Reallocating Multiple Facilities on the Line

Dimitris Fotakis\textsuperscript{a}, Loukas Kavouras\textsuperscript{a,b,1,*}, Panagiotis Kostopanagiotis\textsuperscript{a}, Stratis Skoulakis\textsuperscript{c}, Nikos Zarifis\textsuperscript{d}, Philip Lazos\textsuperscript{e}

\textsuperscript{a}National Technical University of Athens
\textsuperscript{b}Athena Research Centre
\textsuperscript{c}Singapore University of Technology and Design
\textsuperscript{d}University of Wisconsin-Madison
\textsuperscript{e}Sapienza University of Rome

Abstract

We study the $K$-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, 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.

Keywords: Facility Reallocation, $K$-server problem, online optimization

*Corresponding author

Email addresses: fotakis@cs.ntua.gr (Dimitris Fotakis), lkavouras@corelab.ntua.gr (Loukas Kavouras), kostopanagiotis@corelab.ntua.gr (Panagiotis Kostopanagiotis), efratios@sutd.edu.sg (Stratis Skoulakis), Zarifis@wisc.edu (Nikos Zarifis), plazos@gmail.com (Philip Lazos)

1Funding: Supported by a scholarship from the State Scholarships Foundation, co-financed by Greece and the European Union (European Social Fund-ESF).
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 simple and natural model for industrial planning, network design, machine learning, data clustering and computer vision [1, 2, 3, 4]. In its most basic form, the input to a Facility Location setting is a metric space and the location of \( n \) agents. The designer then selects another point in the space to serve as the facility, with the objective being to minimize the sum of distances to the agents. The \( K \)-Facility Location problem, which is also widely known as the \( K \)-median problem, is a straightforward generalization where \( K \) facilities are used instead. The problem then becomes more reminiscent of clustering, with the added subtlety of also having to match facilities with agents.

The underlying assumption so far has usually been that the agent locations are known in advance. In the more traditional setting of city planning this does make sense, but need not be the case in general. For example, in many natural settings such as network design or providing targeted content given preference clusters, the ‘agent locations’ are not known in advance. Motivated by this fact, Meyerson [5] introduced online facility location problems, where agents arrive one-by-one and must be immediately and irrevocably assigned to a facility upon arrival. Moreover, the set of agent locations is revealed in either random or adversarial order. However, all connections are irrevocable and cannot be changed even if new facilities are added, reflecting the constraints of adding physical links in a network.

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 in similar versions by [6] and [7] has been a new tool to analyze the temporal aspects of such networks. In this time dependent variant of facility location, the underlying metric space can change over time. This can be perceived as agents moving through space, but is actually more flexible. Facilities cannot
be moved around however: amongst the set of available facilities, the algorithm 
can select a subset to open at a cost. Then, at each turn, the assignment of 
agents to facilities can be changed. However, this incurs a *switching cost* and 
the objective is to achieve the best tradeoff between the optimal connections of 
agents to facilities and the stability of solutions between consecutive timesteps. 
This switching cost can be interpreted differently depending on the situation 
modelled, but taking it into account can produce significantly different (and 
more ‘reasonable’ looking) clusters than statically finding the best assignment 
each turn.

While the previous models have addressed issues of dynamically maintaining 
a set of facilities (or clusters), the common theme has been that facilities cannot 
move. They can be dissolved and reinstated somewhere else, but the cost is 
usually measured in terms of active facilities at the end. Our contribution is to 
study such settings where facilities can also *move* around.

**Model and Motivation.** In this work, we study the *K*-facility reallocation 
problem on the real line, introduced in [8], where the case of one facility was 
studied thoroughly. In *K*-facility reallocation, *K* facilities are initially lying at 
points \((x_0^0, \ldots, x_K^0)\) of metric space \(\mathcal{M}\). There are *n* agents, also residing on the 
same metric space, that use the facilities for *T* consecutive days. Each day, every 
agent connects to the facility closest to their location and incurs a connection 
cost equal to this distance. Since the agents are free to move around on \(\mathcal{M}\) from 
day to day, the algorithm can also move the facilities accordingly, to keep the 
connection cost low. Naturally, moving a facility is not free, but costs a price 
equal to the distance traversed. 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 over all *T* days is minimized. In the online version of the problem, the 
positions of the agents at each stage \(t \leq T\) are revealed only after determining 
the locations of the facilities at stage \(t - 1\).

The motivating example considered in [8] for \(K = 1\) consists of a political 
party moving along the spectrum from left to right wing, in an attempt to please 
more voters. Extending to \(K\), this setting applies to clustering and advertising:
following [9] from Yahoo Labs, companies often have a limited number of slots to
suggest alternatives to users (such as ads or movie suggestions), given previously
collected data. The users’ preferences gradually change however and the limited
number of suggestions need to stay enticing, without appearing to have abruptly
adjusted to the new information.

The case for the real line was fully characterized by [8], who designed optimal
offline and online algorithms for the case of $K = 1$ and presented a dynamic
programming algorithm for $K \geq 1$ facilities with running time exponential in $K$.
Despite the practical significance and the interesting theoretical properties of
$K$-facility reallocation, its computational complexity and its competitive ratio
(for the online variant) are still hardly understood.

In terms of theory, the online part of this model bears a resemblance to the
celebrated $K$-server problem [10], from online algorithms. In the classical $K$-
server problem, at each turn one of the $K$ servers needs to move on the agents
positions. In $K$-facility reallocation this is not the case: even if no facility shares
a spot with the agent, the connection cost is their distance. Curiously, the
analysis of the online version of $K$-facility reallocation blends techniques from
$K$-server and clustering, which could prove insightful to both communities.

Contribution. In this work, we resolve the computational complexity of $K$-
facility reallocation on the real line and take a first step towards a full under-
standing 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 [8], that
is exponential in $K$. Our algorithm simply finds an optimal solution of a Linear
Programming relaxation (LP) for $K$-facility reallocation, which is an extreme
point of the LP polytope. This can be done in polynomial time. The main
technical contribution is showing that this optimal solution will be integral.

Theorem. The integrality gap of the offline $K$-Facility Reallocation problem
on the real line is 1. In particular, the optimal integral solution can be found in

4
time polynomial in $n$, $K$ and $T$.

Our second main result concerns the online version of the problem with $K = 2$ facilities. We start with the observation that the online $K$-facility reallocation problem with $K \geq 2$ facilities is a natural and interesting extension 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.

**Theorem.** For $K = 2$, there exists a $O(1)$-competitive algorithm for the $K$-Facility Reallocation on the real line.

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 exactly how 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 [10]. We hope that studying the case where $K > 2$ will lead to interesting techniques that could shed some light on online algorithms as well.

**Related Work.** $K$-facility reallocation problem can be seen as multi-winner election (or committee selection) problem in utilitarian voting with single peaked preferences, especially under the Chamberlin-Courant rule. Two papers related to such problems are [11] and [12]. These deal with selecting the best among many possible outcomes in order to maximize the agents’ utility. However, our setting is dynamic in the sense that the agents preferences change between stages, thus the goal is to minimize a social cost function over $T$ stages and we also have to take into account that the solution provided at each stage should
be close to the solution of the previous stage.

We can also 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, \cite{6, 13}. Other examples in this setting are the dynamic sum radii clustering \cite{14} and multi-stage optimization problems on matroids and graphs \cite{15}.

In \cite{16}, a mobile facility location problem was introduced, which can be seen as a one 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 from \( K \)-median problem.

Online facility location problems and variants have been extensively studied in the literature, see \cite{17} for a survey. In \cite{18}, an online model where facilities can be moved with zero cost was studied. Despite achieving a constant competitive ratio, this model has the drawback that a misplaced facility can be moved for free to an optimal point without causing a penalty for the initial error. To remove this obstacle, \cite{19} proposed and a model, where moving a facility incurs a cost proportional to the distance it has moved.

The online variant of the \( K \)-facility reallocation problem is closely related to the \( K \)-server problem, which is one of the most natural online problems. \cite{10} 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 \cite{20}. Other variants of the \( K \)-server problem include the \((H, K)\)-server problem \cite{21, 22}, the infinite server problem \cite{23} and the \( K \)-taxi problem \cite{24, 25}.

\cite{26} proposed an online clustering approach, which stems from Hierarchical Agglomerative Clustering\(^2\). A set of \( k \) clusters is maintained while data points are presented in online fashion. Clusters can be merged, making space for an extra cluster to be used for incoming data. Clusters cannot be split however:

\(^2\)In clustering applications the \( k \)-means objective is often studied, which is slightly different than the \( k \)-median (or sum-of-distances) objective in facility location.
this is both computationally expensive and would change the classification of preexisting data points, which is undesirable in hierarchical clustering. Also, the number of clusters is fixed from the start at \(k\) and it is impossible to 'buy' more of them. This was the first step towards Incremental Facility Location, first studied by [27]. In this version, additional facilities can be opened and pairs of facilities can be merged at any point. The number of facilities does not need to remain fixed throughout and the final cost paid depends only on the number of open facilities at the end. Using techniques from streaming algorithms, the authors presented a constant competitive algorithm for incremental facility location as well as incremental \(k\)-median clustering (also studied by [28]) using \(O(k)\) additional clusters.

The fast increasing volume of available data and the requirement for responsive services has led to yet another approach by [9], namely online clustering algorithms balancing the quality of the clusters with their rate of change over time. The model is semi-online, meaning there is some information such as the length of the stream and the total clustering cost of optimal \(k\)-means. The algorithm proposed achieves a polylog competitive ratio using a polylogarithmically larger fraction of clusters. The authors also present a similar algorithm for the purely online case.

2. Problem Definition and Preliminaries

In this section, we start by presenting \(K\)-facility reallocation problem via an example and then we move to the formal definitions of the concepts, which will appear frequently throughout the paper.

Consider a beach, where two ice cream vendors are to be located for the next three days. The beach is visited by ten customers for the next three days and these customers may change their location on the beach. Naturally, each customer wants to have an ice cream vendor close to him in order to buy ice cream. The goal is to minimize the total distance traveled by the customers plus the total distance traveled by the ice cream vendors. Figure 1 depicts the
The black dots are the customers, which appear in different locations throughout the days.

The previous example is an instance of the \textit{K-facility reallocation problem}, where the number of mobile facilities is two (the ice cream vendors), the number of stages is three (the days) and the number of agents is 10 (the customers). Moreover, the metric space is the real line (the beach) and the variant is the offline if we know all customer locations throughout the days at the beginning of the first day. The problem is online if we learn the customer positions of the next day only after we have served (located the ice cream vendors) the customers of the current day. The following definitions formalize these ideas.

\begin{definition}[\textit{K-facility reallocation problem}] We are given a tuple \( (x^0, C) \) as input. The \( K \) dimensional vector \( x^0 = (x^0_1, \ldots, x^0_K) \) 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 \( \alpha^i_t \) and \( C_t = (\alpha^i_1, \ldots, \alpha^i_t) \)
\end{definition}
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 $x^t = (x^t_1, \ldots, x^t_K)$ is a $K$ dimensional vector that gives the positions of the facilities at stage $t$ and $x^t_k$ is the position of facility $k$ at stage $t$. The cost of the solution $x$ is

$$\text{Cost}(x) = \sum_{t=1}^{T} \left[ \sum_{k=1}^{K} |x^t_k - x^{t-1}_k| + \sum_{i=1}^{n} \min_{1 \leq k \leq K} |\alpha^t_i - x^t_k| \right].$$

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

**Definition 3.** Let $c > 1$ be a real number and let $\text{ALG}(\sigma)$ be the cost of an online deterministic algorithm on a request sequence $\sigma$. The algorithm is called $c$-competitive if there exists a constant $b$ such that

$$\text{ALG}(\sigma) \leq c \cdot \text{OPT}(\sigma) + b$$

holds for any request sequence $\sigma$, where $\text{OPT}(\sigma)$ is the optimal offline algorithm which knows $\sigma$ in advance.

Observe that the competitive ratio of an online algorithm involves an additive constant $b$ in its definition. The additive constant may depend on the initial configuration of the problem (in our case the initial positions of the facilities) but not on the request sequence $\sigma$. This is the standard definition used in $K$-server like problems (see also [29] [30]).

### 3. Polynomial Time Algorithm

In this section, we present an algorithm which computes the optimal solution for $K$-facility reallocation on the line with running time polynomial in $n$, $T$ and
Our algorithm finds an optimal solution for the Linear Program relaxation (LP) of the $K$-facility reallocation, which is an extreme point of this LP polytope. This can be done in polynomial time. Then, we show that this extreme point corresponds to an integral solution for the $K$-facility reallocation problem.

3.1. Formulating the Integer Linear Program

We start by expressing the $K$-facility reallocation problem on the line as an Integer Linear Program. A first difficulty is that the positions on the real line are infinite. We remove this obstacle with help of Lemma 2 from [8], which we restate here.

**Lemma 3.1** (Lemma 2 in [8]). 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 an agent has appeared or a facility was initially lying. This allows us to focus only on solutions which place facilities on at most $K + Tn$ different positions on the real line. Lemma 3.1 provides an exhaustive search algorithm for the problem and is also the basis for the Dynamic Programming approach in [8]. We use this property of the optimal solution to formulate our Integer Linear Program.

More specifically, let $Pos = C_1 \cup \ldots \cup C_T \cup x^0$ denote the set of the finite positions of Lemma 3.1. This set can be equivalently represented 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 on the line 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 $t \in \{1,T\}$: $\zeta^t_{ij} = 1$ if, at stage $t$, agent $i$ connects to a facility located at node $j$, $f^t_{kj} = 1$ if, at stage $t$, facility $k$ is located at node $j$, $S^t_{kjl} = 1$ 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 2.

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 $\sum_{j \in V} f^t_{kj} = 1$ enforces each facility $k$ to be located at exactly one node $j$. The constraint $S^t_k = \sum_{j,l \in V} d(j,l) S^t_{kjl}$ 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 $t$ ($S^t_{kjl} = 1$ iff $f^t_{kl} = 1$ and $f^{t-1}_{kj} = 1$).

We remark that the values of $f^0_{kj}$ are determined by the initial positions of the facilities, which are given by the instance of the problem.

### 3.2. The optimal extreme point solution of the LP is integral

Our algorithm, described in Algorithm 1, simply finds an optimal extreme point for the relaxation of the Integer LP of Figure 2. The rest of this section is dedicated to prove that this extreme point is in fact integral, thus corresponds to an optimal solution for $K$-facility reallocation problem. Formally,

**Theorem 3.1.** Let $Z^*_{LP}$ denote an optimal fractional solution of the LP relaxation for the $K$-facility reallocation problem. Then, the same LP relaxation has $N$ integral solutions $\text{Sol}_p$, $1 \leq p \leq N$, such that

$$\sum_{p=1}^{N} l_p \cdot \text{Sol}_p = Z^*_{LP},$$
(1) \[
\min_{t=1}^{T} \sum_{i \in C} \sum_{j \in V} d(\text{Loc}(i, t), j) \zeta_{ij}^t + \sum_{k \in F} S_k^t
\]

\[
\sum_{j \in V} \zeta_{ij}^t = 1 \quad \forall i \in C, t \in \{1, T\}
\]

\[
\zeta_{ij}^t \leq c_j^t \quad \forall i \in C, j \in V, t \in \{1, T\}
\]

\[
c_j^t = \sum_{k \in F} f_{kj}^t \quad \forall j \in V, t \in \{1, T\}
\]

\[
\sum_{j \in V} f_{kj}^t = 1 \quad \forall k \in F, t \in \{1, T\}
\]

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

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

\[
\sum_{i \in V} S_{kjl}^t = f_{kj}^{t-1} \quad \forall k \in F, j \in V, t \in \{1, T\}
\]

\[
\zeta_{ij}^t, f_{kj}^t, S_{kjl}^t \in \{0, 1\} \quad \forall k \in F, j \in V, t \in \{1, T\}
\]

Figure 2: ILP formulation of \textit{K-facility reallocation problem}. 
where \( l_p \geq 0 \) and \( \sum_{p=1}^{N} l_p = 1 \). Thus, \( Z^*_L \) can be written as a convex combination of the integral solutions \( \text{Sol}_p \).

Theorem 3.1 is the main result of this section and it implies the optimality of our algorithm. Specifically, assume for contradiction that the optimal solution \( \text{Sol}^* \) returned by Algorithm 1 is fractional. Theorem 3.1 states that an optimal fractional solution for the LP relaxation for \( K \)-facility reallocation problem can be written as a convex combination of non trivial feasible solutions. This should also hold for \( \text{Sol}^* \), since it minimizes the value of the LP. But this is a contradiction, since \( \text{Sol}^* \) corresponds to an extreme point and therefore can not be expressed as a non trivial combination of feasible solutions. Thus, \( \text{Sol}^* \) is integral.

The proof of Theorem 3.1 is conducted in two steps and each step is exhibited in subsections 3.3 and 3.4 respectively. In the following subsection, we show that there exists an optimal integral solution for the LP relaxation of Figure 2 in the case, where the values of the LP variables satisfy the following assumption.

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

Although Assumption 1 is very restrictive and it is not generally satisfied, it is the key step for proving the optimality of Algorithm 1. In the upcoming subsections, \( c_j^t, \zeta_{ij}^t, f_{kj}^t, \zeta_{kj}^t, S_{lkj}^t, S_k^t \) will denote the values of these variables in an optimal fractional solution for the relaxation of the ILP (2).

### 3.3. Proving Theorem 3.1 under Assumption 1

Throughout this section, we suppose that Assumption 1 is satisfied; \( f_{kj}^t \) and \( c_j^t \) are either \( 1/N \) or 0, for some positive integer \( N \). If an optimal fractional solution meets these requirements, then we can prove that it can be written as a convex compilation of \( N \) integral solutions with the same cost.

**Definition 4.** \( V_t^+ \) denotes the set of nodes of \( P \) with a positive amount of facility \( (c_j^t) \) at stage \( t \),

\[ j \in V_t^+ \text{ if and only if } c_j^t > 0. \]
Algorithm 1: Algorithm for the offline case

**Data:** Given the initial positions $x^0 = \{x^0_1, \ldots, x^0_K\}$ of the facilities and the positions of the agents $C = \{C_1, \ldots, C_T\}$.

- Construct the path $P$ and the Integer Linear Program (2).
- Find an optimal extreme point for the relaxation of the Integer Linear Program (2).

We remind that since $c^t_j = 1/N$ or 0, $|V_t^+| = K \cdot N$. We also consider the nodes in $V_t^+ = \{Y_t^1, \ldots, Y_{K \cdot N}\}$ to be ordered from left to right. The goal of this subsection is to prove the following lemma.

**Lemma 3.2.** Let $Sol_p$ be the integral solution that at each stage $t$ places the $m$-th facility at the $(m-1)N + p$ node of $V_t^+$ i.e. $Y^t_{(m-1)N+p}$, where $1 \leq p \leq N$ and $Z_{LP}^*$ the optimal fractional solution for LP (2). Then,

$$\frac{1}{N} \sum_{p=1}^{N} \text{Cost}(Sol_p) = \text{Cost}(Z_{LP}^*).$$

Thus, $Z_{LP}^*$ can be written as a convex combination of the integral solutions $Sol_p$, $1 \leq p \leq N$ with coefficients $1/N$ or 0.

The term $m$-th facility refers to the ordering of the facilities on the real line according to their initial positions $\{x^0_1, \ldots, x^0_K\}$. The proof of Lemma 3.2 is quite technically complicated, however it is based on two intuitive observations about the optimal fractional solution.

**Observation 1.** The set of nodes each agent $i$ connects at stage $t$ are consecutive nodes of $V_t^+$. More precisely, there exists a set $\{Y^t_1, \ldots, Y^t_{\ell+N-1}\} \subseteq V_t^+$ such that

$$\sum_{j \in V} d(\text{Loc}(i,t),j)\zeta^t_{ij} = \frac{1}{N} \sum_{h=\ell}^{\ell+N-1} d(\text{Loc}(i,t),Y^t_h).$$

**Proof:** Let an agent $i$ that at some stage $t$ has $x^t_{iY^t_j} > 0, x^t_{iY^t_\ell} < 1/N$ and $x^t_{iY^t_h} > 0$ for some $j < \ell < h$. Assume that $\text{Loc}(i,t) \leq Y^t_\ell$ and to simplify
notation consider $x_\ell = x_{iY_t}, x_h = x_{hY_t}$. Now, increase $x_\ell$ by $\epsilon$ and decrease $x_h$ by $\epsilon$, where $\epsilon = \min(1/N - x_\ell, x_h)$. Then, the cost of the solution is decreased by $(d(\text{Loc}(i, t), h) - d(\text{Loc}(i, t), \ell))\epsilon > 0$, thus contradicting the optimality of the solution. The same argument holds if $\text{Loc}(i, t) \geq Y_t^\ell$. The proof follows since $\sum_{j \in V} \zeta_{ij} = 1$. □

Observation 2. Under Assumption 1, the $m$-th facility places amount of facility $f_{mj}^t = 1/N$ from the $(m-1)N + p$ to the $mN + p - 1$ node of $V_t^+$ i.e. to nodes $\{Y_t^m, \ldots, Y_t^{(m-1)N+p}, \ldots, Y_t^{mN+p-1}\}$.

Observation 2 serves in understanding the structure of the optimal fractional solution under Assumption 1. However, it will be not used in this form in the rest of the section. We use Lemma 3.3 instead, which is roughly a different wording of Observation 2.

Lemma 3.3. Let $S_k^t$ be the optimal fractional moving cost of facility $k$ at stage $t$. Then,

$$
\sum_{t=1}^T \sum_{k \in F} S_k^t = \frac{1}{N} \sum_{t=1}^T \sum_{j=1}^{K \cdot N} d(Y_j^{t-1}, Y_j^t).
$$

Proof: By Assumption 1, $c_{ij}^t = 1/N$ if $j \in V_t^+ = \{Y_1^t, \ldots, Y_K^t\}$ and 0 otherwise. Notice that the connection cost of the optimal fractional solution only depends on the variables $c_{ij}^t$. As a result, $f_{kj}^t, S_k^t, S_{kjl}^t$ must be the optimal solution of the following linear program.

minimize $\sum_{t=1}^T \sum_{k=1}^K S_k^t$

s.t. $\sum_{k \in F} f_{kj}^t = \frac{1}{N} \quad \forall j \in V_t^+, t \in \{1, T\}$

$\sum_{j \in V_t^+} f_{kj}^t = 1 \quad \forall k, t \in \{1, T\}$

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

$\sum_{j \in V_t^{t-1}} S_{kjl}^t = f_{kl}^t \quad \forall k \in F, l \in V_{t-1}^+, t \in \{1, T\}$

$\sum_{l \in V_t^+} S_{kjl}^t = f_{kj}^{t-1} \quad \forall k \in F, j \in V_{t-1}^+, t \in \{1, T\}$
Instead of proving that the minimum cost is \( \frac{1}{N} \sum_{t=1}^{T} \sum_{j=1}^{K \cdot N} d(Y_{j}^{t-1}, Y_{j}^{t}) \) for the LP above, we prove it for the LP’s more convenient relaxation that follows.

\[
\begin{align*}
\text{minimize} & \quad \sum_{t=1}^{T} \sum_{j \in V_{i}^{+}, l \in V_{i}^{+}} d(j, l)F_{jlt} \\
\text{s.t.} & \quad \sum_{t \in V_{i}^{+}} F_{jlt} = \frac{1}{N} \quad \forall j \in V_{i-1}^{+}, t \in \{1, T\} \\
& \quad \sum_{j \in V_{i}^{+}} F_{jlt} = \frac{1}{N} \quad \forall l \in V_{i}^{+}, t \in \{1, T\} 
\end{align*}
\]

(1)

It is easy to prove that the LP (1) is a relaxation of the first by setting \( F_{jlt} = \sum_{k \in F} S_{kjl} \). Moreover, the above LP describes a flow problem between the nodes \( V_{i}^{+} \), where \( F_{jlt} \) is the amount of flow going from node \( j \in V_{i-1}^{+} \) to node \( l \in V_{i}^{+} \) (see Figure 3).

We are ready for the final step of our proof. First, observe that \( F_{Y_{j}^{t-1}Y_{j}^{t}}^{t} \) is a feasible solution for the above LP since \( |V_{i-1}^{+}| = |V_{i}^{+}| = K \cdot N \). If we prove that this assignment minimizes the objective, then we are done. Assume that in the optimal solution, \( F_{Y_{j}^{t-1}Y_{j}^{t}}^{t} < 1/N \). Since \( \sum_{l \in V_{i}^{+}} F_{Y_{j}^{t-1}Y_{j}^{t}}^{t} = \frac{1}{N} \), there exists \( Y_{j}^{t} \) such that \( F_{Y_{j}^{t-1}Y_{j}^{t}}^{t} > 0 \). Similarly, by using the second constraint we obtain that \( F_{Y_{j}^{t-1}Y_{j}^{t}}^{t} > 0 \). Let \( \epsilon = \min(F_{Y_{j}^{t-1}Y_{j}^{t}}^{t}, F_{Y_{j}^{t-1}Y_{j}^{t}}^{t}) \). Observe that if we increase \( F_{Y_{j}^{t-1}Y_{j}^{t}}^{t}, F_{Y_{j}^{t-1}Y_{j}^{t}}^{t} \) by \( \epsilon \) and decrease \( F_{Y_{j}^{t-1}Y_{j}^{t}}, F_{Y_{j}^{t-1}Y_{j}^{t}}^{t} \) by \( \epsilon \), we obtain another feasible solution. The cost difference of the two solutions is \( D = \epsilon(d(Y_{j}^{t-1}, Y_{j}^{t}) + d(Y_{j}^{t-1}, Y_{j}^{t}) - d(Y_{j}^{t-1}, Y_{j}^{t}) - d(Y_{j}^{t-1}, Y_{j}^{t})) \). If we prove that \( D \) is non-negative, we are done. We show the latter using the fact that \( Y_{1}^{t-1} \leq Y_{j}^{t-1} \) and \( Y_{1}^{t} \leq Y_{j}^{t} \). More precisely,

- If \( Y_{1}^{t-1} \leq Y_{1}^{t} \) then \( D \geq 0 \) since \( Y_{1}^{t} \leq Y_{j}^{t} \).
- If \( Y_{1}^{t-1} \geq Y_{1}^{t} \) then \( D \geq 0 \) since \( Y_{1}^{t-1} \leq Y_{j}^{t-1} \).

Until now, we have shown that in the optimal solution, the node \( Y_{1}^{t-1} \) sends all of her flow to the node \( Y_{j}^{t} \). Meaning that \( Y_{1}^{t} \) does not receive flow by any other node apart from \( Y_{1}^{t-1} \). By repeating the same argument, it follows that in the optimal solution each node \( Y_{j}^{t-1} \) sends all of her flow to \( Y_{j}^{t} \). \( \square \)
Now, we will use Lemma 3.3 to prove Lemma 3.4, which shows that the fractional moving cost paid by $Z^*_{LP}$ is the average moving cost paid by the $N$ integral solutions $Sol_p, 1 \leq p \leq N$.

**Lemma 3.4.** Let $S^t_k$ be the moving cost of facility $k$ at stage $t$ in the optimal fractional solution and $MovingCost(Sol_p)$ the total moving cost of the facilities in the integral solution $Sol_p$. Then,

$$\frac{1}{N} \sum_{p=1}^{N} MovingCost(Sol_p) = \sum_{t=1}^{T} \sum_{k \in F} S^t_k,$$

The equation holds for $S^t_k = 1/N$ or $S^t_k = 0$.

**Proof:** By the definition of the solutions $Sol_p$ we have that:

$$\frac{1}{N} \sum_{p=1}^{N} MovingCost(Sol_p) = \frac{1}{N} \sum_{p=1}^{N} \sum_{t=1}^{T} \sum_{m=1}^{K} d(Y^t_{m-1} N+p, Y^t_{m-1} N+p)$$

$$= \frac{1}{N} \sum_{t=1}^{T} \sum_{m=1}^{K} \sum_{p=1}^{N} d(Y^t_{m-1} N+p, Y^t_{m-1} N+p)$$

$$= \frac{1}{N} \sum_{t=1}^{T} K N \sum_{j=1}^{T} d(Y^t_j, Y^t_j)$$

$$= \sum_{t=1}^{T} \sum_{k \in F} S^t_k.$$
where the last equality comes from Lemma 3.3.

Lemma 3.4 states that if we pick be uniformly at random one of the $N$ integral solutions $\{\text{Sol}_p\}_{p=1}^N$, then the expected moving cost that we will pay is equal to the moving cost paid by the optimal fractional solution. Interestingly, the same holds for the connection cost. This is formally stated in Lemma 3.5 and it is where Observation 1 comes into play.

**Lemma 3.5.** Let $\text{ConCost}_t^i(\text{Sol}_p)$ denote the connection cost of agent $i$ at stage $t$ in $\text{Sol}_p$. Then,

$$
\frac{1}{N} \sum_{p=1}^N \text{ConCost}_t^i(\text{Sol}_p) = \sum_{j \in V} d(\text{Loc}(i, t), j) \zeta_{tij}.
$$

The equation holds for $\zeta_{tij} = 1/N$ or $\zeta_{tij} = 0$.

**Proof:** We remind that by Assumption 1, $c_j$ is $1/N$ if $j \in V_i^+$ and 0 otherwise. As a result, in the optimal fractional solution, each agent $i$ finds the $N$ closest to $\text{Loc}(i, t)$ nodes of $V_i^+$ and receives a $1/N$ amount of service from each one of them. Let us call this set $N^t_i$. By Observation 1, the nodes in $N^t_i$ must be consecutive nodes of $V_i^+$ i.e. $N^t_i = \{Y^t_{l}, \ldots, Y^t_{l+N-1}\}$ and

$$
\sum_{j \in V} d(\text{Loc}(i, t), j) \zeta_{tij} = \frac{1}{N} \sum_{j=1}^{l+N-1} d(\text{Loc}(i, t), Y^t_j)/N.
$$

Since $\text{Sol}_p$ puts facilities in the positions $\{Y^t_{(m-1)N+1}\}_{m=1}^K$, there exists a unique node $Y^t_{l(p)} \in N^t_i$ in which $\text{Sol}_p$ puts a facility. $Y^t_{l(p)}$ is the closest node to $\text{Loc}(i, t)$ from all the nodes in which $\text{Sol}_p$ puts a facility. As a result, $\text{ConCost}_t^i(\text{Sol}_p) = d(\text{Loc}(i), Y^t_{l(p)})$. Now, summing over $p$ we get,

$$
\frac{1}{N} \sum_{p=1}^N \text{ConCost}_t^i(\text{Sol}_p) = \frac{1}{N} \sum_{p=1}^N d(\text{Loc}(i), Y^t_{l(p)})
= \sum_{j=l}^{l+N-1} d(\text{Loc}(i), Y^t_j)/N
= \sum_{j \in V} d(\text{Loc}(i, t), j) \zeta_{tij}.
$$
As already mentioned, the proof of Lemma 3.5 crucially makes use of Observation 1. Combining Lemma 3.4 and 3.5 we get that if we pick an integral solution \( \text{Sol}_p \) uniformly at random, the average total cost that we pay is \( Z_{LP}^* \), where \( Z_{LP}^* \) is the optimal fractional cost. More precisely,

\[
\frac{1}{N} \sum_{p=1}^{N} \text{Cost}(\text{Sol}_p) = \frac{1}{N} \sum_{p=1}^{N} [\text{MovingCost}(\text{Sol}_p) + \sum_{t=1}^{T} \sum_{i \in C} \text{ConCost}_t^i(\text{Sol}_p)] \\
= \sum_{t=1}^{T} \sum_{k=1}^{K} S_{k}^t + \sum_{i \in C} \sum_{j \in V} d(\text{Loc}(i,t),j) c_{ij}^t \\
= Z_{LP}^*.
\]

Since \( \text{Cost}(\text{Sol}_p) \geq \text{Cost}(Z_{LP})^* \), we have that \( \text{Cost}(\text{Sol}_1) = \cdots = \text{Cost}(\text{Sol}_N) = \text{Cost}(Z_{LP})^* \). Notice also that the above equations hold when the variables \( S_{k}^t, f_{k}^t, c_{ij}^t \) of \( Z_{LP}^* \) take the values 1/N or 0, therefore we can write \( Z_{LP}^* \) as a convex combination of the \( N \) integral solutions \( \text{Sol}_p \). This concludes the proof of Lemma 3.2.

### 3.4. Proving Theorem 3.1 in the General Case

In this subsection, we show that \( Z_{LP}^* \) can be written as a convex combination of integer solutions even without Assumption 1.

We will make use of Lemma 3.2 to prove Theorem 3.1. As already discussed, Assumption 1 is not satisfied in general by a fractional solution of the relaxation of LP (2). Each \( S_{k}^t, f_{k}^t, c_{ij}^t \) will be either 0 or \( A_{k}^t/N_{k}^t \), for some positive integers \( A_{k}^t, N_{k}^t \). However, each positive \( f_{k}^t \) will have the form \( B_{k}^t/N \), where \( N = \Pi S_{k}^t > 0 \). This is due to the constraint \( f_{k}^t = \sum_{j \in V} S_{k}^t \).

Now, consider the path \( P' = (V', E') \) constructed from path \( P = (V, E) \) as follows: Each node \( j \in V \) is split into \( KN \) copies \( \{j_1, \ldots, j_{KN}\} \) with zero distance between them. Consider also the LP (2), when the underlying path is \( P' = (V', E') \) and at each stage \( t \), each agent \( i \) is located to a node of \( V' \) that is a copy of \( i \)'s original location, \( \text{Loc}'(i,t) = \ell \in V' \), where \( \ell \in \text{Copies}(\text{Loc}(i,t)) \).
Although these are two different LP’s, they are closely related since a solution for the one can be converted to a solution for the other with the exact same cost. This is due to the fact that for all \( j, h \in V \), \( d(j, h) = d(j', h') \), where \( j' \in \text{Copies}(j) \) and \( h' \in \text{Copies}(h) \).

The reason that we defined \( P' \) and the second LP is the following: Given an optimal fractional solution of the LP defined for \( P \), we will construct a fractional solution for the LP defined for \( P' \) with the exact same cost, which additionally satisfies Assumption 1. Then, using Lemma 3.2 we can obtain \( N \) integral solutions for \( P' \) with the same cost. These integral solutions for \( P' \) can be easily converted to integral solutions for \( P \), thus proving the statement of Theorem 3.1.

Given the fractional positions \( \{f_{tij}\}_{t \geq 1} \) of the optimal solution of the LP formulated for \( P = (V, E) \), we construct the fractional positions of the facilities in \( P' = (V', E') \) as follows: If \( f_{kij}^t = B_{kij}/N \), then facility \( k \) puts a \( 1/N \) amount of facility in \( B_{kij} \) nodes of the set \( \text{Copies}(j) = \{j_1, \ldots, j_{KN}\} \) that have a 0 amount of facility. The latter is possible since there are exactly \( KN \) copies of each \( j \in V \) and \( c_j^t \leq K \) (that is the reason we required \( KN \) copies of each node). The values for the rest of the variables are defined in the proof of the following lemma.

Lemma 3.6. Let \( \{f_{kij}^t, c_{jt}^t, S_{kjt}, \zeta_{ijt}\}_{t \geq 1} \) the optimal fractional solution for the LP (1) in the path \( P \). Then, there exists a solution \( \{f_{kij}^t, c_{jt}^t, S_{kjt}, \zeta_{ijt}, S_{kt}^t\}_{t \geq 1} \) of the LP (1) in the path \( P' \) such that:

- its cost is \( Z_{LP}^* \)
- \( f_{kjt}^t = 1/N \) or 0, for each \( \ell \in V' \)
- \( c_{jt}^t = 1/N \) or 0, for each \( \ell \in V' \)
- \( c_j^t = \sum_{\ell \in \text{Copies}(j)} c_{jt}^t \), for each \( j \in V \)

Proof: First, we set values to the variables \( f_{kjt}^t \). Initially, all \( f_{kjt}^t = 0 \). We know that if \( f_{kjt}^t > 0 \), then it equals \( B_{kjt}/N \), for some positive integer \( B_{kjt} \). For
each such $f^t_{kj}$, we find $u_1, \ldots, u_{B^t_{kj}} \in \text{Copies}(j)$ with $f^t_{ku}\neq 0$. Then, we set
\[ f^t_{ku} = 1/N \text{ for } h = \{1, B^t_{kj}\}. \]
Since there are $KN$ copies of each node $j \in V$ and $\sum_{j \in V} f^t_{kj} \leq K$, we can always find sufficient copies of $j$ with $f^t_{ku} = 0$.

When this step is terminated, we are sure that conditions 2 and 3 are satisfied.

We continue with the variables $S^t_{kj\ell}$. Initially, all $S^t_{kj\ell} = 0$. Then, each positive $S^t_{kj\ell}$ has the form $B^t_{kj\ell}/N$. Let $B = B^t_{kj\ell}$ to simplify notation. We now find $B$ copies of $u_1, \ldots, u_B$ of $j$ and $v_1, \ldots, v_B$ of $\ell$ so that
\[
\begin{align*}
&\bullet f^t_{ku_1} = \cdots = f^t_{ku_B} = f^t_{kv_1} = \cdots = f^t_{kv_B} = 1/N. \\
&\bullet S^t_{ku_1} = \cdots = S^t_{ku_B} = S^t_{kv_1} = \cdots = S^t_{kv_B} = 0 \text{ for all } h \in V'.
\end{align*}
\]
We then set $S^t_{ku_1v_1} = \cdots = S^t_{ku_Bv_B} = 1/N$. Again, since $\sum_{\ell \in V} S^t_{kj\ell} = f^t_{kj}$ and $\sum_{j \in V} S^t_{kj\ell} = f^t_{kj\ell}$, we can always find $B^t_{kj\ell}$ pairs of copies of $j$ and $\ell$ that satisfy the above requirements. We can now prove that the movement cost of each facility $k$ is the same in both solutions.

\[
\begin{align*}
\sum_{j \in V} \sum_{\ell \in V} d(j, \ell)S^t_{kj\ell} &= \sum_{j \in V} \sum_{\ell \in V} d(j, \ell)B^t_{kj\ell}/N \\
&= \sum_{j \in V} \sum_{\ell \in V} \sum_{h \in \text{Copies}(j)} \sum_{h' \in \text{Copies}(\ell)} S^t_{khh'}d(h, h') \\
&= \sum_{j \in V'} \sum_{\ell \in V'} S^t_{kj\ell'}d(j', \ell').
\end{align*}
\]

The second equality follows from the fact that $h, h'$ are copies of $j, \ell$ respectively and thus $d(h, h') = d(j, \ell)$.

Finally, set values to the variables $\zeta^t_{ij}$ for each $j \in V'$. Again, each positive $\zeta^t_{ij}$ equals $B^t_{ij}/N$, for some positive integer. We take $B^t_{ij}$ copies of $j$, $u_1, \ldots, u_{B^t_{ij}}$ and set $x^t_{iu_1} = \cdots = x^t_{iu_{B^t_{ij}}} = 1/N$. The connection cost of each agent $i$ remains the same since

\[
\sum_{j \in V} d(\text{Loc}(i, t), j)\zeta^t_{ij} = \sum_{j \in V} d(\text{Loc}(i, t), j)B^t_{ij}/N = \sum_{j \in V} d(\text{Loc}(i, t), j) \sum_{j' \in \text{Copies}(j)} x^t_{ij'},
\]
\[ \sum_{j \in V} \sum_{j' \in \text{Copies}(j)} d(\text{Loc}'(i,t), j') x_{ij}^{t} \]
\[ = \sum_{h \in V'} d(\text{Loc}'(i,t), h) x_{ih}^{t}, \]

where the third equality holds since \( \text{Loc}'(i,t) \in \text{Copies}(\text{Loc}(i,t)) \). \( \square \)

To conclude this section, we explain how to show Theorem 3.1. We have shown in this subsection that an optimal fractional solution \( Z^*_P \) for the LP (2) in \( P \) corresponds to an optimal fractional solution \( Z^*_{P'} \) for the LP (2) in \( P' \) with the same cost. Since \( Z^*_{P'} \) satisfies Assumption [1] it can be written as a convex combination of integral solutions \( \text{Sol}_p, 1 \leq p \leq N \) with coefficients \( 1/N \) or 0 by Lemma 3.2. By Lemma 3.6, \( Z^*_{P} \) can be written as a convex combination of integral solutions with coefficients \( l_{\text{Sol}_p}/N \) or 0, where \( l_{\text{Sol}_p} \) is the number of solutions in path \( P \), which correspond to \( \text{Sol}_p \) (the last bullet in Lemma 3.6).

Since \( Z^*_P \) is an optimal fractional solution for the LP relaxation of the \( K \)-facility reallocation problem of Figure 2, we have shown Theorem 3.1.

4. A Constant Competitive Algorithm for the Online 2-Facility Reallocation Problem

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. We remind that in the online version of the problem, the positions of the agents at each stage \( t \) are revealed only after the online algorithm has determined the locations of the facilities at stage \( t - 1 \). The decisions of the online algorithm are irrevocable and its performance is evaluated using competitive analysis (see Section 2).

The online algorithm, denoted as Algorithm 2, is inspired by the double coverage algorithm proposed for the \( K \)-server problem [10]. The double coverage algorithm solves the \( K \)-server problem on the line optimally in the sense that it achieves a competitive ratio of \( K \), matching the lower bound for this problem.
This simple algorithm performs one of the following steps based on the relevant positions of the facilities and the single agent:

- If the agent is located between two facilities, then it moves these facilities with the same speed towards the agent until the closest one of them reaches the agent.

- Else, the closest facility is moved on the agent.

Notice that the double coverage algorithm moves at most 2 facilities towards the agent. This helps us design the first step of our online algorithm for 2-facility reallocation problem, which performs the following two basic steps.

In Step 1, facilities are initially moved towards the positions of the agents. This step is also performed by the double coverage algorithm with one major difference. That is, we now have \( n \) agents, which define the interval \([\alpha_{t1}, \alpha_{tn}]\). Thus, this step ends, when we reach the leftmost \((\alpha_{t1})\) or the rightmost \((\alpha_{tn})\) agent. 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. Note that this step is not performed if a facility is already inside the interval \([\alpha_{t1}, \alpha_{tn}]\) at the beginning of stage \( t \).

In Step 2, our algorithm determines the final positions of the facilities \(x_{t1}, x_{t2}\). After Step 1, at least one of the facilities is inside the interval \([\alpha_{t1}, \alpha_{tn}]\), 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 \([\alpha_{t1}, \alpha_{tn}]\), 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. We formalize how this choice is performed, introducing some additional notation.
Definition 5.

- $C_t = \{\alpha_1^t, \ldots, \alpha_n^t\}$ denotes the positions of the agents at stage $t$ ordered from left to right.

- If $C$ is a set of positions with $|C| = 2k, k \in \mathbb{N}_{>0}$, then $M_C$ denotes the median interval of the set, which is the interval $[\alpha_{n/2}, \alpha_{n/2+1}]$. 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 just one facility. That is $H(\emptyset) = 0$ and $H(C) = \sum_{\alpha \in C} |\alpha - M_C|$, in case $M_C$ is a single point. In case $M_C$ is an interval, then $H(C) = \sum_{\alpha \in C} |\alpha - b_\alpha|$, where $b_\alpha \in M_C$ and is the nearest point from $\alpha$.

- $C_{1t}^*$ (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 3.

With this notation, we are ready to present our algorithm for online 2-facility reallocation problem.

4.1. The Online Algorithm and a Near Optimal Solution

In this subsection, we present Algorithm 2 which can be seen as a generalization of the double coverage algorithm due to the following two reasons. First, it does not necessarily place a facility on the position of the agent, since it may connect him (or multiple agents) from a different position. Furthermore, the decisions made by the algorithm have also to take into account the connection cost incurred. Therefore, Step 2 of Algorithm 2 tries to achieve a balance between the moving cost and the connection cost in order to be competitive with the optimal offline solution.

We first mention that Algorithm 2 seems much more complicated than it really is (the first two cases are symmetric both in Step 1 and Step 2). In fact,
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,

\[
\text{Cost}(x) \leq 63 \cdot \text{Cost}(x^*) + |x_1^0 - x_2^0|,
\]

where \( x_1^0, x_2^0 \) are the initial positions of the facilities.

Although it is possible to improve the competitive ratio of Algorithm 2 by a much more technically involved analysis, we stress here that it is not possible to turn the result into any constant factor as the following result indicates.

**Lemma 4.1.** Every \( c \)-competitive deterministic algorithm for the \( K \)-facility reallocation problem can be turned to a \( c \)-competitive deterministic algorithm for the \( K \)-server problem.

**Proof.** Consider an instance \( I \) of the \( K \)-server problem with requests \( r_1, \ldots, r_T \) on the line. We construct an instance \( I' \) for the \( K \)-facility reallocation problem with 2 clients on the same metric with requests \( r_1', \ldots, r_T' \), where each \( r_i' \) is an 2-dimensional \( (n = 2) \) of the form \( r_t' = (r_t, r_t) \), \( 1 \leq t \leq T \). This essentially means that, at each round \( t \), 2 clients are requested on the location on the line, where the request of the \( K \)-server problem for round \( t \) lies.

Now assume that we run a \( c \)-competitive algorithm \( ALG' \) for the \( K \)-facility reallocation problem problem on the instance \( I' \). \( ALG' \) can be transformed to an algorithm \( ALG \) for the \( K \)-server problem as follows: Let \( f_t \) denote the facility in \( I' \), which is closer to \( r_t' \) than any other facility and let \( s_t \) be the corresponding server in \( I \). Then, this server moves on \( r_t \) to serve it and then returns to the position, where \( f_t \) lies (\( f_t \) may serve \( r_t' \) from a distance). All other servers are moved to the positions of their corresponding facilities.

Since the initial positions of facilities in \( I' \) and the servers in \( I \) are the same, all the servers will be on the same positions on the line as their corresponding facilities at the end of each round. Next, we analyze the costs paid by \( ALG \).
Algorithm 2: Selecting $x_t^1$ and $x_t^2$

Data: At stage $t \geq 1$ the new agent positions $C_t = \{\alpha_t^1, \ldots, \alpha_t^n\}$ arrive

Step 1: Moving the facilities towards the agents

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

if $z_1 > \alpha_t^n$ then
    move facility 1 to the left until it hits $\alpha_t^n$: $z_1 \leftarrow \alpha_t^n$
end

if $z_2 < \alpha_t^1$ then
    move facility 2 to the right until it hits $\alpha_t^1$: $z_2 \leftarrow \alpha_t^1$
end

if $z_1 < \alpha_t^1$ and $z_2 > \alpha_t^n$ then
    move facility 1 to the right and facility 2 to the left until a facility
    hits $[\alpha_t^1, \alpha_t^n]$:
    $z_1 \leftarrow z_1 + \min(|x_t^{t-1} - \alpha_t^1|, |x_t^{t-1} - \alpha_t^n|)$,
    $z_2 \leftarrow z_2 - \min(|x_t^{t-1} - \alpha_t^1|, |x_t^{t-1} - \alpha_t^n|)$
end

Step 2: Selecting the final position of the facilities

if $\alpha_t^1 \leq z_1 \leq \alpha_t^n$ and $z_2 - \alpha_t^n \geq 3H(C_t)$ then
    put facility 1 to the median of $C_t$: $x_t^1 \leftarrow M_{C_t}$
    move facility 2 to the left by $3H(C_t)$: $x_t^2 \leftarrow z_2 - 3H(C_t)$
end

if $\alpha_t^1 \leq z_2 \leq \alpha_t^n$ and $\alpha_t^1 - z_1 \geq 3H(C_t)$ then
    put facility 2 to the median of $C_t$: $x_t^2 \leftarrow M_{C_t}$
    move facility 1 to the right by $3H(C_t)$: $x_t^1 \leftarrow z_1 + 3H(C_t)$
end

else
    Compute the 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 $O_2$.
    $x_t^1 \leftarrow M_{O_1}, x_t^2 \leftarrow M_{O_2}$
end
and $ALG'$ at each round $t$. All servers except $s_t$ will move the same distance with their corresponding facilities, thus the cost paid for them is the same for $ALG$ and $ALG'$. Regarding $f_t$, assume that it moves $a$ and connects $r'_t$ from distance $b$. Then it pays $a + 2b$ overall to serve both clients. Additionally, the request $r'_t$ is at distance $a + b$ from $f_t$ at the start of round $t$. Then, $s_t$ will pay $a + b$ for moving $s_t$ on the corresponding request $r_t$ and then $b$ to move $s_t$ to the position of $f_t$. Therefore, the cost paid for all servers of $ALG$ at each round is the same with the cost paid for all facilities of $ALG'$. Let $OPT_I$ and $OPT_{I'}$ denote the optimal solutions of $I$ and $I'$ respectively. Since any feasible solution for $I$ is feasible for $I'$, we have that $OPT_{I'} \leq OPT_I$ and therefore $Cost(ALG) = Cost(ALG') \leq c \cdot Cost(OPT_{I'}) \leq c \cdot Cost(OPT_I)$.

The above result rules out the existence of a deterministic algorithm for the 2-facility reallocation problem on the line with competitive ratio lower than 2, since if such an algorithm exists then we can turn it to an algorithm for the 2-server problem with competitive ratio less than 2. But this is not possible, due to the lower bound of 2 for the 2-server problem.

The rest of the section is dedicated to provide a proof of Theorem 4.1. First, we present Lemma 4.2 that is a key component in the subsequent analysis and that reveals the real difficulty of the online 2-facility reallocation problem.

**Lemma 4.2.** Let the optimal solution be $x^*$ and let $C_{1t}^*, C_{2t}^*$ be the set of agents that connect at stage $t$ to facilities 1,2 respectively. Let the solution $y^t = (y^t_1, y^t_2)$ be defined as follows:

\[
y^t_k = \begin{cases} 
M_{C_{kt}^*} & \text{if } C_{kt}^* \neq \emptyset \\
x_k^t & \text{if } C_{kt}^* = \emptyset 
\end{cases}
\]

Then, the following inequality holds:

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

**Proof:** Since $\sum_{t=1}^{T} \sum_{k=1}^{2} H(C_{kt}^*) = \sum_{t=1}^{T} \sum_{k=1}^{2} \sum_{a \in C_{kt}^*} |x_k^t - a|$, we only
have to prove that

\[ \sum_{t=1}^{T} \sum_{k=1}^{2} |y_{tk}^t - y_{tk}^{t-1}| \leq 2 \sum_{t=1}^{T} \sum_{k=1}^{2} [H(C_{kt}^*) + |x_{kt}^t - x_{kt}^{t-1}|] \]

From the triangle inequality, we have that

\[ \sum_{t=1}^{T} \sum_{k=1}^{2} |y_{tk}^t - y_{tk}^{t-1}| \leq \sum_{t=1}^{T} \sum_{k=1}^{2} \left[ |y_{tk}^t - x_{kt}^t| + |y_{tk}^{t-1} - x_{kt}^{t-1}| + |x_{kt}^t - x_{kt}^{t-1}| \right] \]

If \( y_{tk}^t = x_{kt}^t \) and \( y_{tk}^{t-1} = x_{kt}^{t-1} \), then the right hand side of the inequality is simply the optimal moving cost, which is at most \( \text{Cost}(x^*) \). If \( y_{tk}^t \neq x_{kt}^t \) for \( k = 1, 2 \), namely when \( C_{kt}^* \neq \emptyset \), then we can bound the quantity \( \sum_{t=1}^{T} \sum_{k=1}^{2} |y_{tk}^t - x_{kt}^t| \) by the optimal connection cost. Since \( y_{tk}^t \) is the median agent (lies in the median interval of \( C_{kt}^* \) in the case \( |C_{kt}^*| = 2k \)) of \( C_{kt}^* \) in this case, we have that

\[ \sum_{t=1}^{T} \sum_{k=1}^{2} |y_{tk}^t - x_{kt}^t| \leq \sum_{t=1}^{T} \sum_{k=1}^{2} \sum_{a \in C_{kt}^*} |x_{kt}^t - a| \leq \sum_{t=1}^{T} \sum_{k=1}^{2} H(C_{kt}^*). \]

Since the same arguments hold in the case \( y_{tk}^{t-1} \neq x_{kt}^{t-1} \) for \( k = 1, 2 \), we have that:

\[ \sum_{t=1}^{T} \sum_{k=1}^{2} [|y_{tk}^t - x_{kt}^t| + |y_{tk}^{t-1} - x_{kt}^{t-1}|] \leq 2 \sum_{t=1}^{T} \sum_{k=1}^{2} H(C_{kt}^*). \]

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 \( \text{Cost}(x^*) \). The latter is formally stated in Lemma 4.3 and is the main result of this section.
Lemma 4.3. Let $x = \{x^t_1, x^t_2\}_{t \geq 1}$ be the solution produced by Algorithm 2.

Then, the cost paid by solution $x$ at stage $t$, $\sum_{k=1}^{2} |x^t_k - x^{t-1}_k| + \sum_{i=1}^{n} \min_{k=1,2} |x^t_i - \alpha^t_i|$, is at most

$$21 \sum_{k=1}^{2} [H(C^*_k) + |y^t_k - y^{t-1}_k|] + \Phi_t(x^t) - \Phi_{t-1}(x^{t-1}),$$

where $\Phi_t(x_1, x_2) = 2(|x_1 - y^t_1| + |x_2 - y^t_2|) + |x_1 - x_2|$. 

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 $|x^0_1 - x^0_2|$ in Theorem 4.1 depends only on the initial positions of the facilities and follows from the fact that $\Phi_0(x^0) = |x^0_1 - x^0_2|$. Note that the additive term is a constant independent from the request sequence (the client positions $C_t$). As the request sequence grows, the additive term becomes negligible, therefore it is common to define the competitive ratio of an online algorithm as in Section 2 [31, 29].

4.2. Bounding the Cost of the Online Algorithm

In this subsection, we present the proof ideas of Lemma 4.3 which come together with explaining Steps 1 and 2 of our algorithm. Let us start with explaining Step 1.

We remind that Step 1 is performed by Algorithm 2 if both facilities are outside the interval $C_t$ at the beginning of stage $t$. Then, either both facilities are on the same side of $C_t$ or one of them is on the left and the other on the right. Therefore, the online algorithm distinguishes between the three cases depicted in Figure 4 (We show 2 cases since the case with both facilities on the right of the agents is symmetric to the first). Notice that moving with the same speed towards the interval $[a^t_1, a^t_n]$ results to the same moving cost for both facilities; both facilities will move the distance of the facility which is closest to its closest agent.

The following lemma bounds the online cost paid after the execution of Step 1. First, note that since $x^0_1 \leq x^0_2$, then $x^t_1 \leq x^t_2$ by our algorithm construction. Now, assume that $x^t_2 \leq \alpha^t_1$ (second case). Before deciding the exact
positions of the facilities, we can \textit{safely} move facility 2 to the right until reaching \(\alpha_1^t\). The term \textit{safely} means that this moving cost is \textit{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.

\textbf{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,

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

\textbf{Proof:} Assume that \(x_2^{t-1} \leq \alpha_1^t\), then Algorithm 2 will first move facility 2 to \(\alpha_1^t\) \((z_1 = x_1^{t-1}, z_2 = \alpha_1^t)\), 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 \(\Phi\). More specifically, we have that:

\[
\Phi_t(z) - \Phi_{t-1}(x^{t-1}) = \Phi_t(z) - \Phi_t(x^{t-1}) + \Phi_t(x^{t-1}) - \Phi_{t-1}(x^{t-1})
\]

\[
= \Phi_t(z) - \Phi_t(x^{t-1}) + 2 \sum_{k=1}^{2} (|y_k^t - x_k^{t-1}| - |y_k^{t-1} - x_k^{t-1}|)
\]

\[
\leq \Phi_t(z) - \Phi_t(x^{t-1}) + 2 \sum_{k=1}^{2} |y_k^t - y_k^{t-1}|.
\]

In the considered case \(z_1 = x_1^{t-1}, z_2 = \alpha_1^t\), the difference \(\Phi_t(z) - \Phi_t(x^{t-1})\) in the potential function equals the quantity \(2(|y_2^t - \alpha_1^t| - |y_2^{t-1} - x_2^{t-1}|) + |x_1^{t-1} - \alpha_1^t| - |x_1^{t-1} - x_2^{t-1}|.\) By the definition of solution \(y\) in Lemma 4.3, either \(y_1^t\) or \(y_2^t\) lies in the interval \([\alpha_1^t, \alpha_n^t]\). Since either \(y_1^t\) or \(y_2^t\) lies in the interval \([a_1^t, a_2^t]\) and \(y_1^t \leq y_2^t\), we have that \(a_1^t \leq y_2^t.\) Meaning that \(z_2\) is closer to \(y_2^t\) than \(x_2^{t-1}\) and consequently \(2(|y_2^t - \alpha_1^t| - |y_2^{t-1} - x_2^{t-1}|) = -2|x_2^{t-1} - a_1^t|\). Therefore,

\[
\Phi_t(z) - \Phi_t(x^{t-1}) = -2|x_2^{t-1} - a_1^t| + |x_1^{t-1} - \alpha_1^t| - |x_1^{t-1} - x_2^{t-1}| = -|\alpha_1^t - x_2^{t-1}| = -|z_2 - x_2^{t-1}|,
\]

which completes the proof of Lemma 4.4 for this case of Step 1.

Notice that inequality (2) 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. We prove it for the third case of Step 1, since the second case \((x_1^{t-1} \geq \alpha_1^t)\) is just symmetric to the first case.
that either $x_1$ or $x_2$ moves until one of them hits the interval $[a_1^t, a_n^t]$.

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 $[a_1^t, a_n^t]$ until one of them hits the interval.

\[ \text{Figure 4: Step 1 of Algorithm 2 is depicted. After this step, the positions of the facilities are denoted by } z_1, z_2 \text{ in Algorithm 2.} \]

In the third case of Step 1, we have that $x_1^{t-1} < a_1^t$, $x_2^{t-1} > a_n^t$, $z_1 = x_1^{t-1} + \min(|x_1^{t-1} - a_1^t|, |x_2^{t-1} - a_n^t|)$ and $z_2 = x_2^{t-1} - \min(|x_1^{t-1} - a_1^t|, |x_2^{t-1} - a_n^t|)$.

The difference $\Phi_i(z) - \Phi_i(x^{t-1})$ in the potential function equals the quantity $2(|z_1 - y_1^t| - |x_1^{t-1} - y_1^t| + |z_2 - y_2^t| - |x_2^{t-1} - y_2^t|) + |z_1 - z_2| - |x_1^{t-1} - x_2^{t-1}|$. Now, $|z_1 - z_2| - |x_1^{t-1} - x_2^{t-1}| = -2 \min(|x_1^{t-1} - a_1^t|, |x_2^{t-1} - a_n^t|) = -2 \sum_{k=1}^n |x_k - x_k^{t-1}|$. Assume that $y_1^t \in [a_1^t, a_n^t]$, then $2 \sum_{k=1}^n |z_k - y_k^t| - |x_k^{t-1} - y_k^t| \leq 0$ since $|z_1 - y_1^t| - |x_1^{t-1} - y_1^t| = -\min(|x_1^{t-1} - a_1^t|, |x_2^{t-1} - a_n^t|)$ and $|z_2 - y_2^t| - |x_2^{t-1} - y_2^t| \leq \min(|x_2^{t-1} - a_1^t|, |x_2^{t-1} - a_n^t|)$. As a result, inequality (2) holds. Using the same argument in case $y_2^t \in [a_1^t, a_n^t]$ completes the proof.

The proof of Lemma 4.4 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 $C_t = [a_1^t, a_n^t]$ (the latter does not necessarily hold for $x^*$). Finally, the potential function $\Phi_i(x_1, x_2)$ 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).

It is clear, that any reasonable algorithm will move at least one facility inside $C_t$ in order to serve the agents. Lemma 4.4 shows that this moving cost can
be charged to the difference $-\Phi_t(x^{t-1}) + \Phi_t(z)$ in the potential function. Now, we will show that we can charge the cost of the second step of Algorithm 2 to the difference $\Phi_t(x^t) - \Phi_t(z)$. In Step 2, we need to bound the connection cost plus some additional moving cost from the point where the safe move stopped. This is the step, where the final positions of the facilities are determined and also the most challenging one in the analysis, since Algorithm 2 has to decide whether it will serve the agents using both facilities (one of them will definitely serve some of the agents) without knowing what the optimal solution does.

Before we prove the guarantees of Step 2, we will explain the cases exhibited in this step in high level (see Figure 5). In the first case, where the second facility is close to $C_t$, the online algorithm serves the agents using both facilities. Then, Algorithm 2 will pay optimal connection cost and small moving cost, since the first facility is already inside $C_t$ and the second is close to $C_t$. In the second case, where the second facility is far from $C_t$, the agents are served with one facility and the second facility is moved by an appropriate distance towards $C_t$. In this case, the optimal connection cost can be arbitrarily smaller than the connection cost of our online algorithm. However, moving the second facility by an appropriate distance decreases $\Phi_t(x^t) - \Phi_t(z)$ so as to cancel the cost incurred by the online algorithm.

We are now ready to prove Lemma 4.5 which formalizes the guarantees provided by Algorithm 2 after the execution of Step 2. Algorithm 2 keeps a balance between the moving cost of the facilities and the connection cost in order to be competitive with the optimal solution as will become apparent from the analysis.

**Lemma 4.5.** Let $x^t = (x^t_1, x^t_2)$ denote the locations of facilities at stage $t$ after the execution of Step 2. Then,

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

**Proof:** Observe that by Algorithm 2 either $a^t_1 \leq z_1 \leq a^t_n$ or $a^t_1 \leq z_2 \leq a^t_n$. As a result, we need to prove the claim for the following 4 cases:
The first choice of Step 2 is depicted. In this case, the facility initially lying inside the interval \([a'_1, a'_n]\) 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.

We will prove just the first and the second case since the third is symmetric to the first and the forth is symmetric to the second. In case \(a_1 \leq z_1 \leq a_n\) and \(z_2 - a_n \geq 3H(C_t)\), Algorithm 2 puts facility 1 in the median of \(C_t\), namely \(x'_1 = M_{C_t}\) (or \(x'_1 \in M_{C_t}\) in case the number of agents is even), and moves facility 2 to the left by a distance of \(3H(C_t)\) as can be seen below.

First note that \(\sum_{k=1}^{2} H(C_{kt}) \leq H(C_t)\) since \(x'_1 \in M_{C_t}\). Then \(|x'_1 - z_1| \leq |a'_1 - a'_n| \leq H(C_t)\) because both \(x'_1\) and \(z_1\) lie in the interval \([a'_1, a'_n]\) and \(|x'_2 - z_2| = 3H(C_t)\).
3H(C_t) by Algorithm\(^2\) Therefore, we have that the cost of the online algorithm is \(\sum_{k=1}^{2} H(C_{kt}) + |x^t_k - z_k| \leq 5H(C_t)\). By the geometry of this case and the aforementioned bounds,

\[
\Phi_t(x^t) - \Phi_t(z) = 2 \sum_{k=1}^{2} (|x^t_k - y^t_k| - |z_k - y^t_k|) + |x^t_1 - x^t_2| - |z_1 - z_2|
\]

\[
\leq 2 \sum_{k=1}^{2} (|x^t_k - y^t_k| - |z_k - y^t_k|) - 2H(C_t).
\]

Because \(\sum_{k=1}^{2} [H(C_{kt}) + |x^t_k - z_k|] \leq 5H(C_t)\), bounding \(\sum_{k=1}^{2} (|x^t_k - y^t_k| - |z_k - y^t_k|)\) by the optimal connection cost \(\sum_{k=1}^{2} H(C^*_k)\) is now the challenge. The difficulty arises when \(C_{1t} \neq \emptyset\) and \(C_{2t} \neq \emptyset\), where \(\sum_{k=1}^{2} H(C^*_k)\) can be arbitrarily smaller than \(H(C_t)\). As we will see in this case (see also the figure below) \(x^t\) gets closer to \(y^t\) and the term \(\sum_{k=1}^{2} (|x^t_k - y^t_k| - |z_k - y^t_k|)\) becomes negative.

Since \(C^*_{2t} \neq \emptyset\) and \(y^t_2 \in M_{C^*_{2t}}\), we get that \(y^t_2 \leq a^t_1\) and as a result \(|x^t_2 - y^t_2| = |x^t_2 - z_2| - 3H(C_t)\).

\[
\Phi_t(x^t) - \Phi_t(z) \leq 2 \sum_{k=1}^{2} (|x^t_k - y^t_k| - |z_k - y^t_k|) - 2H(C_t)
\]

\[
= 2 (|x^t_1 - y^t_1| - |z_1 - y^t_1|) + 2 (|x^t_2 - z_2| - |z_2 - y^t_2|) - 2H(C_t)
\]

\[
\leq 2|x^t_1 - z_1| - 8H(C_t)
\]

\[
\leq 2H(C_t) - 8H(C_t)
\]

\[
\leq -6H(C_t)
\]

\[
\leq \sum_{k=1}^{2} H(C^*_k) - \sum_{k=1}^{2} |H(C_{kt}) + x^t_k - z_k|.
\]

Now, assume that \(C^*_{1t} = \emptyset\) or \(C^*_{2t} = \emptyset\) meaning that \(\sum_{k=1}^{2} H(C^*_k) = H(C_t)\). As
a result, bounding *everything* by \( H(C_t) \) serves our purpose. More formally,

\[
\Phi_t(x^t) - \Phi_t(z) \leq 2 \sum_{k=1}^{2} (|x^t_k - y^t_k| - |z_k - y^t_k|) - 2H(C_t)
\]

\[
\leq 2 \sum_{k=1}^{2} |x^t_k - z_k| - 2H(C_t)
\]

\[
\leq 6H(C_t)
\]

\[
\leq 11H(C_t) - \sum_{k=1}^{2} \left[ H(C_{kt}) + |x^t_k - z_k| \right]
\]

\[
= 11 \sum_{k=1}^{2} H(C^*_{kt}) - \sum_{k=1}^{2} \left[ H(C_{kt}) + |x^t_k - z_k| \right].
\]

The fourth inequality follows from the fact that \( \sum_{k=1}^{2} H(C_{kt}) + |x^t_k - z_k| \leq 5H(C_t) \).

We now need to treat the second case where \( a_1 \leq z_1 \leq a_n \) and \( z_2 - a_n < 3H(C_t) \). Since Algorithm 2 computes the optimal clustering \((C_{1t}, C_{2t})\) and puts \( x^t_1 \) in the interval \( M_{C_{1t}} \) and \( x^t_2 \) in the interval \( M_{C_{2t}} \), we are ensured that the connection cost of our solution is less than the connection cost of \( y^t \), \( \sum_{k=1}^{2} H(C_{kt}) \leq \sum_{k=1}^{2} H(C^*_{kt}) \), so we are mostly concerned in bounding \( \sum_{k=1}^{2} |x^t_k - z_k| \).

The easy case is when \( \sum_{k=1}^{2} H(C^*_{kt}) = H(C_t) \). A small difference with the previous case is that we don’t know how \( |x^t_2 - z_2| \) is. However, \( z_1, x^t_1, x^t_2 \in [a^1_1, \ldots a^t_n] \) and \( |x^t_2 - z_2| = |x^t_2 - a^t_n| + |a^t_n - z_2| \). Thus, \( |x^t_1 - z_1| + |x^t_2 - a^t_n| \leq H(C_t) \), \( |a^t_n - z_2| \leq 3H(C_t) \) and therefore \( \sum_{k=1}^{2} |H(C_{kt}) + |x^t_k - z_k|| \leq 5H(C_t) \). So we can again bound *everything* by \( H(C_t) \).

\[
\Phi_t(x^t) - \Phi_t(z) = 2 \sum_{k=1}^{2} (|x^t_k - y^t_k| - |z_k - y^t_k|) + |x^t_1 - x^t_2| - |z_1 - z_2|
\]
Things become more complicated, when the connection cost \( \sum C \) is relatively small (\( C|a| \)) and the agent at position \( x \) does not work. However, the solutions \( x^t \) and \( y^t \) will be relatively close in this case. More formally,

\[
\Phi_t(x^t) - \Phi_t(z) = 2 \sum_{k=1}^{2} \left( |x^t_k - y^t_k| - |z_k - y^t_k| \right) + \left| x^t_1 - x^t_2 \right| - \left| z_1 - z_2 \right|
\]

\[
= 2 \sum_{k=1}^{2} |x^t_k - y^t_k| - 2 \sum_{k=1}^{2} |z_k - y^t_k| + \sum_{k=1}^{2} |x^t_k - z_k|
\]

\[
= 2 \sum_{k=1}^{2} |x^t_k - y^t_k| + 2 \sum_{k=1}^{2} \left( |x^t_k - z_k| - |z_k - y^t_k| \right) - 2 \sum_{k=1}^{2} |x^t_k - z_k|
\]

\[
\leq 4 \sum_{k=1}^{2} |x^t_k - y^t_k| - 2 \sum_{k=1}^{2} |x^t_k - z_k|.
\]

We need to upper bound the distance \( \sum_{k=1}^{2} |x^t_k - y^t_k| \). Observe that in the solution \( x^t \), the agent at position \( a^t_1 \) connects to the left facility (facility 1) and the agent at position \( a^t_n \) connects to the right facility (facility 2). \( |x^t_1 - a^t_1| + |x^t_2 - a^t_n| \leq \sum_{k=1}^{2} H(C_{kt}). \) Since \( C_{1t} \neq \emptyset \) and \( C_{2t} \neq \emptyset \), the same holds for the solution \( y^t \). As a result,

\[
\Phi_t(x^t) - \Phi_t(z) \leq 4 \sum_{k=1}^{2} |x^t_k - y^t_k| - \sum_{k=1}^{2} |x^t_k - z_k|
\]

\[
\leq 4 \left( |x^t_1 - a^t_1| + |y^t_1 - a^t_1| + |x^t_2 - a^t_n| + |y^t_2 - a^t_n| \right) - \sum_{k=1}^{2} |x^t_k - z_k|
\]

\[
\leq 4 \sum_{k=1}^{2} \left[ H(C_{kt}) + H(C^*_kt) \right] - \sum_{k=1}^{2} |x^t_k - z_k|
\]
This completes the proof of the performance of our online algorithm Algorithm \( 2 \) and concludes this section.

5. Conclusion and Open Problems

Regarding the offline variant of the \( K \)-facility reallocation, we have resolved its computational complexity, when the metric space is the real line. Therefore, it would be interesting to consider the problem in general metric spaces. Since \( K \)-facility reallocation problem is essentially a dynamic \( K \)-median problem, a main open problem is to design approximation algorithms for this problem as well as to find lower bounds on the approximation ratio of any offline algorithm in general metric spaces. Approximation algorithms have also been designed for other dynamic problems in general metric spaces like the dynamic facility location problem \([13], [6]\) and the dynamic sum radii clustering problem \([14]\).

Turning to the online variant, the main question arising is to design a competitive online algorithm for online \( K \)-facility reallocation problem problem on the line. We have taken the first step towards a full understanding of the competitive ratio for this variant by solving the variant with two facilities. However, the variant with any number of facilities, which is a natural extension of the \( K \)-server problem, seems much more intriguing. For the online \( K \)-facility reallocation problem, we only know a lower bound of \( K \), coming from the \( K \)-server problem and a naive online algorithm, which has \( Kn \) competitive ratio. So, a natural question here is whether the lower bound can be improved or the dependence on \( n \) can be removed from the upper bound. Concluding, it would be also interesting to consider the randomized competitive ratio of the \( K \)-facility reallocation problem.

Acknowledgements

Loukas Kavouras is partially supported by a scholarship from the State Scholarships Foundation, granted by the action “Scholar-
ships 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 Onnasis Foundation. Philip Lazos is partially supported by the ERC Advanced Grant 788893 AMDROMA “Algorithmic and Mechanism Design Research in Online Markets” and the MIUR PRIN project ALGADIMAR “Algorithms, Games, and Digital Markets”.

References


URL https://doi.org/10.1613/jair.5340

URL https://doi.org/10.1613/jair.2566


URL \texttt{http://arxiv.org/abs/1807.06645}


URL \texttt{https://doi.org/10.1145/210118.210128}


URL \texttt{https://doi.org/10.1145/62212.62243}