Reallocating Multiple Facilities on the Line

Dimitris Fotakis[1] , **Loukas Kavouras**[1] , **Panagiotis Kostopanagiotis**[1] , **Philip Lazos**[2] , **Stratis Skoulakis**[1] and **Nikos Zarifis**[1]

[1]National Technical University of Athens

[2]Sapienza University of Rome

fotakis@cs.ntua.gr, {lkavouras, kostopanagiotis}@corelab.ntua.gr, plazos@gmail.com, {sskoul, nzarifis}@corelab.ntua.gr

Abstract

We study the multistage *K-facility reallocation problem* on the real line, where we maintain K facility locations over T stages, based on the stage-dependent locations of n agents. Each agent is connected to the nearest facility at each stage, and the facilities may move from one stage to another, to accommodate different agent locations. The objective is to minimize the connection cost of the agents plus the total moving cost of the facilities, over all stages. *K-facility reallocation* was introduced by [de Keijzer and Wojtczak, 2018], where they mostly focused on the special case of a single facility. Using an LP-based approach, we present a polynomial time algorithm that computes the optimal solution for any number of facilities. We also consider online *K-facility reallocation*, where the algorithm becomes aware of agent locations in a stage-by-stage fashion. By exploiting an interesting connection to the classical *K-server problem*, we present a constant-competitive algorithm for $K = 2$ facilities.

1 Introduction

Facility Location is a classical problem that has been widely studied in both combinatorial optimization and operations research, due to its many practical applications. It provides a natural model for industrial planning, network design, machine learning, data clustering and computer vision [Drezner and Hamacher, 2002; Lazic, 2011; Caragiannis *et al.*, 2016; Betzler *et al.*, 2013]. In its basic form, K-Facility Location instances are defined by the locations of n agents in a metric space. The goal is to find K facility locations so as to minimize the sum of distances of the agents to their nearest facility.

In many natural location and network design settings, agent locations are not known in advance. Motivated by this fact, [Meyerson, 2001] introduced online facility location problems, where agents arrive one-by-one and must be *irrevocably* assigned to a facility upon arrival. Moreover, the fast increasing volume of available data and the need for responsive services has led to *online* clustering algorithms [Liberty *et al.*, 2016], trading off the quality against the consistency of the clusters over time. In practical settings related to online data clustering, new data points arrive, and the decision of clustering some data points together should not be regarded as irrevocable (see e.g., [Fotakis, 2011] and the references therein).

More recently, understanding the dynamics of temporally evolving social or infrastructure networks has been the central question in many applied areas such as viral marketing, urban planning etc. *Dynamic facility location* proposed by [Eisenstat *et al.*, 2014] is a new tool to analyze temporal aspects of such networks. In this time dependent variant of facility location, agents may change their location over time and we look for the best tradeoff between the optimal connections of agents to facilities and the stability of solutions between consecutive timesteps. The stability of the solutions is modeled by introducing an additional moving cost (or switching cost), which has a different definition depending on the particular setting.

In this work, we study the multistage *K-facility reallocation problem* on the real line, introduced by [de Keijzer and Wojtczak, 2018]. In *K-facility reallocation*, K facilities are initially located at $(x_1^0, \ldots, x_K^0)$ on the real line. Facilities are meant to serve n agents for the next T days. At each day, each agent connects to the facility closest to its location and incurs a connection cost equal to this distance. Agent locations may change every day, thus we have to move facilities accordingly so as to reduce the connection cost. Naturally, moving a facility is not for free, but comes with the price of the distance that the facility was moved. Our goal is to specify the exact positions of the facilities at each day so that the total connection cost plus the total moving cost is minimized over all T days. In the online version of the problem, the positions of the clients at each stage t are revealed only after determining the locations of the facilities at stage $t-1$.

For a motivating example, consider a company willing to advertise its products. So, it organizes K advertising campaigns at different locations of a large city for the next T days. Based on planned events, weather forecasts, etc., the company estimates a population distribution over the locations of the city for each day. Then, it decides to compute the best possible campaign reallocation with K campaigns over all days (see also [de Keijzer and Wojtczak, 2018] for more examples).

[de Keijzer and Wojtczak, 2018] fully characterized the optimal offline and online algorithms for the special case of a single facility and presented a dynamic programming algorithm for $K \geq 1$ facilities with running time exponential in K. Despite the practical significance and the interesting theo-

retical properties of K-facility reallocation, its computational complexity and its competitive ratio (for the online variant) are hardly understood.

In this work, we resolve the computational complexity of *K-facility reallocation* on the real line and take a first step towards a full understanding of the competitive ratio for the online variant. More specifically, in Section 3, we present an optimal algorithm with running time polynomial in the combinatorial parameters of *K-facility reallocation* (i.e., n, T and K). This substantially improves on the complexity of the algorithm, presented in [de Keijzer and Wojtczak, 2018], that is exponential in K. Our algorithm solves a Linear Programming relaxation and then *rounds* the *fractional solution* to determine the positions of the facilities.

Interestingly, the extreme points of the K-facility reallocation polytope (as described in in Figure 2) do not need to be integral. This is true even for the polytope of K-facility location (a.k.a. K-median, where T=1) on the line (see e.g., [de Vries *et al.*, 2007]). Moreover, there are examples of classical optimization problems (e.g. Minimum Spanning Trees, Maximum Bipartite Matching) with integral LP formulations, where the the natural time-evolving LP formulations (which are similar to the LP formulation in Figure 2) are not integral and have large integrality gaps [Gupta *et al.*, 2014]. Our main technical contribution is to show that a simple rounding scheme yields an integral solution that has the exact same cost as the optimal fractional one.

Our second main result concerns the *online version* of the problem with $K = 2$ facilities. We start with the observation that online *K-facility reallocation problem* with $K \geq 2$ facilities is a natural and interesting generalization of the classical *K-server problem*, which has been a driving force in the development of online algorithms for decades. The key difference is that, in the *K-server problem*, there is a single agent that changes her location at each stage and a single facility has to be relocated to this new location at each stage. Therefore, the total connection cost is by definition 0, and we seek to minimize the total moving cost.

From a technical viewpoint, the *K-facility reallocation problem* poses a new challenge, since it is *much* harder to track the movements of the optimal algorithm as the agents keep coming. It is not evident at all whether techniques and ideas from the *K-server problem* can be applied to the *K-facility reallocation* problem, especially for more general metric spaces. As a first step towards this direction, we design a constant-competitive algorithm, when $K = 2$. Our algorithm appears in Section 4 and is inspired by the *double coverage algorithm* proposed for the *K-server problem* [Koutsoupias, 2009].

We can cast the *K-facility reallocation problem* as a clustering problem on a temporally evolving metric. From this point of view, *K-facility reallocation problem* is a dynamic *K-median* problem. A closely related problem is the *dynamic facility location problem*, [Eisenstat *et al.*, 2014; An *et al.*, 2017]. Other examples in this setting are the *dynamic sum radii clustering* [Blanchard and Schabanel, 2017] and multi-stage optimization problems on matroids and graphs [Gupta *et al.*, 2014].

In [Friggstad and Salavatipour, 2011], a mobile facility location problem was introduced, which can be seen as a one

$$(1)\quad \min \sum_{t=1}^{T} \left[\sum_{i \in C} \sum_{j \in V} d(\mathrm{Loc}(i,t), j) x_{ij}^t + \sum_{k \in F} S_k^t \right]$$

$$\sum_{j \in V} x_{ij}^t = 1 \qquad \forall i \in C, t \in \{1, T\}$$

$$x_{ij}^t \leq c_j^t \qquad \forall i \in C, j \in V, t \in \{1, T\}$$

$$c_j^t = \sum_{k \in F} f_{kj}^t \qquad \forall j \in V, t \in \{1, T\}$$

$$\sum_{j \in V} f_{kj}^t = 1 \qquad \forall k \in F, t \in \{1, T\}$$

$$S_k^t = \sum_{j,l \in V} d(j,l) S_{kjl}^t \qquad \forall k \in F, t \in \{1, T\}$$

$$\sum_{j \in V} S_{kjl}^t = f_{kl}^t \qquad \forall k \in F, l \in V, t \in \{1, T\}$$

$$\sum_{l \in V} S_{kjl}^t = f_{kj}^{t-1} \qquad \forall k \in F, j \in V, t \in \{1, T\}$$

$$x_{ij}^t, f_{kj}^t, S_{klj}^t \in \{0, 1\} \qquad \forall k \in F, j \in V, t \in \{1, T\}$$

Figure 1: Formulation of *K-facility reallocation*

stage version of our problem. They showed that even this version of the problem is NP-hard in general metric spaces using an approximation preserving reduction to *K-median problem*.

Online facility location problems and variants have been extensively studied in the literature, see [Fotakis, 2011] for a survey. [Divéki and Imreh, 2011] studied an online model, where facilities can be moved with zero cost. As we have mentioned before, the online variant of the *K-facility reallocation problem* is a generalization of the *K-server problem*, which is one of the most natural online problems. [Koutsoupias, 2009] showed a $(2K - 1)$-competitive algorithm for the *K-server problem* for every metric space, which is also K-competitive, in case the metric is the real line [Bartal and Koutsoupias, 2000]. Other variants of the *K-server problem* include the *(H, K)-server problem* [Bansal *et al.*, 2017; Bansal *et al.*, 2018], the *infinite server problem* [Coester *et al.*, 2017] and the *K-taxi problem* [Fiat *et al.*, 1990; Coester and Koutsoupias, 2019].

2 Problem Definition and Preliminaries

Definition 1 (*K-Facility Reallocation*) *We are given a tuple (x^0, C) as input, where the vector $x^0 = (x_1^0, \ldots, x_K^0)$ describes the initial positions of the facilities. The positions of the agents over time are described by $C = (C_1, \ldots, C_T)$. The position of agent i at stage t is α_i^t and $C_t = (\alpha_1^t, \ldots, \alpha_n^t)$ describes the positions of the agents at stage t.*

Definition 2 *A solution of K-Facility Reallocation Problem is a sequence $x = (x^1, \ldots, x^T)$. Each $x^t = (x_1^t, \ldots, x_K^t)$ is a K dimensional vector that gives the positions of the facilities at stage t and x_k^t is the position of facility k at stage t. The cost of the solution x is*

$$Cost(x) = \sum_{t=1}^{T} \left[\sum_{k=1}^{K} |x_k^t - x_k^{t-1}| + \sum_{i=1}^{n} \min_{1 \leq k \leq K} |\alpha_i^t - x_k^t| \right]$$

Given an instance (x^0, C), the goal is to find a solution x that minimizes the $Cost(x)$. The term $\sum_{t=1}^{T}\sum_{k=1}^{K}|x_k^t - x_k^{t-1}|$ describes the cost for moving the facilities from place to place and we refer to it as *moving cost*, while the term $\sum_{t=1}^{T}\sum_{i=1}^{n}\min_{1\le k\le K}|\alpha_i^t - x_k^t|$ describes the connection cost of the agents and we refer to it as *connection cost*.

In the online setting, we study the special case of 2-*facility reallocation problem*. We evaluate the performance of our algorithm using competitive analysis; an algorithm is c-competitive if for every request sequence, its online performance is at most c times worse (up to a small additive constant) than the optimal *offline* algorithm, which knows the entire sequence in advance. Due to space limitations, most proofs are omitted and can be found in https://128.84.21.199/pdf/1905.12379.pdf.

3 Polynomial Time Algorithm

Our approach is a typical LP based algorithm that consists of two basic steps.

- **Step 1:** Expressing the K-*Facility Reallocation Problem* as an Integer Linear Program.
- **Step 2:** Solving *fractionally* the Integer Linear Program and *rounding* the fractional solution to an integral one.

3.1 Formulating the Integer Linear Program

A first difficulty in expressing the K-*Facility Reallocation Problem* as an Integer Linear Program is that the positions on the real line are infinite. We remove this obstacle with help of the following lemma proved in [de Keijzer and Wojtczak, 2018].

Lemma 3.1 *Let (x_0, C) an instance of the K-facility reallocation problem. There exists an optimal solution x^* such that for all stages $t \in \{1, T\}$ and $k \in \{1, K\}$,*

$$x_k^{*t} \in C_1 \cup \ldots \cup C_T \cup x^0$$

According to Lemma 3.1, there exists an optimal solution that locates the facilities only at positions where either a agent has appeared or a facility was initially lying. Lemma 3.1 provides an exhaustive search algorithm for the problem and is also the basis for the *Dynamic Programming* approach in [de Keijzer and Wojtczak, 2018]. We use Lemma 3.1 to formulate our Integer Linear Program.

The set of positions $Pos = C_1 \cup \ldots \cup C_T \cup x^0$ can be represented equivalently by a path $P = (V, E)$. In this path the j-th node corresponds to the j-th leftmost position of Pos and the distance between two consecutive nodes on the path equals the distance of the respective positions on the real line. Now, the *facility reallocation problem* takes the following *discretized form*: We have a path $P = (V, E)$ that is constructed by the specific instance (x^0, C). Each facility k is initially located at a node $j \in V$ and at each stage t, each agent i is also located at a node of P. The goal is to move the facilities from node to node such that the connection cost of the agents plus the moving cost of the facilities is minimized.

To formulate this discretized version as an Integer Linear Program, we introduce some additional notation. Let $d(j, l)$ be the distance of the nodes $j, l \in V$ in P, F be the set of facilities and C be the set of agents. For each $i \in C$, $\text{Loc}(i, t)$ is the node where agent i is located at stage t. We also define the following $\{0, 1\}$-indicator variables for all $t \in \{1, T\}$: $x_{ij}^t = 1$ if, at stage t, agent i connects to a facility located at node j, $f_{kj}^t = 1$ if, at stage t, facility k is located at node j, $S_{kjl}^t = 1$ if facility k was at node j at stage $t-1$ and moved to node l at stage t. Now, the problem can be formulated as the Integer Linear Program depicted in Figure 1.

The first three constraints correspond to the fact that at every stage t, each agent i must be connected to a node j where at least one facility k is located. The constraint $\sum_{j\in V} f_{kj}^t = 1$ enforces each facility k to be located at exactly one node j. The constraint $S_k^t = \sum_{j,l\in V} d(j,l)S_{kjl}^t$ describes the cost for moving facility k from node j to node l. The final two constraints ensure that facility k moved from node j to node l at stage t if and only if facility k was at node j at stage $t-1$ and was at node l at stage t ($S_{kjl}^t = 1$ iff $f_{kl}^t = 1$ and $f_{kj}^{t-1} = 1$).

We remark that the values of f_{kj}^0 are determined by the initial positions of the facilities, which are given by the instance of the problem. The notation x_{ij}^t should not be confused with x_k^t that is the position of facility k at stage t on the real line .

Algorithm 1: Algorithm for the offline case

Data: Given the initial positions $x^0 = \{x_1^0, \ldots, x_K^0\}$ of the facilities and the positions of the agents $C = \{C_1, \ldots, C_T\}$.

- Build the path P and the Integer Linear Program (1).
- Solve the relaxation of the Integer Linear Program (1).
- *Rounding* [1]: For each stage $t \ge 1$:
  - For $m = 1, \ldots, K$, find the node j_m^t such that $\sum_{\ell=1}^{j_m^t - 1} c_\ell^t \le m - 1 \le \sum_{\ell=1}^{j_m^t} c_\ell^t$
  - Set facility m at the respective position of node j_m^t on the line, $x_m^t \leftarrow d(j, 1) + \min_{p\in C_1\cup\ldots\cup C_T\cup x^0} p$.

3.2 Rounding the Fractional Solution

Our algorithm, described in Algorithm 1, is a simple rounding scheme for the *optimal fractional solution* of the ILP of Figure 1. This simple scheme produces an integral solution that has the exact same cost with the optimal fractional solution.

Lemma 3.2 *Let x denote the solution produced by Algorithm 1. Then,*

$$Cost(x) = \sum_{t=1}^{T}\left[\sum_{i\in C}\sum_{j\in V} d(Loc(i,t), j)x_{ij}^t + \sum_{k\in F} S_k^t\right]$$

where x_{ij}^t, S_k^t denote the values of these variables in the optimal fractional solution of the Integer Linear Program (1).

Lemma 3.2 is the main result of this section and it implies the optimality of our algorithm. We remind that by Lemma 3.1 there is an optimal solution that locates facilities only in positions $C_1 \cup \ldots \cup C_T \cup x^0$. This solution corresponds to

an integral solution of our Integer Linear Program, meaning that $Cost(x^*)$ is greater or equal than the cost of the *optimal fractional solution*, which by Lemma 3.2 equals $Cost(x)$.

We dedicate the rest of the section to sketch the key steps towards proving Lemma 3.2. The technical detailes are omitted due to lack of space. Putting technicalities aside, the proof is based on a simple and intuitive observation. For the rest of the section, $c_j^t, x_{ij}^t, f_j^t, S_{kjl}^t$ will denote the values of these variables in the *optimal fractional* solution of the ILP (1).

Definition 3 *V_t^+ denotes the set of nodes of P with a positive amount of facility at stage t. Formally,*

$$j \in V_t^+ \text{ if and only if } c_j^t > 0$$

Nodes in V_t^+ are ordered from left to right and when we refer to the j-th node of V_t^+, we mean the j-th left most node.

Observation 1 *The set of nodes at which **agent** i **connects** at stage t are **consecutive** nodes of V_t^+.*

The term *consecutive nodes of V_t^+* has the following interpretation: every node $j \in V$ between two of the *consecutive nodes of V_t^+* either belongs to the these consecutive nodes of V_t^+ or does not belong to V_t^+. Observation 1 follows from the fact that at any stage t, agent i connects to the nodes of V_t^+ that are closest to $\text{Loc}(i,t)$ and have a total amount of facility (c_j^t) at least 1. It is not hard to see that if these nodes are not *consecutive nodes* of V_t^+, then the connection cost of i at stage t can be decreased, which contradicts the optimality of the *optimal fractional solution*. We use Observation 1 to prove Lemma 3.2 under the following assumption.

Assumption 1 *Let f_{jk}^t and c_j^t be either $1/N$ or 0 for some positive integer N.*

Assumption 1 is not satisfied in general by the fractional solution of the LP (1). Actually, each $S_{kj\ell}^t$ is either 0 or $A_{kj\ell}^t/N_{kj\ell}^t$, for some positive integers $A_{kj\ell}^t, N_{kj\ell}^t$, and each positive f_{kj}^t is of the form B_{kj}^t/N, where $N = \Pi_{S_{kj\ell}^t>0} N_{kj\ell}^t$ (since $f_{kj}^t = \sum_{j\in V} S_{kj\ell}^t$).

However, starting from a fractional solution of the LP (1), we can enforce Assumption 1 for a (possibly exponentially) large N, if we create enough collocated copies of each point (j,t) and equi-distribute the values of f_{jk}^t and c_j^t over them. The crucial observation is that We need Assumption 1 just to simplify the analysis. After we show (i) how to transform a fractional solution so that it satisfies Assumption 1 ; and (ii) that under Assumption 1, Algorithm 1 computes an optimal integral solution, we observe that the facility locations of Algorithm 1 can be derived from the fractional solution in linear time. In the following, show that under Assumption 1, Algorithm 1 computes an integral optimal solution with cost no larger than the cost of the initial fractional solution.

We note that $|V_t^+| = KN$, since there are exactly K facilities. By Assumption 1, each facility k places an amount of facility $1/N$ to exactly N nodes of V_t^+ at each stage t. Now, observe that the connection cost only depends on the values c_j^t, meaning that the *matching* between facilities and nodes of V_t^+ only influences the moving cost of the optimal fractional solution. Thus, the *optimal fractional solution* chooses this *matching* so as to minimize the fractional moving cost. This *optimal matching* is the one presented in Statement 1.

Statement 1 *Let the facilities be ordered from left to right according to their initial positions. The m-th facility places amount of facility $f_{mj}^t = 1/N$ from the $(m-1)N+1$ to the mN node of $V_t^+, \forall t \geq 1$.*

Now, consider the following N integral solutions: The p-th solution locates at stage t, facility m at the $(m-1)N+p$ node of V_t^+. These integral solutions are denoted as $\{Sol_p\}_{p=1}^N$ and notice that Algorithm 1 produces Sol_1.

Let $MovCost_i^t(Sol_p)$, $ConCost_i^t(Sol_p)$ denote the moving cost of the facilities and the connection cost of agent i at stage t in the integral solution Sol_p respectively. By the construction of the above integral solutions and by Statement 1, the following equality follows:

$$\frac{1}{N}\sum_{p=1}^{N} MovCost(Sol_p) = \sum_{t=1}^{T}\sum_{k\in F} S_k^t \tag{1}$$

Now, Observation 1 comes into play: each agent i connects to N *consecutive* nodes of V_t^+. We denote this set of nodes as V_{ti}^+. By the construction of the solutions $\{Sol_p\}_{p=1}^N$, each Sol_p locates *exactly one* facility to *exactly one* node of V_{ti}^+. Moreover, $x_{ij}^t = 1/N$ for all $j \in V_{ti}^+$, since by Assumption 1, $c_j^t = 1/N$ or 0. Using the last two facts, one can prove that:

$$\frac{1}{N}\sum_{p=1}^{N} ConCost_i^t(Sol_p) = \sum_{i\in C}\sum_{j\in V} d(\text{Loc}(i,t),j)x_{ij}^t \tag{2}$$

Summing Equation (2) over all i and t, and adding it to Equation (1) gives us, that $\frac{1}{N}\sum_{p=1}^{N} Cost(Sol_p)$ equals

$$\sum_{t=1}^{T}\left[\sum_{i\in C}\sum_{j\in V} d(\text{Loc}(i,t),j)x_{ij}^t + \sum_{k\in F} S_k^t\right]$$

Since every $Cost(Sol_p)$ is at least the fractional optimal cost, we immediately obtain Lemma 3.2 (Sol_1 is the solution produced by Algorithm 1).

4 A Constant-Competitive Algorithm

In this section, we present an algorithm for the online *2-facility reallocation problem* and we discuss the core ideas that prove its performance guarantee.

Our algorithm (Algorithm 2) consists of two major steps. In Step 1, facilities are initially moved towards the positions of the agents. We remark that in Step 1, the final positions of the facilities at stage t are not yet determined. The purpose of this step is to bring at least one facility close to the agents. This initial moving consists of three cases (see Figure 2), depending only on the relative positions of the facilities at stage $t-1$ and the agents at stage t.

In Step 2, our algorithm determines the final positions of the facilities x_1^t, x_2^t. Notice that after Step 1, at least one of the facilities is inside the interval $[\alpha_1^t, \alpha_n^t]$, meaning that at least one of the facilities is close to the agents. As a result, our algorithm may need to decide between moving the second

facility close to the agents or just letting the agents connect to the facility that is already close to them. Obviously, the first choice may lead to small connection cost, but large moving cost, while the second has the exact opposite effect. Roughly speaking, Algorithm 2 does the following: If the connection cost of the agents, when placing just one facility optimally, is not much greater than the cost for moving the second facility inside $[\alpha_1^t, \alpha_n^t]$, then Algorithm 2 puts the first facility to the position that minimizes the connection cost, if one facility is used. Otherwise, it puts the facilities to the positions that minimize the connection cost, if two facilities are used. The above cases are depicted in Figure 3. We formalize how this choice is performed, introducing some additional notation.

Definition 4
- $C_t = \{\alpha_1^t, \ldots, \alpha_n^t\}$ *denotes the positions of the agents at stage* t *ordered from left to right. If* $|C_t| = 2k, k \in \mathbb{N}_{>0}$*, then* M_C *is the interval* $[\alpha_{n/2}^t, \alpha_{n/2+1}^t]$*. If* $|C| = 2k+1, k \in \mathbb{N}_0$*, then* M_C *is a single point.*
- $H(C)$ *denotes the optimal connection cost for the set* C *when all agents of* C *connect to the just one facility. That is* $H(C) = \sum_{\alpha \in C} |\alpha - M_C|$*. We also define* $H(\emptyset) = 0$*.*
- C_{1t}^* *(resp.* C_{2t}^**) denotes the positions of the agents that connect to facility* 1 *(resp.* 2*) at stage* t *in the optimal solution* x^**.* C_{1t} *(resp.* C_{2t}*) denotes the positions of the agents that connect to facility* 1 *(resp.* 2*) at stage* t *in the solution produced by Algorithm 2.*

Despite its lengthy description, Algorithm 2 is rather simple. Only the last two cases are difficult to handle, and we explain them subsequently. The performance guarantee of Algorithm 2 is formally stated in Theorem 4.1.

Theorem 4.1 *Let* $x = \{x_1^t, x_2^t\}_{t \geq 1}$ *the solution produced by Algorithm 2 and* x^* *the optimal solution. Then,*

$$Cost(x) \leq 63 \cdot Cost(x^*) + |x_1^0 - x_2^0|$$

where x_1^0, x_2^0 *are the initial positions of the servers.*

We remark that the factor 63 in Theorem 4.1 can be improved to 20, but this requires considering some additional cases that are omitted for the sake of exposition. The rest of the section is dedicated to provide a proof sketch of Theorem 4.1. Before proceeding, we present Lemma 4.2 that is a key component in the subsequent analysis and reveals the real difficulty of the online 2-*facility reallocation problem*.

Lemma 4.2 *Let the optimal solution* x^* *and* C_{1t}^*, C_{2t}^* *the set of agents that connect respectively to facility* 1 *and* 2 *at stage* t*. Let the solution* $y^t = (y_1^t, y_2^t)$ *defined as follows:*

$$y_k^t = \begin{cases} M_{C_{kt}^*} & \text{if } C_{kt}^* \neq \emptyset \\ x_k^{*t} & \text{if } C_{kt}^* = \emptyset \end{cases}$$

Then the following inequality holds,

$$\sum_{t=1}^{T} \left[\sum_{k=1}^{2} H(C_{kt}^*) + |y_k^t - y_k^{t-1}| \right] \leq 3 \cdot Cost(x^*)$$

Lemma 4.2 indicates that the real difficulty of the problem is not determining the exact positions of the facilities in the optimal solution, but to determine the *service clusters* that the optimal solution forms. In fact, if we knew the clusters C_{1t}^*, C_{2t}^* then Lemma 4.2 provides us with a 3-approximation algorithm. Obviously, this information cannot be acquired in the online setting, since C_{1t}^*, C_{2t}^* depend on the future positions of the agents that we do not know. We prove that Algorithm 2 has an approximation guarantee of 21 with respect to the solution y, that directly translates to an approximation guarantee of 63 with respect to $Cost(x^*)$. The latter is formally stated in Lemma 4.3 and is the main result of this section.

Lemma 4.3 *Let* $x = \{x_1^t, x_2^t\}_{t \geq 1}$ *be the solution produced by Algorithm 2. Then, the cost paid by solution* x *at stage* t*,* $\sum_{k=1}^{2} |x_k^t - x_k^{t-1}| + \sum_{i=1}^{n} \min_{k=1,2} |x_k^t - \alpha_i^t|$*, is at most*

$$21 \sum_{k=1}^{2} [H(C_{kt}^*) + |y_k^t - y_k^{t-1}|] + \Phi_t(x^t) - \Phi_{t-1}(x^{t-1})$$

where $\Phi_t(x_1, x_2) = 2(|x_1 - y_1^t| + |x_2 - y_2^t|) + |x_1 - x_2|$.

Algorithm 2: Selecting x_1^t and x_2^t

Data: At stage $t \geq 1$ the new positions of the agents $C_t = \{\alpha_1^t, \ldots, \alpha_n^t\}$, ordered from left to right, arrive

Step 1: Moving the facilities towards the agents

$z_1 \leftarrow x_1^{t-1}$, $z_2 \leftarrow x_2^{t-1}$

if $z_1 > \alpha_n^t$ **then**

move facility 1 *to the left until it hits* α_n^t

$z_1 \leftarrow \alpha_n^t$

end

if $z_2 < \alpha_1^t$ **then**

move facility 2 *to the right until it hits* α_1^t

$z_2 \leftarrow \alpha_1^t$

end

if $z_1 < \alpha_1^t$ ***and*** $z_2 > \alpha_n^t$ **then**

move facility 1 *to the right and facility* 2 *to the left until a facility hits* $[\alpha_1^t, \alpha_n^t]$

$z_1 \leftarrow z_1 + \min(|x_1^{t-1} - \alpha_1^t|, |x_2^{t-1} - \alpha_n^t|)$

$z_2 \leftarrow z_2 - \min(|x_1^{t-1} - \alpha_1^t|, |x_2^{t-1} - \alpha_n^t|)$

end

Step 2: Selecting the final position of the facilities

if $\alpha_1^t \leq z_1 \leq \alpha_n^t$ ***and*** $z_2 - \alpha_n^t \geq 3H(C_t)$ **then**

put facility 1 *to the median of* C_t *and move facility* 2 *to the left by* $3H(C_t)$

$x_1^t \leftarrow M_{C_t}$, $x_2^t \leftarrow z_2 - 3H(C_t)$

end

if $\alpha_1^t \leq z_2 \leq \alpha_n^t$ ***and*** $\alpha_1^t - z_1 \geq 3H(C_t)$ **then**

put facility 2 *to the median of* C_t *and move facility* 1 *to the right by* $3H(C_t)$

$x_1^t \leftarrow z_1 + 3H(C_t)$, $x_2^t \leftarrow M_{C_t}$

end

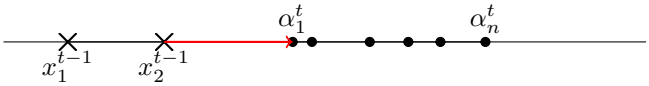
else

Compute the optimal partition (O_1, O_2) *of* C_t *that minimizes the connection cost at stage* t*.*

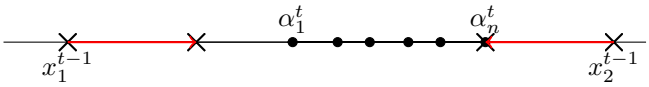
Put facility 1 *to the median of* O_1 *and facility* 2 *to the median of* O_2*.*

$x_1^t \leftarrow M_{O_1}$, $x_2^t \leftarrow M_{O_2}$

end

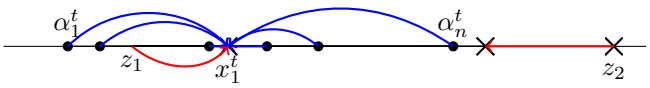


If both facilities 1 and 2 are on the left of the agents, then facility 2 is moved to the right until hitting the position of the leftmost agent (the case with facilities 1 and 2 on the right of agents is symmetric).

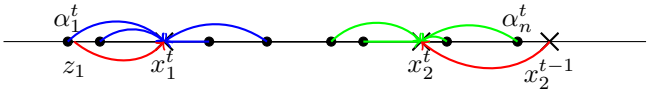


If facility 1 is on the left of the agents and facility 2 is on the right of the agents, then both facilities are moved with the same speed towards the interval $[\alpha_1^t, \alpha_n^t]$ until one of them hits the interval.

Figure 2: Step 1 of Algorithm 2 is depicted.



The first choice of Step 2 is depicted. In this case, the facility initially lying inside the interval $[\alpha_1^t, \alpha_n^t]$ moves to the median of agents. In this position, the connection cost is minimized using one facility.



The second choice of Step 2 is depicted. Facilities are placed to the positions, where the connection cost of the agents is minimized using two facilities.

Figure 3: Step 2 of Algorithm 2 is depicted.

Lemma 4.3 directly implies Theorem 4.1 by applying a telescopic sum over all t and then applying Lemma 4.2. Notice that the additive term $|x_1^0 - x_2^0|$ in Theorem 4.1 comes from the fact that $\Phi_0(x^0) = |x_1^0 - x_2^0|$.

In the rest of the section, we present the proof ideas of Lemma 4.3, which come together with explaining Steps 1 and 2 of our algorithm. In Step 1, note that since $x_1^0 \leq x_2^0$, then $x_1^t \leq x_2^t$ by our algorithm construction. Now, assume that $x_2^{t-1} \leq \alpha_1^t$ (second case). Before deciding the exact positions of the facilities, we can *safely* move facility 2 to the right until reaching α_1^t. The term *safely* means that this moving cost is *roughly* upper bounded by the moving cost $\sum_{k=1}^2 |y_k^t - y_k^{t-1}|$. This *safe moving* applies to all three cases of Step 1 in Algorithm 2 and is formally stated in Lemma 4.4.

Lemma 4.4 *Let $z = (z_1, z_2)$ denote the values of the variables z_1, z_2 after Step 1 of Algorithm 2. Then,*

$$\sum_{k=1}^{2} |z_k - x_k^{t-1}| \leq 2 \sum_{k=1}^{2} |y_k^t - y_k^{t-1}| - \Phi_t(z) + \Phi_{t-1}(x^{t-1})$$

Proof: Assume that $x_2^{t-1} \leq \alpha_1^t$. Then, Algorithm 2 will first move facility 2 to α_1^t ($z_1 = x_1^{t-1}, z_2 = \alpha_1^t$), paying moving cost equal to $|\alpha_1^t - x_2^{t-1}|$. This moving cost can be bounded with the use of the potential function Φ. More specifically, we have that $\Phi_t(z) - \Phi_t(x^{t-1}) + \Phi_t(x^{t-1}) - \Phi_{t-1}(x^{t-1}) \leq \Phi_t(z) - \Phi_t(x^{t-1}) + 2\sum_{k=1}^2 |y_k^t - y_k^{t-1}|$.

In case where $z_1 = x_1^{t-1}, z_2 = \alpha_1^t$, the difference $\Phi_t(z) - \Phi_t(x^{t-1})$ in the potential function equals $2(|y_2^t - \alpha_1^t| - |y_2^t - x_2^{t-1}|) + |x_1^{t-1} - \alpha_1^t| - |x_1^{t-1} - x_2^{t-1}|$. By the definition of solution y in Lemma 4.3, either y_1^t or y_2^t lies in the interval $[\alpha_1^t, \alpha_n^t]$. The latter implies that y_2^t is on the right of α_1^t. As a result we have that $\Phi_t(z) - \Phi_t(x^{t-1}) = -|z_2 - x_2^{t-1}|$, which completes the proof of Lemma 4.4 for this case of Step 1.

Notice that inequality (3) holds for all three cases of Step 1. Thus, one just need to prove that $\Phi_t(z) - \Phi_t(x^{t-1}) \leq -\sum_{k=1}^2 |z_k - x_k^{t-1}|$ for the other two cases. The case $x_1^{t-1} \geq \alpha_n^t$ is just symmetric. Since either either y_1^t or y_2^t is in $[\alpha_1^t, \alpha_n^t]$, it can be easily shown that $\Phi_t(z) - \Phi_t(x^{t-1}) \leq -\sum_{k=1}^2 |z_k - x_k^{t-1}|$ for the third case of Step 1. □

The proof of Lemma 4.4 reveals the basic idea in Step 1. According to the *geometry* of the agents' positions, we can identify a *safe move* whose cost is also paid by solution y for moving the servers. Moreover, this proof reveals why we compare our algorithm with the solution y and not directly with x^*. All these *safe moves* are based on the fact that either y_1^t or y_2^t lies in the $[\alpha_1^t, \alpha_n^t]$ (the latter does not necessarily hold for x^*). Finally, the *potential function* $\Phi_t(x_1, x_2)$ is crucial, since it permits *safe moves*, when all agents are on the right/left of the facilities (first/second case) as well as when they are contained in the interval $[x_1^{t-1}, x_2^{t-1}]$ (third case). This idea was developed for the K-server problem [Koutsoupias, 2009].

Up next, we analyze the ideas of Step 2. We now need to bound the connection cost plus some additional moving cost from the point where *the safe move stopped*.

Lemma 4.5 *Let $x^t = (x_1^t, x_2^t)$ denote the locations of facilities at stage t after the execution of Step 2. Then,*

$$\sum_{k=1}^{2} [H(C_{kt}) + |x_k^t - z_k|] \leq 21 \sum_{k=1}^{2} H(C_{kt}^*) - \Phi_t(x^t) + \Phi_t(z)$$

The proof of Lemma 4.5 is again based on the function $\Phi_t(x_1, x_2)$. The difficult case is when y served the agents using both facilities, since y's connection cost can be arbitrarily smaller than $H(C_t)$. The last case of Step 2 is easier, since Algorithm 2 pays minimum connection cost.

Acknowledgements

The names of the authors are in alphabetical order. Loukas Kavouras is partially supported by a scholarship from the State Scholarships Foundation, granted by the action "Scholarships Grant Programme for second cycle graduate studies", which is co-financed by Greece and the European Union (European Social Fund-ESF) through the Operational Programme "Human Resources Development, Education and Lifelong Learning 2014-2020". Stratis Skoulakis is partially supported by a scholarship of Onassis Foundation.

References

[An *et al.*, 2017] Hyung-Chan An, Ashkan Norouzi-Fard, and Ola Svensson. Dynamic facility location via exponential clocks. *ACM Trans. Algorithms*, 13(2):21:1–21:20, 2017.

[Bansal *et al.*, 2017] Nikhil Bansal, Marek Eliás, Lukasz Jez, and Grigorios Koumoutsos. The (h, k)-server problem on bounded depth trees. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1022–1037. SIAM, 2017.

[Bansal *et al.*, 2018] Nikhil Bansal, Marek Eliás, Lukasz Jez, Grigorios Koumoutsos, and Kirk Pruhs. Tight bounds for double coverage against weak adversaries. *Theory Comput. Syst.*, 62(2):349–365, 2018.

[Bartal and Koutsoupias, 2000] Yair Bartal and Elias Koutsoupias. On the competitive ratio of the work function algorithm for the k-server problem. In *Proceedings of 17th Annual Symposium on Theoretical Aspects of Computer Science*, pages 605–613, 2000.

[Betzler *et al.*, 2013] Nadja Betzler, Arkadii Slinko, and Johannes Uhlmann. On the computation of fully proportional representation. *J. Artif. Intell. Res.*, 47:475–519, 2013.

[Blanchard and Schabanel, 2017] Nicolas K. Blanchard and Nicolas Schabanel. Dynamic sum-radii clustering. In *WALCOM, volume 10167 of Lecture Notes in Computer Science*, pages 30–41. Springer, 2017.

[Caragiannis *et al.*, 2016] Ioannis Caragiannis, Laurent Gourvès, and Jérôme Monnot. Achieving proportional representation in conference programs. In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 144–150. IJCAI/AAAI Press, 2016.

[Coester and Koutsoupias, 2019] Christian Coester and Elias Koutsoupias. The online k-taxi problem. *To appear in STOC 2019*, 2019.

[Coester *et al.*, 2017] Christian Coester, Elias Koutsoupias, and Philip Lazos. The infinite server problem. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 14:1–14:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[de Keijzer and Wojtczak, 2018] Bart de Keijzer and Dominik Wojtczak. Facility reallocation on the line. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 188–194. ijcai.org, 2018.

[de Vries *et al.*, 2007] Sven de Vries, Marc E. Posner, and Rakesh V. Vohra. Polyhedral properties of the k-median problem on a tree. *Math. Prog. Ser. A*, 110(2):261–285, 2007.

[Divéki and Imreh, 2011] Gabriella Divéki and Csanád Imreh. Online facility location with facility movements. *CEJOR*, 19(2):191–200, 2011.

[Drezner and Hamacher, 2002] Zvi Drezner and Horst W. Hamacher. *Facility location - applications and theory*. Springer, 2002.

[Eisenstat *et al.*, 2014] David Eisenstat, Claire Mathieu, and Nicolas Schabanel. Facility location in evolving metrics. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, volume 8573 of *Lecture Notes in Computer Science*, pages 459–470. Springer, 2014.

[Fiat *et al.*, 1990] Amos Fiat, Yuval Rabani, and Yiftach Ravid. Competitive k-server algorithms (extended abstract). In *31st Annual Symposium on Foundations of Computer Science*, pages 454–463, 1990.

[Fotakis, 2011] Dimitris Fotakis. Online and incremental algorithms for facility location. *SIGACT News*, 42(1):97–131, 2011.

[Friggstad and Salavatipour, 2011] Zachary Friggstad and Mohammad R. Salavatipour. Minimizing movement in mobile facility location problems. *ACM Trans. Algorithms*, 7(3):28:1–28:22, 2011.

[Gupta *et al.*, 2014] Anupam Gupta, Kunal Talwar, and Udi Wieder. Changing bases: Multistage optimization for matroids and matchings. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 563–575. Springer, 2014.

[Koutsoupias, 2009] Elias Koutsoupias. The k-server problem. *Computer Science Review*, 3(2):105–118, 2009.

[Lazic, 2011] Nevena Lazic. Message passing algorithms for facility location problems. PhD thesis. University of Toronto, 2011.

[Liberty *et al.*, 2016] Edo Liberty, Ram Sriharsha, and Maxim Sviridenko. An algorithm for online k-means clustering. In Michael T. Goodrich and Michael Mitzenmacher, editors, *Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments, ALENEX 2016, Arlington, Virginia, USA, January 10, 2016*, pages 81–89. SIAM, 2016.

[Meyerson, 2001] Adam Meyerson. Online facility location. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 426–431. IEEE Computer Society, 2001.