The Mobile Server Problem*

Björn Feldkord
Heinz Nixdorf Institut & Computer Science Dept.,
Paderborn University
Fürstenallee 11
Paderborn, Germany
bjoernf@mail.upb.de

Friedhelm Meyer auf der Heide
Heinz Nixdorf Institut & Computer Science Dept.,
Paderborn University
Fürstenallee 11
Paderborn, Germany
fmadh@upb.de

ABSTRACT

We introduce the mobile server problem, inspired by current trends to move computational tasks from cloud structures to multiple devices close to the end user. An example for this are embedded systems in autonomous cars that communicate in order to coordinate their actions.

Our model is a variant of the classical Page Migration Problem. More formally, we consider a mobile server holding a data page. The server can move in the Euclidean space (of arbitrary dimension). In every round, requests for data items from the page pop up at arbitrary points in the space. The requests are served, each at a cost of the distance from the requesting point and the server, and the mobile server may move, at a cost D times the distance traveled for some constant D. We assume a maximum distance m the server is allowed to move per round.

We show that no online algorithm can achieve a competitive ratio independent of the length of the input sequence in this setting. Hence we augment the maximum movement distance of the online algorithms to $(1 + \delta)$ times the maximum distance of the offline solution. We provide a deterministic algorithm which is simple to describe and works for multiple variants of our problem. The algorithm achieves almost tight competitive ratios independent of the length of the input sequence.

KEYWORDS

page migration; online algorithms; competitive analysis; resource augmentation

ACM Reference format:
Björn Feldkord and Friedhelm Meyer auf der Heide. 2017. The Mobile Server Problem. In *Proceedings of SPAA '17, Washington, DC, USA, July 24–26, 2017,* 7 pages.
https://doi.org/10.1145/3087556.3087575

*This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Centre "On-The-Fly Computing" (SFB 901).

SPAA '17, July 24–26, 2017, Washington, DC, USA

ACM ISBN 978-1-4503-4593-4/17/07…$15.00
https://doi.org/10.1145/3087556.3087575

1 INTRODUCTION

Motivated by their large consumption of computational resources, a growing number of applications were shifted from a single machine at some end user to large computing centers. These centers may have networks of processors working on a common memory to execute some common computational task. This development has motivated lots of research problems regarding a static network of machines with some common resources such as memory and bandwidth of common communication channel (see [11] for a survey of scheduling problems for such networks).

Recently, another trend can be observed where the computation is shifting back to the user. This is commonly referred to as "edge computing" and involves a dynamic network with lots of (mobile) devices which are located close to the user. The machines may even be embedded systems such as in autonomous cars which need to share data in order to coordinate. As a result, the structure of the communication network might be much more complex in the sense that devices may join and leave the network and data may be shifted to nodes which are altering their physical location throughout the computation. For examples, see [1, 10].

The Page Migration Problem is a simple model for approaching the problem with shared memory in a cloud computing scenario. In the classical version of this problem we consider a memory page which is shared by multiple processors which are connected in a network. Only one processor can hold the page at a time. Processors request data from the page which incurs cost proportional to the distance within the network. In order to reduce these costs, the page may be moved to another processor; this however incurs cost proportional to the distance in the network times the size of the page.

In this paper, we present with the Mobile Server Problem a simple model which captures both the idea of Page Migration, namely modeling costs for accessing data items from an indivisible data page, and allowing mobility of data servers in order to improve the overall performance of the system.

Abstracting from specific network topologies, we replace the network graph with the Euclidean space, such that, at every step, an arbitrary (finite) number of requests for data items can appear anywhere in the space. In order to improve the overall performance, the mobile server holding the memory page can move to any point in the plane. For the purpose of bounding the time needed for a round, we limit the allowed distance the server can move in a step. The cost for a step is composed of the sum of the distances between the mobile server and the positions of the clients issuing the requests, plus the distance the server moves, weighted with a cost value D.

The Page Migration Problem is naturally often considered as an online problem as data requests may not be known at the start of a larger computation. This is especially true in our scenario where even the participating devices may not be known in advance. Hence, we also consider the Moving Server Problem as an online problem.

We mainly consider the variant where, in a step, a server may move knowing the position of the clients, but answer their requests afterwards. We also discuss other variants.

1.1 Related Work

The Page Migration Problem was first considered as an online problem by Black and Sleator [8] who gave 3-competitive algorithms for arbitrary trees and the complete graph. It was also shown that 3 is the best possible competitive ratio even if the given network just consists of two processors. This lower bound also holds for randomized algorithms against adaptive adversaries.

The first deterministic algorithm for general graphs was given by Westbrook in 1994 [15] which was called *Move-To-Min* as the strategy is to move to the optimal point in regards to the last D requests. The competitive ratio of this algorithm is 7.

For the randomized solutions, there is a simple 3-competitive algorithm called the *Coin-Flip Algorithm* which is simple to describe and also works against adaptive online adversaries. A more involved solution gives a 2.62-competitive algorithm against oblivious adversaries. These algorithms can be found in a paper by Westbrook [15].

An overview over these results including better results for the deterministic case can be found in a survey by Yair Bartal [3].

The Page Migration Problem belongs to the class of *relaxed task systems*. This was used to derive a deterministic solution for the problem with multiple copies of the page. The adaption within the framework is made from an online algorithm for the k-Server Problem [4].

The k-Server Problem can be formulated in terms of the Page Migration Problem with the restriction that requests which appear in the network must be satisfied by moving one of the k identical copies of the page to the location of the request. The competitive ratio of the problem is shown to be $\Omega(k)$. An overview of the most important results can be found in a survey by Elias Koutsoupias [14]. A recent paper by Böckenhauer et. al. also studied the advice complexity of the k-Server Problem where they design a $\frac{1}{1-2\sin(\frac{\pi}{2^b})}$-competitive algorithm reading b bits per time step [5].

In contrast to the k-Server model, Albers and Koga studied a variant with multiple different pages (only one copy per page) but where the processors only can hold one page at a time. The competitive ratio for this model can be bounded by a constant [2].

The Page Migration Problem has already been studied within the Manhattan and Euclidean plane, but without restricting the moving distance of the page in each time step [9, 13]. The Manhattan and Euclidean plane were also considered for the 3-Server Problem for which (almost) optimal online algorithms were given [6].

All the mentioned variants so far assumed a static network where the distances between the nodes do not change over time. Bienkowski et. al. considered a scenario where distances between nodes could change over time. The change in distance was done either by an adversary or determined by a stochastic process. In this scenario the competitive ratio depends both on the size of the page and the number of processors [7].

Since the competitive ratio for our problem does depend on the length of the input sequence, we analyze it using the concept of *resource augmentation*. This technique of giving the online algorithm slightly more power in some sense to get a bounded competitive ratio was first used by Kalyanasundaram and Pruhs for scheduling problems where the online algorithms were given slightly faster processors than the offline solution to improve from an unbounded ratio to a competitive ratio only dependent on the augmentation factor [12].

1.2 Our Contribution

We present the Mobile Server Problem introduced above (and formally described in Section 2). Our first results are lower bounds for the competitive ratio: We prove that no online algorithm can achieve a competitive ratio independent of the number of rounds for this problem.

We therefore consider the problem in a setting where the maximum distance a server may move is augmented by a factor of $(1+\delta)$ for the online algorithm. We show that a simple algorithm is sufficient to achieve a competitive ratio independent of the length of the input sequence for several variants of the problem.

In particular, we describe a deterministic algorithm which is $O(\frac{1}{\delta^{3/2}})$-competitive in the Euclidean Plane when the number of requests per round is fixed. We also briefly sketch how to modify our analysis to work for a varying number of requests per round and for a scenario where requests must be answered before the page can be moved.

To complement these results we show lower bounds for all of the variants which also hold for randomized algorithms against oblivious adversaries. Overall we get tight bounds for the 1-dimensional Euclidean space and almost tight bounds, up to a factor of $\frac{1}{\sqrt{\delta}}$, for arbitrary dimensions.

2 OUR MODEL

The model of the Moving Server Problem adapts some notation from the Page Migration Problem to allow for a better understanding and an easy comparison of the results.

We consider a mobile server holding a memory page, located at a point P_t in the Euclidean Plane at time t. Time is discrete and divided into steps. We refer to the length of an input sequence as the number of time steps denoted by T.

In each time step, clients can request data items from the page. Let r_t be the number of clients requesting data items in step t. These are represented by points $v_{t,1}, \ldots, v_{t,r_t}$ in the plane. The server can move in every time step by a distance of at most m, i.e. $d(P_t, P_{t+1}) \leq m$.

The cost for answering a request issued at position v when the server is located at position P is $d(P, v)$. Moving the server from position P_t to P_{t+1} induces cost $D \cdot d(P_t, P_{t+1})$ for some constant $D \geq 1$. An algorithm operating on an input sequence for the problem can decide in each step where to move the page under the given restriction of the distance.

The total costs of an algorithm *Alg* on a given input sequence are defined as follows:

$$C_{Alg} = \sum_{t=1}^{T} \left(D \cdot d(P_t, P_{t+1}) + \sum_{i=1}^{r_t} d(P_{t+1}, v_{t,i}) \right)$$

Note that in this definition, the page may be moved upon knowing the current requests. The requests are however served after the page has been moved, hence the costs are proportional to the distance to P_{t+1}.

We will discuss the implication of this definition in contrast to moving the page after serving the requests in the following section.

3 LOWER BOUNDS

In this section we show how the different parameters of our model influence the quality of the best possible approximation by an online algorithm.

In this paper we only present a deterministic online algorithm for the Mobile Server Problem, however we formulate the lower bounds for the expected competitive ratio of randomized online algorithms against oblivious adversaries. These lower bounds carry over to the deterministic case and to randomized online algorithms against stronger (adaptive) adversaries.

It should also be noted that these lower bounds hold in the Euclidean space for an arbitrary dimension.

For the lower bounds we use Yao's Min-Max Principle [16] which allows us to construct the lower bounds by generating a randomized sequence and examine the expected competitive ratio of a deterministic online algorithm.

First we show that without use of resource augmentation, there does not exist an online algorithm with a competitive ratio independent of T, even if there is only one request per time step.

Theorem 3.1. *Every randomized online algorithm for the Mobile Server Problem has a competitive ratio of* $\Omega(\sqrt{T}/D)$ *against an oblivious adversary.*

Proof. We consider a sequence of x time steps with one request each on the starting position of the server. Consider two opposite directions from the starting position which we will refer to as left and right. The adversary decides with probability $\frac{1}{2}$ at the beginning of the first step to either move its server a distance m to the left or to the right for the first x time steps. The cost for the adversary is at most $xDm + m \cdot \sum_{i=1}^{x} i \leq xDm + mx^2$ for these first x steps.

After these x steps, with probability $\frac{1}{2}$ the server of the online algorithm has a distance of at least xm to the server of the adversary.

For the remaining $T - x$ steps of the sequence the adversary issues requests on the position of its server and moves it a distance of m towards the same direction it already did during the first x steps. The costs for the adversary are $(T - x)Dm$ while the costs for the online algorithm are at least $(T - x) \cdot xm$ with probability $\frac{1}{2}$. By choosing $x = \sqrt{T}$ the expected competitive ratio is $\Omega(\frac{\sqrt{T}}{D})$. □

In order to have a chance to achieve a competitive ratio independent from T, we augment the maximum distance by which the server may move in every time step for the respective online algorithm.

We consider online algorithms which, in every round, can move their server by a distance of $(1 + \delta)m$ for some $\delta \in (0, 1]$. We do not consider bigger values for δ in this paper since 1 is a natural lower bound for all online problems. Hence asymptotically, for an online algorithm it is sufficient to utilize a distance of at most $2m$ to match the lower bound of $\frac{1}{\delta}$.

Theorem 3.2. *Let R_{min} and R_{max} be the minimum and maximum number of requests per time step. Every randomized online algorithm using an augmented maximum moving distance of at most $(1 + \delta)m$ has an expected competitive ratio of $\Omega(\frac{1}{\delta} \cdot \frac{R_{max}}{R_{min}})$ against an oblivious adversary.*

Proof. We use the same sequence as in the previous theorem to separate the servers of the adversary and the online algorithm:

For x time steps, there are R_{min} requests in every step on the starting position of the server. The adversary moves its server a distance m for x steps to the left or right with probability $\frac{1}{2}$ each, such that the distance between the two servers is at least xm with probability at least $\frac{1}{2}$ after these steps. The costs for the adversary are at most $Dxm + R_{min}mx^2$.

The adversary now issues R_{max} requests at the position of its server and moves it by a distance m in same direction as in the previous steps. This is done for exactly as many time steps as it would take the server of the online algorithm to "catch up" with the server of the adversary when it it is at least a distance xm away from the adversary's server and moves towards it with the maximum distance in each round. The necessary number of steps for that are $\frac{x}{\delta}$, since the distance between the two servers decreases by at most δm in every round.

The costs the online algorithm has to pay, in case the distance between the two servers is at least xm at the beginning of this phase, for serving the requests are minimized when the algorithm moves the server with a maximum distance towards the position of the adversary's server in every time step. The costs for the online algorithm are therefore at least

$$\begin{aligned} R_{max} \cdot \textstyle\sum_{i=1}^{\frac{x}{\delta}} (xm - i\delta m) &= R_{max} \left(\frac{mx^2}{2\delta} - \frac{mx}{2} \right) \\ &\geq \frac{1}{4} \frac{R_{max} mx^2}{\delta} \end{aligned}$$

with probability at least $\frac{1}{2}$ by choosing $x \geq 2\delta$. The adversary pays $\frac{x}{\delta} Dm$ in this phase.

By choosing x sufficiently large, the total costs of the adversary sum up to at most $3R_{min}mx^2$ over both described phases.

The adversary can now repeat the two phases in a circular way arbitrarily often. The costs can be analyzed as above since the one probabilistic choice the adversary does is made independently of the behavior of the online algorithm and its own former behavior. The resulting expected competitive ratio is $\Omega(\frac{R_{max}}{\delta R_{min}})$. □

We observe that as a special case, when $R_{max} = R_{min}$ the lower bound of the competitive ratio becomes independent of the number of requests in each round. In the following section we show that this is indeed possible to achieve in this scenario.

We finally address our decision to allow the algorithms to move the page before answering the requests. Consider the scenario in which an algorithm has to answer the requests before moving the server. It can be shown that the competitive ratio of such algorithms

depend on the number of requests in each time step even if it is fixed throughout the whole sequence.

Theorem 3.3. *If in every time step, requests must be answered before moving the server, every randomized algorithm has an expected competitive ratio of $\Omega(r/D)$ against an oblivious adversary if the number of requests in each time step is fixed to a constant r.*

Proof. Consider the following two time steps: In the first step, r requests are issued at the common position of the servers. The adversary can now move the server to a position such that with probability at least $\frac{1}{2}$, the distance between the two servers is at least m. This can be done by throwing a fair coin and the moving to the left or right as in the previous theorems.

In the second step, r requests are issued at the position of the adversary's page. The two steps may be repeated in a cyclic manner since the random choice of the adversary does not depend on any former time steps.

The costs for the online algorithm for one repetition of these two steps are at least rm with probability at least $\frac{1}{2}$ while the costs of the adversary are at most Dm. □

4 A SIMPLE ALGORITHM

In this section we provide a simple algorithm which achieves an optimal competitive ratio on line segments and a near optimal competitive ratio for the Euclidean plane. We analyze the algorithm in detail only for the case of a fixed r since this is the most insightful case. The upper bounds for the other variants are briefly described in the next section.

The algorithm *Move-to-Center (MtC)* works as follows:

Assume the algorithm has its server located at a point P_{Alg} and receives requests $v_1, \ldots, v_r$. Let c be the point minimizing $\sum_{i=1}^{r} d(c, v_i)$. MtC moves the server towards c for a distance of $\min\{1, \frac{r}{D}\} \cdot d(P_{Alg}, c)$ if this distance is less than $(1+\delta)m$. Otherwise it moves the server a distance of $(1+\delta)m$ towards c.

The remainder of this chapter is devoted to prove the following theorem:

Theorem 4.1. *MtC is $O(\frac{1}{\delta})$-competitive on the infinite line and $O(\frac{1}{\delta^{3/2}})$-competitive in the Euclidean plane using an augmented maximum moving distance of $(1+\delta)m$ for a fixed number r of requests per time step and some $\delta \in (0, 1]$.*

For the analysis, we use a potential function argument. Therefore we fix an arbitrary time step in the input sequence. First, we introduce some notation for this step.

By P_{Alg} and P'_{Alg}, we denote the position of the algorithm's server before and after moving it. P_{Opt} and P'_{Opt} will be used for the optimal server positions before and after moving respectively.

For the requests, we use $v_1, \ldots, v_r$ for the points and c as the point minimizing the sum of distances as above. For the rest of the analysis, we assume that there are r requests on c. This assumption only costs us an additive constant of 1 in the competitive ratio, which can be seen from the following estimation:

$$\begin{aligned} \textstyle\sum_{i=1}^{r} d(P'_{Alg}, v_i) &\leq r \cdot d(P'_{Alg}, c) + \textstyle\sum_{i=1}^{r} d(c, v_i) \\ &\leq r \cdot d(P'_{Alg}, c) + C_{Opt} \end{aligned}$$

where C_{Opt} are the optimal costs in the respective time step.

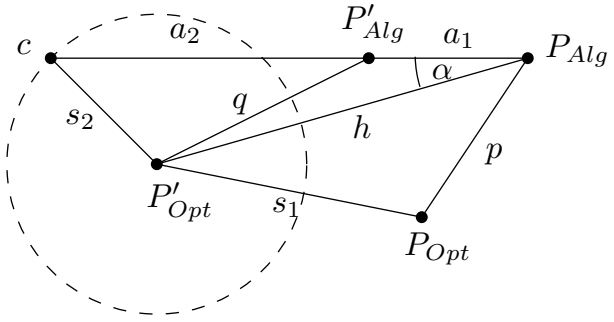

Figure 1: Illustration of relevant points and distances for estimating the potential difference. $h' = d(P_{Opt}, P'_{Alg})$ is omitted for better overview.

For better readability, we define the following abbreviations which we use for the distances and sometimes also for the lines as geometric objects:
$a_1 := d(P_{Alg}, P'_{Alg})$, $a_2 := d(P'_{Alg}, c)$, $s_1 := d(P_{Opt}, P_{Opt'})$,
$s_2 := d(P'_{Opt}, c)$, $p := d(P_{Opt}, P_{Alg})$, $h := d(P'_{Opt}, P_{Alg})$,
$h' := d(P_{Opt}, P'_{Alg})$ and $q := d(P'_{Opt}, P'_{Alg})$.
An illustration can be found in Figure 1.

Using this notation, the costs of the online algorithm are

$$C_{Alg} = Da_1 + ra_2$$

and the optimal costs are

$$C_{Opt} = Ds_1 + rs_2.$$

For the costs of the online algorithm, we often use the following estimation:

$$\begin{aligned} C_{Alg} &= Da_1 + ra_2 \\ &\leq D(p + s_1 + q) + r(q + s_2) \end{aligned} \tag{1}$$

For server locations P_{Alg} and P_{Opt} of the online algorithm and the optimal solution respectively, the potential ϕ is defined as

$$\phi(P_{Opt}, P_{Alg}) := \begin{cases} 8\frac{r}{\delta m} \cdot d(P_{Opt}, P_{Alg})^2, \\ \qquad \text{for } d(P_{Opt}, P_{Alg}) > \delta\frac{Dm}{4r} \\ 2D \cdot d(P_{Opt}, P_{Alg}), \\ \qquad \text{otherwise} \end{cases}.$$

We start with an estimation of the difference $h - q$ which is essential for bounding the potential difference in the Euclidean space.

Lemma 4.2. *If $s_2 \leq \frac{\sqrt{\delta}}{1+\frac{1}{2}\delta} a_2$, then $h - q \geq \frac{1+\frac{1}{2}\delta}{1+\delta} a_1$.*

Proof. We want to get a lower bound for $h - q$ given fixed values for h, s_2 and a_1. q is maximized by choosing the angle α between a_1 and h as large as possible. This can be done by setting the angle between s_2 and a_2 to 90 degrees as shown in Figure 2.

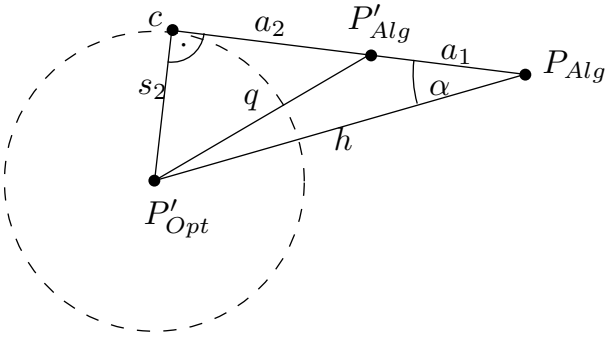

Figure 2: Setting where q is maximized given h, s_2 and a_1.

Making use of the 90 degree angle we get $h^2 = s_2^2 + (a_1 + a_2)^2$ and $q^2 = s_2^2 + a_2^2$. Let $s_2 = \epsilon \cdot a_2$. Then

$$\begin{aligned} h - q &= \sqrt{(\epsilon a_2)^2 + (a_1 + a_2)^2} - \sqrt{(\epsilon a_2)^2 + a_2^2} \\ &= \sqrt{(1+\epsilon^2)a_2^2 + 2a_1a_2 + a_1^2} - \sqrt{1+\epsilon^2}a_2 \\ &\geq \sqrt{(1+\epsilon^2)a_2^2 + 2\sqrt{1+\epsilon^2}a_2\frac{a_1}{\sqrt{1+\epsilon^2}} + \frac{a_1^2}{1+\epsilon^2}} \\ &\quad -\sqrt{1+\epsilon^2}a_2 \\ &= \frac{a_1}{\sqrt{1+\epsilon^2}}. \end{aligned}$$

It therefore holds

$$\epsilon \leq \sqrt{\frac{(1+\delta)^2}{(1+\frac{1}{2}\delta)^2} - 1} \Rightarrow h - q \geq \frac{1+\frac{1}{2}\delta}{1+\delta}a_1.$$

Since

$$\sqrt{\frac{(1+\delta)^2}{(1+\frac{1}{2}\delta)^2} - 1} = \frac{\sqrt{\delta + \frac{3}{4}\delta^2}}{1+\frac{1}{2}\delta} \geq \frac{\sqrt{\delta}}{1+\frac{1}{2}\delta}$$

this implies the lemma. □

4.1 Analysis for $r > D$

In the following, we analyze the algorithm for the case $r > D$ for both the line segment and the Euclidean plane. Most arguments apply to both cases, if there is a difference between the spaces it is stated explicitly in the respective case.

For each of the listed cases we bound the potential difference $\Delta\phi = \phi(P'_{Opt}, P'_{Alg}) - \phi(P_{Opt}, P_{Alg})$ and the costs of the online algorithm as expressions in C_{Opt}.

1. $p \leq \delta\frac{Dm}{4r} \leq \delta m$: The algorithm has the possibility to move its server to the same position as the optimal solution (or closer to c). Hence $a_2 \leq s_2$. If $q \leq \delta\frac{Dm}{4r}$ we have

$$\begin{aligned} \Delta\phi &\leq 2D(q-p) \\ &\leq 2r(s_2 + a_2) - 2Dp \\ &\leq 4C_{Opt} - 2Dp \end{aligned}$$

and

$$\begin{aligned} C_{Alg} &= Da_1 + ra_2 \\ &\leq D(p + s_1 + q) + ra_2 \\ &\leq D(p + s_1) + r(s_2 + 2a_2) \\ &\leq 3C_{Opt} + Dp. \end{aligned}$$

Otherwise it holds $q > \delta\frac{Dm}{4r} \geq p$ and we get

$$\begin{aligned} \Delta\phi &= 8\tfrac{r}{\delta m} \cdot q^2 - 2D \cdot p \\ &\leq 8\tfrac{r}{\delta m}(q^2 - p^2) \\ &= 8\tfrac{r}{\delta m}(q+p)(q-p) \\ &\leq 8\tfrac{r}{\delta m} \cdot 2 \cdot q \cdot (2+\delta)m \\ &\leq 48\tfrac{r}{\delta}(s_2 + a_2) \end{aligned}$$

and

$$\begin{aligned} C_{Alg} &= Da_1 + ra_2 \\ &\leq D(s_1 + 2(s_2 + a_2)) + ra_2 \\ &\leq 5C_{Opt}. \end{aligned}$$

2. $p > \delta\frac{Dm}{4r}$ and $q \leq \delta\frac{Dm}{4r}$: If $p < \delta m$, we have $a_2 \leq s_2$ as before. The potential difference can be estimated by

$$\begin{aligned} \Delta\phi &= 2Dq - 8\tfrac{r}{\delta m}p^2 \\ &\leq 2D(a_2 + s_2) - 2Dp \\ &\leq 4C_{Opt} - 2Dp \end{aligned}$$

and

$$\begin{aligned} C_{Alg} &= Da_1 + ra_2 \\ &\leq D(p + s_1 + q) + ra_2 \\ &\leq C_{Opt} + 2Dp. \end{aligned}$$

Else, we have

$$\begin{aligned} \Delta\phi &= 2Dq - 8\tfrac{r}{\delta m}p^2 \\ &\leq 2Dq - 8rp \\ &\leq -6rp \end{aligned}$$

and with (1) we get

$$C_{Alg} \leq C_{Opt} + 3rp.$$

In all of the following cases, we have $p > \delta\frac{Dm}{4r}$ and $q > \delta\frac{Dm}{4r}$.

3. $q - h \leq -(1+\frac{\delta}{2})m$ or $h' - p \leq -(1+\frac{\delta}{2})m$: Since $s_1 \leq m$, we have $q - p \leq -\frac{\delta}{2}m$. Therefore

$$\begin{aligned} \Delta\phi &\leq 8\tfrac{r}{\delta m}(q+p)(q-p) \\ &\leq 8\tfrac{r}{\delta m}(q+p)(-\tfrac{\delta}{2}m) \\ &\leq -4r(q+p) \end{aligned}$$

and (1) gives us

$$C_{Alg} \leq C_{Opt} + 2r(p+q).$$

4. $q - h > -(1+\frac{\delta}{2})m$ and $h' - p > -(1+\frac{\delta}{2})m$ and $p \geq 4m$: We estimate a_2 in two different ways: First assume our given space is the Euclidean plane. Since $\frac{1+\frac{\delta}{2}}{1+\delta}a_1 \geq (1+\frac{\delta}{2})m$ if $a_1 = (1+\delta)m$, we get $\sqrt{\delta}a_2 \leq s_2$ according to Lemma 4.2.

If the given space is a line segment, $a_1 = (1+\delta)m$ implies that the two algorithms must have moved their servers in opposite directions. This gives us $a_2 \leq s_2$.

For both cases, if $a_1 < (1+\delta)m$, then $a_2 = 0 \leq s_2$.

Now, if $\frac{1}{2}p \leq q$ we may estimate

$$\begin{aligned} \Delta\phi &\leq 8\tfrac{r}{\delta m}(q+p)(q-p) \\ &\leq 8\tfrac{r}{\delta m}(3q)(3m) \\ &\leq 72\tfrac{r}{\delta}(s_2 + a_2) \end{aligned}$$

and

$$\begin{aligned} C_{Alg} &= Da_1 + ra_2 \\ &\leq D(s_1 + 2(s_2 + a_2)) + ra_2 \\ &\leq \tfrac{5}{\sqrt{\delta}}C_{Opt}. \end{aligned}$$

Otherwise we use

$$\begin{aligned}\Delta\phi &\leq 8\tfrac{r}{\delta m}(q+p)(q-p)\\ &\leq 8\tfrac{r}{\delta m}(q+p)(-\tfrac{1}{2}p)\\ &\leq -2\tfrac{r}{\delta}p\end{aligned}$$

and

$$\begin{aligned}C_{Alg} &= Da_1 + ra_2\\ &\leq \tfrac{3}{\sqrt{\delta}}C_{Opt} + Dp.\end{aligned}$$

5. $q - h \leq -(1+\frac{\delta}{2})m$ and $h' - p \leq -(1+\frac{\delta}{2})m$ and $p < 4m$: We get $\sqrt{\delta}a_2 \leq s_2$ and $a_2 \leq s_2$ for the Euclidean plane and the line segment respectively as before. Since $q \leq p + 3m$ we have

$$\begin{aligned}\Delta\phi &\leq 8\tfrac{r}{\delta m}(q+p)(q-p)\\ &\leq 8\tfrac{r}{\delta m}\dot{1}1m\cdot q - 8\tfrac{r}{\delta m}(q+p)\delta\tfrac{Dm}{4r}\\ &\leq 88\tfrac{1}{\delta}(s_2+a_2) - 2D(q+p)\end{aligned}$$

and

$$\begin{aligned}C_{Alg} &= Da_1 + ra_2\\ &\leq \tfrac{3}{\sqrt{\delta}}C_{Opt} + Dp.\end{aligned}$$

In all cases we have $C_{Alg} + \Delta\phi \leq O(\frac{1}{\delta^{3/2}})\cdot C_{Opt}$.

4.2 Analysis for $r \leq D$

For $r \leq D$ we first give a detailed analysis for the Euclidean plane and then briefly describe how to modify the necessary parts to work for the line segment such that the competitive ratio becomes $O(\frac{1}{\delta})$.

1. Let $p \leq \delta\frac{Dm}{4r}$ and $q \leq \delta\frac{Dm}{4r}$. First we consider the case $s_2 \leq \frac{\sqrt{\delta}}{1+\frac{\delta}{2}}a_2$. We bound the potential difference by

$$\begin{aligned}\Delta\phi &= 2D(q-p)\\ &\leq 2D(q-h+h-p)\\ &\leq -2D\tfrac{1+\frac{\delta}{2}}{1+\delta}a_1 + 2Ds_2.\end{aligned}$$

If $a_1 = \frac{r}{D}(a_1+a_2)$ then $\Delta\phi \leq -2r(a_1+a_2)$. We have $C_{Alg} \leq 2r(a_1+a_2)$.

Otherwise $a_1 = (1+\delta)m$ then $\Delta\phi \leq -2D(1+\frac{\delta}{2})m$. In this case $p \leq \delta\frac{Dm}{4r}$ can be used to bound C_{Alg}.

The second case is $s_2 > \frac{\sqrt{\delta}}{1+\frac{\delta}{2}}a_2$. Either $\frac{1}{2}p \leq q$ immediately gives the desired bound or we have $\Delta\phi \leq -Dp$.

We use the same argumentation for $p > \delta\frac{Dm}{4r}$ since in this case $-8\frac{r}{\delta m}p^2 \leq -2Dp$.

2. $p > \delta\frac{Dm}{4r}$ and $q > \delta\frac{Dm}{4r}$: As before we start with the case that $s_2 \leq \frac{\sqrt{\delta}}{1+\frac{\delta}{2}}a_2$. We have

$$\begin{aligned}\Delta\phi &= 8\tfrac{r}{\delta m}(q^2-p^2)\\ &\leq 8\tfrac{r}{\delta m}(q+p)(q-p)\\ &\leq 8\tfrac{r}{\delta m}(q+p)(-\tfrac{1+\frac{\delta}{2}}{1+\delta}a_1 + s_1).\end{aligned}$$

If $a_1 = (1+\delta)m$ then $\Delta\phi \leq -4r(q+p)$ and $C_{Alg} \leq 2r(a_1+a_2) \leq 2C_{Opt} + 2r(p+q)$.

Else, $a_1 = \frac{r}{D}(a_1+a_2)$ and using $a_1 + a_2 \leq 2\frac{Dm}{r}$ we get

$$\begin{aligned}\Delta\phi &\leq 8\tfrac{r}{\delta m}(2(a_1+a_2)+2s_2+s_1)\\ &\quad\cdot(-\tfrac{3r}{4D}(a_1+a_2)+s_1)\\ &\leq -6r(a_1+a_2) + 40\tfrac{D}{\delta}s_1 + 8\tfrac{r}{\delta}s_2.\end{aligned}$$

For the online algorithm we have $C_{Alg} \leq 2r(a_1+a_2)$.

Now consider the case $s_2 > \frac{\sqrt{\delta}}{1+\frac{\delta}{2}}a_2$. If $\frac{1}{2}p \leq q$ we use it to get

$$\begin{aligned}\Delta\phi &= 8\tfrac{r}{\delta m}(q+p)(q-p)\\ &\leq 8\tfrac{r}{\delta m}(3q)(3m)\\ &= 216\tfrac{r}{\delta^{3/2}}s_2\end{aligned}$$

and

$$\begin{aligned}C_{Alg} &\leq 2r(a_1+a_2)\\ &\leq \tfrac{5}{\sqrt{\delta}}C_{Opt}.\end{aligned}$$

Otherwise we have

$$\begin{aligned}\Delta\phi &= 8\tfrac{r}{\delta m}(q+p)(q-p)\\ &\leq 8\tfrac{r}{\delta m}(q+p)(-\tfrac{1}{2}p)\\ &= -D(q+p)\end{aligned}$$

and $C_{Alg} \leq \frac{1}{\sqrt{\delta}}C_{Opt} + D(p+q)$.

Again, the arguments also apply to $q \leq \delta\frac{Dm}{4r}$ due to $-2Dq \leq -8\frac{r}{\delta m}q^2$.

To get the bound for the line segment, in each of the two big cases replace the distinction whether $s_2 \leq \frac{\sqrt{\delta}}{1+\frac{\delta}{2}}a_2$ by $s_2 \leq a_2$. Also the estimation of $q-h$ under the use of Lemma 4.2 may be replaced by $q - h \leq -a_1$.

Again, in all cases we have $C_{Alg} + \Delta\phi \leq O(\frac{1}{\delta^{3/2}})\cdot C_{Opt}$.

5 EXTENSIONS

In this section we briefly describe how to modify the analysis of our algorithm such that the resulting bounds match the lower bounds (up to a factor of $\frac{1}{\sqrt{\delta}}$) for the variants not covered in the previous chapter.

All results mentioned hold for the Euclidean plane (or any $\mathbb{R}^n$ with the Euclidean Distance as a metric) and therefore contain a factor of $\frac{1}{\delta^{3/2}}$. These results can also be applied to the 1-dimensional space where the factor is then reduced to $\frac{1}{\delta}$ by an easy modification of the proof from the previous chapter.

Corollary 5.1. *Let R_{min} and R_{max} be the minimum and maximum number of requests per time step. Algorithm MtC is $O(\frac{R_{max}}{\delta^{3/2}R_{min}})$-competitive utilizing a maximum moving distance of $(1+\delta)m$ if requests are served after the server is moved to a new position.*

Proof. We replace the fixed number of requests r in the potential function by the maximum number of requests R_{max}. In the cases where the potential is used to cancel out the costs of the algorithm this is then still possible. However if the potential difference is positive it may add a term which is $O(\frac{R_{max}}{R_{min}})$ times the optimal costs. □

Corollary 5.2. *Let $r \geq D$ be the fixed number of requests per time step. Algorithm MtC is $O(\frac{r}{D\cdot\delta^{3/2}})$-competitive utilizing a maximum moving distance of $(1+\delta)m$ if requests must be served before the server is moved to a new position.*

Proof. The only change in the proof comes from the fact that the costs of the algorithm now contain $r(a_1+a_2) \leq r(p+s_1+q+a_2)$ for serving the requests where the term rs_1 produces the $\frac{r}{D}$ factor in the competitive ratio. □

6 CONCLUSION

We have seen that a simple, deterministic algorithm is sufficient to get an almost optimal competitive ratio in various versions of our model. For the Euclidean space of dimension 1, we have an asymptotically optimal competitive ratio which can not be beaten even by a randomized algorithm against an oblivious adversary.

For the Euclidean space of higher dimensions, we miss the optimal competitive ratio only by a factor of $\frac{1}{\sqrt{\delta}}$. We conjecture that this remaining gap between the upper and the lower bound can be closed towards the lower bound, but it remains an open problem to design an online algorithm and / or provide an analysis to achieve the better competitive ratio.

It seems an interesting question if the idea of limiting the movement of resources within a time slot also can be applied to other popular models such as the k-Server Problem (effectively turning it into the Page Migration Problem with multiple pages).

It may also be possible to extend existing online problems without a concept of movement by introducing a limited movement like in our model. In problems like the Online Facility Location Problem, this might give possibilities to the online algorithms to slightly improve upon decisions where to open a facility.

Furthermore the concept of allowing only a limited configuration change might be applicable to any problem which belongs to the class of Metrical Task Systems where in every step configurations may be changed to lower costs for answering a certain type of requests which have to be served by the system.

REFERENCES

[1] Arif Ahmed and Ejaz Ahmed. 2016. A Survey on Mobile Egde Computing. In *Proceedings of the 10th IEEE International Conference on Intelligent Systems and Control (ISCO)*. IEEE. https://doi.org/10.1109/ISCO.2016.7727082

[2] Susanne Albers and Hisashi Koga. 1995. Page Migration with Limited Local Memory Capacity. In *Proceedings of the 4th International Workshop on Algorithms and Data Structures (WADS)*. Springer, 147 – 158. https://doi.org/10.1007/3-540-60220-8_58

[3] Yair Bartal. 1998. Distributed Paging. In *Developments from a June 1996 Seminar on Online Algorithms: The State of the Art*. Springer, 97 – 117. http://dl.acm.org/citation.cfm?id=647371.723920

[4] Yair Bartal, Moses Charikar, and Piotr Indyk. 2001. On the Page Migration Problem and Other Relaxed Task Systems. *Theoretical Computer Science* 268, 1 (2001), 43 – 66. https://doi.org/10.1016/S0304-3975(00)00259-0

[5] Hans-Joachim Böckenhauer, Dennis Komm, Rastislav Královič, and Richard Královič. 2017. On the advice complexity of the k-server problem. *J. Comput. System Sci.* 86 (2017), 159 – 170. https://doi.org/10.1016/j.jcss.2017.01.001

[6] Wolfgang W. Bein, Marek Chrobak, and Lawrence L. Larmore. 2002. The 3-server problem in the plane. *Theoretical Computer Science* 289, 1 (2002), 335 – 354. https://doi.org/10.1016/S0304-3975(01)00305-X

[7] Marcin Bienkowski, Jaroslaw Byrka, Miroslaw Korzeniowski, and Friedhelm Meyer auf der Heide. 2009. Optimal algorithms for page migration in dynamic networks. *Journal of Discrete Algorithms* 7, 4 (2009), 545 – 569. https://doi.org/10.1016/j.jda.2008.07.006

[8] David L. Black and Daniel D. Sleator. 1989. *Competitive Algorithms for Replication and Migration Problems*. Technical Report CMU-CS-89-201. Department of Computer Science, Carnegie-Mellon University.

[9] Marek Chrobak, Lawrence L Larmore, Nick Reingold, and Jeffery Westbrook. 1997. Page Migration Algorithms Using Work Functions. *Journal of Algorithms* 24, 1 (1997), 124 – 157. https://doi.org/10.1006/jagm.1996.0853

[10] A. Davis, Jay Parikh, and William E. Weihl. 2004. Edgecomputing: Extending Enterprise Applications to the Edge of the Internet. In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters*. ACM, 180 – 187. https://doi.org/10.1145/1013367.1013397

[11] Adam Janiak, Władysław Janiak, and Maciej Lichtenstein. 2007. Resource management in machine scheduling problems: a survey. *Decision Making in Manufacturing and Services* Vol. 1, no. 1-2 (2007), 59–89.

[12] Bala Kalyanasundaram and Kirk Pruhs. 1995. Speed is as Powerful as Clairvoyance. In *Proceedings of the 36th IEEE Annual Symposium Foundations of Computer Science (FOCS)*. IEEE, 214 – 221. https://doi.org/10.1145/347476.347479

[13] Amanj Khorramian and Akira Matsubayashi. 2016. Uniform Page Migration Problem in Euclidean Space. *Algorithms* 9, 3 (2016). https://doi.org/10.3390/a9030057

[14] Elias Koutsoupias. 2009. The k-server problem. *Computer Science Review* 3, 2 (2009), 105 – 118. https://doi.org/10.1016/j.cosrev.2009.04.002

[15] Jeffery Westbrook. 1994. Randomized Algorithms for Multiprocessor Page Migration. *SIAM J. Comput.* 23, 5 (1994), 951 – 965. https://doi.org/10.1137/S0097539791199796

[16] Andrew Chi-Chin Yao. 1977. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th IEEE Annual Symposium Foundations of Computer Science (FOCS)*. IEEE, 222 – 227. https://doi.org/10.1109/SFCS.1977.24