Vol.2017-AL-161 No.9
2017/1/17

A Simple Algorithm for r-gather-clusterings on the Line

Shin-ichi Nakano[1],[a)]

Abstract: In this paper we study a recently proposed two variants of the facility location problem, called the r-gather-clustering problem and the r-gathering problem.
Given a set C of n points on the plane an *r-gather-clustering* is a partition of the points into clusters such that each cluster has at least r points. The r-gather-clustering problem finds the r-gather-clustering minimizing the maximum radius among the clusters, where the radius of a cluster is the minimum radius of the disk which can cover the points in the cluster. A polynomial time 2-approximation algorithm for the problem is known.
When all C are on the line, an $O(n \log n)$ time algorithm, based on the matrix search method, to find an r-gather-clustering is known. In this paper we give an $O(n \log^* n)$ time algorithm to solve the problem.
We also give an algorithm to solve a similar problem, called the r-gathering problem.

1. Introduction

The facility location problem and many of its variants are studied[5], [6].

In this paper we study recently proposed two variants of the problem, called the r-gather-clustering problem and the r-gathering problem [1], [4].

Given a set C of n points on the plane an *r-gather-clustering* is a partition of the points into clusters such that each cluster has at least r points. The cost of an r-gather-clustering is the maximum radius among the clusters, where the radius of a cluster is the minimum radius of the disk which can cover the points in the cluster. The r-gather-clustering problem [1] is the problem to find the r-gather-clustering minimizing the cost. The problem is NP-complete in general, however a polynomial time 2-approximation algorithm for the problem is known[1]. When all C are on the line, an $O(n \log n)$ time algorithm, based on the matrix search method[2], [7], for the problem is known[3].

In this paper we give an $O(n \log^* n)$ time algorithm to solve the problem, by reducing the problem to the min-max path problem[9] in a weighted directed graph.

Assume that C is a set of residents and we wish to locate emergency shelters for the residents so that each shelter serves r or more residents. Then r-gather clustering problem computes optimal locations for shelters which minimizung the evacuation time span, where each shelter for a cluster is located at the center of the cluster.

In this paper we consider one more similar problem. Given sets C and F of points on the plane an *r-gathering* of C to F is an assignment A of C to *open facilities* $F' \subset F$ such that r or more customers are assigned to each open facility. The cost of an r-gathering is the maximum distance $d(c, f)$ between $c \in C$ and $A(c) \in F'$ among the assignment, which is $\max_{c \in C, A(c) \in F'} \{d(c, A(c))\}$.

Assume that F is a set of possible locations for emergency shelters, and $d(c, f)$ is the time needed for a person $c \in C$ to reach a shelter $f \in F$. Then an r-gathering corresponds to an evacuation assignment such that each opened shelter serves r or more people, and the r-gathering problem finds an evacuation plan minimizing the evacuation time span.

Armon[4] gave a simple 3-approximation algorithm for the r-gathering problem and proves that with the assumption $P \neq NP$ the problem cannot be approximated within a factor of less than 3 for any $r \geq 3$. When all C and F are on the line an $O((|C| + |F|) \log(|C| + |F|))$ time algorithm[3] and an $O(|C| + |F| \log^2 r + |F| \log |F|)$ time algorithm[10] to solve the r-gathering problem are known.

In this paper we give an $O(|C| + r^2|F| \log^* |C|)$ time algorithm to solve the problem, where $\log^* |C|$ is the number of times the log must be iteratively applied before results in less than 1. Since in typical case $r << |F| << |C|$ holds our new algorithm is faster than the known algorithms.

The remainder of this paper is organized as follows. Section 2 gives an algorithm for the r-gather-clustering problem. Section 3 gives an algorithm for the r-gathering problem. Finally Section 4 is a conclusion.

2. r-gather-clustering on the line

In this section we give an algorithm for the r-gather-clustering problem when all points in C are on the line. Let $C = \{c_1, c_2, \cdots, c_n\}$ be points on the horizontal line and we assume they are sorted from left to right. Our idea is to reduce the r-gather-clustering problem to the mix-max path problem in a weighted directed (acyclic) graph[9]. First we have the follow-

1 Gunma University, Kiryu 376-8515, Japan
a) nakano@cs.gunma-u.ac.jp

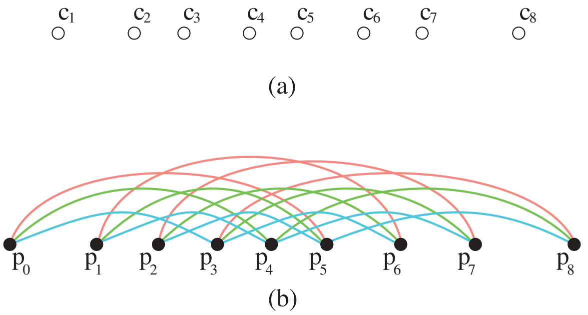

Fig. 1 the weighted directed path D.

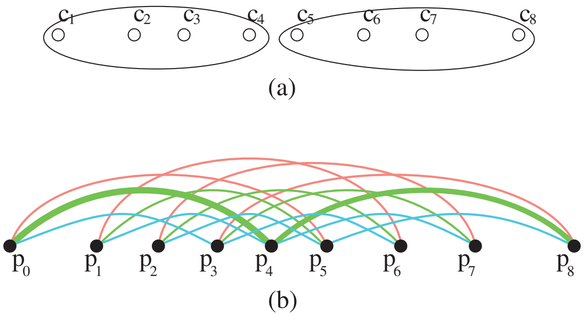

Fig. 2 (a)an r-gather clustering (b)its corresponding min-max path of D.

ing two lemmas.

Lemma 2.1 One can assume the points in each cluster in a solution are consecutive.

Proof. Otherwise repeat swapping some points between the clusters until the condition holds, which never increase the cost. $\mathcal{Q.E.D.}$

Lemma 2.2 One can assume the number of points in each cluster in a solution is at most $2r - 1$.

Proof. Otherwise devide such clusters into two (or more) clusters, respectively, which never increase the cost. $\mathcal{Q.E.D.}$

Then we difine the directed (acyclic) graph $D(V, E)$ and the weight of each edge, as follows.

$$V = \{p_0, p_1, p_2, \cdots, p_n\}$$

$$E = \{(p_i, p_j) | i + r \leq j \leq i + 2r - 1\}$$

See Fig. 1. Note that the number of edges is at most rn. The weight w of an edge $w(p_i, p_j)$ is the half of the distance between c_{i+1} and c_j, and denoted by $w(p_i, p_j)$.

The cost of a directed path from p_0 to p_n is defined by the weight of the edge having the maximum weight in the directed path. *The min-max path* from p_0 to p_n is the directed path from p_0 to p_n with the minimum cost.

Now C has an r-gather-clustering with cost k iff $D(V, E)$ has a directed path from p_0 to p_n with cost k. See Fig. 2.

Thus if we can compute the min-max path in D then it corresponds to the solution of the r-gather-clustering problem. Intuitively, each (directed) edge in the min-max path corresponds to a cluster of an r-gather-clustering.

We can construct the $D(V, E)$ in $O(rn)$ time. Then compute the min-max path from p_0 to p_n in $O(rn \log^* n)$ time, since an $O(|E| \log^* |V|)$ time algorithm for the min-max path problem for a directed graph $D = (V, E)$ is known [9].

Thus we have the following theorem.

Theorem 2.3 One can solve the r-gather-clustering problem in $O(rn \log^* n)$ time, when all points in C are on the line.

3. *r*-gathering

In this section we give an algorithm for the r-gathering problem when all points in C and F are on the line, by reducing the problem to the min-max path problem for a weighted directed graph.

Let $C = \{c_1, c_2, \cdots, c_n\}$ and $F = \{f_1, f_2, \cdots, f_m\}$ be points on the horizontal line and we assume they are sorted from left to right, respectively. Similar to Lemma 2.1 we can assume the points assigned to a facility are consecutive in a solution.

For consecutive three facilities f_{j-1}, f_j and f_{j+1} in F let m_L be the midpoints of f_{j-1} and f_j, and m_R the midpoints of f_j and f_{j+1}. We have the following two lemma.

Lemma 3.1 If C has $2r$ or more points on the left of m_L, then $c_{i'}$ with $i' < i$ is never assigned to f_j in a solution of the r-gathering problem, where c_i is the $2r$-th point in C on or left of m_L.

Proof. Assume for a contradiction such $c_{i'}$ is assigned to f_j. We have two cases.

If the rightmost point assigned to f_j is on the left of m_L then reassigning the points assigned to f_j to f_{j-1} results in a new r-gathering and since it does not increase the cost the resulting r-gathering is also a solution of the given r-gatheing problem.

Otherwise, the rightmost point assigned to f_j is on or right of m_L. Then at least $2r$ points on or left of m_L are assigned to f_j (possibly with other points on the right of m_L) Let C' be the subset of C consisting of the points (1) assigned to f_j, (2) on or left of m_L, and (3) but not the rightmost r points on or left of m_L. Note that $|C'| \geq r$ holds and C' contains $c_{i'}$. Reassigning the points in C' to f_{j-1} results in a new r-gathering and the resulting r-gathering is also a solution since it does not increase the cost. $\mathcal{Q.E.D.}$

Intuitively if $c_{i'}$ is too far form f_j then $c_{i'}$ is never assigned to f_j. Symmetrically we have the following lemma.

Lemma 3.2 If C has $2r$ or more points on the right of m_R, then $c_{i'}$ with $i' > i$ is never assigned to f_j, where c_i is the $2r$-th point in C on or right of m_R.

We have more lemma. Let C' be the set of points between m_L and m_R except the leftmost $2r$ points and the rightmost $2r$ points.

Lemma 3.3 If C has $5r$ or more points between m_L and m_R, then the customers in C' are assigned to f_j in a solution of the r-gathering problem.

Proof. Immediate from the two lemmas above. $\mathcal{Q.E.D.}$

Thus if we can compute the solution for $C - C'$ then appending the assignment from points in C' to f_j results in the solution for C. From now on we assume we have removed every such C' from C.

We have more lemmas for the boundary case. Let m be the midpoints of f_1 and f_2 in F.

Lemma 3.4 If C has $2r$ or more points on the left of m, then each $c_{i'}$ with $i' < i$ is assigned to f_1 in a solution of the r-gathering problem, where c_i is the $2r$-th customer in C on the left of m.

Proof. Immediate from Lemma 3.1. $\mathcal{Q.E.D.}$

Let m be the midpoints of f_{m-1} and f_m in F.

Lemma 3.5 If C has $2r$ or more points on the right of m, then each $c_{i'}$ with $i' > i$ is assigned to f_m in a solution of the r-gathering problem, where c_i is the $2r$-th customer in C on the right of m.

Thus we have the following lemma.

Lemma 3.6 The number of points in C possibly assignning to each facility $f \in F$ is at most $9r$.

Proof. For each f_j with $1 < j < m$ define m_L and m_R as above. The number of points possibly assiging to f_j is (1) at most $2r$ on the left of m_L, (2) at most $2r$ on the right of m_R, and (3) at most $5r$ between m_L and m_R, by the lemmas above. Similar for f_1 and f_m. $\mathcal{Q.E.D.}$

Now we are going to define a weighted directed graph $D(V, E)$ for F and C, and the weight of each edge.

The set of vertices is defined as follows.

$$V = \{p_0, p_1, p_2, \cdots, p_n\}$$

For each facility f_h with $h = 2, 3, \cdots, m-1$ and its possible cluster consisting of points $\{c_{i+1}, c_{i+2}, \cdots c_j\}$ we define an edge (p_i, p_j). So (p_i, p_j) is an edge iff
(1) $i + r \leq j \leq i + 2r - 1$
(2) $i \geq i'$ where i' is the $2r$-th customer on the left of m_L, and
(3) $j \leq j'$ where j' is the $2r$-th customer on the right of m_R,
where m_L and m_R are defined for f_h as in Section 2. Let E_j be the set of edges consisting of edges defined above. Simillary we define E_1 and E_m.

Finally,

$$E = E_1 \cup E_2 \cup \cdots E_m$$

Note that G may contain many multi-edges.

The weight w of an edge (p_i, p_j) for f_h is the maximum of (1) the distance between p_i and f_h, and (2) the distance between p_j and f_h.

The cost of a directed path from p_0 to p_n is defined by the weight of the edge having the maximum weight in the directed path. *The min-max path* from p_0 to p_n is the directed path from p_0 to p_n with the minimum cost.

We need to compute for each f_h the $2r$-th customer on the left of m_L and the $2r$-th customer on the right of m_R. By scanning the line we can compute them for all f_h in $O(|F| + |C|)$ time in toal. Note that each edge in E corresponds to a pair of customers possibly assigning to a common facility. Thus the number of the edges in E is at most $81r^2|F|$ by Lemma 3.6. Thus we can construct $D(V, E)$ in $O(|F| + |C| + 81r^2|F|)$ time in toal.

Similar to Section 2 we have reduced the r-gathering problem to the min-max path problem, and have the following theorem.

Theorem 3.7 When all C and F are on the line one can solve the r-gathering problem in $O(n + r^2 m \log^* n)$ time, where $n = |C|$ and $m = |F|$.

4. Conclusion

In this paper we have presented an algorithm to solve the r-gather clustering problem when all C are on the line. The running time of the algorithm is $O(rn \log^* n)$, where $n = |C|$. We also presented an algorithm to solve the r-gathering problem, which runs in time $O(n + r^2 m \log^* n)$, where $n = |C|$ and $m = |F|$.

Can we design a linear time algorithm for the r-gathering problem when all C and F are on the line?

References

[1] G. Aggarwal, T. Feder, K. Kenthapadi, S. Khuller, R. Panigrahy, D. Thomas and A. Zhu, Achieving anonymity via clustering, Transactions on Algorithms, 6, Article No.49 (2010).
[2] P. Agarwal and M. Sharir, Efficient Algorithms for Geometric Optimization, Computing Surveys, 30, pp.412-458 (1998).
[3] T. Akagi and S. Nakano, On r-gatherings on the Line, Proc.of FAW 2015, LNCS 9130, pp.25-32 (2015).
[4] A Armon, On min-max r-gatherings, Theoretical Computer Science, 412, pp.573-582 (2011).
[5] Z. Drezner, Facility Location: A Survey of Applications and Methods, Springer (1995).
[6] Z. Drezner and H.W. Hamacher, Facility Location: Applications and Theory, Springer (2004).
[7] G. Frederickson and D. Johnson, Generalized Selection and Ranking: Sorted Matrices, SIAM Journal on Computing, 13, pp.14-30 (1984).
[9] H. Gabow and R. Tarjan, Algorithms for Two Bottleneck Optimization Problems, J. of Algorithms, 9, pp.411-417 (1988).
[10] Y. Han and S. Nakano, On r-Gatherings on the Line, Proc. of FCS2016, pp.99-104 (2016).