Online Facility Assignment[0]

Abu Reyan Ahmed[1], Md. Saidur Rahman[2] and Stephen Kobourov[1]

[1] *Dept. of Computer Science, University of Arizona,*
{abureyanahmed@email,kobourov@cs}.arizona.edu
[2] *Dept. of Computer Science and Engineering, Bangladesh University of Engineering and Technology, saidurrahman@cse.buet.ac.bd*

Abstract

We consider the online facility assignment problem, with a set of facilities F of equal capacity l in metric space and customers arriving one by one in an online manner. We must assign customer c_i to facility f_j before the next customer c_{i+1} arrives. The cost of this assignment is the distance between c_i and f_j. The total number of customers is at most $|F|l$ and each customer must be assigned to a facility. The objective is to minimize the sum of all assignment costs. We first consider the case where facilities are placed on a line so that the distance between adjacent facilities is the same and customers appear anywhere on the line. We describe a greedy algorithm with competitive ratio $4|F|$ and another one with competitive ratio $|F|$. We also consider a slightly more general situation where different facilities may have different capacities. Finally, we study a variant of the online facility assignment problem in which the facilities are placed on the vertices of a graph and present two algorithms for that setting.

Keywords: online facility assignment, greedy algorithms

1. Introduction

Let $F = \{f_1, f_2, \cdots, f_{|F|}\}$ be a set of facilities, each with capacity l. We first consider the case when facilities are placed on line L, such that the distance between every pair of adjacent facilities is d, where d is a constant. An input sequence $I = \{c_1, c_2, \cdots, c_n\}$ is a set of n customers who arrive one at a time in an online manner, with c_i corresponding to the location of customer i on the line L. The distance between a customer c_i and a facility f_j is the Euclidean

[0] An earlier version of this paper appears in WALCOM'18 and this is an extended version for the special issue of TCS. Here we provide expanded background about the problem (Related Work) and complete proofs for several claims (e.g., Theorem 1 and Theorem 3). We add new results for a more general setting where different facilities may have different capacities in Section 3.1 and 3.4. We also discuss a capacity-sensitive variant of the greedy algorithm in Section 3.2.

[1] Work on this project was funded in part by NSF grant CCF-AF 1712119.

distance between c_i and f_j. We later consider the case in which the facilities are located on the vertices of a graph $G = (V, E)$ and customers appear on the vertices of G. In that case, the distance between a customer c_i and a facility f_j is equal to the number of edges in the shortest path between c_i and f_j.

Any algorithm for this problem must assign a customer c_i to a facility f_j before the next customer c_{i+1} arrives, where the cost of that assignment is the distance between c_i and f_j. The total number of customers is at most $|F|l$ as every facility can serve at most l customers and each customer must be assigned to a facility. The objective is to minimize the total cost of all assignments. We call this problem the *online facility assignment problem*. This problem arises naturally in different practical applications, such handling online orders for a restaurant with multiple locations, and handling network packets in network with multiple routers.

1.1. Related Work

In the offline version of this problem, the locations of all customers and all facilities are known. The assignment cost of a customer is the distance between the customer and the assigned facility. The goal is to assign all customers to facilities, while maintaining the capacity constraints of the facilities, with the objective of minimizing the total assignment cost. This offline problem can be modeled using the well-studied *transportation problem* [1], which asks for the optimal way to transport a single type of commodity from a set of sources to a set of destinations, subject to facility capacity constraints. The transportation problem and cycle-canceling flows are classic optimization problems usually attributed to work in the 1940's such that of Hitchcock in 1941 [2], but earlier work dates back to 1931 [3]. The problem was also studied by Dantzig [4], Charnes and Cooper [5], Ford and Fulkerson [6], among others, using variations of a linear programming formulation.

The transportation problem can be considered as type of an offline *facility location problem*. In the general setting the input is a set of customers with fixed locations and the goal is to locate one or more facilities that can serve the customers, while minimizing the total cost, which is usually a function of the distances in customer-facility pairs. This problem has many variants, e.g., the k-median problem [7], when there are no costs for opening facilities, but k is an upper bound on the number of facilities that can be opened. The Fermat-Weber problem is considered the first facility location problem, studied as early as the 17th century [8]. Most facility location problems are NP-hard and Shmoys et al. [9] provided the first constant approximation algorithm for the uncapacitated facility location problem. Charikar [10] improved the approximation ratio from 3.16 to 1.728, and Sviridenko [11] further improved it to 1.67. These approximation algorithms can be divided into three categories: rounding algorithms that rely on linear programming, primal-dual, and local search algorithms. Note that all of these results are for the standard offline setting, where demand points (customers) are known ahead of time. Meyerson [12] studied an online version of the problem and provided an $O(\log n)$-competitive algorithm. In this model, a set of demand points appear in online. In order to provide

services to these demands some facility centers have to be opened providing a *facility cost* for each center. Each demand point has to pay a *service cost* when paired with a center. The objective of this problem is to minimize the total facility cost plus service cost. Fotakis [13] presented the first deterministic online algorithm for the same problem which achieves the optimal competitive ratio of $O(\frac{\log n}{\log \log n})$.

A recently proposed facility location variant is the r-gathering problem. An r-gathering of a set of customers C for a set of facilities F is an assignment of C to open facilities $F' \subset F$ such that at least r customers are assigned to each open facility. There is a cost for assigning a customer to a facility and he objective is to minimize the total assignment cost. The r-gathering problem was independently introduced by Karger and Minkoff [14] and by Guha et al. [15] (who called it load-balanced facility-location). Armon [16] describes a simple 3-approximation algorithm for the min-max variant of this problem. Akagi and Nakano [17] provide an $O((|C| + |F|) \log |C| + |F|)$ time algorithm to solve the r-gathering problem when all customers in C and facilities in F are on the real line.

The online facility assignment problem is also related to the k-server problem proposed by Manasse et al. [18], which requires scheduling the movement of a set of k servers, represented as points in a metric space, in order to handle requests that are also in the form of points in the space. For each request, the algorithm must determine which server to move to the requested point, with the goal of minimizing the total distance covered by all servers. Manasse et al. [18] provided a 2-competitive algorithm for the 2-server problem, and a k-competitive algorithm for the k-server problem on $(k+1)$ points in metric spaces. Kleinberg [19] showed a universal lower bound of $(5 + \sqrt{7})/2$ for the competitive ratio of any balancing algorithm for 2 servers. The famous k-server conjecture that any metric space allows for a deterministic k-competitive k-server algorithm is still open. This conjecture played a significant role for the development of competitive analysis. It has been shown to be correct for some special cases, including uniform spaces [20], lines [21], trees[22], weighted stars (all the requests are placed at the leaves of a weighted star) [23], the 3-server problem in the Manhattan plane [24], and spaces with $k + 2$ points [25].

The facility assignment problem can be seen as a generalization of the matching problem [26], where each facility has capacity $l \geq 1$. In the online matching problem, the facilities are correspond to the right side of a bipartite graph. The customers appear in an online manner as vertices of the left side of the graph each customer must be assigned to a facility before the next customer appears. This problem was first introduced by Khuller et al. [27], and independently by Kalyanasundaram et al. [28]. Koutsoupias et al. [29] provided a $O(n)$ competitive algorithm for online matching on a line. Bansal et al. [30] provided a $O(\log^2 n)$ competitive randomized algorithm for the online metric matching problem. Kao et al. [31] provide a randomized lower bound of 4.5911 for online matching on a line. We provide a randomized algorithm, which is $\frac{9}{2}$-competitive for a class of input sequences. Antoniadis et al. [32] describe an $o(n)$-competitive deterministic algorithm for online matching on a line.

Despite similarities, the problems above are different from the online facility assignment problem, which we introduce in this paper. In the facility location problem, customer positions are known ahead of time and we have to place facilities in a subset of given set of locations, whereas in the online facility assignment problem customers appear in an online manner and the facilities are given. The servers in the k-server problem are movable, whereas in the online facility assignment problem the positions of facilities are fixed. In the online matching problem, the facilities have unit capacity, whereas in the online facility assignment problem capacities can be greater than one.

1.2. Our Contributions

We first consider the case where both the facilities F and the customers C are on a line. We propose Algorithm Greedy and show that it has competitive ratio $4|F|$. Introducing randomization in Algorithm Greedy leads to an improved performance of $9/2$ for a special class of input instances. We then describe Algorithm Optimal-Fill and show it has competitive ratio $|F|$.

We next consider the case where both the facilities F and the customers C are located on the vertices of an unweighted graph $G = (V, E)$. We show that Algorithm Greedy has competitive ratio $2|E(G)|$ and Algorithm Optimal-Fill has competitive ratio $|E(G)||F|/r$, where r is the radius of G. Finally, we consider the case where a customer leaves after receiving service at a facility. We define *service time* as the amount of service time needed, and study the facility assignment problem with limited service time.

The rest of this paper is organized as follows. In Section 2 we provide basic definitions. In Section 3 we study the online facility assignment problem on a line. In Section 4 we study the graph version of the problem. In Section 5 we introduce a service time parameter t in our model and show that no deterministic algorithm is competitive when $t = 2$.

2. Preliminaries

A graph $G = (V, E)$ consists of a finite set V of vertices and a finite set E of edges; each edge is an unordered pair of vertices. We often denote the set of vertices G by $V(G)$ and the set of edges by $E(G)$. We say G is *unweighted* if every edge of G has equal weight. Let u and v be two vertices of G. If G has a u, v-path, then the distance from u to v is the length of a shortest u, v-path, denoted by $d_G(u, v)$ or simply by $d(u, v)$. If G has no u, v-path then $d(u, v) = \infty$. The *eccentricity* of a vertex u in G is $max_{v \in V(G)} d(u, v)$ and denoted by $\epsilon(u)$. The *radius* r of G is $min_{u \in V(G)} \epsilon(u)$ and the *diameter* of G is $max_{u \in V(G)} \epsilon(u)$. The *center* of G is the subgraph of G induced by vertices of minimum eccentricity.

In the online facility assignment problem, we are given a set of facilities $F = \{f_1, f_2, \cdots, f_{|F|}\}$ of equal capacity l in a metric space, and an input sequence of customers $I = \{c_1, c_2, \cdots, c_n\}$ which is a set of n customers who arrive one at a time in an online manner, with c_i corresponding to the location of customer i in the given space. We say an input I is *well distributed* if there is at

least one customer between any two adjacent facilities. The capacity of a facility is reduced by one when a customer is assigned to it. We denote the current capacity of facility f_i by $capacity_i$. A facility f_i is called *free* if $capacity_i > 0$. Any algorithm ALG for this problem must assign a customer c_i to a free facility f_j before a new customer c_{i+1} arrives. The cost of this assignment is the distance between c_i and f_j, which is denoted by $distance(f_j, c_i)$. We now define the *cover area* of a facility situated on a line. Consider a facility f_i with two adjacent free facilities f_j and f_k. Let p_1 and (p_2) be the mid-points of (f_i, f_j) and (f_i, f_k). The cover area of f_i is then the line segment p_1 to p_2. The total number of customers is, at most, $|F|l$ and each customer must be assigned to a facility. For any input sequence of customers I, Cost_ALG(I) is defined as the total cost of all assignments made by ALG. The objective is to minimize Cost_ALG(I).

We say an algorithm is *optimal* if, for any input sequence of customers, the total cost of the assignment it provides is the minimum possible. We denote an optimal algorithm by OPT. An online algorithm ALG is c-competitive if there is a constant α such that, for all finite input sequences I,

$$\text{Cost\_ALG}(I) \leq c.\text{Cost\_OPT}(I) + \alpha.$$

The factor c is called the *competitive ratio* of ALG. When the *additive constant* α is less than or equal to zero (i.e., Cost_ALG(I) $\leq c$.Cost_OPT(I)), we may say, for emphasis, that ALG is *strictly* c-competitive. An algorithm is called *competitive* if it attains a constant competitive ratio c. Although c may be a function of the problem parameters, it must be independent of the input I. The infimum over the set of all values c such that ALG is c-competitive is called *the competitive ratio* of ALG and is denoted by $\mathcal{R}$(ALG).

3. Facility Assignment on a Line

Let $F = \{f_1, f_2, \cdots, f_{|F|}\}$ be a set of facilities placed on a line, such that the distance between every pair of adjacent facilities is d, where d is a constant. An input sequence $I = \{c_1, c_2, \cdots, c_n\}$ is a set of n customers who arrive one at a time in an online manner, with c_i corresponding to the location of customer i on the line. In Section 3.1 we describe Algorithm Greedy with competitive ratio $4|F|$. In Section 3.3 we introduce randomization to Algorithm Greedy and show that it is $\frac{9}{2}$-competitive for a special class of input sequences. In Section 3.4 we describe Algorithm Optimal-Fill and show it has competitive ratio $|F|$.

3.1. Algorithm Greedy

Here we describe and analyze the natural greedy algorithm, which assigns each customer to the nearest free facility.

Algorithm Greedy

Input: Customers $I = \{c_1, \cdots, c_n\}$, facilities $F = \{f_1, \cdots, f_{|F|}\}$, capacity l

Output: An assignment of C to F and the total cost of that assignment

```
sum ← 0;
for i ← 1 to |F| do
    capacity_i = l;
for i ← 1 to n do
    min ← ∞;
    index ← −1;
    for j ← 1 to f do
        if capacity_j > 0 and distance(f_j, c_i) < min then
            min ← distance(f_j, c_i);
            index ← j;
    assign c_i to f_index;
    capacity_index ← capacity_index − 1;
    sum ← sum + min;
```

Result: sum is the total cost

We can analyze the online algorithm in the context of a game between an *online player* and a malicious *adversary*. The online player runs the online algorithm on an input created by the adversary. The adversary, based on the knowledge of the online algorithm, constructs the worst possible input (i.e., one that maximizes the competitive ratio). Consider Algorithm Greedy above and the adversary strategy of making an instance very costly for Algorithm Greedy but, at the same time, inexpensive for OPT. The following lemma gives a lower bound for OPT's cost.

Lemma 1. *Let d be the distance between all adjacent facilities. If the assignments of OPT and Algorithm Greedy are not the same, then OPT's cost is at least $\frac{d}{2}$.*

Proof. Let c_x be the first customer for which the assignments of OPT and Algorithm Greedy differ. The optimal cost for assigning c_x is at least $\frac{d}{2}$. Hence the total optimal cost is at least $\frac{d}{2}$. □

The following theorem determines the worst input sequence an adversary can construct for Algorithm Greedy and provides a competitive ratio.

Theorem 1. *Let $F = \{f_1, f_2, \cdots, f_{|F|}\}$ be a set of facilities placed on the line, such that the distance between every pair of adjacent facilities is d, where d is a constant. Then $\mathcal{R}(\text{Algorithm Greedy}) \leq 4|F|$.*

Proof. Recall the definition of a well distributed input sequence, namely that there is at least one customer between any two adjacent facilities. When the metric space is a line, all customers have cost less than d in the optimum

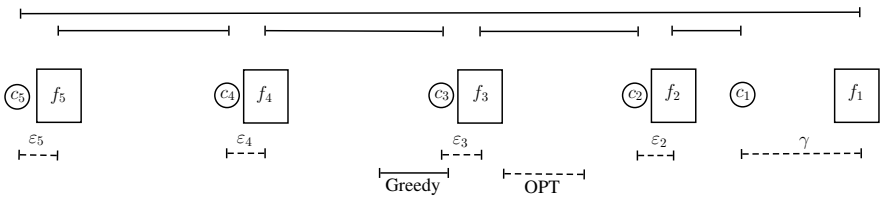

Figure 1: The configurations of Algorithm Greedy and OPT

assignment of a well distributed input sequence. However, if the input sequence is not well distributed, there are some customers with assignment cost greater than d. We consider these two cases separately. For both cases, assume now that the facilities have unit capacity. Later we will also deal with the case for capacity l, where $l > 1$.

Let f_l is the leftmost facility and f_r be the rightmost facility. Consider a customer c who appears to the left of f_l. The distance between c and f_l is $distance(f_l, c)$. Both Cost_Algorithm_Greedy(I) and Cost_OPT(I) must pay the amount $distance(f_l, c)$. The ratio of Cost_Algorithm_Greedy(I) to Cost_OPT(I) increases when $distance(f_l, c)$ decreases. The case when c appears to the right of f_r is analogous. Now consider the case where customers appear between f_l and f_r, since the ratio does not increase if customers appear outside of this range (because both OPT and Algorithm Greedy have to consider the region outside this range).

We first consider the case when all customers have costs less than d in the optimum assignment. In the worst case, the adversary places all the customers very close to the facilities except the first customer c_1 as illustrated in Figure 1. The total cost of Algorithm Greedy is no more than $2|F|d$. In the optimum assignment all customers c_i have cost ε_i except c_1. The cost of the first customer c_1 is γ, where $\gamma > \frac{d}{2}$ (Lemma 1). Then $\frac{\text{Cost\_Algorithm\_Greedy}(I)}{\text{Cost\_OPT}(I)} \leq \frac{2|F|d}{\frac{d}{2}} = 4|F|$.

In the second case, the input sequence is not well distributed. We first provide the intuition that shows why the ratio between Cost_Algorithm_Greedy(I) and Cost_OPT(I) will not be greater than the ratio in the first case. The customers are concentrated in some small areas and the effect of different assignments is limited to only these spanning areas; see Figure 2. If the spanning areas are very small, no algorithm can save that much. In extreme case, when all customers are placed in the same location, all assignments are same.

When the input sequence is not well distributed, k customers have costs greater than d in the optimum assignment. Hence, the total cost of the optimum assignment is at least kd. We have assumed that the customers at distance less than d are assigned with cost zero by the optimal algorithm and there are $|F| - k$ such customers. In the assignment created by Algorithm Greedy, each of these customers would have cost at most d. Note that if any of these customers have cost greater than d, then that assignment A can be easily transformed to an equivalent assignment B with total cost equal to that of the original assignment

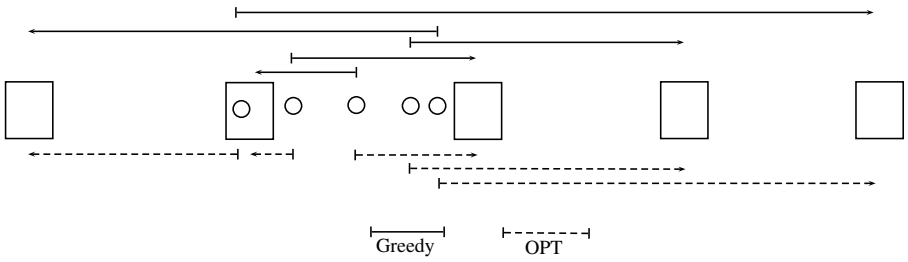

Figure 2: The configurations of Algorithm Greedy and OPT

A and so that $|F| - k$ customers have cost no more than d. The transformation goes one step at a time, as follows. If a customer c_1 assigned to a facility f_1 by OPT has cost less than or equal to d and c_1 is assigned to a facility f_2 in A and has cost greater than d, then we get a new assignment A' by assigning c_1 to f_1 and c_2 to f_2, where c_2 was the customer assigned to f_1 in A. Similarly, we can swap the next pair to get assignment A'', and continue this process until we get the equivalent assignment B. There are $|F| - k$ customers in B with cost at most d and each of the remaining k customers have a cost at most $(|F| - 1)d$. Then $\frac{\text{Cost\_Algorithm\_Greedy}(I)}{\text{Cost\_OPT}(I)} \leq \frac{(|F|-1)dk+(|F|-k)d}{kd} = \frac{(k+1)|F|}{k} - 2$.

In the analysis above we assumed unit capacity; now let each facility have capacity l, where $l > 1$. Suppose that there exists an input sequence of customers I for which the ratio is greater than $4|F|$. We can partition I into $I_1, I_2, \cdots, I_l$ in such a way that the following conditions hold:

- $I_i \cap I_j = \emptyset$ for $1 \leq i, j \leq l$ and $i \neq j$.
- $I_1 \cup I_2 \cup \cdots \cup I_l = I$.
- Exactly one customer from I_i is assigned to a facility f_j for $1 \leq i \leq l$ and $1 \leq j \leq |F|$.

Then there exists a set $I_{max} \in \{I_1, I_2, \cdots, I_l\}$ such that the ratio of the corresponding cost of Algorithm Greedy to the cost of OPT is greater than $4|F|$. If we take a set of facilities with unit capacities and place the customers of I_{max} in the same order as they appear in I, the ratio would be greater than $4|F|$ which is a contradiction to the bound of unit capacity. □

From the proof of Theorem 1 we can observe that the worst case scenario arises when every facility has capacity equal to one. In other words, the online matching problem and the online facility assignment problems are equivalent. However, this is not true for the offline version of these problems. If every facility has capacity equal to one, then the offline problem is just minimum weighted bipartite matching. When the capacity is greater than or equal to one, this problem becomes an instance of the transportation problem.

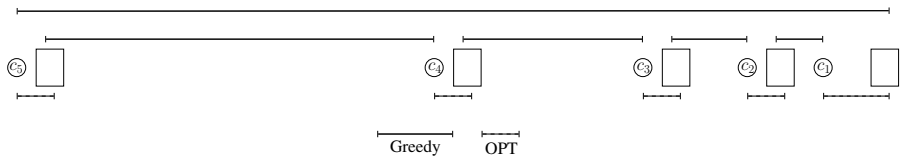

Figure 3: The worst case of Algorithm Greedy when two pairs of adjacent facilities may have different distances

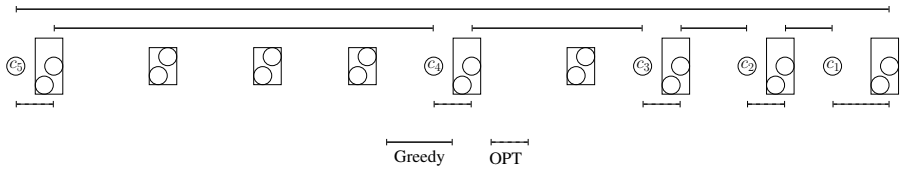

Figure 4: The worst case of Algorithm Greedy when different facilities may have different capacities

Note that this algorithm does not generalize to non-equidistant facilities. In particular, if the distances between adjacent facilities increase exponentially, this algorithm can be forced to pay a factor of $O(2^{|F|})$ more than OPT as shown in Figure 3.

We now consider a more general version of this problem where different facilities may have different capacities. The analysis is similar to the analysis for non-equidistant facilities. We can set the capacities in such a way that the distances between adjacent free facilities increase exponentially and the algorithm can be forced to pay a factor of $O(2^{|F|})$ more than OPT, as shown in Figure 4.

3.2. Capacity Sensitive Greedy Approach

In Section 3.1 we described Algorithm Greedy which assigns a customer to the nearest free facility. Note that the cover area of a facility in Algorithm Greedy is not sensitive to the capacity value, as the distance between a customer and a facility is defined as the Euclidean distance between them. Similarly, the competitive analysis of the algorithm does not depend on the capacity value. The analysis is similar when initial capacity is one or greater than one. The following lemma provides an upper bound for Algorithm Greedy.

Lemma 2. *Given two facilities f_1 and f_2 at distance d on the line $\mathcal{M}$, we have Cost_Algorithm_Greedy$(I) \leq$ Cost_OPT$(I) + \frac{|I|d}{2}$, for any input of customers I.*

Proof. We assume that the facilities have initial capacity equal to l. Hence the number of customers $|I|$ is at most $2l$. In the worst case, the adversary places the first l customers at the midpoint between two facilities. Without loss of generality, Algorithm Greedy assigns these customers to f_1. The adversary places the next

l customers on f_1. Cost_OPT(I) is equal to $\frac{ld}{2}$. Cost_Algorithm_Greedy(I) is equal to $\frac{3ld}{2} = \text{Cost\_OPT}(I) + \frac{|I|d}{2}$. □

We now introduce the idea of capacity sensitive approach. In this approach the cover area of a facility changes with its capacity. The algorithm is given below.

Algorithm Capacity-Sensitive-Greedy

Input: Customers $I = \{c_1, \cdots, c_n\}$, facilities $F = \{f_1, \cdots, f_{|F|}\}$, capacity l

Output: An assignment of C to F and the total cost of that assignment

for $i \leftarrow 1$ **to** f **do**
  $capacity_i = l$;

let d be the distance between adjacent facilities;

for $i \leftarrow 1$ **to** f **do**
  $left\_cover\_area_i = \frac{d}{2}$;
  $right\_cover\_area_i = \frac{d}{2}$;

for $i \leftarrow 1$ **to** n **do**
  $index \leftarrow -1$;
  for $j \leftarrow 1$ **to** f **do**
    if $capacity_j > 0$ *and* c_i *appears in the cover area of* f_j **then**
      $index \leftarrow j$;
  assign c_i to f_{index};
  $capacity_{index} \leftarrow capacity_{index} - 1$;
  let f_l be the left adjacent free facility of f_{index};
  if $capacity_{index} > capacity_l$ **then**
    $left\_cover\_area_{index} = \frac{distance(f_{index}, f_l)}{2^{1+(capacity_{index} - capacity_l)}}$;
    $right\_cover\_area_l = distance(f_{index}, f_l) - left\_cover\_area_{index}$;
  else
    $right\_cover\_area_l = \frac{distance(f_{index}, f_l)}{2^{1+(capacity_l - capacity_{index})}}$;
    $left\_cover\_area_{index} = distance(f_{index}, f_l) - right\_cover\_area_l$;
  similarly update cover area of right adjacent facility;
  $sum \leftarrow sum + distance(f_{index}, c_i)$;

Result: sum is the total cost

We use CSG to denote Algorithm Capacity-Sensitive-Greedy. The following lemma provides an upper bound for CSG.

Lemma 3. *Given two facilities f_1 and f_2 at distance d on the line $\mathcal{M}$, we have $Cost\_CSG(I) \leq Cost\_OPT(I) + \frac{|I|d}{4}$, for any input of customers I.*

Proof. We assume that the facilities have initial capacity equal to l. Hence the number of customers $|I|$ is at most $2l$. In the worst case, the adversary places the first customer c_1 at the midpoint between the two facilities. Let CSG assign the customer to f_1, which without loss of generality is to the left of f_2. Let the

size of the right cover area of f_1 be x before the arrival of c_1. After assigning c_1 to f_1 the size of the right cover area of f_1 becomes $\frac{x}{2}$. The adversary places the next customer c_2 on f_2. The customer c_2 will be assigned to f_2 by CSG and the size of the right cover area of f_1 becomes x again. Note that now the cover areas of both facilities is the same as before any customer appeared. The adversary places customer c_3 again at the midpoint between the two facilities, c_3 gets assigned to f_1, the adversary places customer c_4 on f_2 and CSG assigns it to f_2. Continuing this process, half of the customers assigned to f_1 and the other half to f_2 by CSG. The value of Cost_OPT(I) is equal to $\frac{ld}{4}$. The value of Cost_CSG(I) is equal to Cost_OPT(I) + $\frac{|I|d}{4}$. □

3.3. Algorithm σ-Randomized-Greedy

In this section we introduce randomness to the greedy method of the previous section and show that better competitive ratios can be obtained.With deterministic online algorithms, the adversary knows the full strategy and can select the worst input sequence. This is not possible if ALG is a randomized algorithm. An oblivious adversary must choose a finite input sequence I in advance. ALG is c-competitive against an oblivious adversary if for every such I, E[Cost_ALG(I)] $\leq c$.Cost_OPT(I) + α where α is a constant independent of I, and E[.] is the mathematical expectation operator taken with respect to the random choices made by ALG. Since the offline player does not know the outcomes of the random choices made by the online player, Cost_OPT(I) is not a random variable and there is no need to take its expectation.

We introduce randomness in Algorithm Greedy, described in the previous section, and call the new method Algorithm σ-Randomized-Greedy. Let f_x be the facility which is nearest to customer c_y and let σ be a real number. Then σ-Randomized-Greedy checks whether the distance between c_y and f_x is less than σ and if so then c_y is assigned to f_x. Otherwise, σ-Randomized-Greedy tosses a fair coin before assigning a customer to a facility, choosing the nearest free facility to the right (left) if the coin comes heads (tails).

We will next show that Algorithm σ-Randomized-Greedy performs better than Algorithm Greedy.

Algorithm σ-Randomized-Greedy

Input: Customers $I = \{c_1, \cdots, c_n\}$, facilities $F = \{f_1, \cdots, f_{|F|}\}$, capacity$= l$

Output: An assignment of C to F and the total cost of that assignment

```
sum ← 0;
for i ← 1 to |F| do
    capacity_i = l;
for i ← 1 to n do
    min ← ∞;
    index ← −1;
    for j ← 1 to |F| do
        if capacity_j > 0 and distance(f_j, c_i) < min then
            min ← distance(f_j, c_i);
            index ← j;
    if min ≥ σ then
        randomly select the nearest free facility f_k to the left or right;
        min ← distance(f_k, c_i);
        index ← k;
    assign c_i to f_index;
    capacity_index ← capacity_index − 1;
    sum ← sum + min;
```

Result: sum is the total cost

Theorem 2. *Let I be a well distributed request sequence for Algorithm Greedy. Let γ be the optimal cost for the first customer and ε_i be the optimal cost for i^{th} customer where $i > 1$. If $\sigma > \varepsilon_i$ for all i and $\sigma \leq \gamma$ then Algorithm σ-Randomized-Greedy is $\frac{9}{2}$-competitive for I.*

Proof. Let $F = \{f_1, f_2, \cdots, f_{|F|}\}$ be a set of facilities, such that the distance between every pair of adjacent facilities is d, where d is a constant. Recall that if an input I of customers has the property that all assignments cost less than d in the optimum solution, then I is *well distributed*. The first customer c_1 is placed closer to f_2 (Figure 1) in order to fool Algorithm Greedy. Algorithm Greedy assigns c_1 to f_2. Except the first customer c_1, the adversary places every customer c_k very close to facility f_k. Since Algorithm Greedy has already assigned c_1 to f_2, it can not assign c_2 to the same facility. Similarly, for every customer c_k, Algorithm Greedy assigns it to f_{k+1} although it is very close to f_k. Algorithm σ-Randomized-Greedy overcomes this situation by using randomness. Consider the first customer c_1 who is close to f_2. Algorithm σ-Randomized-Greedy chooses either f_1 or f_2 with equal probability $\frac{1}{2}$. Similarly for every customer c_k, Algorithm σ-Randomized-Greedy chooses either f_{k+1} or f_k with equal probability $\frac{1}{2}$. Then

$$
\begin{aligned}
E[\text{Cost\_}\sigma\text{-Randomized-Greedy}(I)] &= \tfrac{d}{4} + \textstyle\sum_{i=1}^{|F|-2}\{\tfrac{1}{2^{i+1}}(2id - \tfrac{d}{2})\} \\
&\quad + \tfrac{1}{2^{|F|-1}}\{2(|F|-1)d - \tfrac{d}{2}\} \\
&< \tfrac{d}{4} + d\textstyle\sum_{i=1}^{|F|-2}\tfrac{i}{2^i} \\
&< \tfrac{d}{4} + 2d \\
&= \tfrac{9d}{4}
\end{aligned}
$$

Since the optimum cost is at least $d/2$, Algorithm σ-Randomized-Greedy is $\frac{9}{2}$-competitive for I. □

This shows that Algorithm σ-Randomized-Greedy can obtain better (expected) competitive ratios than Algorithm Greedy, for appropriate values of σ. In the theorem above the value of σ is very small compared to d. If a customer c_i is placed beside a facility f_j such that the distance between c_i and f_j is less than σ, then it is assumed that there is no harm to assign c_i to f_j.

3.4. Competitive Analysis of Algorithm Optimal-Fill

In Section 3.1, we showed that Algorithm Greedy can be easily fooled by placing all the customers very close to the facilities except for the first customer. We next describe Algorithm Optimal-Fill, which is more efficient than Algorithm Greedy. The idea is that when a new customer c_i arrives, Algorithm Optimal-Fill finds out facility f_j that would be selected by an optimal assignment of all the customers $c_1, c_2, \cdots, c_i$. Algorithm Optimal-Fill then assigns c_i to f_j.

Algorithm Optimal-Fill

Input: Customers $I = \{c_1, \cdots, c_n\}$, facilities $F = \{f_1, \cdots, f_{|F|}\}$, capacity l

Output: An assignment of C to F and the total cost of that assignment

$sum \leftarrow 0$;

for $i \leftarrow 1$ **to** n **do**

  let f_j be the new facility chosen by an optimal assignment of customers $c_1, c_2, \cdots, c_i$;

  assign c_i to f_j;

  $sum \leftarrow sum + distance(f_j, c_i)$;

Result: sum is the total cost

The following theorem shows that Algorithm Optimal-Fill performs better than deterministic greedy method.

Theorem 3. *Let $F = \{f_1, f_2, \cdots, f_{|F|}\}$ be a set of facilities placed on the line, such that the distance between every pair of adjacent facilities is d, where d is a constant. Then $\mathcal{R}$(Algorithm Optimal-Fill) $\leq |F|$.*

Proof. In the worst case, the adversary can place each customer except the first one on top of a facility, so the cost is zero, while Algorithm Optimal-Fill has to pay for each of these customers. The adversary pays only for the first customer and all others are free, because they are placed on top of their facilities. However, Algorithm Optimal-Fill has to pay at least d for each of them. The

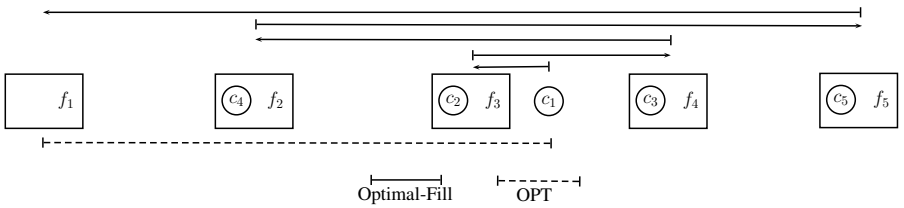

Figure 5: The adversary places the first customer c_1 between f_3 and f_4. Algorithm Optimal-Fill assigns c_1 to f_3 because it is a little bit closer compared to f_4. The adversary now places c_2 exactly on f_3. Algorithm Optimal-Fill assigns c_2 to f_4 because f_3 and f_4 are chosen by an optimal assignment for customers c_1 and c_2. The adversary then places c_3 exactly on f_4. Algorithm Optimal-Fill assigns c_3 to f_2 because f_2 is the new facility chosen by an optimal assignment for customers c_1, c_2 and c_3.

two algorithms (OPT and Optimal-Fill) are illustrated with an example with 5 facilities and 5 customers in Figure 5.

Then $\frac{\text{Cost\_Algorithm\_Optimal-Fill}(I)}{\text{Cost\_OPT}(I)} = \frac{d+2d+\cdots+(|F|-1)d+\frac{d}{2}}{\frac{|F|d}{2}} < |F|$.

We now prove that for any input sequence, the ratio between the cost of Algorithm Optimal-Fill and the cost of OPT is no more than $|F|$ when $|F| > 2$. We can assume that the customers are placed in the leftmost and the rightmost facilities, as described in Theorem 1. We only consider well distributed input sequence and unit capacities, because the analysis for input sequences that are not well distributed and capacities greater than one is the same as that in Theorem 1.

First observe that if at any step Algorithm Optimal-Fill pays a cost x, then the optimal algorithm has to pay at least $\lfloor \frac{x}{d} \rfloor \frac{d}{2}$. The claim is trivially true when $x < d$. Hence, we consider the case when $x \geq d$. We use induction on the number of customers. Consider the first time when Algorithm Optimal-Fill pays a cost greater than or equal to d; see Figure 6. The first two customers are placed in the cover areas of two different facilities. Hence, the assignments of Algorithm Optimal-Fill are same as the assignments of OPT. Both of the assignment costs are less than $\frac{d}{2}$. When the third customer appears, Algorithm Optimal-Fill pays a cost more than d for the first time. Although before the arrival of the third customer, the optimal cost for both of the previous customers were less than $\frac{d}{2}$, now both the optimal assignment have switched with a cost more than $\frac{d}{2}$. The reason is that the optimal cost for the third customer is far less than $\frac{d}{2}$ compared to the optimal cost of the remaining two customers. The new facility used by the optimal algorithm is the rightmost facility. Hence, Algorithm Optimal-Fill has to assign the new customer to the rightmost facility with cost greater that $2d$, whereas Cost_OPT(I) is at least d because of the assignment costs of first two customers. Hence, the claim is true for the first time when Algorithm Optimal-Fill pays a cost greater than or equal to d. Let the cost Algorithm Optimal-Fill has payed for the last customer be x. Note that, $\lfloor \frac{x}{d} \rfloor$ is equal to the number of customers between the last customer and

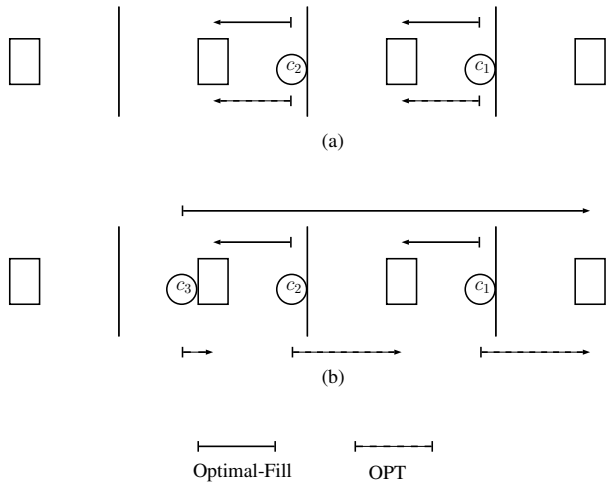

Figure 6: Algorithm Optimal-Fill pays cost greater than or equal to d first time. In Figure (a) we can see that we have two customers c_1 and c_2. The assignment costs of both customers for Algorithm Optima-Fill and the optimal algorithm are same. Both of the assignment costs are less than $\frac{d}{2}$. In Figure (b) a new customer c_3 arrives. As the optimal assignments changes for both c_1 and c_2, Algorithm Optimal-Fill has to pay a cost greater than d for c_3.

the facility to which it has been assigned by Algorithm Optimal-Fill.

Consider a new customer c_k with assignment cost of c_k, greater than or equal to d in Algorithm Optimal-Fill; see Figure 7. Let the optimal cost for customer c_i be $\frac{d}{2} + \epsilon_i$. Hence, the total cost is equal to $\frac{d}{2} + \epsilon_1 + \frac{d}{2} + \epsilon_2 + \cdots + \frac{d}{2} + \epsilon_k = \frac{kd}{2} + \epsilon_1 + \epsilon_2 + \cdots + \epsilon_k$. Let the last customer for which Algorithm Optimal-Fill payed greater than or equal to d be c_j where $j < k$. Let the cost Algorithm Optimal-Fill has payed for the customer c_j be y. According to the induction hypothesis, the optimal algorithm has to pay at least $\lfloor \frac{y}{d} \rfloor \frac{d}{2}$ for customers $c_1, c_2, \cdots, c_{k-1}$. The customers that are located between c_k and the facility f_k in which c_k has been assigned by Algorithm Optimal-Fill have switched their optimal assignments after the arrival of c_k, otherwise Algorithm Optimal-Fill would not assign c_k to f_k. Hence, the optimal assignment cost for customers $c_1, c_2, \cdots, c_{j-1}$ is equal to $\frac{d}{2} - \epsilon_1 + \frac{d}{2} - \epsilon_2 + \cdots + \frac{d}{2} - \epsilon_{j-1} = \frac{(j-1)d}{2} - \epsilon_1 - \epsilon_2 - \cdots - \epsilon_{j-1}$ which is at least $\lfloor \frac{y}{d} \rfloor \frac{d}{2} = \frac{(j-1)d}{2}$. Hence, $\epsilon_1 + \epsilon_2 + \cdots + \epsilon_{j-1} < 0$. The current optimal assignment cost for c_j is $\frac{d}{2} + \epsilon_j$. Here, $\epsilon_j > 0$ because the assignment cost of c_j by Algorithm Optimal-Fill is greater than or equal to d. Let Algorithm Optimal-Fill assign c_j to f_j. Then $\epsilon_j > \epsilon_1 - \epsilon_2 - \cdots - \epsilon_{j-1}$ as customers $c_1, c_2, \cdots, c_{j-1}$ are placed between c_j and f_j. Thus $\epsilon_i \geq 0$ when $j < i < k$ because the assignment cost of c_i by Algorithm Optimal-Fill is less than d. Hence, $\frac{(k-1)d}{2} + \epsilon_1 + \epsilon_2 + \cdots + \epsilon_{k-1} > \frac{(k-1)d}{2}$.

If the maximum assignment cost of Algorithm Optimal-Fill is z, then the total assignment cost of Algorithm Optimal-Fill is no more than $\frac{z(z+1)}{2}$. The cost of

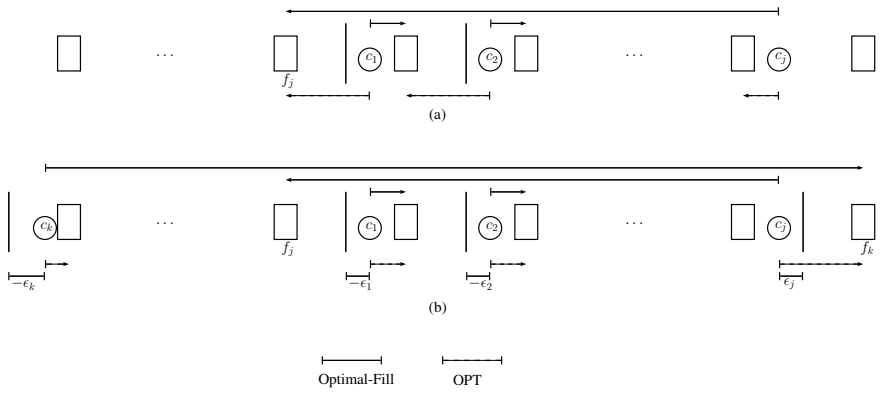

Figure 7: In Figure (a) we can see the situation before the customer c_k appears. When the customer c_k appears, the existing customers $c_1, c_2, \cdots, c_{k-1}$ change their optimal assignments as shown in Figure (b).

optimal assignment is at least $\lfloor \frac{z}{d} \rfloor \frac{d}{2}$, which is approximately $\frac{z}{2}$. Hence the ratio is $z+1$ and the maximum value of z is $|F|-1$, giving us $\mathcal{R}$(Algorithm Optimal-Fill) $\leq |F|$. □

We can generalize the idea of the above proof as shown in the following theorem.

Theorem 4. *Let $F = \{f_1, f_2, \cdots, f_{|F|}\}$ be a set of facilities placed on the line arbitrarily with arbitrary capacities. Then $\mathcal{R}$(Algorithm Optimal-Fill) $\leq |F|$.*

Proof. First consider the case where each facility has the same capacity and the distances between two adjacent facilities are different. The proof is similar to the proof of previous theorem. If Algorithm Optimal-Fill pays a cost x, then the optimal algorithm has to pay approximately $\frac{x}{2}$ in total for the existing customers. Hence, $\mathcal{R}$(Algorithm Optimal-Fill) $\leq |F|$. When facility capacities are different, we have a similar scenario as that in Section 3.1. Hence, the competitive ratio does not change. □

The results above show that the performance of Algorithm Optimal-Fill is better than Algorithm Greedy, since Algorithm Greedy can be as far from OPT as a factor of $O(2^{|F|})$.

4. Facility Assignment on Connected Unweighted Graphs

We now consider the case where the facilities F are placed on the vertices of a connected unweighted graph $G = (V, E)$ and customers arrive one by one in an online manner at vertices of G. We show that Algorithm Greedy has competitive ratio $2|E(G)|$ and Algorithm Optimal-Fill has competitive ratio $|E(G)||F|/r$, where r is the radius of G.

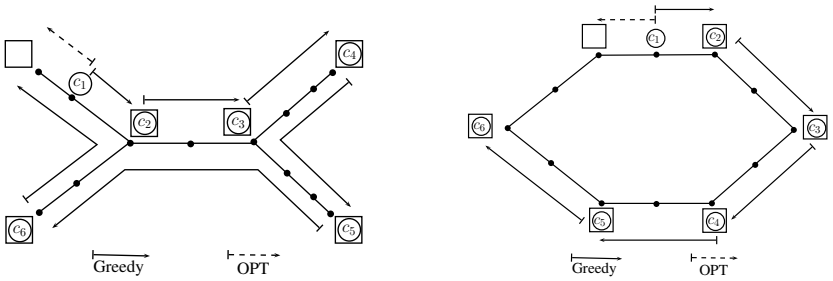

Figure 8: The configurations of Algorithm Greedy and OPT for a tree and a cycle.

4.1. Competitive Analysis of Algorithm Greedy

In Section 3.1 we analyzed Algorithm Greedy on a line. The following theorem describes the performance of Algorithm Greedy in the graph setting.

Theorem 5. *Let $\mathcal{M}$ be a connected unweighted graph. Then $\mathcal{R}$(Algorithm Greedy) $\leq 2|E(\mathcal{M})|$.*

PROOF. We assume that the facilities have unit capacity since the analysis is similar for capacity l, where $l > 1$. Two facilities f_i and f_j are *adjacent* if there exists a path P from f_i to f_j such that no other facilities are situated on P. Recall the definition of a well distributed input sequence: an input I is *well distributed* if there is at least one customer between any two adjacent facilities. We first prove the claim for an input I which is well distributed. Then we show how to transform I to I' such that I' is well distributed and the competitive ratios of I and I' are the same.

We consider two cases; $\mathcal{M}$ is a tree and $\mathcal{M}$ contains at least one cycle. If $\mathcal{M}$ is a tree, we assume that every leaf contains a facility, since $\mathcal{R}$(Algorithm Greedy) does not increase in the other case. In the worst case Cost_Greedy(I) is less than $2|E(\mathcal{M})|$ and Cost_OPT(I) is equal to one as shown in Figure 8. A square box represents a facility and the input customers are shown by their sequence numbers. In this case competitive ratio is $2|E(\mathcal{M})|$.

If $\mathcal{M}$ contains a cycle, $\mathcal{R}$(Algorithm Greedy) does not increase. Consider a set of facilities F placed situated on a cycle. In the worst case Cost_Greedy(I) is less than $|E(\mathcal{M})|$ and Cost_OPT(I) is equal to one, as shown in Figure 8. In this case the competitive ratio is $|E(\mathcal{M})|$.

Now suppose the input sequence I is not well distributed. Let $\mathcal{M}'$ be the minimum subgraph of $\mathcal{M}$ so that all customers are situated on $\mathcal{M}'$. Consider the set of facilities situated on $\mathcal{M}'$. In the worst case the customers assigned to those facilities by Algorithm Greedy incur total cost less than $2|E(\mathcal{M}')|$ and OPT incurs only unit cost. If OPT incurs cost x to assign a customer to a remaining facility, then Algorithm Greedy incurs at most $x + |E(\mathcal{M}')|$ cost to assign a customer to that facility. Hence, Cost_Greedy(I) $\leq$ Cost_OPT(I) $- 1 + |E(\mathcal{M}')|(|E(\mathcal{M})| - |E(\mathcal{M}')|) + 2|E(\mathcal{M}')|$. It follows that if $|E(\mathcal{M}')|$ is small then Algorithm Greedy

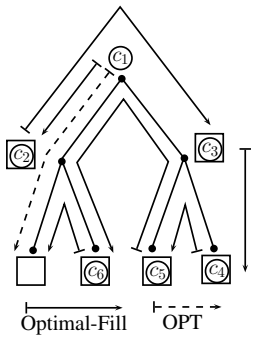

Figure 9: Worst case of Algorithm Optimal-Fill

will perform similar to OPT. The larger the value of $|E(\mathcal{M}')|$ the more well distributed the input I becomes. Hence $\mathcal{R}(\text{Algorithm Greedy}) \leq 2|E(\mathcal{M})|$. □

Theorem 5 immediately yields the following corollary.

Corollary 1. *Let $\mathcal{M}$ be a connected unweighted graph and a set of facilities F is placed on the vertices of $\mathcal{M}$ so that distance between two adjacent facilities is equal. Then $\mathcal{R}(\text{Algorithm Greedy}) \leq 4|F|$.*

4.2. Competitive Analysis of Algorithm Optimal-Fill

In Section 3.4 we showed that Algorithm Optimal-Fill was more efficient than Algorithm Greedy, when the metric space was a line. In the case of a connected unweighted graph, it is not straight-forward to determine whether Algorithm Optimal-Fill is better than Algorithm Greedy. The answer depends on the number of edges, facilities and the radius of the graph. The following theorem describes the performance of Algorithm Optimal-Fill.

Theorem 6. *Let $\mathcal{M}$ be a connected unweighted graph and a set of facilities F is placed on the vertices of $\mathcal{M}$. Then $\mathcal{R}(\text{Algorithm Optimal-Fill}) \leq \frac{|E(\mathcal{M})||F|}{r}$.*

Proof. The proof is similar to the analysis of Theorem 5. It is sufficient to consider the case when $\mathcal{M}$ is a tree and I is well distributed. Let x be a vertex in the center of $\mathcal{M}$ which is not a facility. If no such vertex exists, the first customer c_1 is placed on a vertex which is not a facility and the distance from the center of $\mathcal{M}$ is minimum. Otherwise, c_1 is placed on x. In the worst case, Algorithm Optimal-Fill pays a cost equal to the distance between two facilities for each customer, except the first one (see Figure 9). The adversary pays a cost which is no more than *radius* only for the first customer. Algorithm Optimal-Fill traverses an edge no more than $|F|$ times. Hence, $\mathcal{R}(\text{Algorithm Optimal-Fill})$ is at most $\frac{|E(\mathcal{M})||F|}{r}$.

□

5. Facility Assignment with a Finite Service Time

Until now we have assumed that if a customer c_i is assigned to a facility f_j, then c_i remains there forever. In other words, the service time of an assignment is infinite. Hence a facility with capacity l can provide service to at most l customers. If there are $|F|$ facilities, the total number of customers is limited to $|F|l$. In this section we study the facility assignment problem with a finite service time t. We assume a unit time interval between arrivals of customers. When $t = 1$, the service time is unit. Let c_w be assigned to f_x. and let us consider the case where all facilities have unit capacities ($l = 1$). If c_y is next to c_w then we can also assign c_y to f_x although c_w was assigned to f_x. For unit service time, both Algorithm Greedy and Algorithm Optimal-Fill provide the optimal solution. When the service time is two ($t = 2$), we can not assign c_y to f_x. However, if c_z arrives just after c_y, then we can assign c_z to f_x.

Theorem 7. *Let t be the time needed to provide service to a assigned customer. No deterministic algorithm ALG is competitive for $t = 2$.*

PROOF. Let $I = (c_1, c_2, \cdots, c_n)$ be the input sequence. The adversary places the first customer c_1 between any two adjacent facilities f_i and f_{i+1}. Suppose ALG has assigned c_1 to f_i. The adversary now places $c_2, c_3, \cdots, c_n$ exactly on the facilities assigned for $c_1, c_2, \cdots, c_{n-1}$. The adversary runs the optimal algorithm. It assigns c_1 to f_{i+1}, which incurs cost less than d, the distance between f_i and f_{i+1}. The adversary does not pay any cost for the later assignments, because each customer is placed exactly on a facility. However, ALG pays at least d for each assignment except the first one. □

6. Conclusion

We considered the online facility assignment problem and analyzed several algorithms: Algorithm Greedy, Algorithm σ-Randomized-Greedy and Algorithm Optimal-Fill. We analyzed the performance of these algorithms in two metric spaces: the 1-dimensional line and a simple, connected, unweighted graph. On the line, we made another strong assumption: that the distance between any two adjacent facilities is the same. The algorithms we describe do not generalize to the case when these distances are arbitrary. In Theorem 2, we further assumed that the input sequence of customers is also well distributed. It would be interesting to find out what happens one or both of these assumptions are dropped. In the graph setting we do not make any assumptions about how the facilities are distributed among the vertices, or about how customers are distributed among the vertices. However, our results in this setting are weaker, in the sense that they depend on parameters such as the number of edges in the graph and its radius. A natural question to ask is whether stronger results exist in the graph setting, as well as in other metric spaces.

7. Acknowledgments

We are grateful to Marc Demange for correcting an error in Theorem 3 and to the anonymous reviewers of the conference version of this paper.

References

[1] E. L. Lawler, Combinatorial Optimization: Networks and Matroids, Holt, Rinehart, and Winston, New York, 1976.

[2] F. L. Hitchcock, The distribution of a product from several sources to numerous localities, MIT Journal of Mathematics and Physics 20 (1941) 224–230.

[3] A. Schrijver, On the history of the transportation and maximum flow problems, Mathematical Programming 91 (3) (2002) 437–445.

[4] G. B. Dantzig, Application of simplex method to a transportation problem, John Wiley & Sons, 1951.

[5] A. Charnes, W. W. Cooper, The stepping stone method of explaining linear programming calculations in transportation problems, Management Science 1 (1) (1954) 49–69.

[6] L. Ford Jr, D. R. Fulkerson, Solving the transportation problem, Management Science 3 (1) (1956) 24–32.

[7] K. Jain, V. V. Vazirani, Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation, J. ACM 48 (2) (2001) 274–296.

[8] Z. Drezner, H. W. Hamacher, Facility Location: Applications and Theory, Springer Science & Business Media, 2004.

[9] D. B. Shmoys, E. Tardos, K. Aardal, Approximation algorithms for facility location problems (extended abstract), in: Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing, STOC '97, ACM, New York, NY, USA, 1997, pp. 265–274.

[10] M. Charikar, S. Guha, Improved combinatorial algorithms for the facility location and k-median problems, in: Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, 1999, pp. 378–388.

[11] M. Sviridenko, An improved approximation algorithm for the metric uncapacitated facility location problem, in: W. Cook, A. Schulz (Eds.), Integer Programming and Combinatorial Optimization, Vol. 2337 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2002, pp. 240–257.

[12] A. Meyerson, Online facility location, in: 42nd IEEE Symposium on the Foundations of Computer Science (FOCS), IEEE, 2001, pp. 426–431.

[13] D. Fotakis, On the competitive ratio for online facility location, Algorithmica 50 (1) (2008) 1–57.

[14] D. R. Karger, M. Minkoff, Building Steiner trees with incomplete global knowledge, in: Proceedings 41st Annual Symposium on Foundations of Computer Science, 2000, pp. 613–623. doi:10.1109/SFCS.2000.892329.

[15] S. Guha, A. Meyerson, K. Munagala, Hierarchical placement and network design problems, in: 41st Annual Symposium on the Foundations of Computer Science (FOCS), IEEE, 2000, pp. 603–612.

[16] A. Armon, On min-max r-gatherings, Theoretical Computer Science 412 (7) (2011) 573–582.

[17] T. Akagi, S.-i. Nakano, On r-gatherings on the line, in: J. Wang, C. Yap (Eds.), Frontiers in Algorithmics, Vol. 9130 of Lecture Notes in Computer Science, Springer International Publishing, 2015, pp. 25–32.

[18] M. S. Manasse, L. A. McGeoch, D. D. Sleator, Competitive algorithms for server problems, Journal of Algorithms 11 (2) (1990) 208–230.

[19] J. M. Kleinberg, A lower bound for two-server balancing algorithms, Information Processing Letters 52 (1) (1994) 39 – 43.

[20] D. D. Sleator, R. E. Tarjan, Amortized efficiency of list update and paging rules, Communications of the ACM 28 (2) (1985) 202–208.

[21] M. Chrobak, H. Karloff, T. Payne, S. Vishwanathan, New results on server problems, SIAM Journal on Discrete Mathematics (1990) 291–300.

[22] M. Chrobak, L. L. Larmore, An optimal on-line algorithm for k-servers on trees, SIAM Journal on Computing 20 (1) (1991) 144–148.

[23] Y. Bartal, E. Koutsoupias, On the competitive ratio of the work function algorithm for the k-server problem, Theoretical Computer Science 324 (23) (2004) 337 – 345.

[24] W. W. Bein, M. Chrobak, L. L. Larmore, The 3-server problem in the plane, Theoretical Computer Science 289 (1) (2002) 335 – 354.

[25] E. Koutsoupias, C. Papadimitriou, The 2-evader problem, in: Information Processing Letters, 1996, pp. 473–482.

[26] A. Schrijver, Combinatorial Optimization: Polyhedra and Efficiency, Vol. 24 of Algorithms and Combinatorics, Springer, Berlin, 2003.

[27] S. Khuller, S. G. Mitchell, V. V. Vazirani, On-line algorithms for weighted bipartite matching and stable marriages, Theoretical Computer Science 127 (2) (1994) 255 – 267.

[28] B. Kalyanasundaram, K. Pruhs, Online weighted matching, Journal of Algorithms 14 (3) (1993) 478 – 488.

[29] E. Koutsoupias, A. Nanavati, The online matching problem on a line, in: R. Solis-Oba, K. Jansen (Eds.), Approximation and Online Algorithms, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 179–191.

[30] N. Bansal, N. Buchbinder, A. Gupta, J. S. Naor, An $O(\log^2 k)$-competitive algorithm for metric bipartite matching, Algorithmica 68 (2) (2012) 390–403.

[31] M.-Y. Kao, J. H. Reif, S. R. Tate, Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem, Information and Computation 131 (1) (1996) 63 – 79.

[32] A. Antoniadis, N. Barcelo, M. Nugent, K. Pruhs, M. Scquizzato, A $o(n)$-competitive deterministic algorithm for online matching on a line, in: International Workshop on Approximation and Online Algorithms, Springer, 2014, pp. 11–22.