IEICE TRANS. INF. & SYST., VOL.E100–D, NO.3 MARCH 2017

PAPER Special Section on Foundations of Computer Science —New Trends in Theoretical Computer Science—

On r-Gatherings on the Line[*]

Toshihiro AKAGI[†a)], ***Nonmember and*** **Shin-ichi NAKANO**[†], ***Member***

SUMMARY In this paper we study a recently proposed variant of the facility location problem, called the r-gathering problem. Given an integer r, a set C of customers, a set F of facilities, and a connecting cost $co(c, f)$ for each pair of $c \in C$ and $f \in F$, an *r-gathering* of customers C to facilities F is an assignment A of C to *open facilities* $F' \subseteq F$ such that at least r customers are assigned to each open facility. We give an algorithm to find an *r-gathering* with the minimum cost, where the cost is $\max_{c \in C}\{co(c, A(c))\}$, when all C and F are on the real line.
key words: *algorithm, facility location*

1. Introduction

The facility location problem and many of its variants are studied [5], [6]. In the basic facility location problem we are given (1) a set C of customers, (2) a set F of facilities, (3) an opening cost $op(f)$ for each $f \in F$, and (4) a connecting cost $co(c, f)$ for each $c \in C$ and $f \in F$, then we open a subset $F' \subseteq F$ of facilities and find an assignment A of C to F' such that a designated cost is minimized.

In this paper we study a recently proposed variant of the facility location problem, called the r-gathering problem [4], [9], [10]. An *r-gathering* of customers C to facilities F is an assignment A of C to *open facilities* $F' \subseteq F$ such that at least r customers are assigned to each open facility. This means each open facility has enough number of customers. We assume $|C| \geq r$ holds. Then we define the cost of (the max version of) a gathering as $\max_{c \in C}\{co(c, A(c))\}$. (We assume $op(f) = 0$ for each $f \in F$ in the paper.) The min-max version of the *r-gathering problem* finds an r-gathering having the minimum cost. For the min-sum version see the brief survey in [4].

Assume that F is a set of locations for emergency shelters, and $co(c, f)$ 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 at least r 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 assumption $P \neq NP$ the problem cannot be approximated within a factor of less than 3 for any $r \geq 3$. In this paper we give an $O((n + m) \log(n + m))$ time algorithm, where $n = |C|$ and $m = |F|$, to solve the r-gathering problem when all C and F are on the real line.

The remainder of this paper is organized as follows. Section 2 gives an algorithm to solve a decision version of the r-gathering problem. Section 3 contains our main algorithm for the r-gathering problem. Sections 4, 5 and 6 present more algorithms to solve three similar problems. Finally Sect. 7 is a conclusion.

2. (k, r)-Gathering on the Line

In this section we give a linear time algorithm to solve a decision version of the r-gathering problem [3].

Given customers $C = \{c_1, c_2, \ldots, c_n\}$ and facilities $F = \{f_1, f_2, \ldots, f_m\}$ on the real line (we assume they are distinct coordinates and appear in those order from left to right, respectively) and four numbers i, j, k and r, then problem $P(j, i)$ finds an assignment A of customers $C_i = \{c_1, c_2, \ldots, c_i\}$ to open facilities $F'_j \subseteq F_j = \{f_1, f_2, \ldots, f_j\}$ such that (1) at least r customers are assigned to each open facility, (2) $co(c, A(c)) \leq k$ for each $c \in C_i$ and (3) $f_j \in F'_j$. Here $co(c, f)$ is the distance between $c \in C$ and $f \in F$, (2) implies each customer is assigned to a near facility, and (3) implies the rightmost facility is forced to open. We first remove from F each $f \in F$ having at most $r - 1$ customers in interval $[f - k, f + k]$ (since such f never open), then check if there is a customer $c \in C$ having no $f \in F$ within distance k (since if there is then there is no r-gathering). We can do these in $O(n + m)$ time.

An assignment A of C_i to F_j is called *monotone* if, for any pair $c_{i'}, c_i$ of customers with $i' < i$, $A(c_{i'}) \leq A(c_i)$ holds. In a monotone assignment the interval induced by the assigned customers to a facility never intersects other interval induced by the assigned customers to another facility. We can observe that if $P(j, i)$ has a solution then $P(j, i)$ also has a monotone solution. Also we can observe that if $P(j, i)$ has a solution and $co(c_{i+1}, f_j) \leq k$ then $P(j, i + 1)$ also has a solution. If $P(j, i)$ has a solution for some i then let $s(f_j)$ be the minimum i such that $P(j, i)$ has a solution. Note that (3) $f_j \in F'_j$ implies $c_{s(f_j)}$ is located in interval $[f_j - k, f_j + k]$. We define $P(j)$ to be the problem to find such $s(f_j)$ and a corresponding assignment. If $P(j, i)$ has no solution for every i then we say $P(j)$ has no solution, otherwise we say $P(j)$ has a solution.

Manuscript received March 24, 2016.
Manuscript revised July 24, 2016.
Manuscript publicized December 21, 2016.
[†]The authors are with Gunma University, Kiryu-shi, 376–8515 Japan.
[*]The preliminary version of the paper was presented at 9th International Frontiers of Algorithmics Workshop.
a) E-mail: akagi@nakano-lab.cs.gunma-u.ac.jp
DOI: 10.1587/transinf.2016FCP0007

Lemma 1: For any pair $f_{j'}$ and f_j in F with (1) $j' < j$ and (2) both $P(j')$ and $P(j)$ have solutions, $s(f_{j'}) \leq s(f_j)$ holds.

Proof : Assume otherwise. Then $s(f_{j'}) > s(f_j)$ holds. Modify the assignment corresponding to $s(f_j)$ as follows. Reassign the customers assigned to the facilities between $f_{j'}$ and f_j (including f_j) to $f_{j'}$ then close the facilities between $f_{j'}$ and f_j. The resulting assignment also satisfy the conditions $co(c, A(c)) \leq k$ for each $c \in C$ and each open facility is assigned at least r customers, so it is a solution of $P(j')$ and now $s(f_{j'}) = s(f_j)$. A contradiction. □

Assume that $P(j)$ has a solution and $c_1 < f_j - k$. Then the corresponding solution has one or more facilities except for f_j. Choose the solution of $P(j)$ having the minimum second rightmost open facility, say $f_{j'}$. We say $f_{j'}$ is the *mate* of f_j and write $mate(f_j) = f_{j'}$. Then note that $P(j')$ has a solution. The mate $f_{j'}$ of f_j is one of the following three types.

Type 1: $f_{j'} + k < f_j - k$, the interval $(f_{j'} + k, f_j - k)$ has no customer and the interval $[f_j - k, f_j + k]$ has at least r customers.

Type 2: $f_{j'} + k \geq f_j - k$, $c_{s(f_{j'})} \geq f_j - k$ and the interval $(c_{s(f_{j'})}, f_j + k]$ has at least r customers.

Type 3: $f_{j'} + k \geq f_j - k$, $c_{s(f_{j'})} < f_j - k$ and the interval $[f_j - k, f_j + k]$ has at least r customers.

For each f_j by checking the three conditions above for every possible mate $f_{j'}$ one can design $O(n + m^2)$ time algorithm based on dynamic programming approach. However we can omit the most part of the checks by the following lemma.

Lemma 2: (a) Assume $P(j)$ has a solution. If $P(j + 1)$ also has a solution then $mate(f_j) \leq mate(f_{j+1})$ holds. (b) For $f_j \in F$, if there is $f_{j'}$ such that (i) $P(j')$ has a solution, (ii) $f_{j'} + k \geq f_j - k$ and (iii) $j' < j$ then let f_{min} be $f_{j'}$ with the minimum j'. If $P(j)$ has no solution with the second rightmost open facility f_{min}, then (b1) any $f_{j''}$ satisfying $f_{min} < f_{j''} < f_j$ is not the mate of f_j, and $P(j)$ has no solution, and (b2) $f_{min} \leq mate(f_{j+1})$ holds if $mate(f_{j+1})$ exists.

Proof : (a) Assume otherwise. We have two cases. If $mate(f_{j+1}) < f_j - k$ holds then $mate(f_{j+1})$ is also the mate of f_j, a contradiction. If $mate(f_{j+1}) \geq f_j - k$ holds then by Lemma 1 $mate(f_{j+1})$ is also the mate of f_j, a contradiction. (b1) Immediate from Lemma 1. (b2) Assume otherwise. If $mate(f_{j+1}) + k < f_j - k$ holds then $mate(f_{j+1})$ is also the mate of f_j, a contradiction. If $mate(f_{j+1}) \geq f_j - k$ holds then f_{min} is $mate(f_{j+1})$ not $mate(f_j)$, a contradiction. □

Lemma 2 means after searching for the mate of f_j upto some $f_{j'}$ the next search for the mate of f_{j+1} can start at the $f_{j'}$. Based on the lemma above we can design algorithm **find** (k, r)**-gathering**.

For the preprocessing we compute, for each $f_j \in F$, (1) the index of the first customer in interval $(f_j + k, \infty)$, (2) the index of the first customer in interval $[f_j - k, \infty)$ and (3) the index of the r-th customer in interval $[f_j - k, \infty)$. Also we store the index $s(f_j)$ for each $f_j \in F$. Those need $O(n + m)$ time. After the preprocessing the algorithm runs in $O(m)$ time since $j' \leq j$ always holds the most inner part to compute $s(f_j)$ executes at most $2m$ times.

We have the following lemma.

Lemma 3: One can solve the (k, r)-gathering problem in $O(n + m)$ time.

Algorithm 1 find (k, r)-gathering (C, F, k, r)

```
// Remove never open f //
remove from F each f ∈ F having at most r − 1 customers in its interval
[f − k, f + k]
let F = {f_1, f_2, ..., f_m}
if there is no facility in F then
    return NO
end if
// Check NO solution Case //
if there is a customer which has no facility within distance k then
    return NO
end if
// One open facility Case //
j = 1
while interval [f_j − k, f_j + k] has both c_1 and c_r do
    set s(f_j) to be the index of the r-th customer c_r
    j = j + 1
end while
// Two or more open facilities Case //
j' = 1
while j ≤ m do
    while j' < j and (interval (f_j' + k, f_j − k) has at least one customer
    or P(j') has no solution) do
        j' = j' + 1
    end while
    if j' < j then
        // interval (f_j' + k, f_j − k) has no customer and P(j') has a solution
        //
        if f_j' + k < f_j − k and interval (f_j' + k, f_j − k) has no customer and
        interval [f_j − k, f_j + k] has at least r customers then
            set s(f_j) to be the index of the r-th customer in the interval
            [f_j − k, f_j + k] { // Type 1 // }
        else if f_j' + k ≥ f_j − k and c_s(f_j') ≥ f_j − k and interval (c_s(f_j'), f_j + k]
        has at least r customers then
            set s(f_j) to be the index of the r-th customer in the interval
            (c_s(f_j'), f_j + k] { // Type 2 // }
        else if f_j' + k ≥ f_j − k and c_s(f_j') < f_j − k and interval [f_j − k, f_j + k]
        has at least r customers then
            set s(f_j) to be the index of the r-th customer in the interval
            [f_j − k, f_j + k] { // Type 3 // }
        end if
        // Otherwise P(j) has no solution //
    end if
    j = j + 1
end while
if some f_j with defined s(f_j) has c_n within distance k then
    return YES
else
    return NO
end if
```

3. *r*-Gathering on the Line

In this section we give an $O((n + m)\log(n + m))$ time algorithm to solve the r-gathering problem when all C and F are on the real line.

Our strategy is as follows. First we can observe that the minimum cost k^* of a solution of an r-gathering problem is $co(c, f)$ with some $c \in C$ and some $f \in F$. Since the number of distinct $co(c, f)$ is at most nm, sorting them needs $O(nm \log(nm))$ time. Then find the smallest k such that the (k, r)-gathering problem has a solution by binary search, using the linear-time algorithm in the preceding section $\log(nm)$ times. Those part needs $O((n + m) \log(nm))$ time. Thus the total running time is $O(nm \log(nm))$.

However by using the sorted matrix searching method [7] (See the good survey in [2]) we can improve the running time to $O((n + m) \log(n + m))$. Similar technique is also used in [8], [11] for a fitting problem. Now we explain the detail.

First let M_C be the matrix in which each element is $m_{i,j} = c_i - f_j$. Then $m_{i,j} \geq m_{i,j+1}$ and $m_{i,j} \leq m_{i+1,j}$ always holds, so the elements in the rows and columns are sorted, respectively. Similarly let M_F be the matrix in which each element is $m'_{i,j} = f_j - c_i$. The minimum cost k^* of an optimal solution of an r-gathering problem is some positive element in those two matrices. We can find the smallest k in M_C for which the (k, r)-gathering problem has a solution, as follows.

Let n' be the smallest integer which is (1) a power of 2 and (2) larger than or equal to $\max\{n, m\}$. Then we append the largest element $m_{n',1}$ to M_C as the elements in the lowest rows and the leftmost columns so that the resulting matrix has exactly n' rows and n' columns. Note that the elements in the rows and columns are still sorted respectively. Let M_C be the resulting matrix. Our algorithm consists of stages $s = 1, 2, \ldots, \log n'$, and maintains a set L_s of submatrices of M_C possibly containing k^*. Hypothetically first we set $L_0 = \{M_C\}$. Assume we are now starting stage s.

For each submatrix M in L_{s-1} we partite M into the four submatrices with $n'/2^s$ rows and $n'/2^s$ columns and put them into L_s.

Let k_{min} be the median of the upper-right corner elements of the submatrices in L_s. Then for $k = k_{min}$ we solve the (k, r)-gathering problem. We have the following two cases.

If the (k, r)-gathering problem has a solution then we remove from L_s each submatrix with the upper-right corner element (the smallest element) greater than k_{min}. Since $k_{min} \leq k^*$ holds each removed submatrix has no chance to contain k^*. Also if L_s has several submatrices with the upper-right corner element equal to k_{min} then we remove them except one from L_s. Thus we can remove $|L_s|/2$ submatrices from L_s.

Otherwise if the (k, r)-gathering problem has no solution then we remove from L_s each submatrix with the lower left corner element (the largest element) smaller than k_{min}. Since $k_{min} < k^*$ holds each removed submatrix has no chance to contain k^*. Now we can observe that, for each "chain" of submatrices, which is the sequence of submatrices in L_s with lower-left to upper-right diagonal on the same line, the number of submatrices (1) having the upper right corner element smaller than k_{min} (2) but remaining in L_i is at most one (since the elements on "the common diagonal line" are sorted). Thus, if $|L_s|/2 > D_s$, where $D_s = 2^{s+1}$ is the number of chains plus one, then we can remove at least $|L_s|/2 - D_s$ submatrices from L_s.

Similarly let k_{max} be the median of the lower-left corner elements of the submatrices in L_s, and for $k = k_{max}$ we solve the (k, r)-gathering problem and similarly remove some submatrices from L_s. This ends stage s.

Now after stage $\log n'$ each matrix in $L_{\log n'}$ has just one element, then we can find the k^* using a binary search with the linear-time decision algorithm.

We can prove that at the end of stage s the number of submatrices in L_s is at most $2D_s$, as follows.

First L_0 has 1 submatrix, which is less than $2D_0 = 4$. By induction assume that L_{s-1} has $2D_{s-1} = 2 \times 2^s$ submatrices.

At stage s we first partite each submatrix in L_{s-1} into four submatrices then put them into L_s. Now the number of submatrices in L_s is $4 \times 2D_{s-1} = 4D_s$. We have four cases.

If the (k, r)-gathering problem has a solution for $k = k_{min}$ then we can remove at least a half of the submatrices from L_s, and so the number of the remaining submatrices in L_s is at most $2D_s$, as desired.

If the (k, r)-gathering problem has no solution for $k = k_{max}$ then we can remove at least a half of the submatrices from L_s, and so the number of the remaining submatrices in L_s is at most $2D_s$, as desired.

Otherwise if $|L_s|/2 \leq D_s$ then the number of the submatrices in L_s (even before the removal) is at most $2D_s$, as desired.

Otherwise (1) after the check $k = k_{min}$ we can remove at least $|L_s|/2 - D_s$ submatrices (consisting of too small elements) from L_s, and (2) after the check for $k = k_{max}$ we can remove at least $|L_s|/2 - D_s$ submatrices (consisting of too large elements) from L_s, so the number of the remaining submatrices in L_s is at most $|L_s| - 2(|L_s/2| - D_s) = 2D_s$, as desired.

Thus at the end of stage s the number of submatrices in L_s is always at most $2D_s$.

Now we consider the running time. We implicitly treat each submatrix as the index of the upper right element in M_C and the number of lows. Except for the calls of the linear-time decision algorithm for the (k, r)-gathering problem, we need $O(|L_{s-1}|) = O(D_{s-1})$ time for each stage $s = 1, 2, \ldots, \log n'$, and $D_0 + D_1 + \cdots + D_{\log n'-1} = 2 + 4 + \cdots + 2^{\log n'} < 2 \times 2^{\log n'} = 2n'$ holds, so this part needs $O(n')$ time in total. (Here we use the linear time algorithm to find the median.)

Since each stage calls the linear-time decision algorithm twice this part needs $O(n' \log n')$ time in total.

After stage $s = \log n'$ each matrix has just one element, then we can find the k^* among the $|L_{\log n'}| \leq 2D_{\log n'} = 4n'$ elements using a binary search with the linear-time decision algorithm at most $\log 4n'$ times. This part needs $O(n' \log n')$ time.

Then we similarly find the smallest k in M_F for which the (k, r)-gathering problem has a solution. Finally we output the smaller one among the two.

Thus the total running time is $O((n + m)\log(n + m))$.

Theorem 1: One can solve the r-gathering problem in $O((n + m)\log(n + m))$ time when all C and F are on the real line.

4. *r*-Gather Clustering

In this section we give an algorithm to solve a similar problem by modifying the algorithm in Sect. 3.

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 [1] finds an 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 [1].

When all C are on the real line, in any solution of any r-gather-clustering problem, we can assume that the center of each disk is at the midpoint of some pair of points, and the radius of an optimal r-gather-clustering is the half of the distance between some pair of points in C.

Given C and two numbers k and r the decision version of the r-gather-clustering problem find an r-gather-clustering with maximum radius within k. We can assume that in any solution of the problem the center of each disk is at $c - k$ for some $c \in C$. Thus, by introducing the set of all such points as F, we can solve the decision version of the r-gather-clustering problem as the (k, r)-gathering problem. Using the algorithm in Sect. 2 we can solve the problem in $O(n)$ time.

Now we explain our algorithm to solve the r-gather-clustering problem. First sort C in $O(n \log n)$ time. Let $c_1, c_2, \ldots, c_n$ be the resulting non-decreasing sequences and let M be the matrix in which each element is $m_{i,j} = (c_i - c_j)/2$. Note that the optimal radius is in M and this time M has n rows and n columns. Now $m_{i,j} \geq m_{i,j+1}$ and $m_{i,j} \leq m_{i+1,j}$ holds, so the elements in the rows and columns are sorted, respectively. Then as in Sect. 3 we can find the optimal radius by the sorted matrix searching method. The algorithm calls the decision algorithm $O(\log n)$ times and the decision algorithm runs in $O(n)$ time, and in the stages the algorithm needs $O(n)$ time in total except for the calls. Finally we needs $O(n \log n)$ time for the final binary search. Thus the total running time is $O(n \log n)$.

Theorem 2: One can solve the r-gather-clustering problem in $O(n \log n)$ time when all points C are on the real line.

5. Outlier

In this section we consider a generalization of the r-gathering problem where at most h customers are allowed to be not assigned.

An *r-gathering with h-outliers* of customers C to facilities F is an assignment A of $C \backslash C'$ to open facilities $F' \subseteq F$ such that at least r customers are assigned to each open facility and $|C'| \leq h$. The *r-gathering with h-outliers problem* finds an r-gathering with h-outliers having the minimum cost.

Given customers $C = \{c_1, c_2, \ldots, c_n\}$ and facilities $F = \{f_1, f_2, \ldots, f_m\}$ on the real line and six numbers i, j, k, r, h and h', problem $P(j, i, h')$ finds an r-gathering with h-outliers of $C_i = \{c_1, c_2, \ldots, c_i\} \backslash C'$ to $F'_j \subseteq F_j = \{f_1, f_2, \ldots, f_j\}$ such that (1) at least r customers are assigned to each open facility, (2) $co(c, A(c)) \leq k$ for each $c \in C \backslash C'$, (3) $f_j \in F'_j$ and (4) $|C'| = h'$. For designated j and h' if $P(j, i, h')$ has a solution for some i then let $s(f_j, h')$ be the minimum i such that $P(j, i, h')$ has a solution. We define $P(j, h')$ to be the problem to find such $s(f_j, h')$ and a corresponding assignment.

By dynamic programming approach one can compute $P(j, h')$ for each $j = 1, 2, \ldots, m$ and $h' = 1, 2, \ldots, h$ in $O(n + h^2 m)$ time in total. Then one can decide whether an r-gathering with h-outliers problem has a solution with cost k.

Lemma 4: One can decide whether an r-gathering with h-outliers problem has a solution with cost k in $O(n + h^2 m)$ time.

The minimum cost k^* of a solution of an r-gathering with h-outliers problem is again $co(c, f)$ for some $c \in C$ and some $f \in F$. By the sorted matrix searching method using the $O(n + h^2 m)$ time decision algorithm above one can solve the problem with outliers in $O((n + h^2 m)\log(n + m))$ time.

Theorem 3: One can solve the r-gathering with h-outliers problem in $O((n + h^2 m)\log(n + m))$ time when all C and F are on the real line.

6. New Branch Location

In this section we consider a generalization of the r-gathering problem where some facilities $F^o \subseteq F$ are already forced to open and we wish to find an r-gathering of C to $F' \supseteq F^o$ with the minimum cost. We call this problem the new branch location problem. Note that if $F^o = \phi$ then this is the r-gathering problem.

We solve this problem by dynamic programming, in which we solve the following subproblems systematically. Let $C_i = \{c_1, c_2, \ldots, c_i\} \subseteq C$, $F_j = \{f_1, f_2, \ldots, f_j\} \subseteq F$ and $F^o_j = F_j \cap F^o$. Given C, F, and F^o and four numbers i, j, k and r, then problem $P^o(j, i)$ finds an r-gathering A of C_i to F'_j such that (1) $F^o_j \subseteq F'_j \subseteq F_j$, (2) at least r customers are assigned to each (open) facility, (3) $co(c, A(c)) \leq k$ for each $c \in C_i$, (4) $f_j \in F'_j$. Similar to Sect. 2 we remove from F each $f \in F$ having at most $r - 1$ customers in interval $[f - k, f + k]$, then check if there is a customer $c \in C$ having no $f \in F$ within distance k. If some $f \in F^o$ is removed then there is no solution. We can check these in $O(n + m)$ time.

We need some definitions. If $P^o(j, i)$ has a solution for some i then let $s^o(f_j)$ be the minimum i such that $P^o(j,i)$ has a solution. We define $P^o(j)$ to be the problem to find such $s^o(f_j)$ and a corresponding assignment. If $P^o(j,i)$ has no solution for every i then we say $P^o(j)$ has no solution.

We show that following lemmas, similar to Lemma 1 and Lemma 2 for the ordinary r-gathering problem, are hold.

Lemma 5: For any pair $f_{j'}, f_j \in F$ with (1) $j' < j$, and (2) both $P^o(j')$ and $P^o(j)$ have solutions, $s^o(f_{j'}) \leq s^o(f_j)$ holds.

Proof : We consider the following two cases.
Case 1: There is no facility $f_{j''} \in F^o$ with $j' < j'' < j$.

Assume for a contradiction that $s^o(f_{j'}) > s^o(f_j)$ holds. Modify the assignment corresponding to $s^o(f_j)$ as follows. Reassign the customers assigned to the facilities between $f_{j'}$ and f_j (including f_j) to $f_{j'}$ then close the facilities. The resulting assignment is a solution of $P^o(j')$ and now $s^o(f_{j'}) = s^o(f_j)$. A contradiction.
Case 2: Otherwise. (There is some $f_{j''} \in F^o$ with $j' < j'' < j$.)

Assume for a contradiction that $s^o(f_{j'}) \geq s^o(f_j)$ holds. By the hypothesis, there is some $f_{j''} \in F^o$ such that $j' < j'' < j$.

Modify the assignment corresponding to $s^o(f_j)$ as follows. Reassign the customers assigned to the facilities between $f_{j'}$ and f_j (including f_j) to $f_{j'}$, then close the facilities. The resulting assignment is a solution of $P^o(j')$ and now $s^o(f_{j'}) = s^o(f_j)$. A contradiction. □

Assume that $P^o(j)$ has a solution and $c_1 < f_j - k$. Then the assignment corresponding to the solution of $P^o(j)$ has one or more open facilities except for f_j. Choose the solution of $P^o(j)$ having the minimum second rightmost open facility, say $f_{j'}$. We say $f_{j'}$ is the *mate* of f_j, and write $mate^o(f_j) = f_{j'}$. Then note that $P^o(j')$ has a solution, and interval $(f_{j'}, f_j)$ has no facility in F^o. The mate $f_{j'}$ of f_j is one of the following three types.
type 1: $f_{j'} + k < f_j - k$, the interval $(f_{j'} + k, f_j - k)$ has no customer and the interval $[f_j - k, f_j + k]$ has at least r customers.
type 2: $f_{j'} + k \geq f_j - k$, $c_{s^o(f_{j'})} \geq f_j - k$ and the interval $(c_{s^o(f_{j'})}, f_j - k]$ has at least r customers.
type 3: $f_{j'} + k \geq f_j - k$, $c_{s^o(f_{j'})} < f_j - k$ and the interval $[f_j - k, f_j - k]$ has at least r customers.

Similar to the algorithm in Sect. 2 we have the following lemma.

Lemma 6: (a) Assume $P^o(j)$ has a solution. If $P^o(j+1)$ also has a solution then $mate^o(f_j) \leq mate^o(f_{j+1})$ holds. (b) For $f_j \in F$, if there is $f_{j'}$ such that (i) $P^o(j')$ has a solution, (ii) $f_{j'} + k \geq f_j - k$ and (iii) $j' < j$ then let f_{min} be $f_{j'}$ with the minimum j'. If $P^o(j)$ has no solution with the second rightmost open facility f_{min}, then (b1) any $f_{j''}$ satisfying $f_{min} < f_{j''} < f_j$ is not the mate of f_j, and $P^o(j)$ has no solution, and (b2) $f_{min} \leq mate^o(f_{j+1})$ holds if $mate^o(f_{j+1})$ exists.

Proof : (a) Assume otherwise. We have two cases. If $mate^o(f_{j+1}) + k < f_j - k$ holds then $mate^o(f_{j+1})$ is also the mate of f_j, a contradiction. If $mate^o(f_{j+1}) + k \geq f_j - k$ holds then by Lemma 5 $mate^o(f_{j+1})$ is also the mate of f_j, a contradiction. (b1) Immediate from Lemma 5. (b2) Assume otherwise. If $mate^o(f_{j+1}) + k < f_j - k$ holds then $mate^o(f_{j+1})$ is also the mate of f_j, a contradiction. If $mate^o(f_{j+1}) + k \geq f_j - k$ holds then f_{min} is $mate^o(f_{j+1})$ not $mate^o(f_j)$, a contradiction. □

Lemma 6 means after searching for the mate of f_j upto

Algorithm 2 find new-branch (C, F, F^o, k, r)

// Remove never open f //
remove from F each $f \in F$ having at most $r-1$ customers in its interval $[f-k, f+k]$
let $F = \{f_1, f_2, \ldots, f_m\}$
if there is no facility in F **then**
  return NO
end if
// Check NO solution Case //
if some $f \in F^o$ is removed **then**
  return NO
else
  let $F^o = \{f^o_1, f^o_2, \ldots, f^o_{m^o}\} \subseteq F$
end if
if there is a customer which has no faclity within distance k **then**
  return NO
end if0
// One open facility Case //
$j = 1$
while interval $[f_j - k, f_j + k]$ has both c_1 and c_r **and** interval $(-\infty, f_j)$ has no facility in F^o **do**
  set $s^o(f_j)$ to be the index of the r-th customer c_r
  $j = j + 1$
end while
// Two or more open facilities Case //
$j' = 1$
while $j \leq m$ **do**
  while $j' < j$ **and** (interval $(f_{j'} + k, f_j - k)$ has at least one customer **or** $P^o(j')$ has no solution **or** interval $(f_{j'}, f_j)$ has at least one facility in F^o) **do**
    $j' = j' + 1$
  end while
  if $j' < j$ **then**
    // interval $(f_{j'} + k, f_j - k)$ has no customer and $P^o(j')$ has a solution and interval $(f_{j'}, f_j)$ has no facility in F^o //
    if $f_{j'} + k < f_j - k$ **and** interval $(f_{j'} + k, f_j - k)$ has no customer **and** interval $[f_j - k, f_j + k]$ has at least r customers **then**
      set $s^o(f_j)$ to be the index of the r-th customer in the interval $[f_j - k, f_j + k]$ { // Type 1 // }
    else if $f_{j'} + k \geq f_j - k$ **and** $c_{s^o(f_{j'})} \geq f_j - k$ **and** interval $(c_{s^o(f_{j'})}, f_j + k]$ has at least r customers **then**
      set $s^o(f_j)$ to be the index of the r-th customer in the interval $(c_{s^o(f_{j'})}, f_j + k]$ { // Type 2 // }
    else if $f_{j'} + k \geq f_j - k$ **and** $c_{s^o(f_{j'})} < f_j - k$ **and** interval $[f_j - k, f_j + k]$ has at least r customers **then**
      set $s^o(f_j)$ to be the index of the r-th customer in the interval $[f_j - k, f_j + k]$ { // Type 3 // }
    end if
    // Otherwise $P^o(j)$ has no solution //
  end if
  $j = j + 1$
end while
if some f_j with defined $s^o(f_j)$ has c_n within distance k **then**
  return YES
else
  return NO
end if

some $f_{j'}$ the next search for the mate of f_{j+1} can start at the $f_{j'}$. Based on the lemma above we can design an algorithm. See Appendix for interested readers. The main difference is the condition of the second while, where "or interval $(f_{j'}, f_j)$ has at least one facility in F^o" is appended.

As a preprocessing we also compute, for $f_j \in F$, the index of the facility $f_{j''} \in F^o$ with maximum $j'' < j$, if such $f_{j''}$ exists. It needs $O(m)$ time. Then we decide if interval $(f_{j'}, f_j)$ has a facility in F^o or not. If the indexes computed above for $f_{j'}$ and f_j are identical then interval $(f_{j'}, f_j)$ has no facility in F^o.

We have the following lemma.

Lemma 7: One can solve the new branch problem in $O(n+m)$ time.

The minimum cost k^* of a solution of a new branch problem is $co(c, f)$ with some $c \in C$ and some $f \in F$. By using the sorted matrix searching method in Sect. 3 we can find such k^* in at most $O(\log(n+m))$ rounds, and each round needs $O(n + m)$ time to solve the decision version of the problem.

We have the following theorem.

Theorem 4: One can solve the new branch problem in $O((n + m)\log(n + m))$ time when all C and F are on the real line.

7. Conclusion

In this paper we have presented an algorithm to solve the r-gathering problem when all C and F are on the real line. The running time of the algorithm is $O((n + m)\log(n + m))$. We also presented three more algorithms to solve three similar problems.

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

The preliminary version of the paper appeared in Proc. of FAW 2015.

References

[1] G. Aggarwal, R. Panigrahy, T. Feder, D. Thomas, K. Kenthapadi, S. Khuller, and A. Zhu, "Achieving anonymity via clustering," ACM Transactions on Algorithms, vol.6, no.3, Article 49, 2010.
[2] P.K. Agarwal and M. Sharir, "Efficient algorithms for geometric optimization," ACM Computing Surveys, vol.30, no.4, pp.412–458, 1998.
[3] T. Akagi and S. Nakano, "On (k, r)-gatherings on a road," Proc. of Forum on Information Technology, FIT 2013, RA-001, 2013.
[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.H. Frederickson and D.B. Johnson, "Generalized selection and ranking: sorted matrices," SIAM J. Comput., vol.13, no.1, pp.14–30, 1984.
[8] H. Fournier and A. Vigneron, "Fitting a step function to a point set," Proc. of ESA 2008, LNCS5193, pp.442–453, 2008.
[9] S. Guha, A. Meyerson, and K. Munagala, "Hierarchical placement and network design problems," Proc. of FOCS 2000, pp.603–612, 2000.
[10] D.R. Karger and M. Minkoff, "Building steiner trees with incomplete global knowledge," Proc. of FOCS 2000, pp.613–623, 2000.
[11] J.-Y. Liu, "A randomized algorithm for weighted approximation of points by a step function," Proc. of COCOA 2010, LNCS6508, pp.300–308, 2010.

Toshihiro Akagi received B.E. and M.E. from Gunma University in 2013 and 2015, respectively. He is currently a doctoral student in the Graduate School of Engineering, Gunma University. His research interests include combinatorial algorithms. He is a member of IPSJ.

Shin-ich 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 Toholu 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, JSIAM and ACM.