = h$.
We have two subcases.
If $cost(p_x) = d(p_{h'},p_h) + d(p_{h''},p_h)$, where $p_{h''}$ is the 4-th rightmost point in S. Then $cost(p_x) = d(p_{h'},p_h) + d(p_{h''},p_h) > cost(p_{h'})$. This is a contradiction.
If $cost(p_x) = d(p_{h'},p_h) + d(p_h,p_i)$, then $cost(p_x) = d(p_{h'},p_i) \leq cost(h',h;k-1)$. Thus $cost(p_x) = \min\{ cost(h',h;k-1), d(p_{h'},p_i)\}$ holds.

Case 3: $x = i$.
Since $cost(p_h) < cost(p_i)$, this case will never occur.

Since we compute $\min\{ cost(h',h;k-1), d(p_{h'},p_i)\}$ for every possible h', and choose the maximum one, so the equation computes $cost(h,i;k)$ correctly. □

The number of the subproblems is at most kn^2 and we can compute a solution of each subproblem in $O(n)$ time by Lemma 2. The entire algorithm is shown in Algorithm 1.

Algorithm 1 Find-P2S-dispersion(P,n,k)

```
% Compute P(h,i;3)      (Case k = 3)
for i = 3,4,···,n do
  for h = 2,3,···,i−1 do
    cost(h,i;3) = d(p_1,p_i)
  end for
end for
% Compute P(h,i;k)      (Case k > 4)
for k' = 4,5,···,k do
  for i = k',k'+1,···,n do
    for h = k'−1,k',···,i−1 do
      cost(h,i;k') = 0
      % Compute the maximum cost
      for h' = k'−2,k'−1,···,h−1 do
        cost(h,i;k') = max{cost(h,i;k'), min{cost(h',h;k'−
        1), d(p_h',p_i)}}
      end for
    end for
  end for
end for
% Compute the optimal cost
cost = 0
for h = k−1,k,···,n−1 do
  if cost(h,n;k) > cost then
    cost = cost(h,n;k)
  end if
end for
Output cost
```

We have the following theorem [2].

Theorem 1. *One can solve the P2S-dispersion problem in $O(kn^3)$ time.*

We have the following lemma.

Lemma 3. *$cost(h',h;k-1)$ is a non-decreasing function with respect to h'.*

Proof. Assume otherwise. For a number of p_{h_L}, p_{h_R} in P with $h_L < h_R$, $cost(h_L,h;k-1) > cost(h_R,h;k-1)$ holds. Note that $cost(h',h;k-1)$ is $\min_{p \in S}\{cost(p)\}$. Let S_L be the solution of $P2S(h_L,h;k-1)$ and S' be the set of points derived from S_L by removing p_{h_L} then appending p_{h_R}. Also let p_x be the left neighbour of p_{h_L} in S_L, and p_y be the left neighbour of p_x in S_L. $cost(p_x)$ in S_L is not larger than $cost(p_x)$ in S', and $cost(p_y)$ in S_L is not larger than $cost(p_y)$ in S'. We can also show $cost(p_{h_L})$ in S_L is not larger than $cost(p_{h_R})$ in S', since $cost(p_{h_L})$ in S_L is $\min\{d(p_x,p_h), d(p_y,p_{h_L}) + d(p_x,p_{h_L})\}$ and $cost(p_{h_R})$ in S' is $\min\{d(p_x,p_h), d(p_y,p_{h_R}) + d(p_x,p_{h_R})\}$ and $d(p_y,p_{h_L}) + d(p_x,p_{h_L}) < d(p_y,p_{h_R}) + d(p_x,p_{h_R})$. Thus, $cost(h_L,h;k-1) \leq \min_{p \in S_L}\{cost(p)\} \leq \min_{p \in S'}\{cost(p)\} \leq cost(h_R,h;k-1)$ holds. This is a contradiction. □

Therefore, $\min\{cost(h',h;k-1), d(h',i)\}$ is a non-decreasing function with respect to h' up to a number of points, which is then a decreasing linear function with respect to h', so we can find the maximum one by binary search $\log n$ times.

We have the following theorem [2].

Theorem 2. *One can solve the P2S-dispersion problem in $O(kn^2 \log n)$ time.*

2.2 Matrix search method

In this section, we solved the P2S-dispersion problem using the matrix search method. We first designed an algorithm to solve the decision version of the P2S-dispersion problem.

Given two numbers k and λ, we intend to decide if there exists a subset $S \subset P$ with $|S| = k$ and $cost(S) \geq \lambda$. We refer to the decision problem as the (λ, k)-P2S-dispersion problem.

Lemma 4. *If the answer of (λ, k)-P2S-dispersion problem is YES then one can assume $\{p_1, p_2, p_n\} \subset S$*

Proof. This is similar to the proof of Lemma 1, and has therefore been omitted. □

Algorithm decide-P2S-dispersion shown in Algorithm 2 solves the decision problem.

Algorithm 2 Decide-P2S-dispersion(P, k, λ)

```
s1 = p1, s2 = p2
c = 3
for i = 3, 4, ..., n do
  if d(s_{c-2}, p_i) >= λ then
    s_c = p_i
    c = c + 1
  end if
end for
if c > k then
  return YES
else
  return NO
end if
```

Lemma 5. *Algorithm establishes (λ, k)-P2S-dispersion correctly and determines if there exists a subset $S \subset P$ with $|S| = k$ and $cost(S) \geq \lambda$.*

Proof. Assume otherwise. There exists $S' = \{s'_1, s'_2, \cdots, s'_k\} \subset P$ with $|S'| = k$ and $cost(S') \geq \lambda$, however, the algorithm outputs NO. We assume the points in S' are sorted from left to right. Let $S = \{s_1, s_2, \cdots\}$ be the set of points selected by the algorithm. Now $p_1, p_2 \in S$ holds. By Lemma 4, $p_1 = s'_1, p_2 = s'_2 \in S'$ holds. Let j be the minimum j with $x(s_j) > x(s'_j)$, where $x(s)$ is the coordinate of S. (If j dose not exist, then the algorithm outputs YES, which is a contradiction.) Now, $x(s_{j-1}) \leq x(s'_{j-1})$ and $x(s_{j-2}) \leq x(s'_{j-2})$ hold. We have two cases. If $cost(s'_{j-1})$ in S' is $d(s'_{j-2}, s'_j)$ then $\lambda \leq d(s'_{j-2}, s'_j)$ and $\lambda \leq d(s_{j-2}, s_j)$ holds. This contradicts the choice of s_j in the algorithm, which is either s'_j or specific points left of s'_j would be chosen as s_j. Otherwise, the nearest two points from s'_{j-1} in S' are either s'_{j-3} and s'_{j-2} or s'_j and s'_{j+1}, respectively, and $cost(s'_{j-1}) < d(s'_{j-2}, s'_j)$ holds. As a result, $\lambda \leq cost(s'_{j-1}) < d(s'_{j-2}, s'_j)$ and $\lambda \leq d(s_{j-2}, s_j)$ holds again. This contradicts the choice of s_j in the algorithm. □

Therefore, we have the following theorem.

Theorem 3. *One can solve the the (λ, k)-P2S-dispersion problem in $O(n)$ time.*

The following theorem is known.

Theorem 4. *(Matrix Search [8])*
Let D be a matrix consisting of candidate values for the optimal parameter for a decision problem and each row and column of D are sorted. We assume if the decision problem return YES for parameter λ then for any $\lambda' < \lambda$ the decision problem returns YES. We assume we do not store the entire matrix explicitly, but can access each entry of D in $O(1)$ time. If there is an $O(n)$ time algorithm for the decision problem for parameter λ, one can compute the optimal (maximum) parameter λ in $O(n \log n)$ time.

Let D be the distance matrix in which $d_{ij} = d(p_i, p_j)$. Each row and column of D are sorted and we can compute d_{ij} in $O(1)$ time. With this $O(n)$ time decision algorithm for the (λ, k)-P2S-dispersion problem, and by using the theorem above we can compute the optimal parameter λ for the P2S-dispersion problem in $O(n \log n)$ time.

Theorem 5. *One can solve the P2S-dispersion problem in $O(n \log n)$ time.*

Even though our second algorithm is theoretically faster than our first algorithm, it is difficult to implement. On the other hand our first algorithm is easier to implement.

3. PcS-dispersion problem on a line

In this section we designed an algorithm to solve the PcS-dispersion problem, based on the dynamic programming method, if all points of P are on a horizontal line. We define the subproblem $PcS(h_{c-1}, h_{c-2}, \cdots, h_1, i; k)$ for dynamic programming as follows.

Let P_i be the subset of the points in P located on the left of $p_i \in P$ including p_i, where p_i is the i-th point from the left in P. Given $p_{h_{c-1}}, p_{h_{c-2}}, \cdots, p_{h_1} \in P_i$ and an integer $k \geq c+1$, we intend to find a subset $S \subset P_i$ such that $|S| = k$ and the rightmost c points in S are $p_{h_{c-1}}, p_{h_{c-2}}, \cdots, p_{h_1}$ and p_i, with $h_{c-1} < h_{c-2} < \cdots < h_1 < i$, which as a result, maximize $cost(S)$. This is the subproblem $PcS(h_{c-1}, h_{c-2}, \cdots, h_1, i; k)$. We denote $cost(h_{c-1}, h_{c-2}, \cdots, h_1 \ , i; k)$ as the cost of a solution of PcS($h_{c-1}, h_{c-2}, \cdots, h_1, i; k$). We have the following lemma.

Lemma 6. *If $k = c+1$ then $cost(h_{c-1}, h_{c-2}, \cdots, h_1 \ , i; k) = d(p_1, p_i)$.*

Proof. Then $S = \{p_1, p_{h_{c-1}}, p_{h_{c-2}}, \cdots, p_{h_1}, p_i\}$, similar to Lemma 1. □

Assume $cost(h_{c-1}, h_{c-2}, \cdots, h_1 \ , i; k) = cost(p_x)$ for some $p_x \in S$. Then we have the following four lemmas.

Lemma 7. *$cost(p_x) = c(p_x)$, where $c(p_x)$ is the sum of the c distances from p_x to the nearest $\lceil c/2 \rceil$ points in S lo-*

cating left of p_x and the nearest $\lfloor c/2 \rfloor$ points in S locating right of p_x.

Proof. Assume otherwise. For an integer $g \neq 0$, $cost(p_x)$ is the sum of the distances from p_x to the nearest $\lceil c/2 \rceil + g$ points in S located to the left of p_x and the nearest $\lfloor c/2 \rfloor - g$ points in S located to the right of p_x. First, consider the case in which c is even. If $g > 0$ then $cost(p_x) > cost(p_{x_L})$ holds, where p_{x_L} is the left neighbor of p_x in S. If $g < 0$ then $cost(p_x) > cost(p_{x_R})$ holds, where p_{x_R} is the right neighbor of p_x in S. This is a contradiction. For the case in which c is odd, we can prove this in a similar manner, but with more cases. (Note that if c is odd then $c(p_x)$ equals the sum of the distances from p_y to the nearest $\lfloor c/2 \rfloor$ points in S located to the left of p_y and the nearest $\lceil c/2 \rceil$ points in S located to the right of p_y, where p_y is the left neighbour of p_x in S.) □

Lemma 8. *Let R be the subset of S consisting of $\lfloor c/2 \rfloor$ rightmost points in S. Then $p_x \notin R$.*

Proof. Immediate from Lemma 7. □

Using Lemma 8 when we compute $cost(h_{c-1}, h_{c-2}, \cdots, h_1, i; k)$ which is the minimum over $cost(p)$ for $p \in S$, we can ignore the $\lfloor c/2 \rfloor$ costs $cost(p_i), cost(h_1), \cdots, cost(h_{\lfloor c/2 \rfloor - 1})$.

Lemma 9. *If c is an even integer then $cost(h_{c-1}, h_{c-2}, \cdots, h_1, i; k) = \max_{h'=k-c,k,\cdots,h_{c-1}-1} \min\{cost(h', h_{c-1}, h_{c-2}, \cdots, h_1;\, k-1), c(p_{h_{c/2}})\}$.*

Proof. Let S be the solution to $PcS(h_{c-1}, h_{c-2}, \cdots, h_1, i; k)$, and $p_{h'}$ be the $(c+1)$-th rightmost point in S. $h' \geq k - c$ holds since $|S| = k$, and $S - \{p_i\}$ is a solution of $PcS(h', h_{c-1}, h_{c-2}, \cdots, h_1; k-1)$. Assume $cost(h_{c-1}, h_{c-2}, \cdots, h_1, i; k) = cost(p_x)$ for a number of $p_x \in S$. We have the following three cases.

Case 1: $x < h_{c/2}$.
$cost(p_x) = cost(h', h_{c-1}, h_{c-2}, \cdots, h_1; k-1)$, and $cost(p_x) \leq cost(p_{h_{c/2}}) \leq c(p_{h_{c/2}})$. Thus $cost(p_x) = \min\{cost(h', h_{c-1}, h_{c-2}, \cdots, h_1;\, k-1), c(p_{h_{c/2}})\}$ holds.

Case 2: $x = h_{c/2}$.
$cost(p_x) = c(p_{h_{c/2}}) \leq cost(h', h_{c-1}, h_{c-2}, \cdots, h_1; k-1)$. Thus $cost(p_x) = \min\{cost(h', h_{c-1}, h_{c-2}, \cdots, h_1; k-1), c(p_{h_{c/2}})\}$ holds.

Case 3: $x > h_{c/2}$.
Using Lemma 8, this case never occur. □

Lemma 10. *If c is an odd integer, then $cost(h_{c-1}, h_{c-2}, \cdots, h_1, i; k) = \max_{h'=k-c,k,\cdots,h_{c-1}-1} \min\{cost(h', h_{c-1}, h_{c-2}, \cdots, h_1;\, k-1), c(p_{h_{\lfloor c/2 \rfloor}})\}$.*

Proof. This has been omitted as it is similar to Lemma 9. Note that $c(p_{h_{\lfloor c/2 \rfloor}}) = c'(p_{h_{\lceil c/2 \rceil}})$, where $c'(p_{h_{\lceil c/2 \rceil}})$ is the distances from $p_{h_{\lceil c/2 \rceil}}$ to the nearest $\lfloor c/2 \rfloor$ points in S located to the left of $p_{h_{\lceil c/2 \rceil}}$ and the nearest $\lceil c/2 \rceil$ points in S located to the right of $p_{h_{\lceil c/2 \rceil}}$. □

One can compute $c(p)$ in $O(1)$ time since c is a constant. The number of subproblems is at most kn^c and we can solve each subproblem in $O(n)$ time. Therefore we can solve the PcS-dispersion problem in $O(kn^{c+1})$ time.

We have the following theorem [2].

Theorem 6. *One can solve the PcS-dispersion problem in $O(kn^{c+1})$ time.*

4. Conclusion

In this paper we gave two algorithms for the P2S-dispersion problem. The running time of them are $O(kn^2 \log n)$ and $O(n \log n)$. Also we gave an algorithm to solve the PcS-dispersion problem. The running time of the algorithm is $O(kn^{c+1})$.

We can observe that PcS-dispersion problem has a solution S containing the leftmost $\lfloor c/2 \rfloor$ points and rightmost $\lfloor c/2 \rfloor$ points in P_i. Thus we can assume $p_1, p_2, \cdots, p_{\lfloor c/2 \rfloor} \in S$ and $p_{n-\lfloor c/2 \rfloor-1}, p_{n-\lfloor c/2 \rfloor}, \cdots, p_n \in S$, so we can also solve the PcS-dispersion problem in $O((n-c)^{k-c})$ time by choosing remaining $k - 2\lfloor c/2 \rfloor$ points form $n - 2\lfloor c/2 \rfloor$ points by brute force method for large c.

Acknowledgements

This work was partly supported by Tokyo Metropolitan University Grant-in -Aid for Research on Priority Areas“Research on social big data.”

References

[1] T. Akagi and S. Nakano, Dispersion on the Line, Technical Report, 2016-AL-158-3, IPSJ (2016)
[2] T. Akagi, T. Araki and S. Nakano, Variants of the dispersion problem, Technical Report, 2017-AL-161-8, IPSJ (2017)
[3] T. Akagi, T. Araki and S. Nakano, The LR-dispersion Problem, LA Symposium (2017).
[4] C. Baur and S.P. Feketee, Approximation of Geometric Dispersion Problems, Pro. of APPROX '98, Pages 63-75 (1998).
[5] B. Chandra and M. M. Halldorsson, Approximation Algorithms for Dispersion Problems, J. of Algorithms, 38, pp.438-465 (2001).
[6] Z. Drezner, Facility Location: A Survey of Applications and Methods, Springer (1995).
[7] Z. Drezner and H.W. Hamacher, Facility Location: Applications and Theory, Springer (2004).
[8] G. Frederickson, Optimal Algorithms for Tree Partitioning, Proc. of SODA '91 Pages 168-177 (1991).
[9] R. Hassin, S. Rubinstein and A. Tamir, Approximation Algorithms for Maximum Dispersion, Operation Research Letters, 21, pp.133-137 (1997).
[10] 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).
[11] 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).
[12] M. Sydow, Approximation Guarantees for Max Sum and Max Min Facility Dispersion with Parameterised Triangle Inequality and Applications in Result Diversification, Mathematica Applicanda, 42, pp.241-257 (2014).
[13] D. W. Wang and Yue-Sun Kuo, A study on Two Geometric Location Problems, Information Processing Letters, 28, pp.281-286 (1988).

