Theory of Computing Systems (2021) 65:943–984
https://doi.org/10.1007/s00224-020-10023-8

# Managing Multiple Mobile Resources

**Björn Feldkord**[1] · **Till Knollmann**[1] · **Manuel Malatyali**[1] · **Friedhelm Meyer auf der Heide**[1]

Accepted: 7 December 2020 / Published online: 11 January 2021
© The Author(s) 2021

**Abstract**

We extend the Mobile Server problem introduced in Feldkord and Meyer auf der Heide (TOPC **6**(3), 14:1–14:17 2019) to a model where $k$ identical mobile resources, here named servers, answer requests appearing at points in the Euclidean space. To reduce communication costs, the positions of the servers can be adapted by a limited distance $m_s$ per round for each server. The costs are measured similarly to the classical Page Migration problem: i.e., answering a request induces costs proportional to the distance to the nearest server, and moving a server induces costs proportional to the distance multiplied with a weight $D$. We show that, in our model, no online algorithm can have a constant competitive ratio: i.e., one which is independent of the input length $n$, even if an augmented moving distance of $(1+\delta)m_s$ is allowed for the online algorithm. Therefore we investigate a restriction of the power of the adversary dictating the sequence of requests: We demand *locality of requests*: i.e., that consecutive requests come from points in the Euclidean space with distance bounded by some constant $m_c$. We show constant lower bounds on the competitiveness in this setting (independent of $n$, but dependent on $k$, $m_s$ and $m_c$). On the positive side, we

This article belongs to the Topical Collection: *Special Issue on Approximation and Online Algorithms (2019)*
Guest Editors: Evripidis Bampis and Nicole Megow

A preliminary version of this work appeared at WAOA'19.

✉ Björn Feldkord
bjoernf@hni.upb.de

Till Knollmann
tillk@hni.upb.de

Manuel Malatyali
malatya@hni.upb.de

Friedhelm Meyer auf der Heide
fmadh@hni.upb.de

[1] Heinz Nixdorf Institut & Computer Science Department, Paderborn University, Fürstenallee 11, D-33102 Paderborn, Germany

present a deterministic online algorithm with bounded competitiveness when an augmented moving distance and locality of requests is assumed. Our algorithm simulates any given algorithm for the classical $k$-Page Migration problem as guidance for its servers and extends it by a greedy move of one server in every round. The resulting competitive ratio is polynomial in the number of servers $k$, the ratio between $m_c$ and $m_s$, the inverse of the augmentation factor $1/\delta$ and the competitive ratio of the simulated $k$-Page Migration algorithm. We also show how to directly adapt the Double Coverage algorithm (Chrobak et al. SIAM J. Discrete Math. **4**(2), 172–181 1991) for the $k$-Server problem to receive an algorithm with improved competitiveness on the line.

**Keywords** Online algorithms · $k$-server problem · Resource augmentation

## 1 Introduction

Assume there is a provider offering a service which cannot be realized with a traditional cloud solution due to the high amount of induced network traffic. Instead of one centralized instance, the provider chooses to maintain multiple copies of the service which are located close to the end users such that data only has to be sent over a short distance. However, since the different instances might still have to exchange some data for coordination and maintaining an instance comes with a cost, the number of such instances should be limited. In conclusion, given a network topology, the provider wants to distribute $k$ copies of the service onto nodes of the topology such that they are close to the end user devices to guarantee short response times. Naturally, the user devices appear at different positions in the network over time and the future is unknown to the provider. Each request for the service is answered by the nearest copy. To account for the shifting user base, the provider can move a copy of the service. However, each time a copy is moved the respective service has to be stopped, migrated, and started again which results in the copy being inaccessible while being moved. Therefore, the provider wants to avoid moving a copy over long distances to guarantee high availability.

In a previous work [12], a subset of the authors considered the scenario with only one mobile resource. The scenario described above was modeled based on the classical Page Migration problem [8]: A single resource can be moved between two points $a$ and $b$ for costs $D \cdot d(a, b)$, where $d(a, b)$ is the distance between $a$ and $b$ and $D \geq 1$ is a constant. In every round a request appears at some point $r$, and if the current position of the resource is $p$, it is served for costs $d(p, r)$. The analysis is conducted in the standard framework of online algorithms and competitive analysis. This problem was extended to the Mobile Server problem in [12], which puts a limit on how much the resource (called server) can move in each time step reflecting the idea that the resource can only be moved locally to avoid congestion and have the service available again shortly after the decision to move it.

In our work, we extend the Mobile Server problem to multiple resources: We consider $k$ identical servers located in the Euclidean space (of arbitrary dimension). We use the Euclidean space as an abstraction from concrete topologies and as an

idealization of a fine grained network, where each router or base station is a candidate for holding a service instance. Each of the servers may move a distance of at most $m_s$ per time step. In each time step, a request appears which has to be served by one of the servers by the end of the time step. The cost function is the same as in the Page Migration problem described above, i.e., the cost for serving a request is equal to the distance to the nearest server and moving a server induces cost equal to the distance times some constant $D$. We evaluate our algorithms using competitive analysis, where the costs of an algorithm on an instance is compared to the optimal cost on the same instance. Formally, let $C_{Alg}(I)$ be the costs of an algorithm $Alg$ on an input $I$ and $C_{Opt}(I)$ the minimal possible cost on $I$. Algorithm $Alg$ is $c$-competitive, if $C_{Alg}(I) \leq c \cdot C_{Opt}(I) + a$ for all instances $I$, where $a$ is a constant independent of $I$. If $a = 0$, $Alg$ is strictly $c$-competitive. Our goal is to state strictly competitive online algorithms where the competitive ratio should not depend on the length of the input sequence.

### 1.1 Related Work

Besides being a direct extension of the Mobile Server problem [12], our work builds on and is related to results surrounding the $k$-Server and Page Migration problems. These problems have been examined in many variants and especially for the $k$-Server problem there are many algorithms for special metrics. In this overview we only focus on most relevant results for our problem, which are mostly algorithms with an (asymptotically) optimal competitive ratio.

In the classical $k$-Server problem as introduced by Manasse et al. [18], $k$ identical servers are located in a metric space and requests are answered by moving at least one of the servers to the point of the request. The associated costs are equal to the total distance moved. Manasse et al. showed that no online algorithm could be better than $k$-competitive on every metric with at least $k + 1$ points. They stated as the $k$-Server Conjecture that there is a $k$-competitive online algorithm for every metric space. Further, the conjecture is shown to hold for $k = 2$ and $k = n - 1$ where $n$ is the number of points in the metric space.

Since its introduction, many algorithms have been designed for special cases of the problem. Most notable is the Double-Coverage algorithm [11], which is $k$-competitive on trees. For general metrics, the best known result is the Work-Function algorithm, which is shown to be $2k - 1$-competitive [16]. Although this algorithm seems generally inefficient with respect to runtime and memory, there have been studies showing that an efficient implementation of this algorithm is indeed possible [19, 20]. It was also shown that the algorithm has an optimal competitive ratio of $k$ on line and star metrics, as well as metrics with $k + 2$ points [5]. Recently, an alternative upper bound of $n - 1$ was shown for the algorithm [22] which improves the results for metrics with less than $2k$ points.

The study of randomized online algorithms was initiated by Fiat et al. [13] who gave a $\log(k)$-competitive algorithm for the complete graph. It is speculated that this factor can be obtained for all metrics, however the question is still open. For general metrics, the first algorithm with polylogarithmic competitive ratio was an $\mathcal{O}(\log^3 n \cdot \log^2 k)$-competitive algorithm by Bansal et al. [3]. This was recently improved by

Bubeck et al. [10] who gave an $\mathcal{O}(\log^2 k)$-competitive algorithm for HSTs which can be turned into an $\mathcal{O}(\log^9(k) \cdot \log \log(k))$-competitive one for general metrics by a dynamic embedding of general metrics into HSTs [17].

Regarding the Page Migration problem [8] (also known as File Migration problem), most results focus on online algorithms which handle only a single page. Contrary to the $k$-Server problem, the design of such algorithms is not trivial for the Page Migration problem. To the best of our knowledge, the current best results are a 4-competitive deterministic algorithm by Bienkowski et al. [7] and a collection of randomized algorithms with competitive ratio of at most 3 by Jeffery Westbrook [21]. The most relevant results for our problem are two constructions by Bartal et al. [4] who give both a deterministic and a randomized algorithm which transform a given algorithm for the $k$-Server problem into a deterministic / randomized algorithm for the $k$-Page Migration problem. If the given $k$-Server algorithm is $c$-competitive, the deterministic algorithm is $\mathcal{O}(c^2)$-competitive, the randomized algorithm is $\mathcal{O}(c)$-competitive. Conversely, we use the resulting algorithms as a black box in our constructions.

For our problem, we consider the *locality of requests*, which is a variant where requests accessing the server are in close proximity. Similar variants have also been considered in traditional problems. The idea with these problems is that in practice, requests to memory by a program will often abide certain locality properties. Making these properties part of the model has the potential to lower the achievable competitive ratio and to bring the results for theory and practice closer together. Popular models benefiting from such locality are the List Update problem [1, 2] and the Paging problem [9, 14], which can be regarded as a special case of the $k$-Server problem.

### 1.2 Our Results & Outline of the Paper

In [12] it was already shown that no online algorithm for our problem can be competitive even on the real line and with just $k = 1$ server. As a consequence, we employ the following methods to derive bounds independent of the number of requests for the problem: On the one hand we apply *resource augmentation* as in [12]: i.e., we allow the online algorithm to use a maximum movement distance of $(1+\delta)m_s$. Other than in the case of $k = 1$, this is not enough to receive algorithms with a competitive ratio independent of time. Therefore, on the other hand, we restrict the adversary to the case with *locality of requests*: i.e., we introduce a parameter $m_c$ by which we can define families of instances classified by the maximum distance between two consecutive requests. We show that, for $k \geq 2$, both methods are needed to yield competitive bounds independent of the length of the instance. For $k = 1$, it was shown in [12] that a locality of requests can improve the competitiveness, but is not necessary to achieve a constant upper bound.

The parameters $m_c$ and $m_s$ have a crucial impact on the resulting competitiveness and thus separate simple from hard instances. We are able to show that these parameters seem to naturally describe the problem, since we can prove a lower bound of $\Omega(\frac{m_c}{m_s})$. For fast moving resources ($m_c < (1+\delta)m_s$), our algorithm has an almost optimal competitive ratio when given an optimal $k$-Page Migration

algorithm. For the case of slow moving resources ($m_c \geq (1+\delta)m_s$), we can achieve bounds independent of the length of the input stream. In detail, we obtain a bound of $\mathcal{O}(\frac{1}{\delta^4} \cdot k^2 \cdot \frac{m_c}{m_s} + \frac{1}{\delta^3} \cdot k^2 \cdot \frac{m_c}{m_s} \cdot c(\mathcal{K}))$, where $c(\mathcal{K})$ is the competitiveness of a given $k$-Page Migration algorithm. For the case $D = 1$, which we call the *unweighted problem*, the $k$-Page Migration algorithm can be replaced by a $k$-Server algorithm. Our results for the Euclidean space of arbitrary dimension are listed in Table 1. Note that the parameter $\varepsilon$ is indirectly given as the relative difference between $m_c$ and $m_s$. If $m_c < m_s$, then in the first row we have $\varepsilon > \delta$. Alternatively, if $\delta = 0$, this case still yields an almost optimal upper bound up to a factor of $1/\varepsilon$.

Finally, we construct an algorithm for the line based on the Double Coverage (DC) algorithm for the $k$-Server problem to demonstrate how direct implementations of algorithms, as opposed to the general simulation technique, can help to reduce the resulting competitive ratio.

The paper is structured as follows: A formal definition of our model can be found in Section 2. All relevant lower bounds are established in Section 3, showing that resource augmentation and locality of requests are necessary to obtain competitive algorithms. In terms of upper bounds, we first give an algorithm for the *unweighted problem* in Section 4. The analysis for instances with $m_c < (1+\delta)m_s$ consists of a simple potential function argument found in Section 4.1. The analysis of the other case is much more challenging and is conducted in Section 4.2. The weighted case ($D > 1$) is discussed in Section 5. While the basic approach stays the same, we need to modify the movement of the online algorithm due to the higher movement costs. We show how the algorithm can be adapted and present the resulting competitive ratio following a similar structure as in the unweighted case. Finally, the adaption of the DC algorithm for the line is presented and analyzed in Section 6.

## 2 Model & Notation

In this section we formally describe the model and some common notation used throughout the paper.

Time is considered discrete and divided into time steps $1, \ldots, n$. An input to the $k$-Mobile Server problem is given by a sequence of requests $r_1, \ldots, r_n$ where $r_t$ occurs in time step $t$ and is represented by a point in the Euclidean space of arbitrary dimension. We are given $k$ servers $a_1, \ldots, a_k$ controlled by our online algorithm. At each point in time, one server occupies exactly one point in the Euclidean space. We denote

**Table 1** An overview of the results, using the best known deterministic algorithms for $k$-Server / $k$-Page Migration

| | Lower bound | Unweighted ($D = 1$), Weighted ($D > 1$) Case |
|---|---|---|
| $m_c \leq (1+\delta-\varepsilon)m_s$ | $\Omega(k)$ | $\mathcal{O}(1/\varepsilon \cdot k)$, $\mathcal{O}(1/\varepsilon \cdot k^2)$ |
| $m_c \geq (1+\delta)m_s$ | $\Omega(k + \frac{m_c}{m_s})$ | $\mathcal{O}(1/\delta^4 \cdot k^3 \cdot \frac{m_c}{m_s})$, $\mathcal{O}(1/\delta^4 \cdot k^4 \cdot \frac{m_c}{m_s})$ |

The results in the first row also hold without resource augmentation when $m_c \leq (1-\varepsilon)m_s$

by $a_i^{(t)}$ the position of server $a_i$ at end of time step $t$, and by $d(a, b)$ the Euclidean distance between two points $a$ and $b$. For the distance between two servers $a_i^{(t)}$ and $a_j^{(t)}$ in the same time step $t$, we also use the notation $d_t(a_i, a_j)$. We may also leave out the time $t$ entirely if it is clear from the context.

In each time step $t$, the current request $r_t$ is revealed to the online algorithm. The algorithm may then move each server, such that $d(a_i^{(t-1)}, a_i^{(t)}) \leq m_s$ for all servers $a_i$. The movement incurs cost of $D \cdot \sum_{i=1}^{k} d(a_i^{(t-1)}, a_i^{(t)})$ for a constant $D \geq 1$. The request $r_t$ is then served by a closest server $a_i^{(t)}$, which incurs cost of $d(a_i^{(t)}, r_t)$. Note that the variables indexed with the time $t$ represent the configuration at the end of the time step $t$.

In our model, we consider the locality of requests dictated by a parameter $m_c$ limiting the distance between consecutive requests, i.e., $d(r_t, r_{t+1}) \leq m_c$. We will often refer to the distance which objects move within one time step as *speed*. We also consider a resource augmentation setting, where the maximum distance an online algorithm may move is in fact $(1 + \delta)m_s$ for some $\delta \in (0, 1)$. The cost of our online algorithm is denoted by $C_{Alg}$. We compare the costs of an online algorithm to an offline optimum, whose servers are denoted by $o_1, \ldots, o_k$ and whose cost is $C_{Opt}$.

## 3 Lower Bounds

In this section, we will prove lower bounds for the competitive ratio of our problem. They show the importance both of the resource augmentation and the locality of requests introduced above. All our lower bounds already hold on the line (and therefore in arbitrary dimensions, too). Since our model is an extension of the $k$-Page Migration problem, $\Omega(k)$ is a lower bound for deterministic algorithms inherited from that problem (which itself inherits the bound from the $k$-Server problem, see [4, 18]). Even when $m_c$ is restricted, the lower bound instance can simply be scaled down such that the distance limits are not relevant for the instance. We derive new bounds which hold even for randomized algorithms against oblivious adversaries (and therefore for deterministic algorithms as well).

We start by discussing the model without any restriction on the distance between the requests in two consecutive time steps, i.e., the parameter $m_c$ is unbounded. We also consider the case that there is no resource augmentation: i.e., the maximum movement distance of the online algorithm and of the offline solution are the same. The following lower bound, originally formulated for $k = 1$, carries over from [12]:

**Theorem 1** *Every randomized online algorithm for the Mobile Server problem (with $k = 1$) has a competitive ratio of $\Omega(\frac{\sqrt{n}}{D})$ against an oblivious adversary, where $n$ is the length of the input sequence.*

For more than one server, we obtain an additional bound which cannot be resolved with the help of resource augmentation. In the proofs of the following lower bounds, we use Yao's principle: we construct a probability distribution over inputs and show a lower bound for general deterministic online algorithms (on the same sequence).

According to the principle, the resulting competitive ratio then applies to randomized online algorithms against oblivious adversaries (who may generate sequences adapted to the concrete algorithm).

**Theorem 2** *For $k \geq 2$, every randomized online algorithm for the k-Mobile Server problem has a competitive ratio of at least $\Omega(\frac{n}{Dk^2})$ against an oblivious adversary, where n is the length of the input sequence.*

*Proof* We divide the line to the right of the starting point into $4(k-1)$ segments of size $x \cdot m_s$ each. The segments are divided into $k-1$ groups of size 4. Each group has two inner and two outer segments, where the outer segments neighbor segments of other groups. The adversary now chooses in each group one of the two inner segments uniformly at random. We refer to the middle point in each of the chosen segments as $Z_1, \ldots, Z_{k-1}$. During the first phase, $4kx$ requests appear at the starting point, and the adversary moves one server to $Z_1, \ldots, Z_{k-1}$ each, the last server remains at the starting point. The moving costs for the adversary are $\mathcal{O}(Dk^2 x \cdot m_s)$ (Fig. 1).

In the second phase, on each point $Z_1, \ldots, Z_{k-1}$, $\frac{x}{4}$ requests appear in order of distance to the starting point. If at the first time when a request appears on $Z_i$ the online algorithm does not have one server in the corresponding segment, then the costs for serving requests for the online algorithm are at least $\sum_{i=1}^{\frac{x}{4}} (\frac{x}{2} m_s - i \cdot m_s) = \Omega(x^2 m_s)$. Now we iterate over the groups of segments: Consider the group which contains $Z_1$. At the time of the first request on $Z_1$ the online algorithm either covers both, one or no inner segment of that group. In case of only one covered segment, $Z_1$ lies in the other inner segment with probability $1/2$. Consider a server in one of the inner segments: This server cannot move into a neighboring group within $x/4$ time steps. Hence we can regard the servers which cover inner segments as "used up" for the following groups and hence we may apply the arguments inductively. Let $a$, $b$ and $c$ the number of groups where the online algorithm covers both, one and no inner segment of that group respectively. We have $a + b + c = k - 1$, $2a + b \leq k$ and the expected number of segments for which the online algorithm incurs costs of $\Omega(x^2 m_s)$ are at least $c + \frac{1}{2}b$. It is easy to see that the number of these segments are in $\Omega(k)$: If $c \leq \frac{k}{4} - 1$, then $a + b \geq \frac{3}{4}k$ and hence $b \geq \frac{k}{4}$.

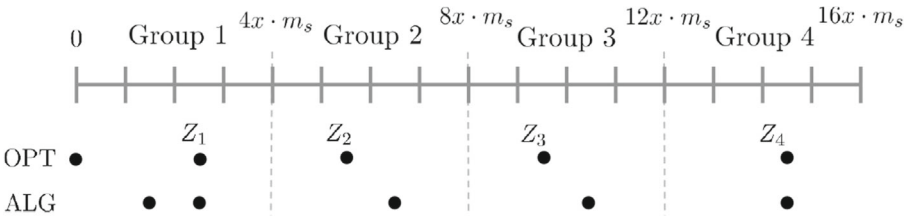


**Fig. 1** The line as used in the proof of Theorem 2. The circles indicate a possible configuration of the servers of the online algorithm and the optimal solution at the beginning of the second phase. The adversary has successfully chosen two segments which the online algorithm does not occupy

For the ratio we compare the costs and get $\frac{\Omega(kx^2 m_s)}{\mathcal{O}(Dk^2 x \cdot m_s)} = \Omega(\frac{x}{Dk}) = \Omega(\frac{n}{Dk^2})$. □

Note that the dependence on $n$ does not disappear in this bound, even if the online algorithm may move its servers a distance of $(1+\delta)m_s$ in each time step: By reducing the number of requests on each point to $\frac{x}{4(1+\delta)}$, the bound gets a term of $\frac{1}{1+\delta}$. This is not sufficient, since we want $\delta$ to be independent of $n$ and especially also be smaller than 1.

Since we often consider input sequences for problems such as ours to be potentially infinite, we deem competitive ratios dependent on the input length undesirable. Hence, as a consequence of the bounds shown so far, we apply two modifications to our model which help us to achieve a competitive ratio independent of the length of the input sequence. We use the concept of resource augmentation just as in [12] to allow the online algorithm to utilize a maximum movement distance of $(1+\delta)m_s$ for some $\delta \in (0, 1)$ as opposed to the distance $m_s$ used by the optimal offline solution. This measure alone does not address the bound from Theorem 2 (the ratio shrinks, but still depends on $n$). Hence, we introduce the locality of requests: We restrict the distance between two consecutive requests to a maximum distance of $m_c$. Note that only restricting the distance between consecutive requests does not remove the dependence on $n$ either, as was shown in [12]. The following theorem can be obtained in a similar way as Theorem 2:

**Theorem 3** *For $k \geq 2$, every randomized online algorithm for the k-Mobile Server problem, where the distance between consecutive requests is bounded by $m_c$, has a competitive ratio of at least $\Omega(\frac{m_c}{m_s})$ against an oblivious adversary.*

*Proof* We use a similar construction as in the proof of Theorem 2, but now divide the line to the right of the starting point into $5(k-1)$ segments of size $x \cdot m_s$ each. The segments are divided into $k-1$ groups of 5. Each group has three inner and two outer segments, where the outer segments neighbor segments of other groups. The adversary now chooses in each group one of the two inner segments, which neighbor an outer segment uniformly at random. We refer to the middle point in each of the chosen segments as $Z_1, \ldots, Z_{k-1}$. During the first phase, $5kx$ requests appear at the starting point, and the adversary moves one server to $Z_1, \ldots, Z_{k-1}$ each, the last server remains at the starting point. The moving costs for the adversary are $\mathcal{O}(Dk^2 x \cdot m_s)$ (Fig. 2).

In the second phase, on each point $Z_1, \ldots, Z_{k-1}$, $\frac{x}{4}$ requests appear in order of distance to the starting point, with requests in between to have a distance $m_c$ between requests, e.g., between $Z_1$ and $Z_2$, there will be $\frac{d(Z_1,Z_2)}{m_c}$ requests. The distance between two points $Z_i$ and $Z_{i+1}$ is at most $8xm_s$, hence the number of requests between them is at most $8x\frac{m_s}{m_c}$. The total cost for requests in this phase for the offline solution is therefore at most $(k-1) \cdot 8xm_s \cdot 8x\frac{m_s}{m_c} = \mathcal{O}(k\frac{x^2 m_s^2}{m_c})$.

The costs of the online algorithm can be bounded similarly as in the previous theorem: As there are two potential choices for each $Z_i$ which are chosen with probability $\frac{1}{2}$ each, $\Omega(k)$ of the chosen segments are initially uncovered when the first request occurs in the respective group of segments. Consider a server of the online algorithm

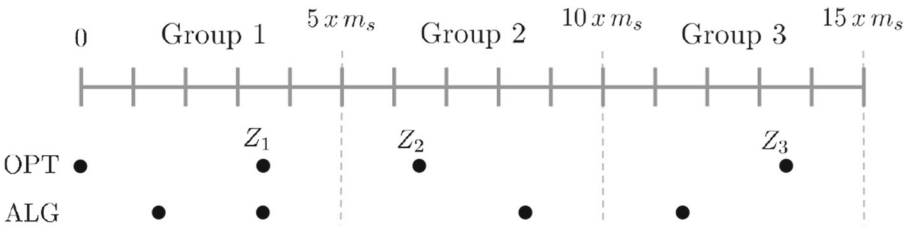


**Fig. 2** The line as used in the proof of Theorem 3. The circles indicate a possible configuration of the servers of the online algorithm and the optimal solution at the beginning of the second phase. The adversary has successfully chosen two segments which the online algorithm does not occupy

which initially covers one of the candidate segments for $Z_i$ but is then moved to cover a different candidate segment. Since there is at least one segment in between, the travel distance is at least $xm_s$. In order to cover the original segment, the online server cannot start moving until the first request occurs in the segment. From this point, there are at most $8x\frac{m_s}{m_c}$ time steps until the first request is in the next segment. This means that for $\frac{m_s}{m_c} \leq \frac{1}{8}$, the online server cannot cover requests in two segments over a distance smaller than $\frac{x}{2}$. The costs for the online algorithm are therefore $\Omega(kx^2m_s)$.

For the ratio we compare the costs and get $\frac{\Omega(kx^2m_s)}{\mathcal{O}(Dk^2x\cdot m_s+k\frac{x^2m_s^2}{m_c})} = \Omega(\frac{m_c}{m_s})$ for sufficiently large $x$. □

From [12], we get a lower bound of $\Omega(1/\delta)$ for $k = 1$ if $m_c \geq m_s$. This result can be extended for larger $k$ as well, using the $k$-dimensional space and adapting the technique accordingly.

## 4 An Algorithm for the Unweighted Problem

In this section we consider the unweighted problem ($D = 1$). Our algorithm does the following: We mainly follow around a simulated $k$-Server algorithm, but always move a closest server greedily towards the request. After formally introducing our algorithm, we will briefly argue why both of these ideas need to be part of it to achieve a competitive ratio independent of the input length.

We use the following notation in this section: Denote by $a_1, \ldots, a_k$ the servers of the online algorithm, $c_1, \ldots, c_k$ the servers of the simulated $k$-server algorithm and $o_1, \ldots, o_k$ the servers of the optimal solution. For an offline server $o_i$, we denote by $o_i^a$ the closest server of the online algorithm to $o_i$ (this might be the same server for multiple offline servers). Furthermore, we denote by $a^*$, $c^*$ and $o^*$ the closest server to the request of the algorithm, the $k$-server algorithm, and the optimal solution respectively. Ties for the closest servers can be broken arbitrarily. For a fixed time step $t$, we add a "$'$" to any variable to denote the state at the end of the current time step, e.g., $a_1 = a_1^{t-1}$ is the position of the server at the beginning of the time step and $a_1' = a_1^t$ is the position at the end of the current step.

Our algorithm *Unweighted-Mobile Servers (UMS)* works as follows:
Take any $k$-Server algorithm $\mathcal{K}$ with bounded competitiveness in the Euclidean space. Upon receiving the next request $r'$, simulate the next step of $\mathcal{K}$. Calculate a minimum weight matching (with the distances as weights) between the servers $a_1, \ldots, a_k$ of the online algorithm and the servers $c'_1, \ldots, c'_k$ of $\mathcal{K}$. There must be a server $c_i$ for which $c'_i = r'$. If the server matched to $c'_i$ can reach $r'$ in this turn, move all servers towards their counterparts in the matching with the maximum possible speed of $(1 + \delta)m_s$. Otherwise, select a server $\tilde{a}$ which is closest to $r'$ and move it to $r'$ with speed at most $(1 + \frac{\delta}{2})m_s$. All other servers move towards their counterparts in the matching with speed $(1 + \delta)m_s$.

In the second case of the algorithm, the server which moves greedily towards the request does so only a distance of $(1 + \frac{\delta}{2})m_s$. This is to guarantee that we always move more towards the simulated algorithm than away from it, and hence in a sense always catch up to it. We briefly want to discuss the fact that both the greedy move and the movement towards the simulated servers are necessary for a bounded competitiveness. For the classical $k$-Server problem, a simple greedy algorithm, which always moves the closest server onto the request has an unbounded competitive ratio. We can show that a simple algorithm which just tries to imitate any $k$-Server algorithm as best as possible is not successful either. Intuitively, the simulated algorithm can move many servers towards the request within one time step and serve the following sequence with them, while the online algorithm needs multiple time steps to get the corresponding servers in position due to the speed limitation. At the same time, the closest server of the online algorithm to the request might be matched to a server further away from the request, and hence it would move that server away from it.

Simple algorithm: Let $\mathcal{K}$ be any given $k$-Server algorithm. The $k$-Mobile Server algorithm does the following: Simulate $\mathcal{K}$. Compute a minimum weight matching (with the distances as weights) between the own servers and the servers of $\mathcal{K}$. Move every server towards the matched server at maximum speed.

**Theorem 4** *For $k \geq 2$, there are competitive $k$-Server algorithms such that the simple algorithm for the $k$-Mobile Server problem does not achieve a competitive ratio independent of $n$.*

*Proof* Consider the following instance: All servers and the request start at the same point on the real line. The requests moves $x$ times to the right by a distance of $m_s$ each. It then moves $y < \frac{x}{4}$ steps to the left again and remains at that point for the remaining $x - 2y$ time steps.

An optimal solution may be to just follow the request around with a single server which induces cost $(x+y)m_s$. Assume the $k$-Server algorithm does the following: As long as the request moves to the right, it gets served by a single server, the requests after that are served by a second server (this $k$-server algorithm would be at most 2-competitive in this instance). As a result, the online algorithm will move one server to the rightmost point in the sequence and then begin to move a second server towards

the request. When the request has reached its final position, the second server of the online algorithm has moved a distance of $ym_s$ to the right and hence it takes $x - 3y$ more time steps for it come closer than a distance of $ym_s$ to the request. The server of the online algorithm who followed the request initially to the rightmost point has now a distance of $ym_s$ to the request. It follows that the costs of the online algorithm are at least $xm_s + (x - 3y)ym_s$. By setting $y = \Theta(\sqrt{x})$, the competitive ratio becomes as large as $\Omega(\sqrt{n})$. □

The remainder of this section is devoted to the analysis of the competitive ratio of the UMS algorithm. In Section 4.1, we first consider the case that the distance between consecutive requests $m_c$ is smaller than the movement speed of the algorithm's servers. This case is easier than the case of slower servers since we can always guarantee that the online algorithm has one server on the position of the request. In the other case ($m_c \geq (1 + \delta)m_s$), described in Section 4.2, we need to extend our analysis to incorporate situations in which our online algorithm has no server near the request although the optimal offline solution might have such a server.

### 4.1 Fast Resource Movement

We first deal with the case that $m_c \leq (1 - \varepsilon) \cdot m_s$ for some $\varepsilon \in (0, 1)$. We show that we can achieve a result independent of the input length, even without resource augmentation. At the end of this section, we briefly discuss how to extend the result to incorporate resource augmentation: i.e., if the online algorithm has a maximum movement distance of $(1 + \delta)m_s$, we handle all cases with $m_c \leq (1 + \delta - \varepsilon) \cdot m_s$.

**Theorem 5** *If $m_c \leq (1-\varepsilon)\cdot m_s$ for some $\varepsilon \in (0, 1)$, the algorithm UMS is $2/\varepsilon \cdot c(\mathcal{K})$-competitive, where $c(\mathcal{K})$ is the competitive ratio of the simulated $k$-server algorithm $\mathcal{K}$.*

*Proof* We assume the servers adapt their ordering $a_1, \ldots, a_k$ according to the minimum matching in each time step. Based on the matching, we define the potential $\psi := \frac{2}{\varepsilon} \cdot \sum_{i=1}^{k} d(a_i, c_i)$. Note that the algorithm reaches the point of $r$ in each time step, and hence only pays for the movement of its servers, i.e., $C_{Alg} = \sum_{i=1}^{k} d(a_i, a_i')$. We assume that $c_1$ is on the request after the current time step, i.e., $c_1' = r'$.

First, consider the case that $a_1$ can reach $r'$ in this time step. Since each server moves directly towards their counterpart in the matching, we have

$$\begin{aligned} \Delta\psi &= \tfrac{2}{\varepsilon} \cdot \textstyle\sum_{i=1}^{k} d(a_i', c_i') - \tfrac{2}{\varepsilon} \cdot \sum_{i=1}^{k} d(a_i, c_i) \\ &\leq \tfrac{2}{\varepsilon} \cdot \textstyle\sum_{i=1}^{k} d(c_i, c_i') - \tfrac{2}{\varepsilon} \cdot \sum_{i=1}^{k} d(a_i, a_i') \\ &= \tfrac{2}{\varepsilon} \cdot C_{\mathcal{K}} - \tfrac{2}{\varepsilon} \cdot C_{Alg}. \end{aligned}$$

Now assume that $a_1$ cannot reach $r'$ in this time step. The server moves at full speed and hence $d(a_1', c_1') - d(a_1, c_1') = -m_s$. Now, let $a_2$ be the server which is at

range at most $m_c$ to $r'$ and does the greedy move possibly away from $c'_2$ onto $r'$. It holds $d(a'_2, c'_2) - d(a_2, c'_2) \leq m_c$. In total, we get

$$\begin{aligned}
\Delta\psi &\leq \tfrac{2}{\varepsilon}\left(\textstyle\sum_{i=1}^{k} d(a'_i, c'_i) - \sum_{i=1}^{k} d(a_i, c'_i)\right) + \tfrac{2}{\varepsilon}\textstyle\sum_{i=1}^{k} d(c_i, c'_i) \\
&\leq \tfrac{2}{\varepsilon}\left(d(a'_1, c'_1) - d(a_1, c'_1) + d(a'_2, c'_2) - d(a_2, c'_2)\right) \\
&\quad - \tfrac{2}{\varepsilon}\textstyle\sum_{i=3}^{k} d(a_i, a'_i) + \tfrac{2}{\varepsilon}\textstyle\sum_{i=1}^{k} d(c_i, c'_i) \\
&\leq -2m_s - \tfrac{2}{\varepsilon}\textstyle\sum_{i=3}^{k} d(a_i, a'_i) + \tfrac{2}{\varepsilon} \cdot C_{\mathcal{K}} \\
&\leq -\textstyle\sum_{i=1}^{k} d(a_i, a'_i) + \tfrac{2}{\varepsilon} \cdot C_{\mathcal{K}}.
\end{aligned}$$

□

We can extend this bound to the resource augmentation scenario, where the online algorithm may move the servers a maximum distance of $(1 + \delta) \cdot m_s$. When relaxing the condition appropriately to $m_c \leq (1 + \delta - \varepsilon) \cdot m_s$, we get the following result:

**Corollary 1** *If $m_c \leq (1 + \delta - \varepsilon) \cdot m_s$ for some $\varepsilon \in (0, 1)$, the algorithm UMS is $\frac{2\cdot(1+\delta)}{\varepsilon} \cdot c(\mathcal{K})$-competitive, where $c(\mathcal{K})$ is the competitive ratio of the simulated $k$-server algorithm $\mathcal{K}$.*

The proof works the same as above by replacing occurrences of $m_s$ by $(1 + \delta)m_s$ and changing the potential to $\frac{2\cdot(1+\delta)}{\varepsilon} \sum_{i=1}^{k} d(a_i, c_i)$.

At first glance, the result seems to become weaker with increasing $\delta$ if $\varepsilon$ stays the same. The reason is that by fixing the absolute distance $\varepsilon$ the relative difference $((1 + \delta)m_s - m_c)/m_s$ between $m_c$ and $(1 + \delta)m_s$ actually decreases: i.e., relatively speaking, $m_c$ gets closer to $(1 + \delta)m_s$. It can be seen that if instead we fix the value of $m_c$ and increase $\delta$, the value of $\varepsilon$ increases by the same amount and hence the competitive ratio tends towards $2 \cdot c(\mathcal{K})$.

### 4.2 Slow Resource Movement

This section considers the case $m_c \geq (1 + \delta)m_s$ and is structured as follows: To support our potential argument, we first introduce a transformation of the simulated $k$-Server algorithm which ensures that the simulated servers are always located near the request. We then introduce an abstraction of the offline solution, reducing it to the positioning of a single server $\hat{o}$ which acts as a reference point for a new potential function. The server $\hat{o}$ approximates the optimal positioning of the servers while at the same time obeying certain movement restrictions necessary in our analysis. Finally, we complete the analysis by combining the new derived potential function with the methods from the previous section.

#### 4.2.1 The k-Server Projection

Our goal is to transform a $k$-Page Migration algorithm $\mathcal{K}$ into a $k$-Page Migration algorithm $\hat{\mathcal{K}}$ which serves the requests of a $k$-Mobile Server instance such that all servers keep relatively close to the current request $r$. We formulate the projection

for general $D \geq 1$ as we will also use it in the next section. Note that using a $k$-Server algorithm for $\mathcal{K}$ also yields a $k$-Server algorithm for $\hat{\mathcal{K}}$, i.e., there will always be a server at the point of the request. For the case $m_c \geq (1+\delta)m_s$, we want our algorithm to use this projection as a simulated algorithm as opposed to a regular $k$-Server algorithm, hence we must ensure that this projection is computable online with the information available to our online algorithm. The servers of $\mathcal{K}$ are denoted as $c_1, \ldots, c_k$ and the servers of $\hat{\mathcal{K}}$ as $\hat{c}_1, \ldots, \hat{c}_k$.

We define two circles around the request $r$: The inner circle $inner(r)$ has a radius of $16kD \cdot m_c$ and the outer circle $outer(r)$ has a radius of $(32kD+1) \cdot m_c$. We will maintain $\hat{c}_i \in outer(r)$ for all $i$ for the entirety of the execution. The time is divided into phases, where the phase starting at time $t$ with the request at point $r_t$ ends on the smallest $t' > t$ such that $d(r_t, r_{t'}) \geq 16kD \cdot m_c$. During a phase the simulated servers move to preserve the following: If $c_i \in inner(r)$, then $\hat{c}_i = c_i$. At the end of the phase the servers move such that additionally, the following holds: If $c_i \notin inner(r)$, then $\hat{c}_i$ is on the boundary of $inner(r)$ such that $d(c_i, \hat{c}_i)$ is minimized. It is obvious that the definition of the algorithm guarantees $\hat{c}_i \in outer(r)$ for all $i$ at each point in time. Intuitively, the upper bound of $\mathcal{O}(k)$ for the factor between the new and previous algorithms stems from instances where the optimal algorithm will only have to send one server along with the request, while the transformed algorithm will always keep all $k$ servers nearby.

**Proposition 1** *For the servers $\hat{c}_1, \ldots, \hat{c}_k$ of $\hat{\mathcal{K}}$ it holds $d(\hat{c}_i, r) \leq (32kD+1) \cdot m_c$ during the whole execution. The costs of $\hat{\mathcal{K}}$ are at most $\mathcal{O}(k)$ times the costs of $\mathcal{K}$.*

*Proof* We define the following potential: $\phi = D \cdot \sum_{i=1}^{k} d(c_i, \hat{c}_i)$. During a phase, the potential decreases every time $\hat{c}_i$ moves to $c_i$ by $D$ times the amount $\hat{c}_i$ moves. Each time $c_i$ moves, $\phi$ increases by at most $D$ times the amount that $c_i$ moves. Let $c_i = c^*$ be a closest server of $\mathcal{K}$ to $r$. If $c_i \in inner(r)$, then $\hat{c}_i = c_i$ and hence the serving costs of the algorithms are the same. Otherwise, $c_i \notin inner(r)$, $\hat{c}_i \in outer(r)$ and hence the serving costs differ at most by a factor of 3.

We show that during each phase, $\mathcal{K}$ has costs of $\Omega(1) \cdot kD^2 \cdot m_c$. Consider the movement of the request from its starting point $r$ to the final point $r'$. We know that $d(r, r') \geq 16kD \cdot m_c$. Imagine drawing a straight line between $r$ and $r'$ and separating it into segments of length $m_c$ by hyperplanes orthogonal to the line. There are now at least $16kD$ such segments. Every server of $\mathcal{K}$ has two segments adjacent to its own. Denote the segments which do not contain a server of $\mathcal{K}$ and are not adjacent to a segment containing such a server *unoccupied segments*. Since there are $16kD$ segments in total and $k$ servers of $\mathcal{K}$, there are at least $(16D-3)k \geq 13kD$ unoccupied segments at the beginning of a phase. Since the maximum movement distance of $r$ is $m_c$, there is at least one request per segment.

The $k$ servers of $\mathcal{K}$ divide the unoccupied segments into at most $k+1$ many groups of segments right next to each other. We now analyze the cost of a group of size $x$. We only consider one half of the group and argue that the other half has at least the

same cost. Requests in the given $x/2$ segments can be served the following way: An adjacent server moves into the first $y$ segments and then serves the remaining $x/2-y$ segments over the distance. The costs incurred are at least $y \cdot Dm_c + \sum_{i=1}^{x/2-y} i \cdot m_c \geq y \cdot Dm_c + \frac{(x/2-y)^2}{2} \cdot m_c$. This term is minimized by setting $y = \frac{x}{2} - D$ which implies cost of at least $\frac{x}{2} Dm_c - \frac{D^2}{2} m_c$. No matter how the $13kD$ unoccupied segments are divided into $k+1$ groups, this gives a total cost of at least $\Omega(1) \cdot kD^2 m_c$.

We can now bound the costs at the end of the phase: The argument when $c_i \in inner(r)$ is the same as before. Otherwise, $\phi$ increases by at most $D \cdot d(\hat{c}_i, \hat{c}'_i) \leq 32kD^2 \cdot m_c$. This yields $C_{\mathcal{K}'} \leq \mathcal{O}(k) \cdot C_{\mathcal{K}}$. □

#### 4.2.2 The Offline Helper

We define a new offline server $\hat{o}$, which approximates the optimal position $o^*$ while managing the role change of $o^*$ in a smooth manner. By $\hat{a}$, we denote the server of the online algorithm with minimal distance to $\hat{o}$. For a formal description of the behavior, we need the following definitions:

- The inner circle $inner_t(o_i)$ contains all points $p$ with $d_t(o_i, p) \leq \frac{\delta^2}{48960k} \cdot d_t(o_i, o_i^a)$.
- The outer circle $outer_t(o_i)$ contains all points $p$ with $d_t(o_i, p) \leq \frac{\delta}{48} \cdot d_t(o_i, o_i^a)$.

Recall that $o_i^a$ is the server of the online algorithm closest to $o_i$. Abusing notation, we also refer to $inner_t(o_i)$ and $outer_t(o_i)$ as distances equal to the radius defined above.

This section is devoted to proving the following:

**Proposition 2** *There exists a virtual server $\hat{o}$ which moves at a speed of at most $(2+\frac{1020k}{\delta}) \cdot m_c$ per time step, for which $d(\hat{a}, \hat{o}) \leq 2 \cdot d(o^*, o^{*a}) + d(a^*, r)$ at all times, and for which the following conditions hold as long as $d_t(o^*, o^{*a}) \geq 2 \cdot 51483 \frac{km_c}{\delta^2}$:*

1. *If $r \in inner(o^*)$ at the end of the current time step, $\hat{o}$ moves at a maximum speed of $(1+\frac{\delta}{8})m_s$, i.e., $r_t \in inner_t(o^*) \Rightarrow d(\hat{o}^{(t-1)}, \hat{o}^{(t)}) \leq (1+\frac{\delta}{8})m_s$.*
2. *If $r \in inner(o^*)$ at the end of the current time step, then $\hat{o} \in outer(o^*)$ at the end of the current time step, i.e., $r_t \in inner_t(o^*) \Rightarrow \hat{o}^{(t)} \in outer_t(o^*)$.*

Before formally describing the algorithm for $\hat{o}$, we will give an intuition on the pattern and why it is useful for the analysis: In essence, $\hat{o}$ follows the point of the request $r$, but will always slow down when the request is near a server (in its inner radius) of the optimal solution. It will then not be able to match the exact position, but be within the outer radius of the same server. Due to the higher speed of $\hat{o}$, it can quickly catch up to $r$ once it leaves the area near the optimal server. This statement will help us in the analysis, since the online algorithm will close the distance towards $\hat{o}$ when it is near an optimal server. We choose the potential to be a function in the distance to $\hat{o}$ and hence can pay for the potentially high costs the algorithm has in such a step. When the request is not near an optimal server however, we can afford to pay into the potential since the optimal solution has high costs in such a step.

In the following, we show that it is possible to define a movement pattern for $\hat{o}$ in a way, such that invariants 1 and 2 of Proposition 2 hold as long as $d(o^*, o^{*a}) \geq 51483\frac{km_c}{\delta^2}$. Otherwise, $\hat{o}$ will simply follow $r$ and restore the properties once $d(o^*, o^{*a}) \geq 2 \cdot 51483\frac{km_c}{\delta^2}$. In order to describe the movement in detail, we introduce the concept of transitions.

In the input sequence and a given optimal solution, we define a *transition* between two steps $t_1 < t_2$, if there are $o_i,\ o_j$ such that $o_i = o^*$ and $r \in inner_{t_1}(o_i)$ at time step $t_1$ and $o_j = o^*$ and $r \in inner_{t_2}(o_j)$ at time step $t_2$. In between these two time steps, $r \notin inner(o^*)$. For such a transition, we define the transition time as $t^* := t_2 - t_1$. If $t^* > inner_{t_1}(o^*)/m_c + 2$, we call this a *long transition*. Otherwise, we call it a *short transition*. We say that $o_i$ *passes* the request after $t_1$ and $o_j$ *receives* the request in $t_2$. The concept is illustrated in Fig. 3.

The behavior of $\hat{o}$ can be computed as follows:

1. During a long transition between time steps $t_1$ and $t_2$, move with speed $d(\hat{o}^{(t-1)},\ \hat{o}^{(t)}) \leq (2 + \frac{1020k}{\delta}) \cdot m_c$ towards $r_t$ during steps $t_1 + 1$ to $t_2 - 2$. In the last two steps $t_2 - 1$ and $t_2$, move such that $\hat{o}^{(t_2-1)} = r_{t_2}$ at time $t_2 - 1$ and do not move in $t_2$ at all. Informally, $\hat{o}$ moves one step ahead of $r$ such that $\hat{o} = r$ after the transition, as soon as $r \in inner(o^*)$.
2. For a sequence of short transitions starting with $o^* = o_i$ in $t_1$, determine which of the following events terminating the current sequence occurs first:

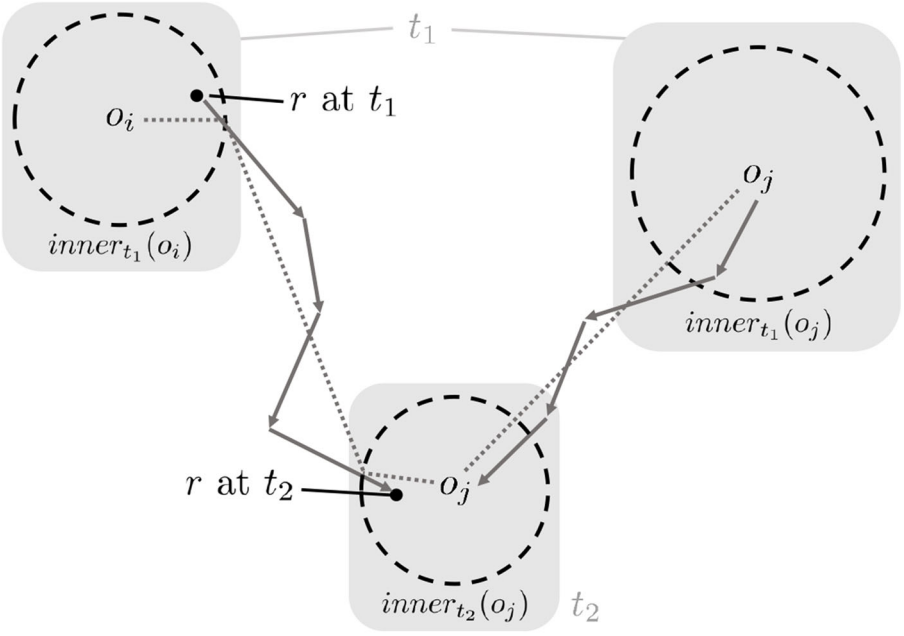


**Fig. 3** Example for a transition from $o_i$ to $o_j$. By definition, $r$ crosses the border of $inner(o_i)$ after time step $t_1$ ($o_i$ passes $r$ after $t_1$). The transition stops at step $t_2$ when $r$ has entered $inner_{t_2}(o_j)$ ($o_j$ receives $r$ in $t_2$). Note that $o_j$'s position and the radius of its inner circle may change from $t_1$ to $t_2$. The distance moved by $r$ is at most $(t_2 - t_1) \cdot m_c$. The dotted line represents the estimation of $d_{t_1}(o_i, o_j)$ used in Lemma 2

(a) A long transition from a server $o_\ell$ to $o_j$ between time $t_2$ and $t_3$ occurs. In this case, $\hat{o}$ simply moves towards $o_\ell^{(t)}$ in each step $t$ with speed at most $(1+\frac{\delta}{8})m_s$ until $t_2$.

(b) A short transition from a server $o_\ell$ to $o_j$ between time $t_2$ and $t_3$ occurs, where at one point prior in the sequence $d(o_j, o^*) > outer(o^*)/3$. If $\hat{o}$ can move straight towards the final position of $o_j$ in $t_3$ with speed $(1+\frac{\delta}{8})m_s$ without ever leaving $outer(o^*)$, then do that. Otherwise move towards a point $p$ with $d(p, o_\ell) = \frac{2\delta}{145} \cdot d(o_\ell, o_\ell^a)$. Among those candidates, $p$ minimizes $d(p, o_\ell^{(t_3)})$. When this point is reached, keep the invariant $d(\hat{o}, o_\ell) = \frac{2\delta}{145} \cdot d(o_\ell, o_\ell^a)$ whenever the final position of $o_j$ is not within $\frac{2\delta}{145} \cdot d(o_\ell, o_\ell^a)$ around $o_\ell$. The position of $\hat{o}$ on the circle around $o_\ell$ should be the one closest to $o_j$'s final position. When $o_j^{(t_3)}$ is inside the circle, the position of $\hat{o}$ should be equal to $o_j^{(t_3)}$.

3. If $d_{t_1}(o^*, o^{*a}) < 51483\frac{km_c}{\delta^2}$, treat the time until $d_{t_2}(o^*, o^{*a}) \geq 2 \cdot 51483\frac{km_c}{\delta^2}$ as a long transition between $t_1$ and $t_2$: i.e., move towards $r$ with speed $(2+\frac{1020k}{\delta}) \cdot m_c$ and skip one step ahead of $r$ during the last 2 time steps. (Steps 1 and 2 are not executed during this time.)

Note that the server $\hat{o}$ is a purely analytical tool and hence the behavior as described above does not have to be computable online.

Our goal is to show that all invariants described in Proposition 2 hold inductively over all transitions. We divide the entire timeline into sequences, where each sequence starts with both $r$ and $\hat{o}$ being in $inner(o^*)$. A sequence ends when one of the events stated in step 2 of the algorithm completes. The following lemma states that the initial condition is restored after every long transition.

**Lemma 1** *If $\hat{o} \in outer_{t_1}(o^*)$ at the beginning of a long transition between $t_1$ and $t_2$, then $\hat{o} \in inner_{t_2}(o^*)$ at the end of the transition.*

*Proof* During the transition time $t^* := t_2 - t_1$, $r$ moves a distance of at most $t^* \cdot m_c$. At the beginning, $\hat{o} \in outer_{t_1}(o^*)$ and $r \in inner_{t_1}(o^*)$, hence $d_{t_1}(\hat{o}, r) \leq d_{t_1}(\hat{o}, o^*) + d_{t_1}(o^*, r) \leq inner_{t_1}(o^*) + outer_{t_1}(o^*)$. During the first $\lceil inner_{t_1}(o^*)/m_c \rceil$ time steps, $\hat{o}$ can catch up to $r$ a distance of

$$\begin{aligned} \frac{inner_{t_1}(o^*)}{m_c} \cdot (1+\tfrac{1020k}{\delta}) \cdot m_c &= inner_{t_1}(o^*) + \tfrac{1020k}{\delta} \cdot inner_{t_1}(o^*) \\ &= inner_{t_1}(o^*) + outer_{t_1}(o^*) \end{aligned}$$

and therefore reaches $r$ (the speed of $\hat{o}$ is an additional $m_c$ higher which accounts for the movement of $r$). Since $t^* > inner_{t_1}(o^*)/m_c + 2$, there are at least 2 time steps remaining where $\hat{o}$ can move to the final position of $r$. □

Our next goal is to analyze a sequence of short transitions. During these transitions, $r$ moves faster than $\hat{o}$ and hence the distance of $\hat{o}$ to $o^*$ increases due to the role change after a transition. The next lemma establishes an upper bound on that increase.

Since we use the lemma in another context as well, the formulation is slightly more general.

**Lemma 2** *Every short transition between $o_i$ in step $t_1$ and $o_j$ in step $t_2$ can increase or reduce the distance of some server $s$, which moves at speed at most $(1+\delta)m_s$, to $o^*$ by at most*
$\min\{6.001 \cdot \frac{\delta^2}{48960k} \cdot d_{t_1}(o^*, o^{*a}) + 8.001m_c\ ,\ 6.002 \cdot \frac{\delta^2}{48960k} \cdot d_{t_2}(o^*, o^{*a}) + 8.002m_c\}$.

*Proof* We consider a short transition from offline server $o_i$ to $o_j$ in between time steps $t_1$ and $t_2$. By definition, $t^* = t_2 - t_1 \leq inner_{t_1}(o_i)/m_c + 2$.

We show that since $o_i$ and $o_j$ must be relatively close together, their distance to the closest server of the online algorithm must be similar. We first upper bound the distance between $o_i$ and $o_j$ in step $t_1$: The request travels a distance of at most $t^* \cdot m_c$ between the two. During this time, $o_j$ could have moved a distance of at most $t^* \cdot m_s$, and the inner radius could have changed by at most $t^* \cdot \frac{\delta}{16}m_s$. Since after the $t^*$ time steps $r$ enters the inner circle of $o_j$, we can use the above information to trace the distance between the two servers and the inner circle's radius of $o_j$ back to time step $t_1$ (see Fig. 3).

With this knowledge, we get

$$\begin{aligned}
d_{t_1}(o_j, o_j^a) &\geq d_{t_1}(o_i, o_i^a) - d_{t_1}(o_i, o_j) \\
&\geq d_{t_1}(o_i, o_i^a) - t^* \cdot (m_c + m_s + \tfrac{\delta}{16}m_s) \\
&\quad - inner_{t_1}(o_i) - inner_{t_1}(o_j) \\
&\geq d_{t_1}(o_i, o_i^a) - 3 \cdot inner_{t_1}(o_i) - inner_{t_1}(o_j) - 4m_c \\
&\geq (1 - 3 \cdot \tfrac{\delta^2}{48960k}) \cdot d_{t_1}(o_i, o_i^a) - \tfrac{\delta^2}{48960k} \cdot d_{t_1}(o_j, o_j^a) - 4m_c \\
\Leftrightarrow\ d_{t_1}(o_j, o_j^a) &\geq \tfrac{1-3\cdot\frac{\delta^2}{48960k}}{1+\frac{\delta^2}{48960k}} \cdot d_{t_1}(o_i, o_i^a) - \tfrac{4}{1+\frac{\delta^2}{48960k}} \cdot m_c \\
\Rightarrow\ d_{t_1}(o_j, o_j^a) &\geq (1 - \tfrac{4}{48960k+1}) \cdot d_{t_1}(o_i, o_i^a) - 4m_c.
\end{aligned}$$

In reverse, we can bound

$$\begin{aligned}
d_{t_1}(o_i, o_i^a) &\geq d_{t_1}(o_j, o_j^a) - d_{t_1}(o_i, o_j) \\
&\geq d_{t_1}(o_i, o_i^a) - 3 \cdot inner_{t_1}(o_i) - inner_{t_1}(o_j) - 4m_c \\
&\geq (1 - \tfrac{\delta^2}{48960k}) \cdot d_{t_1}(o_j, o_j^a) - \tfrac{3\delta^2}{48960k} \cdot d_{t_1}(o_i, o_i^a) - 4m_c \\
\Leftrightarrow\ d_{t_1}(o_i, o_i^a) &\geq \tfrac{1-\frac{\delta^2}{48960k}}{1+\frac{3\delta^2}{48960k}} \cdot d_{t_1}(o_j, o_j^a) - \tfrac{4}{1+\frac{3\delta^2}{48960k}} \cdot m_c \\
\Rightarrow\ d_{t_1}(o_i, o_i^a) &\geq (1 - \tfrac{4}{48960k}) \cdot d_{t_1}(o_j, o_j^a) - 4m_c.
\end{aligned}$$

Since $s$ can move away from $o_j$ during the transition and $o_j$ itself moves at speed at most $m_s$, we get

$$\begin{aligned}
d_{t_2}(s, o_j) &\leq d_{t_1}(s, o_j) + t^* \cdot (2+\delta)m_s \\
&\leq d_{t_1}(s, o_i) + d_{t_1}(o_i, o_j) + t^* \cdot (2+\delta)m_s \\
&\leq d_{t_1}(s, o_i) + t^* \cdot (m_c + m_s + \tfrac{\delta}{16}m_s) + inner_{t_1}(o_i) \\
&\quad + inner_{t_1}(o_j) + t^* \cdot (2+\delta)m_s \\
&\leq d_{t_1}(s, o_i) + 5 \cdot inner_{t_1}(o_i) + inner_{t_1}(o_j) + 8m_c \\
&\leq d_{t_1}(s, o_i) + 5 \cdot \tfrac{\delta^2}{48960k} \cdot d_{t_1}(o_i, o_i^a) + \tfrac{\delta^2}{48960k} \cdot d_{t_1}(o_j, o_j^a) + 8m_c.
\end{aligned}$$

To derive the first bound, we get

$$\begin{aligned} d_{t_2}(s,o_j) &\le d_{t_1}(s,o_i) + 5\cdot\tfrac{\delta^2}{48960k}\cdot d_{t_1}(o_i,o_i^a) + \tfrac{\delta^2}{48960k}\cdot\tfrac{1}{1-\frac{4}{48960k}}\cdot d_{t_1}(o_i,o_i^a) \\ &\quad +(8+\tfrac{\delta^2}{48960k}\cdot\tfrac{4}{1-\frac{4}{48960k}})\cdot m_c \\ &\le d_{t_1}(s,o_i) + 6.001\tfrac{\delta^2}{48960k}\cdot d_{t_1}(o_i,o_i^a) + 8.001 m_c. \end{aligned}$$

For the second bound, we continue with

$$\begin{aligned} d_{t_2}(s,o_j) &\le d_{t_1}(s,o_i) + 5\cdot\tfrac{\delta^2}{48960k}\cdot d_{t_1}(o_i,o_i^a) + \tfrac{\delta^2}{48960k}\cdot d_{t_1}(o_j,o_j^a) + 8m_c \\ &\le d_{t_1}(s,o_i) + (1+\tfrac{5}{1-\frac{4}{48960k+1}})\cdot\tfrac{\delta^2}{48960k}\cdot d_{t_1}(o_j,o_j^a) \\ &\quad +(8+5\cdot\tfrac{\delta^2}{48960k}\cdot\tfrac{4}{1-\frac{4}{48960k+1}})m_c \\ &\le d_{t_1}(s,o_i) + 6.001\cdot\tfrac{\delta^2}{48960k}\cdot d_{t_1}(o_j,o_j^a) + 8.001\cdot m_c. \end{aligned}$$

Next we bound the change in $d(o_j,o_j^a)$ during the transition:

$$\begin{aligned} d_{t_1}(o_j,o_j^a) &\le d_{t_2}(o_j,o_j^a) + t^*\cdot(2+\delta)m_s \\ &\le d_{t_2}(o_j,o_j^a) + 2\cdot inner_{t_1}(o_i) + 4m_c \\ &\le d_{t_2}(o_j,o_j^a) + 2\cdot\tfrac{\delta^2}{48960k}\cdot d_{t_1}(o_i,o_i^a) + 4m_c \\ &\le d_{t_2}(o_j,o_j^a) + 2.001\cdot\tfrac{\delta^2}{48960k}\cdot d_{t_1}(o_j,o_j^a) + 4.001 m_c \\ \Leftrightarrow\ d_{t_1}(o_j,o_j^a) &\le \tfrac{1}{1-2.001\cdot\frac{\delta^2}{48960k}}\cdot d_{t_2}(o_j,o_j^a) + 4.002 m_c. \end{aligned}$$

This gives us

$$\begin{aligned} d_{t_2}(s,o_j) &\le d_{t_1}(s,o_i) + \tfrac{1}{1-2.001\cdot\frac{\delta^2}{48960k}}\cdot 6.001\cdot\tfrac{\delta^2}{48960k}\cdot d_{t_2}(o_j,o_j^a) \\ &\quad +(8.001+6.001\cdot\tfrac{\delta^2}{48960k}\cdot 4.002)\cdot m_c \\ &\le d_{t_1}(s,o_i) + 6.002\cdot\tfrac{\delta^2}{48960k}\cdot d_{t_2}(o_j,o_j^a) + 8.002 m_c. \end{aligned}$$

For the bound of decreasing the distance, the same proof can be applied: Start with $d_{t_2}(s,o_j) \ge d_{t_1}(s,o_j) - t^*\cdot(2+\delta)m_s \ge d_{t_1}(s,o_i) - d_{t_1}(o_i,o_j) - t^*\cdot(2+\delta)m_s$ and use the same estimations as before from there. □

We want to show that $\hat{o} \in inner(o^*)$ holds after a sequence of short transitions is terminated by one of the conditions described in step 2 of the algorithm. During the sequence, we must also show that $\hat{o} \in outer(o^*)$. The main idea for the following lemma is that $o_\ell$ never leaves $outer(o^*)/3$ per definition and hence following it keeps $\hat{o}$ inside $outer(o^*)$.

**Lemma 3** *Consider a sequence of short transitions which is terminated by a long transition. If $\hat{o} \in inner(o^*)$ at the beginning of the sequence, then $\hat{o} \in inner(o^*)$ after the long transition. During the sequence of short transitions, $\hat{o} \in outer(o^*)$.*

*Proof* As in step 2 of the algorithm, we assume the sequence starts at time $t_1$ with $o^* = o_i$, and terminates with a long transition from $o_\ell$ to $o_j$ between time steps $t_2$

and $t_3$. $\hat{o}$ selects the server $o_\ell$ which passes $r$ on to $o_j$ over the long transition and follows it. Since $d(o_\ell, o^*) \leq outer(o^*)/3$ for the duration of the sequence, we have $\hat{o} \in outer(o_\ell)$ at the beginning of the sequence and therefore $\hat{o} \in outer(o_\ell)$ holds for the entire duration. At the beginning, with

$$\begin{aligned} d_{t_1}(o^*, o^{*a}) &\leq d_{t_1}(o^*, o_\ell^a) \\ &\leq d_{t_1}(o^*, o_\ell) + d_{t_1}(o_\ell, o_\ell^a) \\ &\leq \tfrac{\delta}{144} \cdot d_{t_1}(o^*, o^{*a}) + d_{t_1}(o_\ell, o_\ell^a) \\ \Leftrightarrow\ d_{t_1}(o^*, o^{*a}) &\leq \tfrac{1}{1-\frac{\delta}{144}} \cdot d_{t_1}(o_\ell, o_\ell^a) \end{aligned}$$

we get

$$\begin{aligned} d_{t_1}(\hat{o}, o_\ell) &\leq d_{t_1}(\hat{o}, o^*) + d_{t_1}(o^*, o_\ell) \\ &\leq (\tfrac{\delta^2}{48960k} + \tfrac{\delta}{144}) \cdot d_{t_1}(o^*, o^{*a}) \\ &\leq \tfrac{1}{1-\frac{\delta}{144}} \cdot (\tfrac{\delta^2}{48960k} + \tfrac{\delta}{144}) \cdot d_{t_1}(o_\ell, o_\ell^a) \\ &\leq 0.01 \cdot \delta \cdot d_{t_1}(o_\ell, o_\ell^a). \end{aligned}$$

Furthermore, since $\hat{o}$ at least holds its relative distance to $o_\ell$, during any step $t$ during the sequence,

$$\begin{aligned} d_t(\hat{o}, o^*) &\leq d_t(\hat{o}, o_\ell) + d_t(o_\ell, o^*) \\ &\leq \tfrac{d_{t_1}(\hat{o}, o_\ell)}{d_{t_1}(o_\ell, o_\ell^a)} \cdot d_t(o_\ell, o_\ell^a) + d_t(o_\ell, o^*) \\ &\leq 0.01 \cdot \delta \cdot d_t(o_\ell, o_\ell^a) + \tfrac{\delta}{144} \cdot d_t(o^*, o^{*a}) \\ &\leq 0.01 \cdot \delta \cdot d_t(o_\ell, o^{*a}) + \tfrac{\delta}{144} \cdot d_t(o^*, o^{*a}) \\ &\leq 0.01 \cdot \delta \cdot (d_t(o_\ell, o^*) + d_t(o^*, o^{*a})) + \tfrac{\delta}{144} \cdot d_t(o^*, o^{*a}) \\ &\leq 0.01 \cdot \delta \cdot (\tfrac{\delta}{144} \cdot d_t(o^*, o^{*a}) + d_t(o^*, o^{*a})) + \tfrac{\delta}{144} \cdot d_t(o^*, o^{*a}) \\ &\leq \tfrac{\delta}{48} \cdot d_t(o^*, o^{*a}) \end{aligned}$$

and therefore $\hat{o} \in outer_t(o^*)$ during the whole sequence. By Lemma 1, we have $\hat{o} \in inner(o^*)$ after the long transition. □

We show with the help of Lemma 2 that during the sequence of transitions, $\hat{o}$ does not lose too much distance to $o^*$, while $o_j$, since at one point $d(o_j, o^*) > outer(o^*)/3$, takes enough time to get into position for a short transition such that $\hat{o}$ can reach the final position of $o_j$ in time.

**Lemma 4** *Consider a sequence of short transitions which is terminated by a short transition from $o_\ell$ to $o_j$, where at one point prior in the sequence $d(o_j, o^*) > outer(o^*)/3$. If $\hat{o} \in inner(o^*)$ at the beginning of the sequence and $d(o^*, o^{*a}) \geq 51483\frac{km_c}{\delta^2}$ at all times, then $\hat{o} \in inner(o^*)$ after the transition to $o_j$. During the sequence, $\hat{o} \in outer(o^*)$.*

*Proof* We assume the sequence starts at time $t_1$ with $o^* = o_i$, and terminates with a short transition from $o_\ell$ to $o_j$ between time steps $t_2$ and $t_3$.

We first consider the case that $\hat{o}$ would run outside $outer(o^*)$ if it moved directly to its target point. First, we need to show that $d(\hat{o}, o_\ell) = \frac{2\delta}{145} \cdot d(o_\ell, o_\ell^a)$ is reached before this happens. In the beginning, it holds

$$\begin{aligned} d_{t_1}(\hat{o}, o_\ell) &\leq d_{t_1}(\hat{o}, o^*) + d_{t_1}(o^*, o_\ell) \\ &\leq (\tfrac{\delta^2}{48960k} + \tfrac{\delta}{144}) \cdot d_{t_1}(o^*, o^{*a}). \end{aligned}$$

With

$$\begin{aligned} d(o^*, o^{*a}) &\leq d(o^*, o_\ell) + d(o_\ell, o_\ell^a) \\ &\leq \tfrac{\delta}{144} \cdot d(o^*, o^{*a}) + d(o_\ell, o_\ell^a) \\ \Leftrightarrow\ (1 - \tfrac{\delta}{144}) \cdot d(o^*, o^{*a}) &\leq d(o_\ell, o_\ell^a) \end{aligned}$$

we get $d_{t_1}(\hat{o}, o_\ell) \leq \frac{1}{1-\frac{\delta}{144}} \cdot (\frac{\delta^2}{48960k} + \frac{\delta}{144}) \cdot d_{t_1}(o_\ell, o_\ell^a) < \frac{2\delta}{145} \cdot d_{t_1}(o_\ell, o_\ell^a)$.

Now assume $d(\hat{o}, o_\ell) \leq \frac{2\delta}{145} \cdot d(o_\ell, o_\ell^a)$. Then

$$\begin{aligned} d(\hat{o}, o^*) &\leq d(\hat{o}, o_\ell) + d(o_\ell, o^*) \\ &\leq \tfrac{2\delta}{145} \cdot d(o_\ell, o_\ell^a) + \tfrac{\delta}{144} \cdot d(o^*, o^{*a}) \\ &\leq \tfrac{2\delta}{145} \cdot (d(o_\ell, o^*) + d(o^*, o^{*a})) + \tfrac{\delta}{144} \cdot d(o^*, o^{*a}) \\ &\leq \tfrac{2\delta}{145} \cdot (1 + \tfrac{\delta}{144}) \cdot d(o^*, o^{*a}) + \tfrac{\delta}{144} \cdot d(o^*, o^{*a}) \\ &\leq \tfrac{\delta}{48} \cdot d(o^*, o^{*a}), \end{aligned}$$

meaning $\hat{o} \in outer(o^*)$ for the duration of the sequence. Taking the negation of that statement it also follows that $d(\hat{o}, o_\ell) = \frac{2\delta}{145} \cdot d(o_\ell, o_\ell^a)$ is reached before $\hat{o} \notin outer(o^*)$.

Note that $\hat{o}$ can maintain the point at the fixed distance to $o_\ell$ which is closest to the final position of $o_j$: Imagine the radius $\frac{2\delta}{145} \cdot d(o_\ell, o_\ell^a)$ stays fixed and only $o_\ell$ moves by at most $m_s$. Then the point at the fixed radius closest to $o_j^{(t_3)}$ only changes by at most $m_s$. Afterwards the radius changes by at most $3m_s \cdot \frac{2\delta}{145} < \frac{\delta}{20} m_s$ and hence the movement speed of $(1 + \frac{\delta}{8})m_s$ is sufficient.

We now need to determine that at the final time step $t_3$, $d_{t_3}(o_j, o_\ell) \leq \frac{2\delta}{145} \cdot d_{t_3}(o_\ell, o_\ell^a)$. Apply Lemma 2 by setting $s = o_\ell$ and we can bound

$$\begin{aligned} d_{t_3}(o_j, o_\ell) &\leq 6.002 \cdot \tfrac{\delta^2}{48960k} \cdot d_{t_3}(o^*, o^{*a}) + 8.002 m_c \\ &\leq \tfrac{1}{1-\frac{\delta}{144}} \cdot (\tfrac{6.002\delta^2}{48960k} + \tfrac{8.002\delta^2}{43170}) \cdot d_{t_3}(o_\ell, o_\ell^a) \\ &< \tfrac{2\delta}{145} \cdot d_{t_3}(o_\ell, o_\ell^a). \end{aligned}$$

Now assume it holds true that $\hat{o}$ can move straight towards the final position of $o_j$ with speed $(1 + \frac{\delta}{8})m_s$ without ever leaving $outer(o^*)$. In this case, we compute a path which constitutes an upper bound on the distance $\hat{o}$ has to traverse, using the following definition:

**Definition 1** (**Transition Path**) Assume $o_m = o^*$ at time step $t$ and $o_n = o^*$ at some later time step $t'$. Consider the path constructed as follows. Start at the position of $o_m$ in time step $t$. Let $t_i$ be the last time step before $t'$ in which $o_m = o^*$ and $r \in inner(o_m)$. The first part of the path goes from $o_m$'s position at time step $t$ to

$o_m$'s position in time step $t_i$. Afterwards, a short transition from $o_m$ to some other server $o_x$ between time step $t_i$ and $t_j$ occurs, in which case our path goes from $o_m$ in $t_i$ to $o_x$ in $t_j$. Continue the procedure recursively until $o_n$ in time step $t'$ is reached. We call the constructed path a $(t, t')$-*transition path*. See Fig. 4 for an illustration of one of the recursion steps.

Now consider the $(t_1, t_3)$-transition path. The distance traveled by $\hat{o}$ is bounded by the distance of $\hat{o}$ to $o^*$ at time $t_1$ plus the length of the transition path. The former has a length of $d_{t_1}(\hat{o}, o^*) \leq \frac{\delta^2}{48960k} \cdot d_{t_1}(o^*, o^{*a})$.

To upper bound the length of the $(t_1, t_3)$- transition path, we divide it into two types of edges (excluding the first edge): The first type is between the same offline server in different time steps. If the total time is $\hat{t} = t_3 - t_1$, the maximum distance induced is $\hat{t} \cdot m_s$.

The second type of edges are between different offline servers and represent a short transition. By construction, there are at most $k$ such edges. With the help of Lemma 2 we may upper bound the length of an edge by $6.001 \cdot \frac{\delta^2}{48960k} \cdot d_{t'}(o^*, o^{*a}) + 8.001 m_c$, where $t'$ is the time the transition begins (in the lemma, set $s$ to a static server at the position of the server who passes the request at time $t'$).

The distance $d(o^*, o^{*a})$ can change in two ways over time: It changes due to the movement of the servers or due to a role change of $o^*$, where it suffices to consider only those short transitions included in our constructed path. Let $t'_1, \ldots, t'_k$ be the points in time where the short transitions inducing the second type edges begin. We can upper bound their total length as $\sum_{i=1}^{k}(6.001 \cdot \frac{\delta^2}{48960k} \cdot d_{t'_i}(o^*, o^{*a}) + 8.001 m_c)$. Assuming the highest possible distance for each of the $d_{t'_i}(o^*, o^{*a})$, we get the total distance of the movement during the sequence added to the original length, which is $d_{t_1}(o^*, o^{*a}) + \hat{t} \cdot (2 + \delta) m_s$, for the first transition. The transitions after that build inductively on the resulting lengths. Define $T_0 := d_{t_1}(o^*, o^{*a}) + \hat{t} \cdot (2 + \delta) m_s$. The

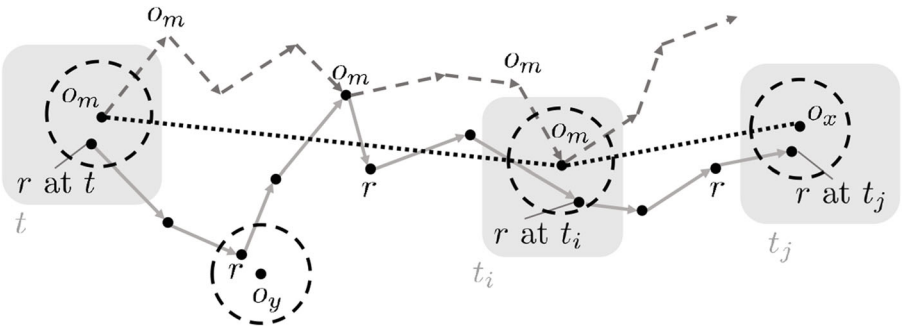


**Fig. 4** The construction of a transition path. The transition path is marked by black points, while the movement of $o_m$ is depicted by dashed arrows. The movement of $r$ is marked by the gray arrows. Starting at the position of $o_m$ at $t$, the last time step $t_i$ is identified at which $o_m = o^*$ and $r \in inner(o_m)$. Note that the role of $o^*$ might change multiple times between $t$ and $t_i$

first edge length is upper bounded by $A_1 := \frac{6.001\delta^2}{48960k} \cdot T_0 + 8.001m_c$, the resulting value for $d(o^*, o^{*a})$ is $T_1 := T_0 + A_1$. In general, $A_i := \frac{6.001\delta^2}{48960k} \cdot T_{i-1} + 8.001m_c$ and $T_i := T_{i-1} + A_i = T_0 + \sum_{j=1}^{i} A_j$. We can bound the total increase by

$$\begin{aligned} \textstyle\sum_{i=1}^{k} A_i &= \textstyle\sum_{i=1}^{k} \left( \frac{6.001\delta^2}{48960k} \cdot (T_0 + \sum_{j=1}^{i-1} A_j) + 8.001m_c \right) \\ &\leq k \cdot \tfrac{6.001\delta^2}{48960k} \cdot (T_0 + \textstyle\sum_{j=1}^{k} A_j) + k \cdot 8.001m_c \\ \Leftrightarrow\ (1 - \tfrac{6.001\delta^2}{48960}) \cdot \textstyle\sum_{i=1}^{k} A_i &\leq \tfrac{6.001\delta^2}{48960} \cdot T_0 + 8.001km_c \\ \Rightarrow\ \textstyle\sum_{i=1}^{k} A_i &\leq 0.0002\delta^2 \cdot T_0 + 8.002km_c. \end{aligned}$$

The total path length may hence bounded by $\hat{t} \cdot m_s + 0.0002\delta^2 \cdot (d_{t_1}(o^*, o^{*a}) + \hat{t} \cdot (2+\delta)m_s) + 8.002km_c + \frac{\delta^2}{48960k} \cdot d_{t_1}(o^*, o^{*a})$.

For comparison, we lower bound the time it takes $o_j$ to move into position such that a short transition can occur. Take the last time step $t$ where $o_j \notin outer_t(o^*)/3 \Rightarrow d_t(o_j, o^*) > \frac{\delta}{144} \cdot d_t(o^*, o^{*a})$. We may assume that $t = t_1$, otherwise the travel time for $o_j$ simply increases. For a short transition between time steps $t_2$ and $t_3$ to $o_j$ to occur, we need $r \in inner_{t_2}(o_\ell)$, $r \in inner_{t_3}(o_j)$ and $t^* := t_3 - t_2 \leq inner_{t_2}(o_\ell)/m_c + 2$. We have $d_{t_2}(o_j, o_\ell) \leq t^* \cdot (m_c + m_s + \frac{\delta}{16}m_s) + inner_{t_2}(o_\ell) + inner_{t_2}(o_j)$ (see Fig. 3 and the proof of Lemma 2).

With $d_{t_2}(o_j, o_j^a) \leq d_{t_2}(o_j, o_\ell) + d_{t_2}(o_\ell, o_\ell^a)$ we get

$$\begin{aligned} d_{t_2}(o_j, o_\ell) &\leq inner_{t_2}(o_j) + inner_{t_2}(o_\ell) + t^* \cdot 2m_c \\ &\leq \tfrac{\delta^2}{48960k} \cdot d_{t_2}(o_j, o_j^a) + 3 \cdot inner_{t_2}(o_\ell) + 4m_c \\ &\leq \tfrac{4\cdot\delta^2}{48960k} \cdot d_{t_2}(o_\ell, o_\ell^a) + \tfrac{\delta^2}{48960k} \cdot d_{t_2}(o_j, o_\ell) + 4m_c \\ \Leftrightarrow\ (1 - \tfrac{\delta^2}{48960k}) \cdot d_{t_2}(o_j, o_\ell) &\leq \tfrac{4\cdot\delta^2}{48960k} \cdot d_{t_2}(o_\ell, o_\ell^a) + 4m_c \\ \Rightarrow\ d_{t_2}(o_j, o_\ell) &\leq \tfrac{4.001\cdot\delta^2}{48960k} \cdot d_{t_2}(o_\ell, o_\ell^a) + 4.001m_c. \end{aligned}$$

Comparing the distances at $t_1$ and $t_2$, we conclude that $d_{t_1}(o_j, o^*) - d_{t_2}(o_j, o^*) \geq \frac{\delta}{144} \cdot d_{t_1}(o^*, o^{*a}) - 4.001 \cdot \frac{\delta^2}{48960k} \cdot d_{t_2}(o^*, o^{*a}) - 4.001m_c$.

In order to lower bound the number of time steps $\hat{t} := t_2 - t_1$ needed for bridging that distance, we first examine the change in $d(o^*, o^{*a})$. Recall that $o^* = o_i$ in $t_1$ and $o^* = o_\ell$ in $t_2$. We can represent the movement of $o^*$ with the $(t_1, t_2)$-transition path. The distance $d(o^*, o^{*a})$ can change in two ways over time: It changes due to the movement of the servers or due to a role change of $o^*$, where it suffices to consider only those short transitions included in our constructed path. If we set the beginnings of the short transitions at time steps $t'_1, \ldots, t'_k$, we get the upper bound similar to before:

$$\begin{aligned} d_{t_2}(o^*, o^{*a}) &\leq d_{t_1}(o^*, o^{*a}) + \hat{t} \cdot (2+\delta)m_s \\ &\quad + \textstyle\sum_{i=1}^{k} (6.001 \cdot \tfrac{\delta^2}{48960k} \cdot d_{t'_i}(o^*, o^{*a}) + 8.001m_c) \\ &\leq d_{t_1}(o^*, o^{*a}) + \hat{t} \cdot (2+\delta)m_s \\ &\quad + 0.0002\delta^2 \cdot (d_{t_1}(o^*, o^{*a}) + \hat{t} \cdot (2+\delta)m_s) + 8.002km_c \end{aligned}$$

Continuing from above, we have

$$\begin{aligned} d_{t_1}(o_j, o^*) - d_{t_2}(o_j, o^*) &\geq \tfrac{\delta}{144} \cdot d_{t_1}(o^*, o^{*a}) - 4.001 \cdot (\tfrac{\delta^2}{48960k} d_{t_2}(o^*, o^{*a}) + m_c) \\ &\geq \tfrac{\delta}{144} \cdot d_{t_1}(o^*, o^{*a}) - \tfrac{4.001\delta^2}{48960k} \cdot (1.0002 \cdot (d_{t_1}(o^*, o^{*a}) \\ &\quad + \hat{t} \cdot (2+\delta) m_s) + 8.002 k m_c) - 4.001 m_c. \end{aligned}$$

Now we consider the ways in which $d(o_j, o^*)$ shrinks: The first is the movement of $o_j$ and $o^*$, reducing the distance by at most $2m_s$ per time step: i.e., if the entire sequence lasts $\hat{t}$ steps, the maximum reduction is $\hat{t} \cdot 2m_s$. The other way is by the role change of $o^*$. Note that above, we just accounted for the change of the distance $d(o^*, o^{*a})$ due to the role change, and not for the change of $d(o_j, o^*)$. Lemma 2 gives us that the distance of $o_j$ to any server decreases by at most $6.001 \cdot \frac{\delta^2}{48960k} \cdot d_t(o^*, o^{*a}) + 8.001 m_c$. This decrease is maximized the same as above, i.e., $0.0002\delta^2 \cdot (d_{t_1}(o^*, o^{*a}) + \hat{t} \cdot 2m_s) + 8.002 k m_c$.

We can now lower bound the number of time steps it takes to complete the sequence: It is bounded by the minimum time $\hat{t}$, such that

$$\begin{aligned} & \hat{t} \cdot 2m_s + 0.0002\delta^2 \cdot (d_{t_1}(o^*, o^{*a}) + \hat{t} \cdot 2m_s) + 8.002 k m_c \\ \geq\ & \tfrac{\delta}{144} \cdot d_{t_1}(o^*, o^{*a}) - \tfrac{4.001\delta^2}{48960k} \cdot (d_{t_1}(o^*, o^{*a}) + \hat{t} \cdot (2+\delta) m_s \\ & + 1.0002 \cdot (d_{t_1}(o^*, o^{*a}) + \hat{t} \cdot (2+\delta) m_s) + 8.002 k m_c) - 4.001 m_c \\ \Leftrightarrow\ & \hat{t} \cdot 2.0004 m_s + \tfrac{4.001\delta^2}{48960k} \cdot 2.0002 \cdot \hat{t} \cdot (2+\delta) m_s \\ \geq\ & \tfrac{\delta}{144} \cdot d_{t_1}(o^*, o^{*a}) - \tfrac{4.001\delta^2}{48960k} \cdot 2.0002 \cdot d_{t_1}(o^*, o^{*a}) - 0.0002\delta^2 \cdot d_{t_1}(o^*, o^{*a}) \\ & - 8.002 k m_c - \tfrac{4.001\delta^2}{48960k} \cdot 8.002 k m_c - 4.001 m_c \\ \Rightarrow\ & 2.0009 \cdot \hat{t} \cdot m_s \geq 0.0065\delta \cdot d_{t_1}(o^*, o^{*a}) - 12.0047 k m_c. \end{aligned}$$

To finish the proof, we show that $\hat{o}$ has enough time to reach its destination by comparing the lower bound of the time $o_j$ takes to move into position to the upper bound of the travel path of $\hat{o}$:

$$\begin{aligned} & \hat{t} \cdot (1 + \tfrac{\delta}{8}) \cdot m_s \\ \geq\ & \hat{t} \cdot m_s + 0.0002\delta^2 \cdot (d_{t_1}(o^*, o^{*a}) + \hat{t}(2+\delta) \cdot m_s) + 8.002 k m_c \\ & + \tfrac{\delta^2}{48960k} \cdot d_{t_1}(o^*, o^{*a}) \\ \Leftrightarrow\ & \hat{t} \cdot (1 + \tfrac{\delta}{8}) \cdot m_s - (1 + 0.0006\delta^2) \cdot \hat{t} \cdot m_s \\ \geq\ & (0.0002\delta^2 + \tfrac{\delta^2}{48960k}) \cdot d_{t_1}(o^*, o^{*a}) + 8.002 k m_c \\ \Leftarrow\ & \hat{t} \cdot (\tfrac{\delta}{8} - 0.0006\delta^2) m_s \\ \geq\ & (0.0002\delta^2 + \tfrac{\delta^2}{48960k}) \cdot d_{t_1}(o^*, o^{*a}) + 8.002 k m_c \\ \Leftarrow\ & \tfrac{1}{2.0009 \cdot m_s} \cdot (0.0065\delta \cdot d_{t_1}(o^*, o^{*a}) - 12.0047 k m_c) \cdot (\tfrac{\delta}{8} - 0.0006\delta^2) m_s \\ \geq\ & (0.0002\delta^2 + \tfrac{\delta^2}{48960k}) \cdot d_{t_1}(o^*, o^{*a}) + 8.002 k m_c \\ \Leftarrow\ & 0.0004\delta^2 \cdot d_{t_1}(o^*, o^{*a}) - 0.75\delta k m_c \\ \geq\ & (0.0002\delta^2 + \tfrac{\delta^2}{48960k}) \cdot d_{t_1}(o^*, o^{*a}) + 8.002 k m_c \\ \Leftarrow\ & 0.00017\delta^2 \cdot d_{t_1}(o^*, o^{*a}) \geq (8.002 + 0.75\delta) k m_c \\ \Leftarrow\ & d_{t_1}(o^*, o^{*a}) \geq 51483 k \tfrac{m_c}{\delta^2} \end{aligned}$$

□

Our analysis of the movement pattern of $\hat{o}$ leads directly to the following lemma, in which we mostly need to argue that either $\hat{o} \in outer(o^*)$ or $\hat{o} = r$.

**Lemma 5** *During the execution of the algorithm, $d(\hat{a}, \hat{o}) \leq 2 \cdot d(o^*, o^{*a}) + d(a^*, r)$ as long as the algorithm is in step 1 or 2.*

*Proof* We argue that $\hat{o} \in outer(o^*)$ or $\hat{o} = r$. We have demonstrated that during a sequence of short transitions, $\hat{o}$ never leaves $outer(o^*)$. It remains to show that the statement holds during a long transition. We observe $\hat{o}$ during the transition time $t^* = t_2 - t_1$. Before the first step, $r \in inner_{t_1}(o^*)$ and $\hat{o} \in outer_{t_1}(o^*)$. We have already shown that for $t^* \geq inner_{t_1}(o^*)/m_c$, $\hat{o}$ catches up to $r$ within the time $t^*$ in the proof of Lemma 1. Expressed in distance, $\hat{o}$ catches up to $r$ when $r$ is a distance of $inner_{t_1}(o^*)$ outside the inner circle of $o^*$. We show that at this time, $r$ is still in $outer(o^*)$: Let $\hat{t} = \lceil inner_{t_1}(o^*)/m_c \rceil$. We have

$$\begin{aligned} d_{t_1+\hat{t}}(r, o^*) &\leq d_{t_1}(r, o*) + \hat{t} \cdot 2m_c \\ &\leq \tfrac{\delta^2}{48960k} \cdot d_{t_1}(o^*, o^{*a}) + 2 \cdot inner_{t_1}(o^*) + 2m_c \\ &\leq 3 \cdot \tfrac{\delta^2}{48960k} \cdot d_{t_1}(o^*, o^{*a}) + 2m_c. \end{aligned}$$

With

$$\begin{aligned} d_{t_1+\hat{t}}(o^*, o^{*a}) &\geq d_{t_1}(o^*, o^{*a}) - \hat{t} \cdot (2+\delta)m_s \\ &\geq d_{t_1}(o^*, o^{*a}) - 2 \cdot inner_{t_1}(o^*) - 2m_c \\ \Leftrightarrow \quad d_{t_1+\hat{t}}(o^*, o^{*a}) + 2m_c &\geq (1 - \tfrac{2\delta^2}{48960k}) \cdot d_{t_1}(o^*, o^{*a}) \\ \Rightarrow 2 \cdot d_{t_1+\hat{t}}(o^*, o^{*a}) + 4m_c &\geq d_{t_1}(o^*, o^{*a}) \end{aligned}$$

we get

$$\begin{aligned} d_{t_1+\hat{t}}(r, o^*) &\leq \tfrac{6\delta^2}{48960k} \cdot d_{t_1+\hat{t}}(o^*, o^{*a}) + 3m_c \\ &\leq \tfrac{\delta}{48} \cdot d_{t_1+\hat{t}}(o^*, o^{*a}) \end{aligned}$$

as long as $d_{t_1+\hat{t}}(o^*, o^{*a}) > 145m_c$.

This implies that at all times, either $\hat{o} \in outer(o^*)$ or $r = \hat{o}$.

We now turn to the claim of the lemma. If $\hat{o} \in outer(o^*)$, then $d(\hat{a}, \hat{o}) \leq d(o^{*a}, \hat{o}) \leq 2 \cdot d(o^*, o^{*a})$. If $\hat{o} = r$, then $\hat{a} = a^*$ and therefore $d(\hat{a}, \hat{o}) = d(a^*, r)$. □

So far we have shown that all claims of Proposition 2 hold as long as the algorithm is not in step 3. It remains to analyze step 3 of the algorithm, using similar arguments as for analyzing the long transitions earlier.

**Lemma 6** *After the execution of step 3 it holds $\hat{o} = r$. Furthermore, $d(\hat{a}, \hat{o}) \leq 2 \cdot d(o^*, o^{*a}) + d(a^*, r)$ during step 3 of the algorithm.*

*Proof* We define time steps $t_1$ and $t_2$ such that they encompass step 3 of the algorithm: i.e., $t_1$ and $t_2$ are chosen minimal such that $d_{t_1}(o^*, o^{*a}) < 51483\frac{m_c}{\delta^2}$ and $d_{t_2}(o^*, o^{*a}) \geq 2 \cdot 51483\frac{m_c}{\delta^2}$. Since $d(o^*, o^{*a})$ changes by at most $(2+\delta)m_s \leq 2m_c$ in each time step, $t_2 - t_1 \geq 25741.5 \cdot \frac{1}{\delta^2}$.

If at time $t_1$, the procedure is in a long transition, the algorithm already follows $r$ and can continue as usual (the result for the long transition holds independently of $d(o^*, o^{*a})$). Otherwise, we have $\hat{o}, r \in outer(o^*)$. Hence $d_{t_1}(\hat{o}, r) \leq \frac{\delta}{24} \cdot d_{t_1}(o^*, o^{*a}) \leq \frac{51483}{24} \cdot \frac{m_c}{\delta}$. The server $\hat{o}$ catches up to $r$ a distance of at least $(1 + \frac{1020k}{\delta}) \cdot m_c$ per time step. Clearly, $(t_2 - t_1) \cdot (1 + \frac{1020k}{\delta}) \cdot m_c > \frac{51483}{24} \cdot \frac{m_c}{\delta}$ and therefore $\hat{o} = r$ at time $t_2$.

The second claim, $d(\hat{a}, \hat{o}) \leq 2 \cdot d(o^*, o^{*a}) + d(a^*, r)$ can be shown the same way as in the previous lemma, where it is clear that $r$ is reached before $d(o^*, o^{*a})$ falls below $145 m_c$. □

#### 4.2.3 Algorithm Analysis

We now turn our attention back to the analysis of the UMS algorithm. In the following, we assume $\mathcal{K}$ to be a $k$-Server algorithm as described in Section 4.2.1. We use a potential composed of two major parts which balance the main ideas of our algorithm against each other: $\phi$ will measure the costs of the greedy strategy, while $\psi$ will cover the matching to the simulated $k$-Server algorithm.

Let $\hat{o}$ be an offline server which fulfills the invariants stated in Proposition 2. Recall that $\hat{a}$ denotes the currently closest server of the online algorithm to $\hat{o}$. The first part of the potential is then defined as

$$\phi := \begin{cases} 4 \cdot d(\hat{a}, \hat{o}) & \text{if } d(\hat{a}, \hat{o}) \leq 107548 \cdot \frac{km_c}{\delta^2} \\ 4 \cdot \frac{1}{\delta m_s} d(\hat{a}, \hat{o})^2 - A & \text{if } 107548 \cdot \frac{km_c}{\delta^2} < d(\hat{a}, \hat{o}) \end{cases}$$

with $A := 4 \cdot (\frac{1}{\delta m_s}(107548 \frac{km_c}{\delta^2})^2 - 107548 \frac{km_c}{\delta^2})$.

For the second part, we set

$$\psi := Y \cdot \frac{m_c}{\delta m_s} \sum_{i=1}^{k} d(a_i, c_i)$$

where the online servers $a_i$ are always sorted such that they represent a minimum weight matching to the simulated servers $c_i$. We choose $Y = \Theta(\frac{k}{\delta^2})$ to be sufficiently large.

If we understand $\phi$ as a function in $d(\hat{a}, \hat{o})$, then we can rewrite it as

$$\phi(d(\hat{a}, \hat{o})) = \max\{4 \cdot d(\hat{a}, \hat{o}), 4 \cdot \frac{1}{\delta m_s} d(\hat{a}, \hat{o})^2 - A\}.$$

Hence, when estimating the potential difference $\Delta\phi = \phi(d(\hat{a}', \hat{o}')) - \phi(d(\hat{a}, \hat{o}))$, we can upper bound it by replacing the term $\phi(d(\hat{a}, \hat{o}))$ with the case identical to $\phi(d(\hat{a}', \hat{o}'))$. This mostly reduces estimating $\Delta\phi$ to bounding the difference $d(\hat{a}', \hat{o}') - d(\hat{a}, \hat{o})$.

For some of our estimations we use a slightly altered result from [12] (simply replace $\delta$ by $\frac{\delta}{2}$). Note that while this lemma might hold for some other metrics as well, it explicitly requires the Euclidean space in the proof provided in [12].

**Lemma 7** *Let $s$ be some server with $d(s', r') \leq \frac{\sqrt{\delta}}{4} \cdot d(a_i', r')$ and $a_i$ moves towards $r'$ a distance of $d(a_i, a_i')$, then $d(a_i, s') - d(a_i', s') \geq \frac{1+\frac{1}{4}\delta}{1+\frac{1}{2}\delta} d(a_i, a_i')$.*

We start the analysis by bounding the second potential difference $\Delta\psi$. The bounds can be obtained by similar arguments as in the proof of Theorem 5.

**Lemma 8** $\Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} \cdot C_{\mathcal{K}} - \sum_{i=1}^{k} d(a_i, a_i')$.

*Proof* Assume that $a^* = a_1$. Every other server $a_i$ moves towards its counterpart $c_i$, hence

$$\begin{aligned}\Delta\psi &\leq Y \cdot \tfrac{m_c}{\delta m_s} \sum_{i=1}^{k} (d(a_i', c_i') - d(a_i, c_i)) \\ &\leq Y \cdot \tfrac{m_c}{\delta m_s} \left( d(a_1', c_1') - d(a_1, c_1) + \sum_{i=2}^{k} (d(c_i, c_i') - d(a_i, a_i')) \right).\end{aligned}$$

Now, if $\mathcal{K}$ serves the request with $c_1$, i.e., $c_1' = r'$, then

$$\Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} \sum_{i=i}^{k} (d(c_i, c_i') - d(a_i, a_i')).$$

Otherwise, $\mathcal{K}$ serves the request with another server (assume $c_2$). Since $a_2$ was not chosen as $a^*$, it moves the full distance of $(1+\delta)m_s$ and hence

$$\begin{aligned}\Delta\psi &\leq Y \cdot \tfrac{m_c}{\delta m_s} \Big( d(a_1, a_1') + d(c_1, c_1') + d(c_2, c_2') - d(a_2, a_2') \\ &\quad + \sum_{i=3}^{k} (d(c_i, c_i') - d(a_i, a_i')) \Big) \\ &\leq Y \cdot \tfrac{m_c}{\delta m_s} \left( \sum_{i=1}^{k} d(c_i, c_i') - \tfrac{\delta}{2} m_s - \sum_{i=3}^{k} d(a_i, a_i') \right).\end{aligned}$$

The lemma follows by setting $Y \geq 8$, as $d(a_1, a_1') + d(a_2, a_2') \leq 4m_s$. □

**Lemma 9** *If $d(a^{*'}, r') > 0$, then $\Delta\psi \leq Y \frac{m_c}{\delta m_s} \cdot C_{\mathcal{K}} - \sum_{i=1}^{k} d(a_i, a_i') - \frac{Y-4}{2} m_c$.*

*Proof* We assume $a^* = a_1$. Since $d(a^{*'}, r') > 0$, we have $d(a_1, a_1') = (1 + \frac{\delta}{2})m_s$. If $r$ is served by $c_1$, then

$$\begin{aligned}\Delta\psi &= Y \cdot \tfrac{m_c}{\delta m_s} \sum_{i=1}^{k} (d(a_i', c_i') - d(a_i, c_i)) \\ &\leq Y \cdot \tfrac{m_c}{\delta m_s} \sum_{i=1}^{k} (d(c_i, c_i') - d(a_i, a_i')) \\ &\leq Y \cdot \tfrac{m_c}{\delta m_s} C_{\mathcal{K}} - \tfrac{m_c}{\delta m_s} \sum_{i=1}^{k} d(a_i, a_i') - (Y-1) \cdot \tfrac{m_c}{\delta m_s} (1 + \tfrac{\delta}{2}) m_s.\end{aligned}$$

If $r$ is served by a different server of $\mathcal{K}$ (assume $c_2$), then

$$\begin{aligned}\Delta\psi &= Y \cdot \tfrac{m_c}{\delta m_s} \sum_{i=1}^{k} (d(a_i', c_i') - d(a_i, c_i)) \\ &\leq Y \cdot \tfrac{m_c}{\delta m_s} \sum_{i=1}^{k} d(c_i, c_i') - \tfrac{m_c}{\delta m_s} \sum_{i=3}^{k} d(a_i, a_i') - Y \cdot \tfrac{m_c}{\delta m_s} \cdot \tfrac{\delta m_s}{2} \\ &\leq Y \cdot \tfrac{m_c}{\delta m_s} C_{\mathcal{K}} - \sum_{i=1}^{k} d(a_i, a_i') - \tfrac{Y-4}{2} m_c.\end{aligned}$$

This term is larger then the former one for sufficiently large $Y$. □

Now consider the case that $r' \notin inner(o^{*'})$. We have

$$\begin{aligned}d(a^{*'}, r') &\leq d(o^{*a'}, r') \\ &\leq d(o^{*'}, o^{*a'}) + d(o^{*'}, r') \\ &\leq (\tfrac{48960k}{\delta^2} + 1) \cdot d(o^{*'}, r').\end{aligned}$$

The movement cost are canceled by $\Delta\psi$ as in Lemma 8. It only remains to bound the possible increase of $\phi$. We use $d(\hat{a}', \hat{o}') - d(\hat{a}, \hat{o}) \leq (3 + \frac{1020k}{\delta}) \cdot m_c$.

**Lemma 10** *If $r' \notin inner(o^{*'})$, then $\Delta\phi \leq \mathcal{O}(\frac{k^2}{\delta^4} \frac{m_c}{m_s}) \cdot C_{Opt}$.*

*Proof* 1. $d(\hat{a}', \hat{o}') \leq 107548 \cdot \frac{km_c}{\delta^2}$:

$$\Delta\phi \leq 4 \cdot d(\hat{a}', \hat{o}') \leq 8 \cdot d(o^{*'}, o^{*a'}) + 4 \cdot d(a^{*'}, r') \leq (12 \cdot \tfrac{48960k}{\delta^2} + 4) \cdot d(o^{*'}, r').$$

2. $107548 \cdot \frac{km_c}{\delta^2} < d(\hat{a}', \hat{o}')$:

$$\begin{aligned}\Delta\phi &\leq \tfrac{4}{\delta m_s}(d(\hat{a}', \hat{o}')^2 - d(\hat{a}, \hat{o})^2) \\ &\leq \tfrac{4}{\delta m_s}(d(\hat{a}', \hat{o}')^2 - (d(\hat{a}', \hat{o}') - (3 + \tfrac{1020k}{\delta}) \cdot m_c)^2) \\ &\leq \mathcal{O}(\tfrac{k}{\delta}) \cdot \tfrac{m_c}{\delta m_s} d(\hat{a}', \hat{o}') \\ &\leq \mathcal{O}(\tfrac{k^2}{\delta^3}) \cdot \tfrac{m_c}{\delta m_s} d(o^{*'}, r').\end{aligned}$$

□

In all of the above, the competitive ratio is bounded by

$$\mathcal{O}(\frac{k^2}{\delta^3}) \cdot \frac{m_c}{\delta m_s} + Y \cdot \frac{m_c}{\delta m_s} \cdot c(\mathcal{K}).$$

Finally, we consider the case $r' \in inner(o^{*'})$. When $d(a^*, r') > 102970 \frac{km_c}{\delta^2}$, we use Lemma 7 to obtain the following:

**Lemma 11** *If $d(a^*, r') > 102970 \frac{km_c}{\delta^2}$ and $r' \in inner(o^{*'})$, then $d(\hat{a}', \hat{o}') - d(\hat{a}, \hat{o}) \leq -\frac{\delta}{8} m_s$.*

*Proof* By our construction of the simulated $k$-Server algorithm, we have $d(c_i', r') \leq 9km_c \leq \frac{\delta^2}{9724} \cdot d(a^{*'}, r')$ for all $i$. Furthermore,

$$\begin{aligned} & & d(o^{*'}, o^{*a'}) &\leq d(o^{*'}, a^{*'}) \\ & & &\leq d(o^{*'}, r') + d(r', a^{*'}) \\ &\Leftrightarrow & (1 - \tfrac{\delta^2}{48960k}) \cdot d(o^{*'}, o^{*a'}) &\leq d(r', a^{*'}). \end{aligned}$$

Hence

$$\begin{aligned} d(c_i', \hat{o}') &\leq d(c_i', r') + d(r', o^{*'}) + d(o^{*'}, \hat{o}') \\ &\leq \tfrac{\delta^2}{9724} \cdot d(a^{*'}, r') + (\tfrac{\delta}{48} + \tfrac{\delta^2}{48960k}) \cdot d(o^{*'}, o^{*a'}) \\ &\leq 0.021\delta \cdot d(a^{*'}, r') \\ &\leq 0.021\delta \cdot d(a_i', r') \end{aligned}$$

and with Lemma 7, we get $d(a_i', \hat{o}') - d(a_i, \hat{o}') \leq -\frac{1+\frac{1}{4}\delta}{1+\frac{1}{2}\delta} d(a_i, a_i')$ for all $i$.

In order to bound the movement of $\hat{o}$, we need to show that $d(o^*, o^{*a}) \geq 2 \cdot 51483 \frac{km_c}{\delta^2}$.

We use

$$\begin{aligned} & & d(a^*, r') &\leq m_c + d(a^{*'}, r') \\ & & &\leq m_c + d(o^{*a'}, r') \\ & & &\leq m_c + (1 + \tfrac{\delta^2}{48960k}) \cdot d(o^{*'}, o^{*a'}) \\ &\Leftrightarrow & \tfrac{1}{1+\frac{\delta^2}{48960k}} (d(a^*, r') - m_c) &\leq d(o^{*'}, o^{*a'}). \end{aligned}$$

The bound follows from $d(a^*, r') > 102970 \frac{km_c}{\delta^2}$.

From Proposition 2 we get $d(\hat{o}, \hat{o}') \leq (1 + \frac{\delta}{8}) m_s$ and therefore

$$\begin{aligned} d(\hat{a}', \hat{o}') - d(\hat{a}, \hat{o}) &\leq -\tfrac{1+\frac{1}{4}\delta}{1+\frac{1}{2}\delta} d(a_i, a_i') + d(\hat{o}, \hat{o}') \\ &\leq -(1 + \tfrac{\delta}{4}) m_s + (1 + \tfrac{\delta}{8}) m_s \\ &\leq -\tfrac{\delta}{8} m_s \end{aligned}$$

where $i$ is chosen such that $a_i$ is closest to $\hat{o}'$. □

With this lemma, $\phi$ can be used to cancel the costs of the algorithm in case of a high distance to $r$.

**Lemma 12** *If $r' \in inner(o^{*'})$, then $C_{Alg} + \Delta\phi + \Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} \cdot C_{\mathcal{K}} + 2 \cdot d(o^{*'}, r')$.*

*Proof* 1. $d(\hat{a}', \hat{o}') \leq 107548 \cdot \frac{km_c}{\delta^2}$: We use

$$\begin{aligned} d(a^{*'}, r') &\leq d(\hat{a}', r') \\ &\leq d(\hat{a}', \hat{o}') + d(\hat{o}', r') \\ &\leq d(\hat{a}', \hat{o}') + 2 \cdot \tfrac{\delta}{48} \cdot d(o^{*'}, o^{*a'}) \\ &\leq (1 + \tfrac{2\delta}{47}) \cdot d(\hat{a}', \hat{o}') \end{aligned}$$

to get $C_{Alg} + \Delta\phi \leq 6 \cdot d(\hat{a}', \hat{o}') + \sum_{i=1}^{k} d(a_i, a_i')$. Furthermore, $\Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} C_{\mathcal{K}} - \sum_{i=1}^{k} d(a_i, a_i') - \frac{Y-4}{2} m_c$ due to Lemma 9. In total, $C_{Alg} + \Delta\phi + \Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} \cdot C_{\mathcal{K}}$ with $Y \geq \Omega(\frac{k}{\delta^2})$.

2. $107548 \cdot \frac{k m_c}{\delta^2} < d(\hat{a}', \hat{o}')$: We show that the condition of Lemma 11 applies:

$$\begin{aligned} d(\hat{a}', \hat{o}') &\leq d(a^{*'}, \hat{o}') \\ &\leq d(a^{*'}, a^*) + d(a^*, r') + d(r', \hat{o}') \\ &\leq m_c + d(a^*, r') + 2 \cdot \tfrac{\delta}{48} \cdot d(o^{*'}, o^{*a'}) \\ &\leq m_c + d(a^*, r') + \tfrac{2}{47} \cdot d(\hat{a}', \hat{o}') \\ \Leftrightarrow \tfrac{45}{47} \cdot d(\hat{a}', \hat{o}') - m_c &\leq d(a^*, r') \\ \Rightarrow 102970 \tfrac{k m_c}{\delta^2} &\leq d(a^*, r') \end{aligned}$$

Hence the lemma gives us

$$\begin{aligned} \Delta\phi &\leq 4 \cdot \tfrac{1}{\delta m_s} \left( d(\hat{a}', \hat{o}')^2 - d(\hat{a}, \hat{o})^2 \right) \\ &\leq 4 \cdot \tfrac{1}{\delta m_s} \left( d(\hat{a}', \hat{o}')^2 - (d(\hat{a}', \hat{o}') + \tfrac{\delta}{8} m_s)^2 \right) \\ &= -d(\hat{a}', \hat{o}'). \end{aligned}$$

Furthermore, we have

$$\begin{aligned} C_{Alg} &\leq d(\hat{a}', r') + \textstyle\sum_{i=1}^{k} d(a_i, a_i') \\ &\leq d(\hat{a}', \hat{o}') + d(\hat{o}', o^{*'}) + d(o^{*'}, r') + \textstyle\sum_{i=1}^{k} d(a_i, a_i') \\ &\leq d(\hat{a}', \hat{o}') + (1 + \tfrac{\delta}{48}) \cdot d(o^{*'}, r') + \textstyle\sum_{i=1}^{k} d(a_i, a_i') \end{aligned}$$

and $\Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} \cdot C_{\mathcal{K}} - \sum_{i=1}^{k} d(a_i, a_i')$ due to Lemma 8. In total, we get $C_{Alg} + \Delta\phi + \Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} \cdot C_{\mathcal{K}} + 2 \cdot d(o^{*'}, r')$.

□

The resulting competitive ratio of $Y \cdot \frac{m_c}{\delta m_s} \cdot c(\mathcal{K}) + 2$ is less than the $\mathcal{O}(\frac{k^2}{\delta^3}) \cdot \frac{m_c}{\delta m_s} + Y \cdot \frac{m_c}{\delta m_s} \cdot c(\mathcal{K})$ bound from the former set of cases. Accounting for the loss due to the transformation of the simulated $k$-Server algorithm, we obtain the following result:

**Theorem 6** *If $m_c \geq (1 + \delta) m_s$, the algorithm UMS is $\mathcal{O}(\frac{1}{\delta^4} \cdot k^2 \cdot \frac{m_c}{m_s} + \frac{1}{\delta^3} \cdot k^2 \cdot \frac{m_c}{m_s} \cdot c(\mathcal{K}))$-competitive, where $c(\mathcal{K})$ is the competitive ratio of the simulated $k$-server algorithm $\mathcal{K}$.*

## 5 Extension to the Weighted Problem

In this section we consider our general model in which the movement costs are weighted with a factor $D > 1$. We assume throughout the section that $D \geq 2$ for convenience in the analysis. In case $D < 2$, we may just apply the algorithm from the previous section, whose costs increase by at most a factor of 2 as a result.

The main difference to the unweighted case is that our algorithm uses a $k$-Page Migration algorithm as guidance, whose best competitive ratio in the deterministic

case so far is a factor $\Theta(k)$ worse than that of a $k$-Server algorithm for general metrics. The analysis is slightly more involved since unlike in the $k$-Server problem, a $k$-Page Migration algorithm does not always have to have one page at the point of the request. In case of small distances to $r$, the movement costs have to be balanced against the serving costs by scaling down the movement distance by a factor of $D$. Throughout this section, we use the same notation as for the unweighted version.

Our algorithm *Weighted-Mobile Servers (WMS)* works as follows:
Take any $k$-Page Migration algorithm $\mathcal{K}$. Upon receiving the next request $r'$, simulate the next step of $\mathcal{K}$. Calculate a minimum weight matching (with the distances as weights) between the servers $a_1, \ldots, a_k$ of the online algorithm and the pages $c'_1, \ldots, c'_k$ of $\mathcal{K}$. Select a closest server $\tilde{a}$ to $r'$ and move it to $r'$ at most a distance $\min\{m_c, \frac{1}{D}(1-\varepsilon)\cdot d(\tilde{a}, r')\}$ in case $m_c \leq (1+\delta-\varepsilon)m_s$ and at most $\min\{(1+\frac{\delta}{2})m_s, \frac{1}{D}(1-\frac{\delta}{2})\cdot d(\tilde{a}, r')\}$ in case $m_c \geq (1+\delta)m_s$. All other servers $a_i$ move towards their counterparts in the matching $c'_i$ with speed $\min\{(1+\delta)m_s, \frac{1}{D}\cdot d(\tilde{a}, r')\}$. If another server than $\tilde{a}$ is closer to $r'$ after movement, then move all servers towards their counterpart in the matching with speed $m_s$ instead.

The remainder of this section is devoted to the analysis of the WMS algorithm and is structured similar to Section 4.

We start by analyzing the case that $m_c \leq (1-\varepsilon)\cdot m_s$ for some $\varepsilon \in (0, \frac{1}{2}]$. For $\varepsilon \geq \frac{1}{2}$, our algorithm simply assumes $\varepsilon = \frac{1}{2}$. It can be easily verified that this does not hinder the analysis.

**Theorem 7** *If $m_c \leq (1-\varepsilon)\cdot m_s$ for some $\varepsilon \in (0, \frac{1}{2}]$, the algorithm WMS is $\sqrt{2}\cdot 11/\varepsilon \cdot c(\mathcal{K})$-competitive, where $c(\mathcal{K})$ is the competitive ratio of the simulated $k$-Page Migration algorithm $\mathcal{K}$.*

*Proof* We assume the servers adapt their ordering $a_1, \ldots, a_k$ according to the minimum matching in each time step. Based on the matching, we define the following potential: $\psi := \sqrt{2}\cdot\frac{4D}{\varepsilon}\sum_{i=1}^{k} d(a_i, c_i)$. We observe that in all time steps it holds $d(a^*, r) \leq \frac{D}{1-\varepsilon}\cdot m_c \leq 2Dm_c$. This is because the distance does not increase if the movement towards $r$ is $m_c$, and this is done as soon as $m_c$ is less or equal $\frac{1}{D}(1-\varepsilon)\cdot d(\tilde{a}, r')$ at the beginning of the time step. We fix a time step and assume $\tilde{a} = a_1$.

First examine the case that $\tilde{a}$ moves towards its matching partner instead of $r'$. Then $\Delta\psi \leq \sqrt{2}\frac{4D}{\varepsilon}\sum_{i=1}^{k} d(c_i, c'_i) - \sqrt{2}\frac{4D}{\varepsilon}\sum_{i=1}^{k} d(a_i, a'_i)$ and $C_{Alg} = D\cdot\sum_{i=1}^{k} d(a_i, a'_i) + d(a^{*'}, r') \leq D\cdot\sum_{i=1}^{k} d(a_i, a'_i) + 2Dm_c$. Consider the server which is matched to $c^{*'}$: Either it reaches $c^{*'}$ or it moves a distance of $m_s$. In the first case $d(a^{*'}, r') \leq d(c^{*'}, r')$ which gives a competitive ratio of $\sqrt{2}\frac{4}{\varepsilon}\cdot c(\mathcal{K})$ immediately. In the latter case, there is a server $a_j$ such that $d(a_j, a'_j) = m_s$ and hence $\Delta\psi \leq \sqrt{2}\frac{4D}{\varepsilon}\sum_{i=1}^{k} d(c_i, c'_i) - \sqrt{2}\frac{D}{\varepsilon}\sum_{i=1}^{k} d(a_i, a'_i) - \sqrt{2}\frac{3D}{\varepsilon}m_s$ which implies a competitive ratio of at most $\sqrt{2}\frac{4}{\varepsilon}\cdot c(\mathcal{K})$ as well.

Now assume $\tilde{a} = a_1$ moves towards $r'$ and hence $a^{*'} = a'_1$. We have $d(a'_1, c'_1) - d(a_1, c'_1) \leq \min\{m_c, \frac{1}{D}(1-\varepsilon) \cdot d(\tilde{a}, r')\}$. In all of the following cases, we make use of

$$\begin{aligned}\Delta\psi &= \sqrt{2}\tfrac{4D}{\varepsilon}\left(\textstyle\sum_{i=1}^{k} d(a'_i, c'_i) - \sum_{i=1}^{k} d(a_i, c_i)\right)\\ &\leq \sqrt{2}\tfrac{4D}{\varepsilon}\textstyle\sum_{i=1}^{k} d(c_i, c'_i) + \sqrt{2}\tfrac{4D}{\varepsilon}\left(\sum_{i=1}^{k} d(a'_i, c'_i) - \sum_{i=1}^{k} d(a_i, c'_i)\right).\end{aligned}$$

We distinguish the following cases with respect to the positioning of the pages of $\mathcal{K}$:

1. $d(a^{*'}, r') \leq d(c^{*'}, r')$:
   Since we assume $D \geq 2$, we have
   $$\begin{aligned} D \cdot d(a_1, a'_1) &\leq d(a_1, r')\\ &\leq d(a_1, a'_1) + d(a'_1, r')\\ \Rightarrow \tfrac{D}{2} \cdot d(a_1, a'_1) &\leq d(c^{*'}, r').\end{aligned}$$
   It follows $C_{Alg} \leq 3 \cdot d(c^{*'}, r') + D \cdot \sum_{i=2}^{k} d(a_i, a'_i)$ and
   $\Delta\psi \leq \sqrt{2}\frac{4D}{\varepsilon}\sum_{i=1}^{k} d(c_i, c'_i) + \sqrt{2}\frac{8}{\varepsilon} \cdot d(c^{*'}, r') - D \cdot \sum_{i=2}^{k} d(a_i, a'_i)$.
2. $d(a^{*'}, r') > d(c^{*'}, r')$ and $c^{*'} = c'_1$:
   We know that
   $d(a'_1, c'_1) - d(a_1, c'_1) \leq -\frac{1}{\sqrt{2}} \cdot d(a_1, a'_1) = -\frac{1}{\sqrt{2}} \cdot \min\{m_c, \frac{1}{D}(1-\varepsilon) \cdot d(\tilde{a}, r')\}$
   and hence
   $\Delta\psi \leq \sqrt{2}\frac{4D}{\varepsilon}\sum_{i=1}^{k} d(c_i, c'_i) - \frac{4D}{\varepsilon} \cdot \min\{m_c, \frac{1}{D}(1-\varepsilon) \cdot d(\tilde{a}, r')\} - D \cdot \sum_{i=2}^{k} d(a_i, a'_i)$.
   If $d(a_1, a'_1) = m_c$ then $C_{Alg} \leq 3Dm_c + D \cdot \sum_{i=2}^{k} d(a_i, a'_i)$, otherwise
   $C_{Alg} \leq 2 \cdot d(\tilde{a}, r') + D \cdot \sum_{i=2}^{k} d(a_i, a'_i)$.
3. $d(a^{*'}, r') > d(c^{*'}, r')$ and $c^{*'} \neq c'_1$:
   We assume $c^{*'} = c'_2$. It must hold $a'_2 \neq c'_2$ and hence $d(c'_2, a'_2) - d(c'_2, a_2) \leq -\min\{m_s, \frac{1}{D} \cdot d(\tilde{a}, r')\}$.
   In the case $d(a_2, a'_2) = \frac{1}{D} \cdot d(\tilde{a}, r')$, it holds
   $d(a'_1, c'_1) - d(a_1, c'_1) + d(a'_2, c'_2) - d(a_2, c'_2) \leq -\frac{\varepsilon}{D} \cdot d(\tilde{a}, r')$. This gives us
   $\Delta\psi \leq \sqrt{2}\frac{4D}{\varepsilon}(\sum_{i=1}^{k} d(c_i, c'_i) - \frac{\varepsilon}{D} \cdot d(\tilde{a}, r') - \sum_{i=3}^{k} d(a_i, a'_i))$.
   With $C_{Alg} = d(a'_1, r') + D \cdot \sum_{i=1}^{k} d(a_i, a'_i) \leq 3 \cdot d(\tilde{a}, r') + D \cdot \sum_{i=3}^{k} d(a_i, a'_i)$
   the bound follows.
   In case $d(a_2, a'_2) = m_s$, we have $d(a'_1, c'_1) - d(a_1, c'_1) + d(a'_2, c'_2) - d(a_2, c'_2) \leq m_c - m_s \leq -\varepsilon m_s$. Similar as before,
   $\Delta\psi \leq \sqrt{2}\frac{4D}{\varepsilon}(\sum_{i=1}^{k} d(c_i, c'_i) - \varepsilon m_s - \sum_{i=3}^{k} d(a_i, a'_i))$ and
   $C_{Alg} \leq 4Dm_s + D \cdot \sum_{i=3}^{k} d(a_i, a'_i)$.

□

We can extend this bound to the resource augmentation scenario, where the online algorithm may move the servers a maximum distance of $(1+\delta)\cdot m_s$. When relaxing the condition appropriately to $m_c \leq (1+\delta-\varepsilon)\cdot m_s$, then we get the following result:

**Corollary 2** *If $m_c \leq (1+\delta-\varepsilon)\cdot m_s$ for some $\varepsilon \in (0, \frac{1}{2}]$, the algorithm WMS is $\frac{\sqrt{2}\cdot 11\cdot(1+\delta)}{\varepsilon}\cdot c(\mathcal{K})$-competitive, where $c(\mathcal{K})$ is the competitive ratio of the simulated $k$-Page Migration algorithm $\mathcal{K}$.*

From here on we assume $\mathcal{K}$ to be a $k$-Page Migration algorithm obtained from the transformation in Section 4.2.1. The offline helper and its invariants as stated in Proposition 2 do not depend on the simulated algorithm and therefore all insights gained from Section 4.2.2 are still valid. We use a potential composed of two major parts just as for the unweighted case.

Let $\hat{o}$ be an offline server which fulfills the invariants stated in Proposition 2. The first part of the potential is then defined as

$$\phi := \begin{cases} 4\cdot d(\hat{a},\hat{o}) & \text{if } d(\hat{a},\hat{o}) \leq 107548D\cdot\frac{km_c}{\delta^2} \\ 4\cdot\frac{1}{\delta m_s}d(\hat{a},\hat{o})^2 + A & \text{if } 107548D\cdot\frac{km_c}{\delta^2} < d(\hat{a},\hat{o}) \end{cases}$$

with $A := 4\cdot(107548D\frac{km_c}{\delta^2} - \frac{1}{\delta m_s}(107548D\frac{km_c}{\delta^2})^2)$.

For the second part, we set

$$\psi := Y\cdot D\frac{m_c}{\delta m_s}\sum_{i=1}^{k} d(a_i, c_i)$$

where the online servers $a_i$ are always sorted such that they represent a minimum weight matching to the simulated servers $c_i$. We choose $Y = \Theta(\frac{k}{\delta^2})$ to be sufficiently large.

We begin by analyzing $\psi$, reusing ideas from the proof of Theorem 7.

**Lemma 13** $\Delta\psi \leq \mathcal{O}(1)\cdot Y\cdot\frac{m_c}{\delta m_s}\cdot C_{\mathcal{K}} - D\cdot\sum_{i=1}^{k} d(a_i, a_i')$.

*Proof* Assume that $a^* = a_1$. Every other server $a_i$ moves towards its counterpart $c_i$, hence

$$\begin{aligned}\Delta\psi &\leq Y\cdot D\tfrac{m_c}{\delta m_s}\sum_{i=1}^{k}(d(a_i', c_i') - d(a_i, c_i)) \\ &\leq Y\cdot D\tfrac{m_c}{\delta m_s}\left(d(a_1', c_1') - d(a_1, c_1) + \sum_{i=2}^{k}(d(c_i, c_i') - d(a_i, a_i'))\right).\end{aligned}$$

First examine the case that $\tilde{a}$ moves towards its matching partner instead of $r'$. Then $\Delta\psi \leq Y\cdot D\frac{m_c}{\delta m_s}\cdot\sum_{i=1}^{k} d(c_i, c_i') - Y\cdot D\frac{m_c}{\delta m_s}\cdot\sum_{i=1}^{k} d(a_i, a_i')$.

Now assume $\tilde{a} = a_1$ moves towards $r'$. We have $d(a_1, a_1') \leq \min\{(1+\frac{\delta}{2})m_s, \frac{1}{D}(1-\frac{\delta}{2})\cdot d(\tilde{a}, r')\}$. We distinguish the following cases with respect to the positioning of the pages of $\mathcal{K}$:

1. $d(a^{*'}, r') \leq d(c^{*'}, r')$:

Since we assume $D \geq 2$, we have

$$\begin{aligned} D \cdot d(a_1, a_1') &\leq d(a_1, r') \\ &\leq d(a_1, a_1') + d(a_1', r') \\ \Rightarrow \tfrac{D}{2} \cdot d(a_1, a_1') &\leq d(c^{*'}, r'). \end{aligned}$$

It follows $\Delta\psi \leq Y D \frac{m_c}{\delta m_s} \cdot \sum_{i=1}^{k} d(c_i, c_i') + Y \frac{m_c}{\delta m_s} \cdot d(c^{*'}, r') - D \cdot \sum_{i=2}^{k} d(a_i, a_i')$.

2. $d(a^{*'}, r') > d(c^{*'}, r')$ and $c^{*'} = c_1'$:
We know that $d(a_1', c_1') - d(a_1, c_1') \leq -\frac{1}{\sqrt{2}} \cdot d(a_1, a_1')$ and hence
$\Delta\psi \leq \sqrt{2}\frac{4D}{\varepsilon} \sum_{i=1}^{k} d(c_i, c_i') - \frac{4D}{\varepsilon} \cdot d(a_1, a_1') - D \cdot \sum_{i=2}^{k} d(a_i, a_i')$.
3. $d(a^{*'}, r') > d(c^{*'}, r')$ and $c^{*'} \neq c_1'$:
We assume $c^{*'} = c_2'$. It must hold $a_2' \neq c_2'$ and hence
$d(c_2', a_2') - d(c_2', a_2) \leq -\min\{m_s, \frac{1}{D} \cdot d(\tilde{a}, r')\}$. This gives us
$d(a_1', c_1') - d(a_1, c_1') + d(a_2', c_2') - d(a_2, c_2') \leq -\varepsilon \cdot d(a_2, a_2')$. It follows

$$\begin{aligned} \Delta\psi &\leq \sqrt{2}\tfrac{4D}{\varepsilon}\left(\sum_{i=1}^{k} d(c_i, c_i') - \varepsilon \cdot d(a_2, a_2') - \sum_{i=3}^{k} d(a_i, a_i')\right) \\ &\leq \sqrt{2}\tfrac{4D}{\varepsilon} \sum_{i=1}^{k} d(c_i, c_i') - D \cdot \sum_{i=1}^{k} d(a_i, a_i'). \end{aligned}$$

□

**Lemma 14** *If $d(a^{*'}, r') > d(c^{*'}, r')$, then*
$\Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} C_{\mathcal{K}} - D \cdot \sum_{i=1}^{k} d(a_i, a_i') - \frac{Y-4}{2} D \frac{m_c}{\delta m_s} \cdot \min\{m_s, \frac{1}{D} \cdot d(\tilde{a}, r')\}$.

*Proof* Assume that $a^* = a_1$. Every other server $a_i$ moves towards its counterpart $c_i$, hence

$$\begin{aligned} \Delta\psi &\leq Y \cdot D \tfrac{m_c}{\delta m_s} \sum_{i=1}^{k} (d(a_i', c_i') - d(a_i, c_i)) \\ &\leq Y \cdot D \tfrac{m_c}{\delta m_s} \left( d(a_1', c_1') - d(a_1, c_1) + \sum_{i=2}^{k} (d(c_i, c_i') - d(a_i, a_i')) \right). \end{aligned}$$

First examine the case that $\tilde{a}$ moves towards its matching partner instead of $r'$. Then

$$\begin{aligned} \Delta\psi &\leq Y \cdot D \tfrac{m_c}{\delta m_s} \cdot \sum_{i=1}^{k} d(c_i, c_i') - Y \cdot D \tfrac{m_c}{\delta m_s} \cdot \sum_{i=1}^{k} d(a_i, a_i') \\ &\leq Y \cdot \tfrac{m_c}{\delta m_s} C_{\mathcal{K}} - D \cdot \sum_{i=1}^{k} d(a_i, a_i') - (Y-1) \cdot D \tfrac{m_c}{\delta} \end{aligned}$$

since the server matched to $c^{*'}$ moves the full distance.

Now assume $\tilde{a}$ moves towards $r'$. If $c^{*'} = c_1'$, we know that $d(a_1', c_1') - d(a_1, c_1') \leq -\frac{1}{\sqrt{2}} \cdot d(a_1, a_1')$ and hence
$\Delta\psi \leq Y \cdot D \frac{m_c}{\delta m_s} \sum_{i=1}^{k} d(c_i, c_i') - D \frac{m_c}{\delta m_s} \cdot \sum_{i=1}^{k} d(a_i, a_i') - \frac{Y}{\sqrt{2}} \cdot D \frac{m_c}{\delta m_s} d(a_1, a_1')$ with
$d(a_1, a_1') = \min\{(1 + \frac{\delta}{2})m_s, \frac{1}{D}(1 - \frac{\delta}{2}) \cdot d(\tilde{a}, r')\}$.

Otherwise, we assume $c^{*'} = c'_2$. It must hold $a'_2 \neq c'_2$ and hence $d(c'_2, a'_2) - d(c'_2, a_2) \leq -\min\{m_s, \frac{1}{D} \cdot d(\tilde{a}, r')\}$. This gives us $d(a'_1, c'_1) - d(a_1, c'_1) + d(a'_2, c'_2) - d(a_2, c'_2) \leq -\frac{\delta}{2} \cdot d(a_2, a'_2)$. It follows

$$\begin{aligned}\Delta\psi &\leq Y \cdot D\frac{m_c}{\delta m_s}\left(\sum_{i=1}^{k} d(c_i, c'_i) - \frac{\delta}{2} \cdot d(a_2, a'_2) - \sum_{i=3}^{k} d(a_i, a'_i)\right)\\ &\leq Y \cdot D\frac{m_c}{\delta m_s}\sum_{i=1}^{k} d(c_i, c'_i) - D \cdot \sum_{i=1}^{k} d(a_i, a'_i) - \frac{Y-4}{2} D\frac{m_c}{\delta m_s} \cdot d(a_2, a'_2).\end{aligned}$$

□

Now consider the case that $r' \notin inner(o^{*'})$. We have

$$\begin{aligned}d(a^{*'}, r') &\leq d(o^{*a'}, r')\\ &\leq d(o^{*'}, o^{*a'}) + d(o^{*'}, r')\\ &\leq (\tfrac{48960k}{\delta^2} + 1) \cdot d(o^{*'}, r').\end{aligned}$$

The movement costs are canceled by $\Delta\psi$ as in Lemma 13. The increase of $\phi$ can be bound with Lemma 10. In all of the above, the competitive ratio is bounded by $\mathcal{O}(\frac{k^2}{\delta^3}) \cdot \frac{m_c}{\delta m_s} + Y \cdot \frac{m_c}{\delta m_s} \cdot c(\mathcal{K})$.

Finally, we consider the case $r' \in inner(o^{*'})$. As in the previous Section, whenever $d(a^*, r') > 102970D\frac{km_c}{\delta^2}$, we make use of Lemma 7 to obtain the following result, which then helps us bound $\Delta\phi$:

**Lemma 15** *If $d(a^*, r') > 102970D\frac{km_c}{\delta^2}$ and $r' \in inner(o^{*'})$, then $d(\hat{a}', \hat{o}') - d(\hat{a}, \hat{o}) \leq -\frac{\delta}{8}m_s$.*

*Proof* By our construction of the simulated $k$-Page Migration algorithm, we have $d(c'_i, r') \leq 33Dkm_c \leq \frac{\delta^2}{2652} \cdot d(a^{*'}, r')$ for all $i$. Furthermore,

$$\begin{aligned}d(o^{*'}, o^{*a'}) &\leq d(o^{*'}, a^{*'})\\ &\leq d(o^{*'}, r') + d(r', a^{*'})\\ \Leftrightarrow\ (1 - \tfrac{\delta^2}{48960k}) \cdot d(o^{*'}, o^{*a'}) &\leq d(r', a^{*'}).\end{aligned}$$

Hence

$$\begin{aligned}d(c'_i, \hat{o}') &\leq d(c'_i, r') + d(r', o^{*'}) + d(o^{*'}, \hat{o}')\\ &\leq \tfrac{\delta^2}{2652} \cdot d(a^{*'}, r') + (\tfrac{\delta}{48} + \tfrac{\delta^2}{48960k}) \cdot d(o^{*'}, o^{*a'})\\ &\leq 0.022\delta \cdot d(a^{*'}, r')\\ &\leq 0.022\delta \cdot d(a'_i, r')\end{aligned}$$

and with Lemma 7, $d(a'_i, \hat{o}') - d(a_i, \hat{o}') \leq -\frac{1+\frac{1}{4}\delta}{1+\frac{1}{2}\delta} d(a_i, a'_i)$ follows for all $i$.

In order to bound the movement of $\hat{o}$, we need to show that $d(o^*, o^{*a}) \geq 2 \cdot 51483\frac{km_c}{\delta^2}$. We use

$$\begin{aligned}d(a^*, r') &\leq m_c + d(a^{*'}, r')\\ &\leq m_c + d(o^{*a'}, r')\\ &\leq m_c + (1 + \tfrac{\delta^2}{48960k}) \cdot d(o^{*'}, o^{*a'})\\ \Leftrightarrow\ \tfrac{1}{1+\frac{\delta^2}{48960k}}(d(a^*, r') - m_c) &\leq d(o^{*'}, o^{*a'}).\end{aligned}$$

The bound follows from $d(a^*, r') > 102970\frac{km_c}{\delta^2}$.

From Proposition 2 we get $d(\hat{o}, \hat{o}') \leq (1 + \frac{\delta}{8})m_s$ and therefore

$$\begin{aligned} d(\hat{a}', \hat{o}') - d(\hat{a}, \hat{o}) &\leq -\tfrac{1+\frac{1}{4}\delta}{1+\frac{1}{2}\delta} d(a_i, a_i') + d(\hat{o}, \hat{o}') \\ &\leq -(1 + \tfrac{\delta}{4})m_s + (1 + \tfrac{\delta}{8})m_s \\ &\leq -\tfrac{\delta}{8}m_s \end{aligned}$$

where $i$ is chosen such that $a_i$ is closest to $\hat{o}'$. □

**Lemma 16** *If $r' \in inner(o^{*'})$, then $C_{Alg} + \Delta\phi + \Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} \cdot C_{\mathcal{K}} + 2 \cdot d(o^{*'}, r')$.*

*Proof* 1. $d(\hat{a}', \hat{o}') \leq 107548D \cdot \frac{km_c}{\delta^2}$:

First consider the case $d(a^{*'}, r') \leq d(c^{*'}, r')$. With Lemma 13 we can bound the movement costs of the algorithm. Furthermore, we use

$$\begin{aligned} d(o^{*'}, o^{*a'}) &\leq d(o^{*'}, \hat{a}') \\ &\leq d(o^{*'}, \hat{o}') + d(\hat{o}', \hat{a}') \\ &\leq \tfrac{\delta}{48} \cdot d(o^{*'}, o^{*a'}) + d(\hat{o}', \hat{a}') \\ \Leftrightarrow (1 - \tfrac{\delta}{48}) \cdot d(o^{*'}, o^{*a'}) &\leq d(\hat{o}', \hat{a}') \end{aligned}$$

to get

$$\begin{aligned} d(\hat{o}', \hat{a}') &\leq d(\hat{o}', a^{*'}) \\ &\leq d(a^{*'}, r') + d(r', o^{*'}) + d(o^{*'}, \hat{o}') \\ &\leq d(a^{*'}, r') + 2 \cdot \tfrac{\delta}{48} \cdot d(o^{*'}, o^{*a'}) \\ &\leq d(a^{*'}, r') + \tfrac{2\delta}{47} \cdot d(\hat{o}', \hat{a}') \\ \Rightarrow d(\hat{o}', \hat{a}') &\leq 2 \cdot d(a^{*'}, r'). \end{aligned}$$

Hence $\Delta\phi \leq 4 \cdot d(\hat{a}', \hat{o}') \leq 8 \cdot d(c^{*'}, r')$. In total, $C_{Alg} + \Delta\phi + \Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} \cdot C_{\mathcal{K}}$ with $Y \geq 9$.

Otherwise, Lemma 14 applies which gives us

$\Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} C_{\mathcal{K}} - D \cdot \sum_{i=1}^{k} d(a_i, a_i') - \frac{Y-4}{2} D \frac{m_c}{\delta m_s} \cdot \min\{m_s, \frac{1}{D} \cdot d(\tilde{a}, r')\}$. We may either use $C_{Alg} + \Delta\phi \leq 9 \cdot d(\tilde{a}, r') + D \cdot \sum_{i=1}^{k} d(a_i, a_i')$, or

$$\begin{aligned} d(a^{*'}, r') &\leq d(\hat{a}', r') \\ &\leq d(\hat{a}', \hat{o}') + d(\hat{o}', r') \\ &\leq d(\hat{a}', \hat{o}') + 2 \cdot \tfrac{\delta}{48} \cdot d(o^{*'}, o^{*a'}) \\ &\leq (1 + \tfrac{2\delta}{47}) \cdot d(\hat{a}', \hat{o}') \end{aligned}$$

gives us

$C_{Alg} + \Delta\phi \leq 6 \cdot d(\hat{a}', \hat{o}') + D \cdot \sum_{i=1}^{k} d(a_i, a_i') \leq 6 \cdot 91414D \cdot \frac{km_c}{\delta^2} + D \cdot \sum_{i=1}^{k} d(a_i, a_i')$.

In any case, $C_{Alg} + \Delta\phi + \Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} \cdot C_{\mathcal{K}}$ with $Y \geq \Omega(\frac{k}{\delta^2})$.

2. $107548D \cdot \frac{km_c}{\delta^2} < d(\hat{a}', \hat{o}')$:

We show that the condition of Lemma 15 applies:

$$\begin{aligned} d(\hat{a}', \hat{o}') &\leq d(a^{*'}, \hat{o}') \\ &\leq d(a^{*'}, a^*) + d(a^*, r') + d(r', \hat{o}') \\ &\leq m_c + d(a^*, r') + 2 \cdot \tfrac{\delta}{48} \cdot d(o^{*'}, o^{*a'}) \\ &\leq m_c + d(a^*, r') + \tfrac{2}{47} \cdot d(\hat{a}', \hat{o}') \\ \Leftrightarrow \quad \tfrac{45}{47} \cdot d(\hat{a}', \hat{o}') - m_c &\leq d(a^*, r') \\ \Rightarrow \quad 102970 D \tfrac{k m_c}{\delta^2} &\leq d(a^*, r') \end{aligned}$$

Hence the lemma gives us

$$\begin{aligned} \Delta\phi &\leq 4 \cdot \tfrac{1}{\delta m_s} \left( d(\hat{a}', \hat{o}')^2 - d(\hat{a}, \hat{o})^2 \right) \\ &\leq 4 \cdot \tfrac{1}{\delta m_s} \left( d(\hat{a}', \hat{o}')^2 - (d(\hat{a}', \hat{o}') + \tfrac{\delta}{8} m_s)^2 \right) \\ &= -d(\hat{a}', \hat{o}'). \end{aligned}$$

Furthermore, we have

$$\begin{aligned} C_{Alg} &\leq d(\hat{a}', r') + D \cdot \textstyle\sum_{i=1}^{k} d(a_i, a_i') \\ &\leq d(\hat{a}', \hat{o}') + d(\hat{o}', o^{*'}) + d(o^{*'}, r') + D \cdot \textstyle\sum_{i=1}^{k} d(a_i, a_i') \\ &\leq d(\hat{a}', \hat{o}') + (1 + \tfrac{\delta}{48}) \cdot d(o^{*'}, r') + D \cdot \textstyle\sum_{i=1}^{k} d(a_i, a_i') \end{aligned}$$

and $\Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} \cdot C_{\mathcal{K}} - D \cdot \sum_{i=1}^{k} d(a_i, a_i')$ due to Lemma 14. In total, we get $C_{Alg} + \Delta\phi + \Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} \cdot C_{\mathcal{K}} + 2 \cdot d(o^{*'}, r')$.

□

The resulting competitive ratio $Y \cdot \frac{m_c}{\delta m_s} \cdot c(\mathcal{K}) + 2$ is less than the $\mathcal{O}(\frac{k^2}{\delta^3}) \cdot \frac{m_c}{\delta m_s} + Y \cdot \frac{m_c}{\delta m_s} \cdot c(\mathcal{K})$ bound from the former set of cases. Accounting for the loss due to the transformation of the simulated $k$-Page Migration algorithm, we obtain the following upper bound:

**Theorem 8** *If $m_c \geq (1+\delta) m_s$, the algorithm WMS is $\mathcal{O}(\frac{1}{\delta^4} \cdot k^2 \cdot \frac{m_c}{m_s} + \frac{1}{\delta^3} \cdot k^2 \cdot \frac{m_c}{m_s} \cdot c(\mathcal{K}))$-competitive, where $c(\mathcal{K})$ is the competitive ratio of the simulated $k$-Page Migration algorithm $\mathcal{K}$.*

## 6 Improved Competitiveness on the Line

In this section, we show how to use our offline abstraction from Section 4.2.2 to analyze an algorithm not based on the simulation of an existing $k$-Page Migration algorithm. We take the well known Double Coverage algorithm for the line and adapt it to our setting with restricted movement. Note that we will only state the algorithm and its analysis for $D = 1$. It is easy to see how to extend it to an arbitrary $D \geq 1$.

Our algorithm *Restricted Double Coverage (RDC)* works as follows:

Let $a_1, \ldots, a_k$ be the servers ordered by their position on the line from left to right. If the new request $r'$ is to the left of $a_1$ or to the right of $a_k$, move the respective server $a_i$ a distance of $\min\{d(a_i, r'), (1+\delta)m_s\}$ towards $r'$. Otherwise, $r'$ is located

between two servers $a_i$ and $a_{i+1}$. In this case, let $a_j$ be a closest server to $r'$. Move $a_i$ and $a_{i+1}$ a distance of $\min\{d(a_j, r'), (1+\delta)m_s\}$ towards $r'$.

The following analysis, specifically the potential function $\psi$ and its use in the analysis, is adapted from [11]. The overall structure is the same as in the previous sections.

We use the abstraction $\hat{o}$ from Proposition 2 and use as before the potential

$$\phi := \begin{cases} 4 \cdot d(\hat{a}, \hat{o}) & \text{if } d(\hat{a}, \hat{o}) \leq 107548 \cdot \frac{km_c}{\delta^2} \\ 4 \cdot \frac{1}{\delta m_s} d(\hat{a}, \hat{o})^2 + A & \text{if } 107548 \cdot \frac{km_c}{\delta^2} < d(\hat{a}, \hat{o}) \end{cases}$$

with $A := 4 \cdot (107548 \frac{km_c}{\delta^2} - \frac{1}{\delta m_s}(107548 \frac{km_c}{\delta^2})^2)$.

To handle shorter distances to the request, we use an adaption of the potential used in the proof for DC:

$$\psi := X \cdot \frac{m_c}{m_s} \sum_{i<j} d(a_i, a_j) + Y \cdot \frac{m_c}{m_s} \sum_{i=1}^{k} d(a_i, o_i)$$

where both the online servers $a_i$ and the optimal servers $o_i$ are always sorted from left to right. It is easy to see that every offline solution can be transferred such that the servers never change their ordering. We choose $X = \Theta(\frac{k}{\delta^2})$, $Y = \Theta(\frac{k^2}{\delta^2})$ to be sufficiently large.

Similar to the original analysis we obtain the following result:

**Lemma 17** $\Delta\psi \leq 2Y \frac{m_c}{m_s} \cdot C_{Opt} - \min\{\frac{1}{2}Y, X\} \cdot \frac{m_c}{m_s} \cdot \sum_{i=1}^{k} d(a_i, a_i')$.

*Proof* We distinguish two major cases: First, assume the request is either to the left of $a_1$ or to the right of $a_n$. Both are analogous, hence we deal only with $r'$ being to the left of $a_1$.

If $o_1'$ is not the closest server of the optimal solution to $r'$, then it is to the left of $r'$ and hence $d(o_1', a_1') - d(o_1', a_1) \leq -d(a_1, a_1')$. Otherwise,

$$\begin{aligned} d(o_1', a_1') - d(o_1', a_1) &\leq d(o_1', r') + d(a_1', r') - (d(a_1, r') - d(r', o_1')) \\ &\leq 2d(o_1', r') - d(a_1, a_1'). \end{aligned}$$

For the potential it therefore holds

$$\begin{aligned} \Delta\psi &= X \cdot (k-1)\tfrac{m_c}{m_s} \cdot d(a_1, a_1') + Y \tfrac{m_c}{m_s} \left( d(o_1', a_1') - d(o_1', a_1) + \sum_{i=1}^{k} d(o_i, o_i') \right) \\ &\leq 2Y \cdot \tfrac{m_c}{m_s} \cdot C_{Opt} + X \cdot (k-1)\tfrac{m_c}{m_s} \cdot d(a_1, a_1') - Y \cdot \tfrac{m_c}{m_s} d(a_1, a_1') \\ &\leq 2Y \cdot \tfrac{m_c}{m_s} \cdot C_{Opt} - \tfrac{1}{2}Y \cdot \tfrac{m_c}{m_s} d(a_1, a_1') \end{aligned}$$

for $Y \geq 2kX$.

In the second major case, the request $r'$ is in between $a_i$ and $a_{i+1}$. We assume that $a_i$ is closest to $r'$, the other case is analogous. Both servers move $\min\{d(a_i, r'), (1+\delta)m_s\}$ towards $r'$.

Consider first the term $X \cdot \frac{m_c}{m_s} \sum_{i<j} d(a_i, a_j)$. Servers $a_i$ and $a_{i+1}$ decrease their distance by $2d(a_i, a_i')$. For the servers right of $a_{i+1}$, $a_i$ decreases the distance by

$d(a_i, a_i')$ and $a_{i+1}$ moves away from the same servers by the same distance. The same argument applies for the servers left of $a_i$. Hence
$X \cdot \frac{m_c}{m_s} \sum_{i<j} d(a_i', a_j') - X \cdot \frac{m_c}{m_s} \sum_{i<j} d(a_i, a_j) \leq -2X \frac{m_c}{m_s} \cdot d(a_i, a_i')$.

For the second term $Y \cdot \frac{m_c}{m_s} \sum_{i=1}^{k} d(a_i, o_i)$. If $o_i'$ is to the right of $r'$ or $o_{i+1}'$ to the left of $r'$ then $\sum_{i=1}^{k} d(a_i', o_i') - \sum_{i=1}^{k} d(a_i, o_i') \leq 0$. Otherwise, let $j \in \{i, i+1\}$ such that $o^{*'} = o_j'$. Then

$$\begin{aligned} d(o_j', a_j') - d(o_j', a_j) &\leq d(o_j', r') + d(a_j', r') - (d(a_j, r') - d(r', o_j')) \\ &\leq 2d(o_j', r') - d(a_j, a_j'). \end{aligned}$$

Since $d(o_\ell', a_\ell') - d(o_\ell', a_\ell) \leq d(a_\ell, a_\ell')$, we have
$d(o_i', a_i') - d(o_i', a_i) + d(o_{i+1}', a_{i+1}') - d(o_{i+1}', a_{i+1}) \leq 2d(o^{*'}, r')$.

Summarizing this case, we got

$$\begin{aligned} \Delta\psi &\leq -2X \tfrac{m_c}{m_s} \cdot d(a_i, a_i') + 2Y \tfrac{m_c}{m_s} \cdot d(o^{*'}, r') + Y \tfrac{m_c}{m_s} \cdot \sum_{i=1}^{k} d(o_i, o_i') \\ &\leq 2Y \tfrac{m_c}{m_s} \cdot C_{Opt} - X \cdot \tfrac{m_c}{m_s} \textstyle\sum_{i=1}^{k} d(a_i, a_i'). \end{aligned}$$

□

Now consider the case that $r' \notin inner(o^{*'})$. We have $d(a^{*'}, r') \leq d(o^{*a'}, r') \leq d(o^{*'}, o^{*a'}) + d(o^{*'}, r') \leq (\frac{48960k}{\delta^2} + 1) \cdot d(o^{*'}, r')$. The movement costs are canceled by $\Delta\psi$ as in Lemma 17. The increase of $\phi$ can be bounded as in Lemma 10. In all of the above, the competitive ratio is bounded by $\mathcal{O}(\frac{k^2}{\delta^3} \cdot \frac{m_c}{\delta m_s} + Y \cdot \frac{m_c}{m_s})$.

Finally, we consider the case $r' \in inner(o^{*'})$. In contrast to the previous sections, we can simplify the use of $\phi$ due to the line metric and the DC algorithm:

**Lemma 18** *If $d(a^*, r') > 102970 \frac{km_c}{\delta^2}$ and $r' \in inner(o^{*'})$, then $d(\hat{a}', \hat{o}') - d(\hat{a}, \hat{o}) \leq -\frac{\delta}{2} m_s$.*

*Proof* Since we are on the line, it is guaranteed by the algorithm that the servers which move in the current time step move directly towards $\hat{o}'$, and hence we get $d(a_i', \hat{o}') - d(a_i, \hat{o}') = -d(a_i, a_i') = -(1+\delta)m_s$.

In order to bound the movement of $\hat{o}$, we need to show that $d(o^*, o^{*a}) \geq 2 \cdot 51483 \frac{km_c}{\delta^2}$. We use

$$\begin{aligned} d(a^*, r') &\leq m_c + d(a^{*'}, r') \\ &\leq m_c + d(o^{*a'}, r') \\ &\leq m_c + (1 + \tfrac{\delta^2}{48960k}) \cdot d(o^{*'}, o^{*a'}) \\ \Leftrightarrow \tfrac{1}{1+\frac{\delta^2}{48960k}} (d(a^*, r') - m_c) &\leq d(o^{*'}, o^{*a'}). \end{aligned}$$

The bound follows from $d(a^*, r') > 102970 \frac{km_c}{\delta^2}$.

From Proposition 2 we get $d(\hat{o}, \hat{o}') \leq (1 + \frac{\delta}{8})m_s$ and therefore

$$\begin{aligned} d(\hat{a}', \hat{o}') - d(\hat{a}, \hat{o}) &\leq -d(a_i, a_i') + d(\hat{o}, \hat{o}') \\ &\leq -(1+\delta)m_s + (1 + \tfrac{\delta}{8})m_s \\ &\leq -\tfrac{\delta}{2}m_s \end{aligned}$$

where $i$ is chosen to be closest to $\hat{o}'$. □

**Lemma 19** *If $r' \in inner(o^{*'})$, then $C_{Alg} + \Delta\phi + \Delta\psi \leq 4Y\frac{m_c}{m_s} \cdot C_{Opt}$.*

*Proof* 1. $d(\hat{a}', \hat{o}') \leq 107548 \cdot \frac{km_c}{\delta^2}$: We use

$$\begin{aligned} d(a^{*'}, r') &\leq d(\hat{a}', r') \\ &\leq d(\hat{a}', \hat{o}') + d(\hat{o}', r') \\ &\leq d(\hat{a}', \hat{o}') + 2 \cdot \tfrac{\delta}{48} \cdot d(o^{*'}, o^{*a'}) \\ &\leq (1 + \tfrac{2\delta}{47}) \cdot d(\hat{a}', \hat{o}'). \end{aligned}$$

to get $C_{Alg} + \Delta\phi \leq 6 \cdot d(\hat{a}', \hat{o}') + \sum_{i=1}^{k} d(a_i, a_i')$. Furthermore,

$$\begin{aligned} \Delta\psi &\leq 2Y\tfrac{m_c}{m_s} \cdot C_{Opt} - \min\{\tfrac{1}{2}Y, X\} \cdot \tfrac{m_c}{m_s} \cdot \textstyle\sum_{i=1}^{k} d(a_i, a_i') \\ &\leq 2Y\tfrac{m_c}{m_s} \cdot C_{Opt} - \textstyle\sum_{i=1}^{k} d(a_i, a_i') - (\min\{\tfrac{1}{2}Y, X\} - 1) \cdot m_c \end{aligned}$$

due to Lemma 17. In total, $C_{Alg} + \Delta\phi + \Delta\psi \leq 2Y\frac{m_c}{m_s} \cdot C_{Opt}$ with $X, Y \geq \Omega(\frac{k}{\delta^2})$.

2. $107548 \cdot \frac{km_c}{\delta^2} < d(\hat{a}', \hat{o}')$: We show that the condition of Lemma 18 applies:

$$\begin{aligned} d(\hat{a}', \hat{o}') &\leq d(a^{*'}, \hat{o}') \\ &\leq d(a^{*'}, a^*) + d(a^*, r') + d(r', \hat{o}') \\ &\leq m_c + d(a^*, r') + 2 \cdot \tfrac{\delta}{48} \cdot d(o^{*'}, o^{*a'}) \\ &\leq m_c + d(a^*, r') + \tfrac{2}{47} \cdot d(\hat{a}', \hat{o}') \\ \Leftrightarrow\ \tfrac{45}{47} \cdot d(\hat{a}', \hat{o}') - m_c &\leq d(a^*, r') \\ \Rightarrow\ 102970\tfrac{km_c}{\delta^2} &\leq d(a^*, r'). \end{aligned}$$

Hence the lemma gives us

$$\begin{aligned} \Delta\phi &\leq 4 \cdot \tfrac{1}{\delta m_s}\left(d(\hat{a}', \hat{o}')^2 - d(\hat{a}, \hat{o})^2\right) \\ &\leq 4 \cdot \tfrac{1}{\delta m_s}\left(d(\hat{a}', \hat{o}')^2 - (d(\hat{a}', \hat{o}') + \tfrac{\delta}{2}m_s)^2\right) \\ &= -4 \cdot d(\hat{a}', \hat{o}'). \end{aligned}$$

Furthermore, we have

$$\begin{aligned} C_{Alg} &\leq d(\hat{a}', r') + \textstyle\sum_{i=1}^{k} d(a_i, a_i') \\ &\leq d(\hat{a}', \hat{o}') + d(\hat{o}', o^{*'}) + d(o^{*'}, r') + \textstyle\sum_{i=1}^{k} d(a_i, a_i') \\ &\leq d(\hat{a}', \hat{o}') + (1 + \tfrac{\delta}{48}) \cdot d(o^{*'}, r') + \textstyle\sum_{i=1}^{k} d(a_i, a_i') \end{aligned}$$

and $\Delta\psi \leq 2Y\frac{m_c}{m_s} \cdot C_{Opt} - \min\{\frac{1}{2}Y, X\} \cdot \frac{m_c}{m_s} \cdot \sum_{i=1}^{k} d(a_i, a_i')$ due to Lemma 17. In total, we get $C_{Alg} + \Delta\phi + \Delta\psi \leq 4Y\frac{m_c}{m_s} \cdot C_{Opt}$.

□

The resulting competitive ratio $\mathcal{O}(1) \cdot Y \cdot \frac{m_c}{m_s}$ is less than the $\mathcal{O}(\frac{k^2}{\delta^3} \cdot \frac{m_c}{\delta m_s} + Y \cdot \frac{m_c}{m_s})$ bound from the former set of cases. We therefore conclude:

**Theorem 9** *The RDC algorithm is $\mathcal{O}(\frac{k^2}{\delta^4} \cdot \frac{m_c}{m_s})$-competitive.*

## 7 Open Problems

The gap between the upper and lower bound is closely related to the question of the deterministic upper bound for $k$-Page Migration: Not only would an $\mathcal{O}(k)$-competitive algorithm for $k$-Page Migration directly improve the bound for $D > 1$, it could also give an idea how to improve the analysis of the greedy step in our algorithm, such that the costly transformation of the simulated algorithm would no longer be needed. This would potentially reduce the upper bound by another factor of $k$.

We can argue from previous work that we have a lower bound of $\Omega(1/\delta)$ for when the dimension of the space is as large as $k$. It would be an interesting problem to explore the problem without this resource augmentation on the line, and whether a competitive algorithm exists in this case for $k > 1$. For higher dimensions, intuitively there should be a lower bound depending on $1/\delta$ but decreasing with $k$, as there are more initial places the algorithm can cover as a "guess", and hence smaller distances between the adversary and the algorithm can be achieved at the time where the target position of the request is revealed.

In Section 6 we have shown how to circumvent the use of the transformation by stating an explicit algorithm for the problem. For the original $k$-Server problem, the DC algorithm also works for trees. Similarly, we believe this result is also extendable to continuous trees, where the servers can occupy any place on the paths within the tree. Furthermore, it would be interesting to adopt other algorithms as well if possible. If one wants to use the scheme of analysis we used here, there needs to be a potential function for the algorithm which cancels the cost in every time step where the distance to the requests is small such that our offline abstraction cannot be utilized yet.

If $\Omega(k^2)$ is a lower bound for $k$-Page Migration, this carries over to our model as well. We believe that the main algorithmic idea is suitable to reach an asymptotically optimal competitive ratio, but it remains an open problem to derive a proof of that. The high constants in our proofs are partially due to allowing easier argumentation in certain segments of the proof. There is however also great potential in reducing constants by trying to extend the potential analysis to operate in longer phases instead of doing a step-by-step analysis.

If we allow randomization, we can get $k$-Page Migration algorithms with polylogarithmic competitive ratio from [4]. As discussed in the related work section, the question of the best possible competitive ratio of randomized algorithms for the $k$-Server problem is still open, however we know that a result polylogarithmic in $k$ can be achieved [17]. As our construction is entirely deterministic, apart from potentially the simulated algorithm, it would be interesting whether randomization can be used to significantly improve the competitive ratio. The

desired result would be an algorithm with a competitive ratio polylogarithmic in $k$.

Finally, it would be interesting to know whether a competitive ratio independent of time can be achieved if instead of restricting the distance between consecutive requests, we would analyze the problem under a weak adversary who has less servers than the online algorithm. This problem is also considered for the classical $k$-Server problem [15], where the question of the competitive ratio is also still unresolved. In our problem, this extension could not just replace the restriction to the parameter $m_c$, but also reduce the competitive ratio with respect to the number of servers. For Euclidean metrics, not much is known in this regard, with only a recent bound showing that no matter how high the difference in the number of servers, the dependence on the number of optimal servers can never be removed [6].

**Acknowledgments** This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Centre "On-The-Fly Computing" under the project number 160364472 — SFB 901/3.

**Funding** Open Access funding enabled and organized by Projekt DEAL.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommonshorg/licenses/by/4.0/.

## References

1. Albers, S., Lauer, S.: On list update with locality of reference. J. Comput. Syst. Sci. **82**(5), 627–653 (2016). https://doi.org/10.1016/j.jcss.2015.11.005
2. Angelopoulos, S., Dorrigiv, R., López-Ortiz, A.: List update with locality of reference. In: Proceedings of the 8th Latin American Symposium on Theoretical Informatics (LATIN), pp. 399–410. Springer (2008). https://doi.org/10.1007/978-3-540-78773-0_35
3. Bansal, N., Buchbinder, N., Madry, A., Naor, J.: A polylogarithmic-competitive algorithm for the k-server problem. J. ACM **62**(5), 40:1–40:49 (2015). https://doi.org/10.1145/2783434
4. Bartal, Y., Charikar, M., Indyk, P.: On page migration and other relaxed task systems. Theor. Comput. Sci. **268**(1), 43–66 (2001). https://doi.org/10.1016/S0304-3975(00)00259-0
5. Bartal, Y., Koutsoupias, E.: On the competitive ratio of the work function algorithm for the k-server problem. Theor. Comput. Sci. **324**(2–3), 337–345 (2004). https://doi.org/10.1016/j.tcs.2004.06.001
6. Bienkowski, M., Byrka, J., Coester, C., Jez, L.: Unbounded lower bound for k-server against weak adversaries. In: Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC), pp. 1165–1169. ACM (2020). https://doi.org/10.1145/3357713.3384306
7. Bienkowski, M., Byrka, J., Mucha, M.: Dynamic beats fixed: On phase-based algorithms for file migration. ACM Trans. Algorithms **15**(4), 46,1–46,21 (2019). https://doi.org/10.1145/3340296
8. Black, D.L., Sleator, D.D.: Competitive algorithms for replication and migration problems. Tech. Rep. CMU-CS-89-201, Department of Computer Science Carnegie-Mellon University (1989)
9. Borodin, A., Irani, S., Raghavan, P., Schieber, B.: Competitive paging with locality of reference. J. Comput. Syst. Sci. **50**(2), 244–258 (1995). https://doi.org/10.1006/jcss.1995.1021

10. Bubeck, S., Cohen, M.B., Lee, Y.T., Lee, J.R., Madry, A.: k-server via multiscale entropic regularization. In: Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC), pp. 3–16 (2018). https://doi.org/10.1145/3188745.3188798
11. Chrobak, M., Karloff, H.J., Payne, T.H., Vishwanathan, S.: New results on server problems. SIAM J. Discrete Math. **4**(2), 172–181 (1991). https://doi.org/10.1137/0404017
12. Feldkord, B., Meyer auf der Heide, F.: The mobile server problem. TOPC **6**(3), 14:1–14:17 (2019). https://doi.org/10.1145/3364204
13. Fiat, A., Karp, R.M., Luby, M., McGeoch, L.A., Sleator, D.D., Young, N.E.: Competitive paging algorithms. J. Algorithms **12**(4), 685–699 (1991). https://doi.org/10.1016/0196-6774(91)90041-V
14. Fiat, A., Mendel, M.: Truly online paging with locality of reference. In: 38th Annual Symposium on Foundations of Computer Science (FOCS), pp. 326–335. IEEE Computer Society (1997). https://doi.org/10.1109/SFCS.1997.646121
15. Koutsoupias, E.: Weak adversaries for the k-server problem. In: Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS), pp. 444–449 (1999). https://doi.org/10.1109/SFFCS.1999.814616
16. Koutsoupias, E., Papadimitriou, C.H.: On the k-server conjecture. J. ACM **42**(5), 971–983 (1995). https://doi.org/10.1145/210118.210128
17. Lee, J.R.: Fusible hsts and the randomized k-server conjecture. In: Proceedings of the 59th Annual Symposium on Foundations of Computer Science (FOCS), pp. 438–449 (2018). https://doi.org/10.1109/FOCS.2018.00049
18. Manasse, M.S., McGeoch, L.A., Sleator, D.D.: Competitive algorithms for server problems. J. Algorithms **11**(2), 208–230 (1990). https://doi.org/10.1016/0196-6774(90)90003-W
19. Rudec, T., Baumgartner, A., Manger, R.: A fast work function algorithm for solving the k-server problem. CEJOR **21**(1), 187–205 (2013). https://doi.org/10.1007/s10100-011-0222-7
20. Rudec, T., Manger, R.: A fast approximate implementation of the work function algorithm for solving the k-server problem. CEJOR **23**(3), 699–722 (2015). https://doi.org/10.1007/s10100-014-0349-4
21. Westbrook, J.R.: Randomized algorithms for multiprocessor page migration. SIAM J. Comput. **23**(5), 951–965 (1994). https://doi.org/10.1137/S0097539791199796
22. Zhang, W., Cheng, Y.: A new upper bound on the work function algorithm for the k-server problem. Journal of Combinatorial Optimization. https://doi.org/10.1007/s10878-019-00493-z (2019)

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.