PADERBORN UNIVERSITY
The University for the Information Society

Mobile Resource Allocation

Dissertation

In partial fulfillment of the requirements for the degree of

Doctor rerum naturalium (Dr. rer. nat.)

at the Faculty of Computer Science,
Electrical Engineering and Mathematics
at Paderborn University

submitted by
BJÖRN FELDKORD

Reviewers:

- Prof. Dr. Friedhelm Meyer auf der Heide,
  Paderborn University
- Dr. habil. Marcin Bieńkowski,
  Wrocław University

Zusammenfassung

Die vorliegende Arbeit behandelt Ressourcenallokationsprobleme für Anwendungen, welche hauptsächlich von mobilen Nutzern ausgeführt werden. Ressourcen werden dabei nah am Nutzer platziert, wie beispielwiese in den Basisstationen des mobilen Netzwerkes. Die Performanz von Anwendungen für den Benutzer hängt unter anderem von Latenzen ab, welche durch eine entsprechende Platzierung gewährleistet werden müssen. So muss die Konfiguration der Ressourcen einerseits stetig angepasst werden, andererseits müssen die entsprechenden Änderungen leicht berechenbar und schnell dürchführbar sein um eine hohe Qualität des entsprechenden Services zu gewährleisten.

Wir behandeln zwei grundlegende Modelle, die sich mit der Platzierung mobiler Ressourcen beschäftigen: In unserem *Mobile Server Problem* untersuchen wir eine fixe Anzahl Ressourcen welche vor Beantwortung einer eintreffenden Anfrage über eine kurze Distanz verschoben werden können. Wir geben für dieses Problem Online Algorithmen an, welche auf vorhandenen Methoden zu ähnlichem Problemen wie k-Server und Page Migration aufbauen, und kompetitive Faktoren asymptotisch nah an einem optimalem Ergebnis erreichen.

Unser zweites Problem ist eine Erweiterung des *Online Facility Location Problems*, in welchem wir dem Online Algorithmus erlauben, die Positionen seiner Facilities über die Zeit zu korrigieren. Dabei limitieren wir die Korrektur sowohl durch damit verbundene, proportionale Kosten, also auch durch eine fest Schranke pro Zeitschritt. Wir geben Online Algorithmen an, welche einen kompetitiven Faktor unabhängig von der Anzahl der Zeitschritte bzw. der Anfragen erreichen und asymptotisch optimal auf der Linie sind.

Abstract

This thesis covers the topic of resource allocation problems that are tailored to scenarios primarily involving mobile users. The resources are placed close to the users, e.g., at the base stations of a mobile network. The performance of applications accessed by the users depends, among other things, on latencies which have to be guaranteed by a suitable placement of the resources. This implies that the configuration of resources must be continuously adapted. However, these changes must be easily computable and quickly applicable to maintain a high service quality.

We propose two basic models, which deal with the placement of mobile resources: Our first model is called the *Mobile Server problem*, where we are concerned with the placement of a fixed number of resources. These resources can be moved over a short distance before answering an incoming request. For this problem, we propose online algorithms based on the methods used in similar problems such as the k-Server and Page Migration problems, and prove asymptotically almost optimal competitive ratios.

The second problem is an extension of the *Online Facility Location problem*, where we allow an online algorithm to correct its positions of the facilities over time. The movement distance is limited implicitly through a cost proportional to the distance as well as directly through a fixed upper bound per time step. We propose online algorithms that achieve competitive ratios independent of time and the number of clients. The results are asymptotically optimal on the line.

Preface

When I started to work in research about four and a half years ago, I had no idea what this thesis would be about, except for a few basic concepts I picked up during my studies. Luckily, since then I met a number of people who have helped me move forward and who I would like to thank at this opportunity.

First of all, I would like to thank my advisor Prof. Dr. Friedhelm Meyer auf der Heide for giving me the opportunity to work at his Algorithms and Complexity group and for always pushing me in the right direction. I would also like to thank Prof. Dr. Marcin Bieńkowski for agreeing to review this thesis and joining the commission, as well as all the other members of the commission. Further thanks goes to my coauthors from various publications; it was a pleasure working with you. I would like to thank all my colleagues at the HNI for our amazing time together, be it at work or at less serious occasions like the social events of our AG Seminar. A special shout-out goes to the colleagues from my office: Daniel, Manuel and Max, who provided the necessary motivation to push on when needed, each in their own way.

Finally, I would like to thank my family for all their support during this project.

Björn Feldkord
September 2019

Contents

1 Introduction 11

1.1 Technical Background 13

1.2 Related Work 15

2 The Mobile Server Problem 19

2.1 Formal Model 20

2.2 Summary of Results 21

2.3 One Server, Multiple Requests 21

2.3.1 Lower Bounds 21

2.3.2 The Move-to-Center Algorithm 23

2.3.3 Locality of Requests 32

2.4 Multiple Servers, One Request 34

2.4.1 Lower Bounds 34

2.4.2 An Algorithm for the Unweighted Problem 37

2.4.3 Extension to the Weighted Problem 53

3 Online Facility Location with Mobile Facilities 63

3.1 Formal Model 64

3.2 Summary of Results 65

3.3 Algorithms 65

3.3.1 Technical Overview 66

3.3.2 A Randomized Algorithm for Unlimited Movement 67

3.3.3 Extension to Limited Movement 70

3.3.4 Extension to Higher-Dimensional Spaces 71

3.4 Lower Bounds 73

4 Conclusion & Open Problems 75

1. Introduction

Consider a network of processors working on common data. The data is organized in memory pages, where each page can be located in the local memory of any processor, but only one copy may exist at a time. Processors who want to access the page but do not have it in their local memory must communicate over the network with the processor holding the page. This communication induces costs; to reduce it, an algorithm may decide to move the page to a different processor. However, this also induces costs which may be high due to the size of the page, and hence moving the page only pays off if it reduces the costs of multiple requests. An algorithm for the problem is evaluated by the total cost incurred by communication and the movement of the page.

The above problem, commonly referred to as *Page Migration*, belongs to the broad class of resource allocation problems. These models are of interest from both a theoretical and a practical point of view: While they capture many challenges occurring in different distributed systems in a simple model, they still maintain the basic combinatorial complexity behind the practical systems and therefore require non-trivial solutions. This thesis covers resource allocation problems occurring in large distributed systems: i.e., systems that consist of multiple, heterogeneous entities (also clients or nodes) forming a dynamic network. The clients request access to shared resources in the network, which must be allocated by an algorithm. In our problems, we consider a shared resource that is used throughout the execution of some task and not exclusively bound to a single entity. Typical examples would be shared files that can be read from or written to, (limited) storage used by multiple clients, or remotely accessed web-services that are executed on specific servers. Our models focus on the placement of resources to reduce communication: i.e., the usage of resources requires communication with a resource that in turn incurs costs dependent on the distance communicated. An algorithm solving the allocation problems has to make decisions concerning the location of available resources such that the costs for communication and for the placement of the resources are minimized.

The models discussed in this thesis are inspired by an emerging trend in computing regarding applications which are traditionally associated with *cloud computing*. 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 (the "cloud"). Various resource management problems arise in these computing centers. As an example, data used in common by multiple processors must be allocated to be accessible by all the processors who work on it. If multiple copies of the data exists, consistency problems in addition to the question of optimal placement may arise. This problem is captured in the *File Allocation problem* [12], which in turn contains different subproblems which are of independent interest from both a theoretical and practical standpoint. The most relevant of these subproblems for the research in this thesis is the aforementioned Page Migration (also known as *File Migration*) problem [22], which deals with assigning a single page (or file) to different places in a network such that the costs for access and migration are minimized. Due to the relation of the formal model to one of our problems, Page Migration and the associated results are described in more detail in Section 1.2.

Other examples for problems of theoretical interest occurring in large computing centers are scheduling problems, where multiple machines organized in a network use common resources such as memory and bandwidth of a common communication channel (e.g. [50, 53] for recent examples). From the user side, resources in computing centers may be utilized by renting them for a certain time. This was formally modeled by the *Parking Permit problem* [64] as a basic model for leasing infrastructure, and then applied to various resources allocation problems (e.g. [1, 4, 65]).

The usage of few, large computing centers has, among many advantages, also some major drawbacks: Due to the distances of the end user to the cloud, latency guarantees that are critical for some applications cannot be given. The network infrastructure is also not sufficient to handle applications where a lot of necessary data is generated at the user devices and then uploaded to the cloud. Future tasks such as the coordination of automated driving would require too much data to be sent to cloud servers and processed there to receive a timely answer. A new architecture has been proposed for solving these problems, commonly referred to as *edge computing* or *fog computing*. Here, computation is not done in a few large computing centers, but closer to the user, e.g., at the base stations for the wireless network. The high mobility of the users necessitates applications to be very flexible if they want to be able to adapt to the shifting demands and maintain a high service quality. An overview of the main motivations, applications and important problems for research in the area of edge computing can be found in various surveys covering the topic (e.g. [66, 71, 78]).

In this thesis, we focus on resource allocation problems adopted from models initially designed for the aforementioned cloud computing scenario. We alter the models to fit systems that are closer to the edge computing scenario: i.e., we assume demands are issued by mobile users who expect low latencies for their applications. Resources need to be allocated such that the configuration adapts quickly to the demands of the users in order to meet those expectations. On the other hand, we allow a high flexibility on the configurations: i.e., our algorithms can choose between a potentially infinite number of system states (placements) to adapt to the users. These aspects hopefully contribute a first step towards integrating the aspects of mobile users and the dynamic applications into classical questions of theoretical computer science.

Our problems all belong to the class of *online problems*, where the input, which in our models are clients who want to access resources, arrives over time. The clients' requests must be served in the same time step as they pose their request, ensuring minimal delays. The use of *competitiveness* (the worst-case ratio between the cost of the online algorithm and an optimal solution) as a metric ensures that our algorithms perform reasonably well, even in scenarios where nothing is known of the appearing clients in advance. In our example of Page Migration from the beginning, this would mean that the online algorithm has to make the choice of where to place the page at each time without knowing anything about the future access pattern of the processors beforehand. The algorithm is evaluated by comparing the cost during a worst-case input to that of an optimal solution (knowing the whole input in advance). It is obvious that while taking worse-case inputs as a measurement ensures that the resulting performance is guaranteed for all inputs, some results might be very misleading since a bad competitive ratio might be induced by very specific inputs that hardly appear in practice. In our work, we use the technique of *resource augmentation* to show that the worst-case bounds in our models are due to the exploitation of a specific property of the problem which disappears if the online algorithm is given slightly more power than the offline solution it is compared to. In that sense, when facing such problems, simply using resources with enhanced capabilities such as faster processors or larger storage can help to atone for shortcomings due to the lack of knowledge about the future. A formal description of online algorithms, competitiveness and resource augmentation can be found in the following section.

The two major models discussed in this thesis focus on the aspect of the mobility of resources and users. The first model, titled the *Mobile Server problem* discussed in Chapter 2, deals with the placement of a fixed number of mobile resources, which can adapt their position over time with a limited speed. It is an extension of the classical Page Migration problem, which becomes more challenging through the introduction of the limited movement. The algorithms build heavily on the work done for the Page Migration and k-Server problems. This necessary background will be discussed in Section 1.2. The main result for this model is a competitive online algorithm that can be easily implemented given an existing k-Page Migration algorithm.

The second model is an extension of the *Online Facility Location problem*, where facilities are placed by an algorithm over time to serve clients that are connected to the facilities. In the classical version of this model, facilities cannot be moved once they are placed by the online algorithm. As a result, small errors in the facility placement result in a large competitive ratio in specific instances. We extend the model by allowing the facilities to move for cost as in the Mobile Server problem and show, that the competitive ratio significantly improves: i.e., its dependence on the number of clients disappears. The main result is an algorithm based on the randomized algorithm by Meyerson [63] which combines it with the approach taken for the Mobile Server problem. We formally introduce the model and present the results in Chapter 3.

Both models isolate a specific aspect of the overall resource allocation scenario as described above. In Chapter 4, we discuss ways of extending the models to cover more aspects of the scenario and how to connect it with other research done in the area.

1.1 Technical Background

In this section, we introduce some necessary definitions regarding the technical framework for the problems discussed in the thesis. Readers familiar with the basic concepts of online computation may skip the corresponding paragraphs without affecting their understanding of the rest of the thesis.

Online Algorithms and Competitive Ratio

All problems considered in this thesis belong to the class of online problems. In these problems, the input is not known in advance to an algorithm, but given to it over time. Dependent on the concrete problem, the algorithm has to make decisions only on the basis of the input seen so far, but which will affect the future as well. By making decisions over time, the online algorithm creates a solution for the entire input, which is evaluated by a cost function. Online algorithms are evaluated on the basis of their competitive ratio, introduced by Sleator and Tarjan [73]. The competitive ratio is the maximum ratio between the cost of the solution created by the online algorithm and the cost of the best possible solution for the same input, taken over all possible inputs. Formally, let A be an online algorithm, $A(I)$ be the solution of A given input I and $C_{A(I)}$ be the cost of the solution $A(I)$. In the same way, let $Opt(I)$ and $C_{Opt(I)}$ be the optimal solution to input I and $C_{Opt(I)}$ its cost. Finally, let S be the set of all possible inputs. Then, A is called α-competitive if

$$C_{A(I)} \leq \alpha \cdot C_{Opt(I)} + \beta$$

for all $I \in S$ and some constant β independent of I. If $\beta = 0$ we call A strictly α-competitive. Note, that in this thesis we allow α to be a function in the input length, as it is unavoidable for some of the problems. In the literature, algorithms for which there is no α independent of the input length are sometimes called *not competitive*. To avoid confusion, we will not use this term for our models.

For the evaluation of randomized online algorithms, the concept of *adversaries* is used to describe different ways of regulating the power of randomization. An adversary is viewed as an entity that dictates the input of the problem and also creates a solution to the created instance. A paper by Ben-David et al. [16] discusses three types of adversaries whose strength can be directly related. The most commonly used adversary, and also the adversary we mostly use in this thesis, is the *oblivious adversary*. The oblivious adversary creates the input only on the basis of the description of the online algorithm, without knowing the outcome of the random experiments the online algorithm does. This description of competitiveness is equivalent to the following: A randomized algorithm A is called α-competitive against an oblivious adversary, if

$$E[C_{A(I)}] \leq \alpha \cdot C_{Opt(I)} + \beta$$

for all $I \in S$ and some constant β independent of I. The expectation is taken over the random choices of A.

The other two types of adversaries are adaptive to the random choices of the online algorithm: The *adaptive-online adversary* creates the instance step-by-step and learns of the random choices of the algorithm in each time step. However, the adversary also has to create its solution to the instance online: i.e., it has to make the same decisions as the online algorithm in each time step. The *adaptive-offline adversary*, on the other hand, creates the instance in the same way as the adaptive-online adversary, but only has to create a solution after the creation of the instance is completed. The paper by Ben-David et al. [16] lays out the connection between these three adversaries. The most important one is that there is a total order on them with respect to the possible competitive ratio: For a given problem, if there is an α-competitive algorithm against an adaptive-offline adversary, there also is an α-competitive algorithm against an adaptive-online adversary which in turn implies the existence of an α-competitive algorithm against an oblivious adversary. The best possible competitive ratio for deterministic online algorithms and randomized algorithms against adaptive-offline algorithms is the same: i.e., randomization brings no advantage over a deterministic algorithm against an adaptive-offline adversary.

Lower Bounds & Yao's Principle

When proving lower bounds on the competitive ratio, it must be argued that there is no online algorithm that can achieve a better competitive ratio than the desired bound. This can be particularly hard when dealing with randomized algorithms that possibly make a lot of randomized choices throughout the computation. An established technique to show lower bounds for randomized algorithms against an oblivious adversary, is the use of *Yaos' Minimax Principle* [77]. We make use of this technique in both our models. The principle, originally formulated for runtime, states that the best possible (expected) performance of randomized algorithms over all inputs is the same as the best possible (expected) performance of deterministic algorithms over all probability distributions over the inputs. On the basis of this, a lower bound on the competitive ratio for online algorithms against oblivious adversaries can be shown by lower bounding the expected cost ratio of all deterministic online algorithms against a constructed distribution over the inputs. In order to properly apply the principle, we need to argue that for our instances the number of possible algorithms for our given distribution is essentially finite. This is non-trivial in our models since we use Euclidean metrics as a basis. We will argue that the principle nevertheless applies in the respective theorems.

Resource Augmentation

Some online problems have hard instances for which online algorithms cannot achieve a constant competitive ratio: i.e., a competitive ratio independent of the size of the given input sequence. This does not always mean the problem is generally hard, but the high competitiveness is caused by only a few specific inputs, which abuse a certain property in the problem definition. Resource augmentation is a concept that tries to improve the performance of online algorithms in hard instances by granting them additional power which the offline solution they are compared to cannot utilize. This technique of giving the online algorithm slightly more power to improve the 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 [51].

In both of the models in this thesis, we make use of resource augmentation to get competitive ratios independent of the number of time steps in the input. In the first model (Mobile Server problem), we give the resources of the online algorithm a higher movement speed than the resources of the offline algorithm. In the second model (Online Facility Location), we give the originally static resources the potential to move in order to improve the positioning. For the first case, this can be interpreted as trading the latency caused by the resource reconfiguration against the communication cost, i.e., sacrificing service quality for lower cost. In the second case, it illustrates that allowing small corrections to the online solution leads to a much better outcome. One could also say that adding more flexibility to the resource configuration improves the overall result.

Note that there are other concepts that offer a refined view on worst-case instances as well: Applying *smoothed analysis* [74] to online algorithms, the *smoothed competitiveness* [14] of an algorithm is evaluated by perturbing the input values of a problem at random and taking the expected (w.r.t. the random perturbation) competitive ratio. Another concept is the notion of *semi-random input streams* [48], where the order of the input is changed to something in between an adversarial and completely random order (while the values themselves are still determined by an adversary). In both concepts, randomness is used to make worst-case instances of a problem very unlikely to occur. In contrast, the adversary can still fully determine the instance in the resource augmentation model.

1.2 Related Work

We review the most important results for relevant problems connected to the models discussed later. Our first model, the Mobile Server problem, can be viewed as a generalization of both the k-Server and the Page Migration problems. Some of the difficulties occurring in the Mobile Server problem also need to be solved in these related problems, and hence we reuse ideas contained in the solutions to these problems for our model. We also use algorithms for the k-Server and Page Migration problems as black-box simulations in our algorithms, hence it is useful to know some of the algorithms and their competitive ratio.

Our second model is a variant of the Online Facility Location problem and existing results on the competitive ratio for the classical version of the problem motivate our variant. In addition, our algorithms draw inspiration from the randomized solution for the problem mentioned here.

k-Server Problem

The k-Server problem [60] was introduced as a generalization of paging and other online problems. Given is a metric space and k servers placed within. The input being processed over time is a sequence of requests, each occurring at a point in the metric space. The requests must be served by moving one server to the point of the request. The optimization goal is to minimize the total distance moved.

The k-Server problem is mostly considered in the online setting, since it can be efficiently solved offline by dynamic programming or a reduction to the Mincost-Maxflow problem. Manasse et al. [60] 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.

For general metrics, the best known result is the *Work-Function algorithm* (WFA), which was shown to be $2k-1$-competitive by Koutsoupias and Papadimitriou [56]. Although this algorithm seems generally inefficient in case of runtime and memory, there have been studies showing that an efficient implementation of this algorithm is indeed possible [68, 69]. 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 by Bartal and Koutsoupias [13].

Since its introduction, many algorithms have been designed for special cases of the problem. Most notable is the *Double-Coverage algorithm* by Chrobak and Larmore [28], which is k-competitive on trees. Like our algorithms in this paper, the Double-Coverage Algorithm is memoryless in the sense that it does not need to remember the past request sequence to answer the current request. For the Euclidean plane, Bein et al. constructed a deterministic algorithm that slightly improves over the competitive ratio of WFA for $k = 3$ servers to 4.243 [15].

The study of randomized algorithms was initiated by Fiat et al. [38] 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 a polylogarithmic competitive ratio was an $\mathcal{O}(\log^3 n \cdot \log^2 k)$-competitive algorithm by Bansal et al. [7]. This was recently improved by Bubeck et al. [25] who gave an $\mathcal{O}(\log^2 k)$-competitive algorithm for Hierarchical Separated Trees (HSTs). Using a different technique, Buchbinder et al. [26] achieved similar results. These algorithms can be turned into an $\mathcal{O}(\log n \cdot \log^2 k)$-competitive one for general metrics by an embedding of general metrics into HSTs [33] often used in literature. Lee presented an alternative embedding which results in a $\mathcal{O}(\log^9 k \cdot \log \log k)$-competitive algorithm in general metrics [58], using the algorithm in [25] as a basis.

The k-Server problem can be formulated within the general framework of *metrical task systems* [23] whose competitive ratio in general depends on the number of states n in the system. While this bound is linear in the deterministic case, Bartal et al. showed that there is an $\mathcal{O}(log^6 n)$-competitive randomized algorithm for general metrics [10]. A recent paper by Bubeck et al. improves this bound to $\mathcal{O}(log^2 n)$ [24], based on the technique from [25]. Schäfer and Sivadasan utilized the concept of smoothed analysis to show that small perturbations of the request cost can lead to a significant improvement of the competitiveness for certain topologies of the underlying state graph [70].

The dependence of the competitive ratio of the k-Server problem on the number of servers leads to the question whether increasing the number of servers of the online algorithm in comparison to the optimal solution improves the competitive ratio towards a constant. This was first considered by Koutoupias, who showed that in the case of a uniform metric the competitive ratio indeed decreases as the difference between the number of servers for the online algorithm and the optimal solution increases [55]. Despite different algorithms that achieve a competitive ratio independent of the number of servers for special metrics (often parametrized with the size of the metric) [9, 30], no $o(k)$-competitive algorithm has been found for general metrics (here k denotes the number of servers for the optimal solution). Interestingly, for the established optimal Double-Coverage algorithm, the competitive ratio even slightly increases in the resource augmentation setting [8].

Due to its popularity, the k-Server problem has been studied in many variants that are mostly unrelated to the problems discussed in this thesis such as having the maximum instead of the sum of travel distances as an optimization goal [76] and being able to reject requests for a penalty, which then do not have to be served [21]. Our general definition of the Mobile Server problem allows for multiple requests per time step, something which Bienkowski and Kutylowski also considered [20]. However, their model has no limit on travel distances and each request needs to be served by its own server, even if multiple requests occupy the same position.

Page Migration Problem

The Page Migration problem is a simple model for approaching the management of shared data in a cloud computing scenario. In the classical version of this problem we consider a memory page that is shared by multiple processors connected in a network. Only one processor can hold the page at a time. Processors request data from the page which incurs a 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 a cost proportional to the distance in the network times the size D of the page. The offline version of this problem can be solved efficiently by dynamic programming.

The Page Migration problem was first considered as an online problem by Black and Sleator [22] 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. For deterministic algorithms, the best known lower bound is slightly above 3 [61, 62].

The first result for deterministic algorithms was given by Westbrook in 1994 [75], who showed that there exists a $3(1+\phi)$-competitive algorithm, where $\phi \approx 1.62$ is the golden ratio. The result is derived by a transformation of randomized algorithms and the relation of different adversaries. The first explicit deterministic algorithm was called *Move-To-Min* as the strategy is to move to the optimal point in regards to the last D requests [6]. The competitive ratio of this algorithm is 7. The currently best deterministic result is a 4-competitive algorithm by Bienkowski et al. [19].

Considering randomized algorithms, 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 $(1+\phi)$-competitive algorithm against oblivious adversaries. These algorithms can be found in a paper by Westbrook [75].

In addition to the above results in general metrics, the Page Migration problem has also been studied in the Euclidean space explicitly. The best competitive ratio for the Euclidean space and $D = 1$ is 2.75 [52]. For large D, the randomized algorithm by Westbrook can be used to obtain a deterministic algorithm for which the competitive ratio tends to $1+\phi$ [29] (but is worse than the previous result for $D = 1$).

The Page Migration problem belongs to the class of *relaxed task systems*, which is a type of metrical task system that can be related to another metrical task system, in this case the k-Server problem. This was used to derive algorithms for the problem with multiple copies of the page (k-Page Migration). The adaption within the framework results in an $\mathcal{O}(k)$-competitive randomized and an $\mathcal{O}(k^2)$-competitive deterministic algorithm [11].

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

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 in which 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 [18].

(Online) Facility Location Problem

Facility Location is a well-researched area that captures a variety of allocation problems arising from applications such as the placement of warehouses to clustering problems. In the Uncapacitated Metric Facility Location problem, we are given a set of n clients and a set of possible facility locations which both are part of the same metric space. For each of the possible facility locations, we are also given an opening cost. The goal is to select a subset of the facility locations such that the sum of the implied opening costs plus the sum of the distances of each client to the closest facility is minimized. In this thesis we only consider uncapacitated and metric variants, and hence will omit these terms in the following.

Since the problem is NP-hard, many approximation algorithms have been designed. We mainly focus on the online variant and mention only the most important results for the offline variant here. An overview of many results and the different techniques can be found in the survey by Shmoys [72]. For general metrics, the currently best algorithm reaches an approximation factor of 1.488 [59], which is close to the best known lower bound of 1.463 which holds under the assumption that $NP \notin DTIME(n^{\mathcal{O}(\log\log n)})$ [47]. For the Euclidean plane, there exists a randomized approximation scheme [5]. In contrast, the lower bound we show for our variant even holds for the line, and for the general online variant the lower bound for the line is asymptotically the same as for general metrics. As a consequence, we are not necessarily interested in the best constant and use other algorithms as inspirations for the online variant. One prominent example is a fast approximation algorithm by Jain and Vazirani [49] based on the primal-dual method, which has inspired some of the algorithms used for the online [40] and leasing variants [65].

The Online Facility Location problem was introduced by Meyerson [63] who gave a simple randomized algorithm for the problem which achieves a competitive ratio of $\mathcal{O}(\frac{\log n}{\log\log n})$. Fotakis later proved that this competitive ratio is asymptotically optimal and also gave a deterministic online algorithm achieving the same competitive ratio, fixing the ratio to $\Theta(\frac{\log n}{\log\log n})$ for both the deterministic and the randomized case [41]. In addition to the complex deterministic algorithm which achieves the optimal competitive ratio, there are deterministic algorithms that are much simpler to implement and achieve an almost optimal competitive ratio of $\mathcal{O}(\log n)$ [3, 40].

In the same paper where the Online Facility Location problem was introduced, Meyerson also showed that it is possible to achieve a constant competitive ratio if the clients are given in a random order [63]. This result was later refined by Lang [57], who showed that the competitive ratio can be fixed to $\Theta(\frac{\log t}{\log\log t})$ for a t-bounded adversary. That is, an adversary generates a semi-random input stream from a random order of the (adversarially chosen) clients by permuting the input stream such that for each element a, less than t elements originally placed before a are then placed after a.

Other works in the area of Online Facility Location tackle the lower bound of the problem by allowing the algorithm to make some adjustment to the solution over time. The model closest to our work is by Divéki and Imreh [32] who allow an online algorithm to arbitrarily change the position of already opened facilities without incurring any additional costs. The requests can also be reassigned to different facilities for free in each time step. They achieve a competitive ratio of at most 2 for arbitrary metrics when given black-box access to optimal algorithms for facility location and k-median. Otherwise their competitive ratio depends on the approximation ratios of the algorithms for these problems.

Another variant in which some of the facility placements can be revoked completely is the Incremental Facility Location problem [39]: Motivated by applications to clustering, existing facilities may be merged into one (for no additional cost) while the clients that were previously assigned to the merged facilities are then all assigned to the same facility. The algorithm does not need to pay for facilities that are deleted in this way. In this setting, a constant competitive ratio can be achieved. A survey by Fotakis [42] covers various algorithms for both the Online and the Incremental Facility Location problem.

There exist a variety of other models of mobility in Facility Location. The Facility Reallocation model by de Keijzer and Wojtczak [31] is actually close to the k-Page Migration problem: A fixed number of facilities can be moved for a cost proportional to the distance to improve the connection cost to the clients. All clients are present in all time steps and change their location over time. Hence the model is equivalent to k-Page Migration with $D = 1$ and multiple requests in each time step. For the online variant, [31] only focuses on the case of one facility. A recent paper by Fotakis et al. exploits a connection to the k-Server problem to receive an online algorithm for two facilities [43].

Other concepts of movement related to Facility Location include models where the number of facilities is fixed and they have to be moved to service clients that appear over time [46], a variant in which both the clients and facilities may be moved in a network such that each client arrives at the node of a facility [44], and a mobile version in which a set of clients moves over time and the movement for a fixed set of facilities has to be determined to fulfill a given property [17]. In all of these models, the focus lies on the movement of a fixed number of facilities, and thus the aspect of placement of new facilities is not contained in them. Hence the techniques used for these problems are of limited use for the Facility Location problem we discuss in Chapter 3, in which the initial placement (and the cost for the facilities) are still a major concern.

2. The Mobile Server Problem

In this chapter, with the Mobile Server problem we present a simple model that focuses on managing a fixed number of identical resources (e.g., a fixed number of copies of a data page) that are requested by mobile devices not known beforehand. We refer to the resources as servers in accordance with models such as the k-Server problem and allow them to be moved to different positions (e.g., other base stations or even mobile devices in the network), but only a limited distance in the network before the next round of requests must be served. The distance limitation on the movement incorporates the necessity of low latencies into the model: Having the resources close to the user as described in the introduction is done in order to reduce latencies for critical applications. It therefore makes sense that the reconfiguration of resources is done quickly and hence the necessary data should also only be transmitted locally instead over the entire network.

We formulate the problem in the Euclidean space as a metric. Modeling the clients as nodes within a static network topology is not feasible since we want to cover a scenario with a large number of mobile devices that can change their position and join or leave the network at all times. In our model, the servers can also be moved to any point in the Euclidean space. While this might not be entirely realistic for most applications, there are still some scenarios that come close to it: Mobile relay stations can be used in scenarios such as natural disasters in order to provide network access to the respective area, which might be necessary to quickly collect data about the incident to assist the local helpers. For future applications, it is feasible to consider automated, flying drones being equipped to not only provide network access, but also provide resources for important local applications. Even for a more traditional computing scenario, having the entire space as an option provides a good starting point for the research of models like this, which could then of course be extended in the future to a limited number of options.

As a cost function, we choose the same model as for the classical Page Migration problem: The servers can be moved for a cost proportional to the distance and a weight $D \geq 1$. The requests are served for costs equal to their distance to the nearest server. This makes our results directly comparable to popular models in the same area, namely the k-Server and Page Migration problems.

In the following, we formally define the model and its variants and then give an overview of the results we achieved.

Chapter Basis

The model, algorithms and analysis in this chapter are based on the following publications:

> *Björn Feldkord and Friedhelm Meyer auf der Heide. The Mobile Server Problem. In: ACM Transactions on Parallel Computing (TOPC), 6(3): 14:1 - 14:17, 2019. Cf. [37].*
> *A preliminary version appeared at SPAA'17, cf. [35].*
>
> *Björn Feldkord, Till Knollmann, Manuel Malatyali and Friedhelm Meyer auf der Heide. Managing Multiple Mobile Resources. In: Proceedings of the 17th Workshop on Approximation and Online Algorithms (WAOA), 2019 (accepted). Cf. [34].*

2.1 Formal Model

In this section, we formally define the model of the k-Mobile Server problem and establish the notation used throughout this chapter.

Time is considered discrete and divided into time steps $1,\dots,T$. An input to the k-Mobile Server problem is given by a sequence of requests $v_{1,1},\dots,v_{1,r_1},\dots,v_{T,1},\dots,v_{T,r_T}$ where each of the r_t requests $v_{t,1},\dots,v_{t,r_t}$ occur in time step t and are represented by points in the Euclidean space of arbitrary dimension. We are given k servers $a_1,\dots,a_k$ controlled by our online algorithm. At each point in time, one server occupies exactly one point in the Euclidean space. We denote 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 requests $v_{t,1},\dots,v_{t,r_t}$ are revealed to the online algorithm. The algorithm may then move each server, such that $d(a_i^{(t-1)},a_i^{(t)}) \le m_s$ for all servers a_i. The movement incurs costs of $D\cdot\sum_{i=1}^k d(a_i^{(t-1)},a_i^{(t)})$ for a constant $D \ge 1$. The requests $v_{t,1},\dots,v_{t,r_t}$ are then served by the closest server $a_i^{(t)}$, which incurs costs of $d(a_i^{(t)},\ v_{t,j})$. Note that the variables indexed with the time t represent the configuration at the end of the time step t. The total cost of an algorithm Alg on a given input sequence can hence be expressed as follows:

$$C_{Alg} = \sum_{t=1}^{T}\left(D\cdot\sum_{i=1}^{k} d(a_i^{(t-1)},a_i^{(t)}) + \sum_{j=1}^{r_t}\min_{i\in\{1,\dots,k\}} d(a_i^{(t)},v_{t,j})\right)$$

We will also discuss a variant we refer to as the Answer-First variant, where the requests have to be served before moving the servers. The cost of the algorithm in this case is

$$C_{Alg} = \sum_{t=1}^{T}\left(\sum_{j=1}^{r_t}\min_{i\in\{1,\dots,k\}} d(a_i^{(t-1)},r_{t,j}) + D\cdot\sum_{i=1}^{k} d(a_i^{(t-1)},a_i^{(t)})\right).$$

We will observe that this small modification of the problem has a huge impact on the best possible competitive ratio.

In our model, we also consider the *locality of requests* dictated by a parameter m_c limiting the distance between consecutive requests in case the number of requests is fixed for all time steps, i.e., $d(v_{t,i},v_{t+1,i}) \le m_c$. 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 $\delta\in(0,1)$. We compare the costs of an online algorithm to an offline optimum, whose servers are denoted by $o_1,\dots,o_k$ and whose cost is C_{Opt}.

2.2 Summary of Results

The results presented in the following are divided into two major parts: The first part focuses on the problem with just a single server, but allows for multiple requests to occur in one time step. In the second part, we discuss the problem with $k \geq 2$ servers, but allow only one request per time step to occur. Combining both an arbitrary number of servers and requests per time step is an open problem we further comment on in the conclusion of this thesis.

In both major parts, we first establish lower bounds on the competitive ratio for any randomized online algorithm against an oblivious adversary. As previously explained, this implies the same lower bound for the other adversary models as well, including a lower bound for deterministic algorithms. We prove that no online algorithm can achieve a competitive ratio independent of the number of rounds for the problem, even if $k = 1$ and the metric is the real line. 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. For the version with only one server, the competitive ratio then no longer depends on time, and only depends on the factor between the minimum and maximum number of requests in each time step and the inverse of the augmentation factor $1/\delta$. For $k \geq 2$, we show that resource augmentation alone is not sufficient to get a competitive ratio independent of time. Hence, we restrict the adversary to a *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. For these instances, we show a lower bound dependent on the ratio between m_c and m_s.

On the positive side, we show for $k = 1$ 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 that is $\mathcal{O}(1/\delta^{3/2})$-competitive in the Euclidean space when the number of requests per round is fixed. We also briefly sketch how to modify our analysis to work for the Answer-First variant and a varying number of requests per round.

In case $k \geq 2$, we construct a deterministic algorithm for the scenario in which locality of requests is given. 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.

2.3 One Server, Multiple Requests

In this section we deal with the case $k = 1$. For this problem we can obtain almost tight competitive ratios for the resource augmentation setting, in which the server can move a distance of $(1+\delta)m_s$. The lower bounds in this section obviously carry over to the next section with multiple servers when setting the number of requests to 1. The online algorithm and its analysis presented here, namely the potential function, contain some important ideas which will be reused in the problem for multiple servers later.

2.3.1 Lower Bounds

We first show how the different parameters of our model influence the quality of the best possible approximation by an online algorithm. It should be noted that the following lower bounds hold in the Euclidean space for an arbitrary dimension. We state all of them on the real line to demonstrate that a lower dimension does not simplify the problem.

For the lower bounds, we use Yao's principle [77] as described in Section 1.2. We construct a distribution of inputs by describing an adversary who issues the requests and may toss a fair coin to make binary decisions. Note that first, for a fixed input length there are only a finite number of different inputs that are being generated. Second, in all of the bounds the real line can be discretized into steps of size δm_s: Since the requests in all our bounds will always be located on those discrete points, we can restrict algorithms to the points with a loss of at most a factor 2. Hence, the number of different deterministic algorithms for a fixed input length is also finite, and the principle is applicable.

First we show that without the use of resource augmentation, there exists no online algorithm with a competitive ratio independent of the number of time steps T.

Theorem 2.3.1 Every randomized online algorithm for the Mobile Server problem has a competitive ratio of $\Omega(\sqrt{T}/D)$ against an oblivious adversary, even if there is only one request per time step.

Proof. 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_s to the left or right for the first x time steps. The cost for the adversary is at most $xDm_s + m_s \cdot \sum_{i=1}^{x} i \leq xDm_s + x^2 m_s$ 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_s 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_s towards the same direction it already did during the first x steps. The costs for the adversary are $(T-x)Dm_s$, while the costs for the online algorithm are at least $(T-x) \cdot xm_s$ with probability $\frac{1}{2}$. By choosing $x = \sqrt{T}$ the expected competitive ratio is as large as $\Omega(\sqrt{T}/D)$. ■

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 that, in every round, can move their server by a distance of $(1+\delta)m_s$ for some $\delta \in (0,1]$. Note that 1 is a lower bound on the competitive ratio of every algorithm regardless of the movement distance: The adversary can simply create an instance in which utilizing a higher movement distance than m_s does not decrease the algorithm's cost.

Theorem 2.3.2 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 $(1+\delta)m_s$ has an expected competitive ratio of $\Omega(\frac{1}{\delta} \cdot \frac{r_{max}}{r_{min}})$ against an oblivious adversary.

Proof. We use a similar sequence as in the proof of 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_s for x steps to the left or right with probability $1/2$ each, such that the distance between the two servers is at least xm_s with a probability of at least $1/2$ after these steps. The costs for the adversary are at most $Dxm_s + r_{min}x^2 m_s$.

The adversary now issues r_{max} requests at the position of its server and moves it by a distance m_s in the 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_s away from the adversary's server and moves towards it with the maximum distance in each round. The necessary number of steps for that is x/δ, since the distance between the two servers decreases by at most δm_s in every round.

The costs the online algorithm has to pay for serving the requests, in case the distance between the two servers is at least xm_s at the beginning of this phase, 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

$$\begin{aligned} r_{max} \cdot \sum_{i=1}^{\frac{x}{\delta}} (xm_s - i\delta m_s) &= r_{max}\left(\frac{x^2 m_s}{\delta} - \delta m_s \sum_{i=1}^{\frac{x}{\delta}} i\right) \\ &= r_{max}\left(\frac{x^2 m_s}{2\delta} - \frac{xm_s}{2}\right) \\ &\geq \frac{1}{4}\frac{r_{max}x^2 m_s}{\delta} \end{aligned}$$

by choosing $x \geq 2\delta$. These costs occur with probability $\frac{1}{2}$. The adversary pays $\frac{x}{\delta}Dm_s$ in this phase. By choosing $x \geq \frac{D}{\delta}$, the total costs of the adversary sum up to at most $3r_{min}x^2m_s$ over both described phases.

The adversary can now repeat the two phases arbitrarily often to get a bound for infinitely long sequences. 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 $\Omega(\frac{1}{\delta} \cdot \frac{r_{max}}{r_{min}})$. ■

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 server before answering the requests. Consider the scenario in which an algorithm has to answer the requests before moving the server (Answer-First variant). It can be shown that the competitive ratio of such algorithms depends on the number of requests in each time step even if it is fixed throughout the whole sequence. Note that the following theorem also holds for the resource augmentation scenario.

Theorem 2.3.3 In the Answer-First variant, 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 a probability of at least $\frac{1}{2}$, the distance between the two servers is at least m_s. This can be done by tossing a fair coin and then 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 server. The two steps may be repeated arbitrarily often 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_s with a probability of at least $\frac{1}{2}$, while the costs of the adversary are at most Dm_s. ■

The bound from Theorem 2.3.3 holds in addition to the bound from Theorem 2.3.1 in the case of no resource augmentation and to the bound from Theorem 2.3.2 in the scenario with resource augmentation.

2.3.2 The Move-to-Center Algorithm

In this section we provide a simple algorithm that achieves an optimal competitive ratio on the real line and a near optimal competitive ratio in the Euclidean space with arbitrary dimension. Our algorithm tries to imitate classical, phase-based solutions of the Page Migration problem such as the Move-to-Min algorithm [6], but incorporates the fact that it has to adhere to a movement constraint. Instead of waiting for D requests in total to arrive and then moving to the middle point of these requests, we move the server a $1/D$-fraction towards each request individually.

Algorithm 1 — Move-to-Center (MtC). Assume the algorithm has its server located at a point a and receives requests $v_1, \dots, v_r$. Let c be the point minimizing $\sum_{i=1}^{r} d(c, v_i)$. If c is not unique, pick one minimizing $d(a,c)$. MtC moves the server towards c for a distance of $\min\{1, \frac{r}{D}\} \cdot d(a,c)$ if this distance is less than $(1+\delta)m_s$. Otherwise it moves the server a distance of $(1+\delta)m_s$ towards c.

The goal of the analysis is to prove the following theorem:

Theorem 2.3.4 Let r_{min} and r_{max} be the minimum and maximum number of requests per time step. Algorithm *MtC* is $\mathcal{O}(\frac{1}{\delta} \cdot \frac{r_{max}}{r_{min}})$-competitive on the infinite line and $\mathcal{O}(\frac{1}{\delta^{3/2}} \cdot \frac{r_{max}}{r_{min}})$-competitive in the Euclidean space with arbitrary dimension using an augmented maximum moving distance of $(1+\delta)m_s$ for some $\delta \in (0,1]$.

We first analyze the algorithm for the case where some fixed number r of requests appear per time, i.e., $r_{max} = r_{min} =: r$. We then describe how the result of this analysis implies the results for the general case and for the Answer-First variant in Section 2.3.2. Note that in our proof we do not optimize the constants and instead focus on readability.

We first show that it is sufficient to analyze a simplified version of the problem in which the r requests occur exactly on the point c picked by MtC in every round.

Lemma 2.3.5 If MtC is α-competitive in an instance where all requests $v_{t,1}, \dots, v_{t,r_t}$ in a time step t occur on a single point c_t, MtC is $2\alpha+1$-competitive in an instance where c_t is the closest center of the requests in the respective time step.

Proof. Fix a time step and let $v_1, \dots, v_r$ be the requests in an instance where c is the center closest to the server of MtC and c' the center closest to the server of the optimal solution OPT when serving the requests. Let C_{Opt} and C_{Alg} be the costs of the optimal solution and the MtC algorithm in this instance and C'_{Opt} and C'_{Alg} be the respective costs in the simplified instance, where all requests occur on c, for serving the requests (in the current time step). Note that since c and c' are optimal centers we have $C_{Opt} \geq \sum_{i=1}^{r} d(c, v_i) = \sum_{i=1}^{r} d(c', v_i)$. Let o be the position of the optimal server in the simplified instance and a be the position of the algorithm's server (which is the same in both instances). We have $C'_{Opt} = r \cdot d(o,c) \leq r \cdot d(o,c') \leq \sum_{i=1}^{r} d(o, v_i) + \sum_{i=1}^{r} d(c', v_i) \leq 2 \cdot C_{Opt}$. For the algorithm, we have $C_{Alg} = \sum_{i=1}^{r} d(a, v_i) \leq r \cdot d(a,c) + \sum_{i=1}^{r} d(c, v_i) \leq C'_{Alg} + C_{Opt}$. Therefore $\frac{C_{Alg}}{C_{Opt}} \leq 2\frac{C'_{Alg}}{C'_{Opt}} + 1$. ■

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

In accordance with Section 2.1 we denote by a and a' the position of the algorithm's server before and after moving it. o and o' will be used for the optimal server positions before and after moving, respectively.

For better readability, we define the following abbreviations which we use for the distances: $a_1 := d(a,a')$, $a_2 := d(a',c)$, $s_1 := d(o,o')$, $s_2 := d(o',c)$, $p := d(o,a)$, $h := d(o',a)$ and $q := d(o',a')$. An illustration can be found in Figure 2.1.

Using this notation, the cost of the online algorithm is

$$C_{Alg} = D \cdot a_1 + r \cdot a_2$$

and the optimal cost is

$$C_{Opt} = D \cdot s_1 + r \cdot s_2.$$

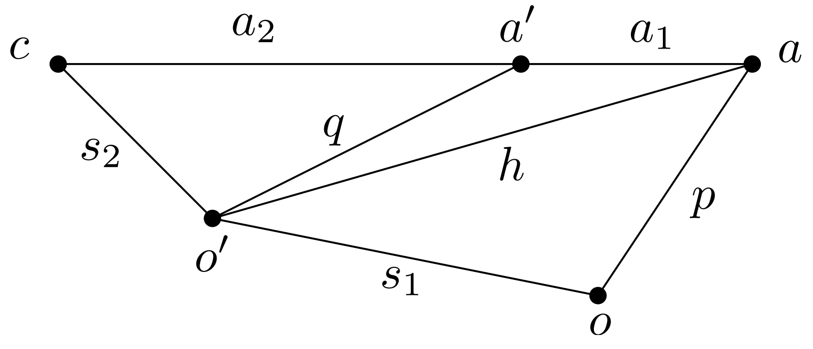

Figure 2.1: Illustration of relevant points and distances for estimating the potential difference.

We start with an estimation of the difference $d(o',a) - d(o',a') = h - q$ which is essential for utilizing our potential function in the upcoming parts of the analysis. The lemma describes the progression of the algorithm's server towards the optimal server when the optimal server is close to the center of the requests.

Lemma 2.3.6 If $d(o',c) \leq \frac{\sqrt{\delta}}{2} d(a',c)$, then $d(o',a) - d(o',a') \geq \frac{1+\frac{1}{2}\delta}{1+\delta} d(a,a')$.

Proof. We want to get a lower bound for $h - q$ given a_1. If we assume fixed values for h and s_2, then the value of q is maximized by choosing the angle α between the lines connecting a with a' and a with o' as large as possible. This can be done by setting the angle between the lines from c to o' and c to a' to 90 degrees as shown in Figure 2.2. Note that no matter the dimension of the underlying space, a, a' and c will always form a line and hence together with o' they lie on a plane.

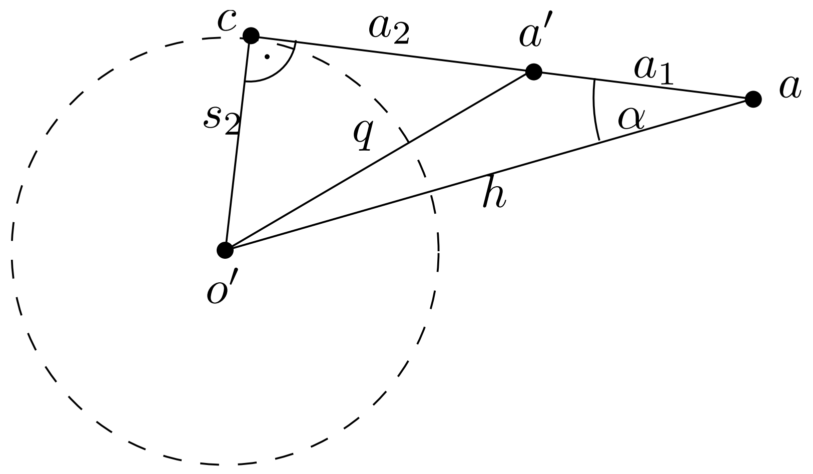

Figure 2.2: Setting where q is maximized given h, s_2 and a_1. Fixing the positions of a and a', the only decision left is where to place c on the circle.

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 $s_2 = \varepsilon \cdot a_2$ for some $\varepsilon \in (0,1)$. Then

$$\begin{aligned}
h-q &= \sqrt{(\varepsilon a_2)^2 + (a_1+a_2)^2} - \sqrt{(\varepsilon a_2)^2 + a_2^2} \\
&= \sqrt{(1+\varepsilon^2)a_2^2 + 2a_1a_2 + a_1^2} - \sqrt{1+\varepsilon^2}a_2 \\
&\geq \sqrt{(1+\varepsilon^2)a_2^2 + 2\sqrt{1+\varepsilon^2}a_2\tfrac{a_1}{\sqrt{1+\varepsilon^2}} + \tfrac{a_1^2}{1+\varepsilon^2}} - \sqrt{1+\varepsilon^2}a_2 \\
&= \sqrt{\left(\sqrt{1+\varepsilon^2}a_2 + \tfrac{a_1}{\sqrt{1+\varepsilon^2}}\right)^2} - \sqrt{1+\varepsilon^2}a_2 \\
&= \tfrac{a_1}{\sqrt{1+\varepsilon^2}}.
\end{aligned}$$

It therefore holds

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

Since

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

this implies the lemma. ■

For server locations a and o of the online algorithm and the optimal solution, respectively, the potential ϕ is defined as

$$\phi(o,a) := \begin{cases} 4D \cdot d(o,a) & \text{if } d(o,a) \leq \delta\frac{Dm_s}{4r} \\ 16\frac{r}{\delta m_s} \cdot d(o,a)^2 & \text{otherwise.} \end{cases}$$

Note that this function is equal to $\phi(o,a) = \max\{4D \cdot d(o,a)\ ,\ 16\frac{r}{\delta m_s} \cdot d(o,a)^2\}$. The linear part of the potential is similar to the function used in the analysis of many Page Migration algorithms, while the quadratic part captures the situation when the algorithm's server is far away from the optimal server similar as in the lower bound instances of the problem. In these cases, the algorithm incurs a cost quadratic in the distance to the optimum before it catches up to it, which is reflected in the potential. For each of the following cases we bound the potential difference $\Delta\phi = \phi(o',a') - \phi(o,a)$ and the cost of the online algorithm as functions of C_{Opt}.

Analysis for $r > D$

In the following, we analyze the algorithm in the case $r > D$ for both the real line and the Euclidean space of higher dimension. Most arguments apply to any dimension; if there is a special argument needed for the better bound on the line it is stated explicitly in the respective case.

1. $p \leq \delta\frac{Dm_s}{4r} \leq \delta m_s$:
Since the algorithm will either move its server onto c or by a distance of $(1+\delta)m_s$, it will be the same distance or closer to c than the optimal solution after the current time step, i.e., $a_2 \leq s_2$. If $q \leq \delta\frac{Dm_s}{4r}$ we have

$$\begin{aligned}
\Delta\phi &= 4D \cdot (q-p) \\
&\leq 4r \cdot (s_2 + a_2) - 4D \cdot p \\
&\leq 8 \cdot C_{Opt} - 4D \cdot p
\end{aligned}$$

and

$$\begin{aligned} C_{Alg} &= D\cdot a_1 + r\cdot a_2 \\ &\leq D\cdot(p+s_1+q)+r\cdot a_2 \\ &\leq D\cdot(p+s_2)+r\cdot(s_2+2a_2) \\ &\leq 3\cdot C_{Opt}+D\cdot p. \end{aligned}$$

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

$$\begin{aligned} \Delta\phi &= 16\frac{r}{\delta m_s}\cdot q^2 - 4D\cdot p \\ &\leq 16\frac{r}{\delta m_s}\cdot(q^2-p^2) \\ &= 16\frac{r}{\delta m_s}\cdot(q+p)\cdot(q-p) \\ &\leq 16\frac{r}{\delta m_s}\cdot 2q\cdot(2+\delta)m_s \\ &\leq 96\frac{r}{\delta}\cdot(s_2+a_2) \\ &\leq \frac{192}{\delta}\cdot C_{Opt} \end{aligned}$$

and

$$\begin{aligned} C_{Alg} &= D\cdot a_1 + r\cdot a_2 \\ &\leq D\cdot(p+s_1+q)+r\cdot a_2 \\ &\leq D\cdot s_1+r\cdot(2q+a_2) \\ &\leq D\cdot s_1+r\cdot(2s_2+3a_2) \\ &\leq 5\cdot C_{Opt}. \end{aligned}$$

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

$$\begin{aligned} \Delta\phi &= 4D\cdot q - 16\frac{r}{\delta m_s}\cdot p^2 \\ &\leq 4D\cdot(a_2+s_2)-4D\cdot p \\ &\leq 8\cdot C_{Opt}-4D\cdot p \end{aligned}$$

and

$$\begin{aligned} C_{Alg} &= D\cdot a_1 + r\cdot a_2 \\ &\leq D\cdot(p+s_1+q)+r\cdot a_2 \\ &\leq C_{Opt}+2D\cdot p. \end{aligned}$$

Else, $p > \delta m_s$, and we have

$$\begin{aligned} \Delta\phi &= 4D\cdot q - 16\frac{r}{\delta m_s}\cdot p^2 \\ &\leq 4D\cdot q - 16r\cdot p \\ &\leq -12r\cdot p \end{aligned}$$

and

$$\begin{aligned} C_{Alg} &= D\cdot a_1 + r\cdot a_2 \\ &\leq D\cdot(p+s_1+q)+r\cdot(q+s_2) \\ &\leq C_{Opt}+3r\cdot p. \end{aligned}$$

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

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

$$\begin{aligned} \Delta\phi &= 16\frac{r}{\delta m_s}\cdot(q^2-p^2) \\ &= 16\frac{r}{\delta m_s}\cdot(q+p)\cdot(q-p) \\ &\leq 16\frac{r}{\delta m_s}\cdot(q+p)\cdot(-\frac{\delta}{2}m_s) \\ &\leq -8r\cdot(q+p) \end{aligned}$$

complemented by

$$\begin{aligned} C_{Alg} &= D\cdot a_1 + r\cdot a_2 \\ &\leq D\cdot(p+s_1+q)+r\cdot(s_2+q) \\ &\leq C_{Opt}+2r\cdot(p+q). \end{aligned}$$

3b. $q-h > -(1+\frac{\delta}{2})m_s$ and $p \geq 4m_s$:
We estimate a_2 in two different ways: First we establish a bound for an arbitrary dimension. If $a_1 = (1+\delta)m_s$, then $h-q < (1+\frac{\delta}{2})m_s \leq \frac{1+\frac{\delta}{2}}{1+\delta}a_1$. According to Lemma 2.3.6, this implies $\frac{\sqrt{\delta}}{2}a_2 \leq s_2$.

If the given space is the real line, $a_1 = (1+\delta)m_s$ directly implies $a_2 \leq s_2$ since o' cannot be located between a' and c (otherwise $q-h = -(1+\delta)m_s$).

For any dimension, if $a_1 < (1+\delta)m_s$, then $a_2 = 0 \leq s_2$.

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

$$\begin{aligned} \Delta\phi &= 16\tfrac{r}{\delta m_s}\cdot(q^2-p^2) \\ &= 16\tfrac{r}{\delta m_s}\cdot(q+p)\cdot(q-p) \\ &\leq 16\tfrac{r}{\delta m_s}\cdot 3q\cdot(a_1+s_1) \\ &\leq 16\tfrac{r}{\delta m_s}\cdot 3q\cdot 3m_s \\ &\leq 144\tfrac{r}{\delta}\cdot(s_2+a_2) \\ &\leq \tfrac{432}{\delta^{3/2}}\cdot C_{Opt} \end{aligned}$$

and

$$\begin{aligned} C_{Alg} &= D\cdot a_1 + r\cdot a_2 \\ &\leq D\cdot(p+s_1+q)+r\cdot a_2 \\ &\leq D\cdot(s_1+3q)+r\cdot a_2 \\ &\leq D\cdot(s_1+3(s_2+a_2))+r\cdot a_2 \\ &\leq \tfrac{10}{\sqrt{\delta}}\cdot C_{Opt}. \end{aligned}$$

Both estimations can be reduced by a factor of $\frac{1}{\sqrt{\delta}}$ when dealing with the infinite line, using $a_2 \leq s_2$.
In case $\frac{1}{2}p > q$ we use

$$\begin{aligned} \Delta\phi &= 16\tfrac{r}{\delta m_s}\cdot(q^2-p^2) \\ &= 16\tfrac{r}{\delta m_s}\cdot(q+p)\cdot(q-p) \\ &\leq 16\tfrac{r}{\delta m_s}\cdot(q+p)\cdot(-\tfrac{1}{2}p) \\ &\leq -4D\cdot p \end{aligned}$$

and

$$\begin{aligned} C_{Alg} &= D\cdot a_1 + r\cdot a_2 \\ &\leq D\cdot(p+s_1+q)+r\tfrac{2}{\sqrt{\delta}}\cdot s_2 \\ &\leq D\cdot(p+s_1+a_2+s_2)+r\tfrac{2}{\sqrt{\delta}}\cdot s_2 \\ &\leq \tfrac{5}{\sqrt{\delta}}\cdot C_{Opt}+D\cdot p. \end{aligned}$$

3c. $q-h > -(1+\frac{\delta}{2})m_s$ and $p < 4m_s$:
We get $\frac{\sqrt{\delta}}{2}a_2 \le s_2$ and $a_2 \le s_2$ for arbitrary dimension and the real line, respectively, as before. Since $q \le s_1 + p + a_1 \le p + 3m_s$ we have

$$\begin{aligned}
\Delta\phi &= 16\tfrac{r}{\delta m_s}\cdot(q^2-p^2)\\
&= 16\tfrac{r}{\delta m_s}\cdot(q+p)\cdot(q-p)\\
&\le 16\tfrac{r}{\delta m_s}\cdot(q+p)\cdot q - 16\tfrac{r}{\delta m_s}\cdot(q+p)\cdot p\\
&\le 16\tfrac{r}{\delta m_s}\cdot 11m\cdot q - 16\tfrac{r}{\delta m_s}\cdot(q+p)\cdot\delta\tfrac{Dm_s}{4r}\\
&\le 176\tfrac{r}{\delta}\cdot(s_2+a_2) - 4D\cdot(q+p)\\
&\le \tfrac{528}{\delta^{3/2}}\cdot C_{Opt} - 4D\cdot(q+p)
\end{aligned}$$

and

$$\begin{aligned}
C_{Alg} &= D\cdot a_1 + r\cdot a_2\\
&\le \tfrac{5}{\sqrt{\delta}}\cdot C_{Opt} + D\cdot p.
\end{aligned}$$

Again, both estimations can be reduced by a factor of $\frac{1}{\sqrt{\delta}}$ for the 1-dimensional space.
In all cases we have $C_{Alg}+\Delta\phi \le \mathcal{O}(\frac{1}{\delta^{3/2}})\cdot C_{Opt}$ for arbitrary dimensions and $C_{Alg}+\Delta\phi \le \mathcal{O}(\frac{1}{\delta})\cdot C_{Opt}$ for the 1-dimensional Euclidean space.

Analysis for $r \le D$

For $r \le D$ we first give a detailed analysis for arbitrary dimensions and then briefly describe how to modify the necessary parts to work for the 1-dimensional space so that the competitive ratio becomes $\mathcal{O}(\frac{1}{\delta})$.

1. $q \le \delta\frac{Dm_s}{4r}$:
Note that $-16\frac{r}{\delta m_s}\cdot p^2 \le -4D\cdot p$ in case $p > \delta\frac{Dm_s}{4r}$ and hence $\Delta\phi \le 4D\cdot(q-p)$ for all values of p.

1a. $s_2 \le \frac{\sqrt{\delta}}{2}a_2$:
Using Lemma 2.3.6, we bound the potential difference by

$$\begin{aligned}
\Delta\phi &\le 4D\cdot(q-p)\\
&\le 4D\cdot(q-h+h-p)\\
&\le -4D\tfrac{1+\frac{\delta}{2}}{1+\delta}\cdot a_1 + 4D\cdot s_1.
\end{aligned}$$

Note that either $a_1 = \frac{r}{D}(a_1+a_2)$ or $a_1 = (1+\delta)m_s$. If $a_1 = \frac{r}{D}(a_1+a_2)$, then $\Delta\phi \le -2r\cdot(a_1+a_2)+4\cdot C_{Opt}$. We have $C_{Alg} \le D\cdot\frac{r}{D}(a_1+a_2)+ra_2 \le 2r\cdot(a_1+a_2)$.

Otherwise, $a_1 = (1+\delta)m_s$ which gives us $\Delta\phi \le -4D(1+\frac{\delta}{2})m_s + 4\cdot C_{Opt}$. In this case $q \le \delta\frac{Dm_s}{4r}$ can be used to get

$$\begin{aligned}
C_{Alg} &= D\cdot a_1 + r\cdot a_2\\
&\le D(1+\delta)m_s + r\cdot(\delta\tfrac{Dm_s}{4r}+s_2)\\
&\le 2D(1+\delta)m_s + C_{Opt}.
\end{aligned}$$

1b. $s_2 > \frac{\sqrt{\delta}}{2}a_2$:
If $\frac{1}{2}p \le q$ then

$$\begin{aligned}
\Delta\phi &\le 4D\cdot(q-p)\\
&\le 4D\cdot(s_1+p+a_1-p)\\
&\le 4D\cdot s_1 + 4r\cdot(a_1+a_2)\\
&\le 4D\cdot s_1 + 4r\cdot(p+s_1+q+a_2)\\
&\le 8D\cdot s_1 + 4r\cdot(3q+a_2)\\
&\le 8D\cdot s_1 + 4r\cdot(3s_2+4a_2)\\
&\le \tfrac{44}{\sqrt{\delta}}\cdot C_{Opt}
\end{aligned}$$

and

$$\begin{aligned}
C_{Alg} &= D\cdot a_1 + r\cdot a_2\\
&\le r\cdot(a_1+a_2)+r\cdot a_2\\
&\le r\cdot(p+s_1+q)+2r\cdot a_2\\
&\le r\cdot(s_1+3q)+2r\cdot a_2\\
&\le r\cdot(s_1+3(s_2+a_2))+2r\cdot a_2\\
&\le \tfrac{11}{\sqrt{\delta}}\cdot C_{Opt}.
\end{aligned}$$

Else $\frac{1}{2}p > q$ and we have

$$\begin{aligned}
\Delta\phi &\le 4D\cdot(q-p)\\
&\le -2D\cdot p
\end{aligned}$$

and

$$\begin{aligned}
C_{Alg} &= D\cdot a_1 + r\cdot a_2\\
&\le D\cdot(p+s_1+q)+r\cdot a_2\\
&\le \tfrac{2}{\sqrt{\delta}}\cdot C_{Opt}+\tfrac{3}{2}D\cdot p.
\end{aligned}$$

2. $q > \delta\frac{Dm_s}{4r}$:
Note that $-4D\cdot p \le -16\frac{r}{\delta m_s}\cdot p^2$ in case $p\le \delta\frac{Dm_s}{4r}$ and hence $\Delta\phi \le 16\frac{r}{\delta m_s}\cdot(q^2-p^2)$ for all values of p.

2a. $s_2 \le \frac{\sqrt{\delta}}{2}a_2$:
We have

$$\begin{aligned}
\Delta\phi &\le 16\tfrac{r}{\delta m_s}\cdot(q^2-p^2)\\
&= 16\tfrac{r}{\delta m_s}\cdot(q+p)\cdot(q-p)\\
&\le 16\tfrac{r}{\delta m_s}\cdot(q+p)\cdot\left(-\tfrac{1+\frac{\delta}{2}}{1+\delta}a_1+s_1\right).
\end{aligned}$$

If $a_1=(1+\delta)m_s$ then $\Delta\phi\le -8r\cdot(q+p)$ and

$$\begin{aligned}
C_{Alg} &= D\cdot a_1 + r\cdot a_2\\
&\le 2r\cdot(a_1+a_2)\\
&\le 2r\cdot(q+s_1+p+s_2+q)\\
&\le 2\cdot C_{Opt}+2r\cdot(p+q).
\end{aligned}$$

Else, $a_1=\frac{r}{D}(a_1+a_2) < (1+\delta)m_s$ and hence $a_1+a_2\le 2\frac{Dm_s}{r}$. We get

$$\begin{aligned}
\Delta\phi &\le 16\tfrac{r}{\delta m_s}\cdot(q^2-p^2)\\
&= 16\tfrac{r}{\delta m_s}\cdot(q+p)\cdot(q-p)\\
&\le 16\tfrac{r}{\delta m_s}\cdot(q+p)\cdot s_1 - 16\tfrac{r}{\delta m_s}\cdot(q+p)\cdot\tfrac{1+\frac{\delta}{2}}{1+\delta}\cdot a_1\\
&\le 16\tfrac{r}{\delta m_s}\cdot(s_1+a_1+2q)\cdot s_1 - 16\tfrac{r}{\delta m_s}\cdot\delta\tfrac{Dm_s}{4r}\cdot\tfrac{1}{2}\tfrac{r}{D}\cdot(a_1+a_2)\\
&\le 16\tfrac{r}{\delta m_s}\cdot(s_1+a_1+2s_2+2a_2)\cdot s_1 - 2r\cdot(a_1+a_2)\\
&\le 16\tfrac{r}{\delta m_s}\cdot(s_1+4\tfrac{Dm_s}{r})\cdot s_1 + 16\tfrac{r}{\delta m_s}\cdot 2s_2\cdot s_1 - 2r\cdot(a_1+a_2)\\
&\le 80\tfrac{D}{\delta}\cdot s_1 + 32\tfrac{r}{\delta}\cdot s_2 - 2r\cdot(a_1+a_2)\\
&\le \tfrac{80}{\delta}\cdot C_{Opt} - 2r\cdot(a_1+a_2).
\end{aligned}$$

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

2b. $s_2 > \frac{\sqrt{\delta}}{2} a_2$:
If $\frac{1}{2}p \leq q$ we use this to get

$$\begin{aligned}
\Delta\phi &= 16\tfrac{r}{\delta m_s} \cdot (q+p) \cdot (q-p) \\
&\leq 16\tfrac{r}{\delta m_s} \cdot 3q \cdot 3m_s \\
&\leq 16\tfrac{r}{\delta m_s} \cdot 3(a_2+s_2) \cdot 3m_s \\
&= 432\tfrac{r}{\delta^{3/2}} \cdot s_2
\end{aligned}$$

and

$$\begin{aligned}
C_{Alg} &= D \cdot a_1 + r \cdot a_2 \\
&\leq 2r \cdot (a_1 + a_2) \\
&\leq 2r \cdot (p + s_1 + q + a_2) \\
&\leq 2r \cdot (s_1 + 3q + a_2) \\
&\leq 2r \cdot (s_1 + 3s_2 + 4a_2) \\
&\leq \tfrac{22}{\sqrt{\delta}} \cdot C_{Opt}.
\end{aligned}$$

Otherwise, we have $\frac{1}{2}p > q$ and therefore

$$\begin{aligned}
\Delta\phi &= 16\tfrac{r}{\delta m_s} \cdot (q+p) \cdot (q-p) \\
&\leq 16\tfrac{r}{\delta m_s} \cdot (q+p) \cdot (-q) \\
&= -4D \cdot (q+p)
\end{aligned}$$

and $C_{Alg} \leq D \cdot (q + s_1 + p) + r \cdot a_2 \leq \frac{2}{\sqrt{\delta}} \cdot C_{Opt} + D \cdot (p+q)$.

To get the bound for the line, in each of the two big cases replace the distinction whether $s_2 \leq \frac{\sqrt{\delta}}{2} a_2$ by $s_2 \leq a_2$. Also the estimation of $q-h$ under the use of Lemma 2.3.6 may be replaced by $q - h \leq -a_1$.

Again, in all cases we have $C_{Alg} + \Delta\phi \leq \mathcal{O}(\frac{1}{\delta^{3/2}}) \cdot C_{Opt}$ for the case of an arbitrary dimension and $C_{Alg} + \Delta\phi \leq \mathcal{O}(\frac{1}{\delta}) \cdot C_{Opt}$ for the 1-dimensional case.

General Case and Answer-First Variant

So far we assumed that the number of requests per time step is fixed to a constant r. We now briefly describe how to modify the analysis such that the result for general r_{min} and r_{max} follows. Furthermore we show that the Move-To-Center algorithm is also almost optimal in the Answer-First variant.

The general result can be derived as follows: 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 cost of the algorithm this is then still possible. However, if the potential difference is positive it may add a term that is $\mathcal{O}(\frac{r_{max}}{r_{min}})$ times the optimal cost. This concludes the proof of Theorem 2.3.4.

Theorem 2.3.7 Let $r \geq D$ be the fixed number of requests per time step. Algorithm MtC is $\mathcal{O}(\frac{1}{\delta^{3/2}} \cdot \frac{r}{D})$-competitive utilizing a maximum moving distance of $(1+\delta)m_s$ in the Answer-First variant.

Proof. We relate the cost of the algorithm in the Answer-First variant to the cost for the same request sequence in the original model. The optimal solution may be assumed to be the same for both model variants given the same request sequence, since we can insert r dummy requests on the starting point of the server at the beginning of the sequence which allows the optimal solution to operate the same as in the Answer-First model and only has additional costs of at most rm_s. This will also not change the behavior of the *MtC* algorithm. However, the cost of the online algorithm will increase by $r \cdot d(a,a')$ for each step.

In case $r \leq D$, the additional term $r \cdot d(a,a')$ in the cost of the algorithm can be estimated via $r \cdot d(a,a') \leq D \cdot d(a,a')$ which implies an increase of the algorithm's cost by at most a factor 2, since this is equal to the movement cost. For the case $r > D$, the additional term in the cost of the algorithm $r \cdot d(a,a') \leq \frac{r}{D} \cdot D \cdot d(a,a')$ implies the cost of the algorithm increases by a factor of at most $2\frac{r}{D}$. ∎

2.3.3 Locality of Requests

We turn our attention to instances where we demand a locality of requests: i.e., the number of requests is fixed and the sequence can be represented as r agents traveling a distance of at most m_c per time step. Note that the assumption we made in the original analysis of the *MtC* algorithm still holds: Since each agent posing the requests moves by at most m_c, there always is a center point c minimizing the sum of distances to all agents and which is a distance of at most m_c to the previous center picked by *MtC*. Hence, we may still assume with only a constant factor loss, that the r requests all occur at the chosen center c of the *MtC* algorithm.

We first show that the competitive ratio depends on T if m_c is larger than m_s if we do not make use of resource augmentation.

Theorem 2.3.8 Let $m_c > m_s$. No randomized online algorithm can be better than $\Omega(\frac{\sqrt{T}}{D}\frac{(m_c - m_s)}{m_c})$-competitive against an oblivious adversary when consecutive requests (by the same agent) are within a distance m_c.

Proof. The construction is similar to the bound in Theorem 2.3.1. We divide the input sequence into two major phases. In the first phase, for some value x chosen later, the adversary moves the server $x \cdot \frac{m_c}{m_s}$ rounds by a distance of m_s in one of two opposite directions determined by a fair coin. The requesting agent stays at the starting point and moves to the position of the adversary only for the last x time steps. The costs for the adversary in this step are at most $Dxm_c + x^2\frac{m_c^2}{m_s}$. By the end of this phase, with probability 1/2 the online algorithm has a distance of at least $x(m_c - m_s)$ to the agent and the adversary, since it can only move x steps towards the agent after the outcome of the random experiment is revealed.

In the second phase the adversary and the agent continue to move in the same direction by a distance of m_s in each round. Hence the costs of the online algorithm are at least $(T - x\frac{m_c}{m_s})x(m_c - m_s)$ while the costs of the adversary are at most $D(T - x\frac{m_c}{m_s})m_s$. By setting $x = \sqrt{T} \cdot \frac{m_s}{m_c}$ we get a competitive ratio of $\Omega(\frac{\sqrt{T}}{D}\frac{(m_c - m_s)}{m_c})$. ∎

Since this model is a restriction of the general model, Theorem 2.3.4 directly gives us that by using an augmented maximum moving distance of $(1+\delta)m_s$ for the online algorithm, we can achieve a competitive ratio independent of T.

Corollary 2.3.9 MtC is $\mathcal{O}(\frac{1}{\delta^{3/2}})$-competitive, when consecutive requests are bounded by a distance m_c, using an augmented maximum moving distance of $(1+\delta)m_s$ for some $\delta \in (0,1]$.

Until now, we only discussed the case $m_c > m_s$. Note that in case $m_c \leq m_s$ standard solutions to the Page Migration problem still do not apply, since they require moving to a specific point after collecting a batch of requests. This point may still lie outside the allowed moving distance m_s. However, we can show that the MtC algorithm used to solve our original problem achieves a constant competitive ratio.

The algorithm in our case does the following: Upon receiving the requests with center point c, move the server a distance of $\min\{m_s, \min\{\frac{r}{D}, 1\} \cdot d(a,c)\}$ towards c.

For the analysis, we use a simpler version of Lemma 2.3.6 which we state separately since it will also be useful in the next section.

Lemma 2.3.10 If $d(o',c) \leq d(a',c)$, then $d(o',a) - d(o',a') \geq \frac{1}{\sqrt{2}} d(a,a')$.

Proof. Replace ε with 1 at the end of the first estimation in the proof of Lemma 2.3.6. ■

Theorem 2.3.11 Let $m_c \leq m_s$. The MtC algorithm is $\mathcal{O}(1)$-competitive when consecutive requests are bounded by a distance m_c.

Proof. We define the potential $\phi(a,o) := 2^{\frac{3}{2}} D \cdot d(a,o)$. We use the same notation as in the proof of Theorem 2.3.4, shown in Figure 2.1. Hence

$$\Delta\phi = 2^{\frac{3}{2}} D \cdot (q-p) \leq 2^{\frac{3}{2}} D \cdot (q-h+s_1).$$

1. $r > D$:
Since we always move the server onto c we have $a_2 = 0$. We hence have

$$C_{Alg} = Da_1 \leq D(p+s_1+q) \leq D(p+s_1)+rs_2 = Dp + C_{Opt}$$

and

$$\Delta\phi = 2^{\frac{3}{2}} D(q-p) \leq 2^{\frac{3}{2}} rs_2 - Dp = 2^{\frac{3}{2}} C_{Opt} - Dp$$

where the first inequality is due to $q \leq a_2 + s_2$.

2. $r \leq D$:
We first note that $a_2 \leq Dm_s$: In each round, the distance to c increases by at most m_c, and decreases by m_s as soon as the distance to the center is at least $\frac{D}{r} m_s$. We distinguish two cases:

If $s_2 \leq a_2$, then $\Delta\phi \leq 2^{\frac{3}{2}} Ds_1 - 2Da_1$ by Lemma 2.3.10. Either $a_1 = \frac{r}{D}(a_1+a_2)$ and hence

$$C_{Alg} + \Delta\phi = D \cdot \frac{r}{D}(a_1+a_2) + ra_2 + \Delta\phi \leq 2^{\frac{3}{2}} Ds_1$$

or $a_1 = m_s$ and

$$C_{Alg} + \Delta\phi \leq 2Dm_s + \Delta\phi \leq 2^{\frac{3}{2}} Ds_1$$

since $a_2 \leq Dm_s$.

For the second case, which is $a_2 < s_2$, we make use of

$$\begin{aligned} C_{Alg} &\leq r(a_1+2a_2) \\ &\leq r(p+s_1+q+2a_2) \\ &\leq rp + 4C_{Opt}. \end{aligned}$$

If $q \leq (1 - \frac{1}{2^{3/2}})p$ then $\Delta\phi \leq -Dp$ and

$$C_{Alg} + \Delta\phi \leq 4C_{Opt}.$$

Else

$$\begin{aligned} \Delta\phi &= 2^{\frac{3}{2}} D(q-p) \\ &\leq 2^{\frac{3}{2}} D(s_1+p+a_1-p) \\ &\leq 2^{\frac{3}{2}} Ds_1 + 2^{\frac{3}{2}} r(a_1+a_2) \\ &\leq 2^{\frac{3}{2}} Ds_1 + 2^{\frac{3}{2}} r(p+s_1+q+a_2) \\ &\leq 2^{\frac{5}{2}} Ds_1 + 2^{\frac{3}{2}} r(3q+a_2) \\ &\leq 2^{\frac{5}{2}} Ds_1 + 2^{\frac{3}{2}} r(3s_2+4a_2) \\ &\leq 36C_{Opt} \end{aligned}$$

and

$$C_{Alg} \le rp + 4C_{Opt} \le 8C_{Opt}.$$

■

2.4 Multiple Servers, One Request

We now turn our attention to the scenario in which we have multiple (k) servers, but only one request per time step. Having more than one server increases the challenge of the problem significantly, as illustrated by the lower bounds shown in the following section. Our algorithms for the model rely on the simulation of existing algorithms for the k-Server and k-Page Migration problems which they use as a guidance for the placement. This takes away some of the challenges of our model which are the same as in these problems, allowing us to focus on the difficulties introduced by the limited speed of the servers.

2.4.1 Lower Bounds

The following lower bounds investigate the additional challenges introduced into the problem when dealing with $k \ge 2$ servers. In contrast to the model with only one server, they show the importance of both the resource augmentation and the locality of requests in order to achieve a competitive ratio independent of time. As before, 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 [11, 60]). Even when the maximum distance between two consecutive requests 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 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 bound of $\Omega(\sqrt{T}/D)$ from Theorem 2.3.1 obviously still applies in this case. For more than one server, we obtain an additional bound that cannot be resolved with the help of the speed augmentation.

Theorem 2.4.1 For $k \ge 2$, every randomized online algorithm for the k-Mobile Server problem has a competitive ratio of at least $\Omega(\frac{T}{Dk^2})$.

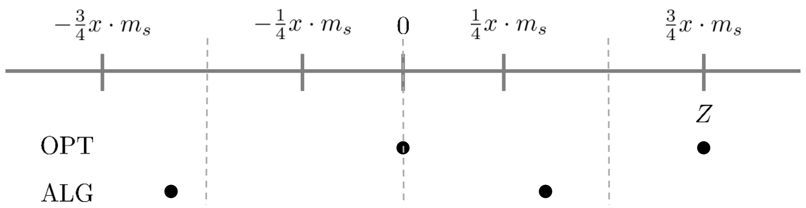

Figure 2.3: The line as used in the proof of Theorem 2.4.1 for $k = 2$. The circles indicate a possible configuration of the servers of the online algorithm and the optimal solution at the beginning of the second phase. The four segments are indicated by the dashed lines. The adversary has successfully chosen a segment that the online algorithm does not occupy.

Proof. We first explain the proof for $k = 2$ in detail and then describe how to extend it to $k > 2$ servers. All servers start on the same position on the real line, which we identify with 0. The input proceeds in two phases. In the first phase, there are $x \cdot m_s$ requests on point 0. At the start of the second phase, choose one of the following points uniformly at random: $-\frac{3}{4}x \cdot m_s, -\frac{1}{4}x \cdot m_s, \frac{1}{4}x \cdot m_s, \frac{3}{4}x \cdot m_s$. We refer to this point as Z. In the second phase, issue $x/8$ requests on Z.

The optimal solution moves one server to Z during the first phase and has costs of at most $Dx \cdot m_s$. Since the online algorithm only has two servers, both of its servers have a distance of at least $\frac{x}{8} \cdot m_s$ to Z with a probability of at least $1/2$: Divide the line into four segments of size $\frac{1}{4}x \cdot m_s$. The online algorithm can occupy at most two of theses segments (cf. Figure 2.3). As a consequence, the expected costs for the online algorithm in the second phase are at least $\frac{1}{2}\sum_{i=1}^{\frac{x}{8}}(\frac{1}{8}x \cdot m_s - i \cdot m_s) \geq \frac{x^2}{264}m_s$. The cost ratio is then $\Omega(x/D) = \Omega(T/D)$.

For $k > 2$ servers, split 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 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 $Z_1, \dots, Z_{k-1}$. During the first phase, $4kx$ requests appear at the starting point, and the adversary moves one server to $Z_1, \dots, Z_{k-1}$ each, the last server remains at the starting point. The moving costs for the adversary are $\mathcal{O}(Dk^2x \cdot m_s)$.

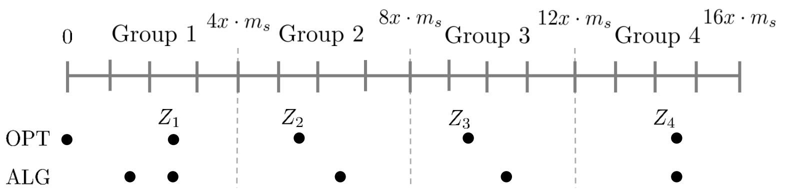

Figure 2.4: The line as used in the proof of Theorem 2.4.1 for $k > 2$ (here $k = 5$). The circles indicate a possible configuration of the servers of the online algorithm and the optimal solution at the beginning of the second phase. The groups are indicated by the dashed lines. The adversary has successfully chosen two inner segments that the online algorithm does not occupy.

In the second phase, on each point $Z_1, \dots, Z_{k-1}$, $x/4$ requests appear in order of distance to the starting point. As before, 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 $\Omega(x^2 m_s)$. Now we iterate over the groups of segments: Consider the group that 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 that cover inner segments as "used up" for the following groups and we therefore may apply the arguments inductively. Let a, b and c be 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 $a + \frac{1}{2}b$. It is easy to see that the number of these segments are in $\Omega(k)$.

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

As a consequence of this lower bound, we apply the two modifications to our model mentioned previously, 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 the case of one server: i.e., we 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 2.4.1 (the ratio shrinks, but still depends on T). Hence, we also use the locality of requests, i.e., restrict the distance between two consecutive requests to a maximum distance of m_c. Note that only restricting the distance between consecutive requests does also not remove the dependence on T, as was shown in the previous section (see Theorem 2.3.8). The following theorem can be obtained in a similar way as Theorem 2.4.1:

Theorem 2.4.2 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 $\Omega(\frac{m_c}{m_s})$.

Proof. The proof follows the structure of the proof of Theorem 2.4.1. We first describe the bound for $k = 2$ and then extend it to $k > 2$ servers. All servers start on the same position denoted by 0 on the real line. The input is given in three phases. In the first phase, x requests are issued consecutively on point 0. At the beginning of the second phase, choose uniformly at random one of the points $-\frac{3}{4}x \cdot m_s, -\frac{1}{4}x \cdot m_s, \frac{1}{4}x \cdot m_s, \frac{3}{4}x \cdot m_s$. Let the chosen point be Z. Now move the request by m_c towards Z in each time step until Z is reached. In the third phase, issue $x/8$ requests consecutively at Z.

Observe that the request needs at most $x \cdot \frac{m_s}{m_c}$ time steps to get to Z. The optimal solution moves one of its servers in the first phase to Z and has a movement cost of at most $Dx \cdot m_s$. Since the distance between the request and an optimal server in each step in the second phase is at most $\frac{x}{2} \cdot m_s$, the costs of the optimal solution in this phase are bounded by $\frac{x^2}{2} \cdot \frac{m_s^2}{m_c}$. The optimal solution does not incur costs in the third phase.

Since the point Z is unknown to the online algorithm, both servers of the online algorithm have a distance of at least $\frac{1}{8}xm_s$ to Z after the first phase with a probability of at least $1/2$. From here on, we assume that this is this the case. After the second phase the distance of an online server to the request is at least $\frac{1}{8}xm_s - x \cdot \frac{m_s^2}{m_c} =: y \cdot m_s$. The costs for serving requests in the third phase is minimized for the online algorithm, if it moves with speed m_s towards Z in each time step. The induced costs are at least $\sum_{i=0}^{y}(y-i) \cdot m_s \geq \frac{y^2}{2}m_s = \frac{1}{2}(\frac{1}{8} - \frac{m_s}{m_c})^2 x^2 m_s \geq \Omega(x^2 m_s)$ if m_c is sufficiently large.

In total, the competitive ratio is $\frac{\Omega(x^2 m_s)}{Dxm_s + x^2/2 \cdot m_s^2/m_c} = \Omega(\frac{m_c}{m_s})$ for sufficiently large x.

Now consider the case of $k > 2$ servers. We use a similar construction as in the proof of Theorem 2.4.1, but now split 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 that 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 $\mathcal{O}(Dk^2 x \cdot m_s)$.

In the second phase, on each point $Z_1, \ldots, Z_{k-1}$, $x/4$ requests appear in order of distance to the starting point, with requests in between when it moves over the line. The latter type of requests induce costs for the adversary of $\mathcal{O}(k\frac{x^2 m_s^2}{m_c})$ if $\frac{m_c}{m_s}$ is sufficiently large (same argument as above). The costs of the online algorithm can be bounded as in the previous theorem, with the additional argument that while the request moves past the first potential choice for a Z_i, any server covering this segment does not get to the second potential candidate in time. With this, the costs for the online algorithm are still $\Omega(kx^2 m_s)$.

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

2.4.2 An Algorithm for the Unweighted Problem

In this section we consider the unweighted problem ($D = 1$). Our algorithm does the following: We mainly attempt to imitate a simulated k-Server algorithm, but always move the closest server greedily towards the request.

We use the following notation in this section: Denote by $a_1, \dots, a_k$ the servers of the online algorithm, $c_1, \dots, c_k$ the servers of the simulated k-Server algorithm and $o_1, \dots, o_k$ 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. 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.

Algorithm 2 — Unweighted-Mobile Servers (UMS). Take any k-Server algorithm $\mathscr{K}$ with bounded competitiveness in the Euclidean space. Upon receiving the next request v', simulate the next step of $\mathscr{K}$. Calculate a minimum weight matching (with the distances as weights) between the servers $a_1, \dots, a_k$ of the online algorithm and the servers $c_1', \dots, c_k'$ of $\mathscr{K}$. There must be a server c_i for which $c_i' = v'$. If the server matched to c_i' can reach v' in this turn, move all servers by a distance $(1+\delta)m_s$ towards their counterparts in the matching. Otherwise, select the server $\tilde{a}$ that is closest to v' and move it to v' a distance of at most $(1+\frac{\delta}{2})m_s$. All other servers move the maximum distance $(1+\delta)m_s$ towards their counterparts in the matching.

We briefly want to discuss the fact that both steps of our algorithm are necessary for a competitiveness independent of T. 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 that just tries to imitate any k-Server algorithm as best as possible is also not successful. 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.

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

Theorem 2.4.3 For $k \geq 2$, there are k-Server algorithms with constant competitiveness such that the corresponding simple algorithm for the k-Mobile Server problem does not achieve a competitive ratio independent of T.

Proof. Consider the following instance: All servers and the request start at the same point on the real line. The request 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 upper bound for the optimal solution can be obtained by just following the request around with a single server which induces the 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 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 requires $x - 3y$ more time steps for it to 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 now has 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 $\Omega(\sqrt{T})$. ■

Note that the bound above makes use of a hypothetical k-Server algorithm which does not always move the closest server to the request onto the request. If the simulated k-Server algorithm does always move the closest server onto the request, e.g., such as the well-known Double-Coverage algorithm [27], then the bound does not apply.

The remainder of this section is devoted to the analysis of the competitive ratio of the UMS algorithm. 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 ($m_c \geq (1+\delta)m_s$), 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.

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. Afterwards, 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 2.4.4 If $m_c \leq (1-\varepsilon)\cdot m_s$ for some $\varepsilon \in (0,1)$, the algorithm UMS is $2/\varepsilon \cdot c(\mathscr{K})$-competitive, where $c(\mathscr{K})$ is the competitive ratio of the simulated k-Server algorithm $\mathscr{K}$.

Proof. We assume the servers adapt their ordering $a_1,\ldots,a_k$ according to the minimum matching in each time step. On the basis of 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 v in each time step, and hence only pays for the movement of its servers, i.e., $C_{Alg} = \sum_{i=1}^k d(a_i,a_i')$. For the simulated algorithm we assume, that c_1 is on the request after the current time step, i.e., $c_1' = v'$.

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

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

Now assume that a_1 cannot reach v' 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 that is at range at most m_c to v' and does the greedy move possibly away from c_2' onto v'. It holds $d(a_2',c_2') - d(a_2,c_2') \leq m_c$. In total, we get

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

We have $C_{Alg} + \Delta\psi \leq \frac{2}{\varepsilon}\cdot C_{\mathscr{K}}$ in both cases from which the competitive ratio follows. ■

We can extend the above 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 2.4.5 If $m_c \leq (1+\delta-\varepsilon)\cdot m_s$ for some $\varepsilon \in (0,1)$, UMS is $\frac{2\cdot(1+\delta)}{\varepsilon}\cdot c(\mathscr{K})$-competitive, where $c(\mathscr{K})$ is the competitive ratio of the simulated k-Server algorithm $\mathscr{K}$.

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 δ if ε stays the same. The reason is that by fixing ε 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 δ, the value of ε increases by the same amount and hence the competitive ratio tends towards $2\cdot c(\mathscr{K})$.

Slow Resource Movement

We now consider the case $m_c \geq (1+\delta)m_s$. The analysis 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. This ensures that in a case where all online servers are far away from the request, moving towards the simulated servers also reduces the distance to the optimal solution if the optimal servers are close to 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 obeys 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.

The k-Server Projection

Our goal is to transform a k-Server algorithm $\mathscr{K}$ into a k-Server algorithm $\hat{\mathscr{K}}$, which serves the requests of a k-Mobile Server instance such that all servers keep relatively close to the current request v. 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 $\mathscr{K}$ are denoted as $c_1,\dots,c_k$ and the servers of $\hat{\mathscr{K}}$ as $\hat{c}_1,\dots,\hat{c}_k$.

We define two circles around v:

- The inner circle $\mathit{inner}(v)$ consists of all points p with $d(p,v) \leq 4k\cdot m_c$.
- The outer circle $\mathit{outer}(v)$ consists of all points p with $d(p,v) \leq (8k+1)\cdot m_c$.

We will maintain $\hat{c}_i \in \mathit{outer}(v)$ for all $i = 1\dots k$ during the entire execution. The time is divided into phases: A phase starting at time t with the request at position v_t ends at the minimum time $t' > t$ with the request at $v_{t'}$, when $d(v_t, v_{t'}) \geq 4k\cdot m_c$. During a phase the simulated servers move to preserve the following:

- If $c_i \in \mathit{inner}(v)$, then $\hat{c}_i = c_i$.

At the end of the phase the servers move such that the following holds:

- If $c_i \in \mathit{inner}(v)$, then $\hat{c}_i = c_i$.
- If $c_i \notin \mathit{inner}(v)$, then $\hat{c}_i$ is on the boundary of $\mathit{inner}(v)$ such that $d(c_i,\hat{c}_i)$ is minimized.

Proposition 2.4.6 For the servers $\hat{c}_1, \dots, \hat{c}_k$ of $\hat{\mathscr{K}}$ it holds $d(\hat{c}_i, v) \leq (8k+1) \cdot m_c$ during the whole execution. The costs of $\hat{\mathscr{K}}$ are at most $\mathscr{O}(k)$ times the costs of $\mathscr{K}$.

Proof. Take a phase between time steps t_1 and t_2. At the beginning, $\hat{c}_i \in inner(v_t)$ per definition. Now, if $\hat{c}_i$ is never moved for time steps $t_1 \leq t \leq t_2$ in the phase we have $d(\hat{c}_i, v_t) \leq d(\hat{c}_i, v_{t_1}) + d(v_{t_1}, v_t) \leq 4k \cdot m_c + (4k+1) \cdot m_c = (8k+1) \cdot m_c$ and hence $\hat{c}_i \in outer(v_t)$. If $\hat{c}_i$ is moved into $inner(v)$ during the phase, the same argument applies between each of the movements of $\hat{c}_i$.

For the estimation of the cost, we define the following potential function: $\phi = \sum_{i=1}^{k} d(c_i, \hat{c}_i)$. During a phase, the potential decreases every time $\hat{c}_i$ moves to c_i by the same distance $\hat{c}_i$ moves. Each time c_i moves, ϕ increases by at most the distance that c_i moves.

We show that during each phase, $\mathscr{K}$ moves its servers by a total distance of at least $k \cdot m_c$. Consider the movement of the request from its starting point v_{t_1} to the final point v_{t_2}. We know that $(4k+1) \cdot m_c \geq d(v_{t_1}, v_{t_2}) \geq 4k \cdot m_c$. Imagine drawing a straight line between v_{t_1} and v_{t_2} and separating it into segments of length m_c by hyperplanes orthogonal to the line. There are now at least $4k$ such segments. Since the maximum movement distance of v is m_c, there is at least one request per segment.

We consider the configuration of $\mathscr{K}$ at the beginning of the phase. Every server of $\mathscr{K}$ has two segments adjacent to its own. We call the segments that do not contain a server of $\mathscr{K}$ and are not adjacent to a segment containing such a server *unoccupied segments*. Since there are $4k$ segments in total and k servers of $\mathscr{K}$, there are at least k unoccupied segments. For any unoccupied segment it holds that a server of $\mathscr{K}$ has to move at least m_c to answer a request in the segment since it needs to cross the entire neighboring segment. Thus, for at least k segments, the servers of $\mathscr{K}$ incur costs of at least m_c, implying a total movement cost of at least $k \cdot m_c$.

We can now bound the costs at the end of a phase: The argument when $c_i \in inner(v)$ is the same as before. Otherwise, ϕ increases by at most $d(\hat{c}_i, \hat{c}_i') \leq (4k+1) \cdot m_c$. Summing up over all i, this yields $C_{\hat{\mathscr{K}}} \leq \mathscr{O}(k) \cdot C_{\mathscr{K}}$. ■

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 $d_t(o_i, p) \leq \frac{\delta}{48} \cdot d_t(o_i, o_i^a)$.

Abusing notation, we also refer to $inner_t(o_i)$ and $outer_t(o_i)$ as distances equal to the radius defined above. The next part of the analysis is devoted to proving the following:

Proposition 2.4.7 There exists a virtual server $\hat{o}$ that moves at a speed of at most $(2 + \frac{1020k}{\delta}) \cdot m_c$ per time step, for which $d(\hat{a}, \hat{o}) \leq 2 \cdot d(o^*, o^{*a}) + d(a^*, v)$ 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 $v \in inner(o^*)$ at the end of the current time step, $\hat{o}$ moves at a maximum speed of $(1 + \frac{\delta}{8})m_s$: i.e., $v_t \in inner_t(o^*) \Rightarrow d(\hat{o}^{(t-1)}, \hat{o}^{(t)}) \leq (1 + \frac{\delta}{8})m_s$.
2. If $v \in inner(o^*)$ at the end of the current time step, then $\hat{o} \in outer(o^*)$ at the end of the current time step: i.e., $v_t \in inner_t(o^*) \Rightarrow \hat{o}^{(t)} \in outer_t(o^*)$.

In the following, we show that it is possible to define a movement pattern for $\hat{o}$ in such a way, that conditions 1 and 2 of Proposition 2.4.7 hold as long as $d(o^*, o^{*a}) \geq 51483 \frac{km_c}{\delta^2}$. Once the distance $d(o^*, o^{*a})$ drops below $51483 \frac{km_c}{\delta^2}$, $\hat{o}$ will simply follow v 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 $v \in inner_{t_1}(o_i)$ at time step t_1 and $o_j = o^*$ and $v \in inner_{t_2}(o_j)$ at time step t_2. In between these two time steps, $v \notin inner(o^*)$. For such a transition, we define the transition time as $t^* := t_2 - t_1$. If $t^* > \frac{inner_{t_1}(o^*)}{m_c} + 2$, 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 Figure 2.5.

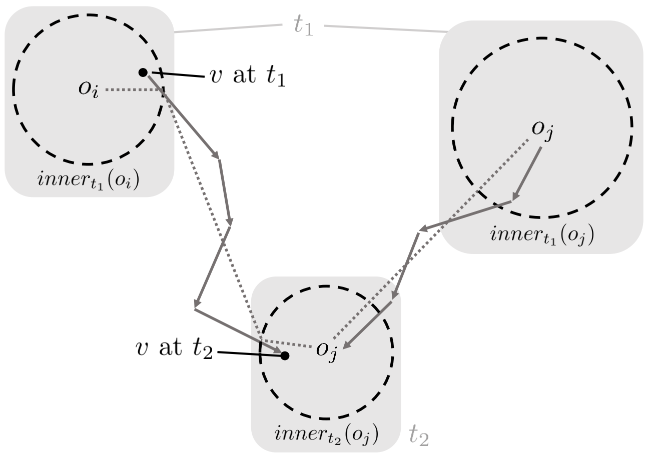

Figure 2.5: Example for a transition from o_i to o_j. By definition, v crosses the border of $inner(o_i)$ after time step t_1 (o_i passes v after t_1). The transition stops at step t_2 when v has entered $inner_{t_2}(o_j)$ (o_j receives v in t_2). Note that o_j's position and the radius of its inner circle may change from t_1 to t_2. The distance moved by v 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 2.4.9.

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 v_t whenever $v_t \notin inner_t(o^*)$. In the last two steps $t_2 - 1$ and t_2, move such that $\hat{o}^{(t_2-1)} = v_{t_2}$ at time $t_2 - 1$ and do not move in t_2 at all. Informally, $\hat{o}$ moves one step ahead of v such that $\hat{o} = v$ after the transition, as soon as $v \in inner(o^*)$.

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:

   (a) A long transition from a server o_ℓ 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 a speed of at most $(1 + \frac{\delta}{8})m_s$ until t_2.

   (b) A short transition from a server o_ℓ to o_j between time t_2 and t_3 occurs, where at one point prior in the sequence $d(o_j, o^*) > \frac{outer(o^*)}{3}$. 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_ℓ. Again, the position of $\hat{o}$ on the circle around o_ℓ should be the one closest to o_j's final position $o_\ell^{(t_3)}$. 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 v with speed $(2 + \frac{1020k}{\delta}) \cdot m_c$ and skip one step ahead of v 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 2.4.7 hold inductively over all transitions. We divide the entire timeline into sequences where each sequence starts with both v 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 2.4.8 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 $t^* := t_2 - t_1$, v moves a distance of at most $t^* \cdot m_c$. At the beginning, $\hat{o} \in outer_{t_1}(o^*)$ and $v \in inner_{t_1}(o^*)$, hence

$$d_{t_1}(\hat{o}, v) \leq d_{t_1}(\hat{o}, o^*) + d_{t_1}(o^*, v) \leq inner_{t_1}(o^*) + outer_{t_1}(o^*).$$

During the first $\lceil inner_{t_1}(o^*)/m_c \rceil$ time steps, $\hat{o}$ can catch up to v a distance of

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

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

Our next goal is to analyze a sequence of short transitions. During these transitions, v 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 2.4.9 Every short transition between o_i in step t_1 and o_j in step t_2 can increase the distance of some server s, which moves at a speed of at most $(1+\delta)m_s$, to o^* by at most $\min\{6.001 \frac{\delta^2}{48960k} \cdot d_{t_1}(o^*, o^{*a}) + 8.001 m_c\ ,\ 6.002 \cdot \frac{\delta^2}{48960k} \cdot d_{t_2}(o^*, o^{*a}) + 8.002 m_c\}$.
Likewise, s decreases its distance to o^* by at most
$\min\{6.001 \frac{\delta^2}{48960k} \cdot d_{t_1}(o^*, o^{*a}) + 8.001 m_c\ ,\ 6.002 \cdot \frac{\delta^2}{48960k} \cdot d_{t_2}(o^*, o^{*a}) + 8.002 m_c\}$.

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, $t^* = t_2 - t_1 \leq \frac{inner_{t_1}(o_i)}{m_c} + 2$.

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 v 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 Figure 2.5):

$$\begin{aligned} d_{t_1}(o_i, o_j) &\leq d(o_i^{(t_1)}, o_j^{(t_2)}) + d(o_j^{(t_2)}, o_j^{(t_1)}) \\ &\leq inner_{t_1}(o_i) + t^* \cdot m_c + inner_{t_2}(o_j) + t^* \cdot m_s \\ &\leq inner_{t_1}(o_i) + t^* \cdot m_c + inner_{t_1}(o_j) + t^* \cdot \frac{\delta}{16} m_s + t^* \cdot m_s. \end{aligned}$$

With this knowledge, we get

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

In reverse, we can bound

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

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

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

To derive the first bound, we get

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

For the second bound, we continue with

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

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

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

This gives us

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

For the bound of decreasing the distance, the same proof can be applied: Start with

$$d_{t_2}(s,o_j) \geq d_{t_1}(s,o_j) - t^* \cdot (2+\delta)m_s \geq d_{t_1}(s,o_i) - d_{t_1}(o_i,o_j) - t^* \cdot (2+\delta)m_s$$

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 $d(o_\ell, o^*) \leq \frac{outer(o^*)}{3}$ per definition and hence following it keeps $\hat{o}$ inside $outer(o^*)$.

Lemma 2.4.10 Consider a sequence of short transitions that 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_ℓ to o_j between time steps t_2 and t_3. $\hat{o}$ selects the server o_ℓ which passes v on to o_j over the long transition and follows it. Since $d(o_\ell, o^*) \leq \frac{outer(o^*)}{3}$ 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

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

we get

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

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

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

and therefore $\hat{o} \in outer_t(o^*)$ during the whole sequence. By Lemma 2.4.8, we have $\hat{o} \in inner(o^*)$ after the long transition. ■

For the second case of step 2, we show with the help of Lemma 2.4.9 that during the sequence of transitions, $\hat{o}$ does not lose too much distance to o^*. Furthermore, the server o_j, since at one point $d(o_j, o^*) > \frac{outer(o^*)}{3}$, 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 2.4.11 Consider a sequence of short transitions that is terminated by a short transition from o_ℓ to o_j, where at one point prior in the sequence $d(o_j, o^*) > \frac{outer(o^*)}{3}$. 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_ℓ 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 $d(\hat{o}, o_\ell) \leq \frac{2\delta}{145} \cdot d(o_\ell, o_\ell^a)$:

$$d_{t_1}(\hat{o}, o_\ell) \leq d_{t_1}(\hat{o}, o^*) + d_{t_1}(o^*, o_\ell) \leq \left(\frac{\delta^2}{48960k} + \frac{\delta}{144}\right) \cdot d_{t_1}(o^*, o^{*a}).$$

With

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

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_ℓ 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_ℓ 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 $3m_s \cdot \frac{2\delta}{145} < \frac{\delta}{20} m_s$, 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 2.4.9 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 that constitutes an upper bound on the distance $\hat{o}$ has to traverse, using the following definition:

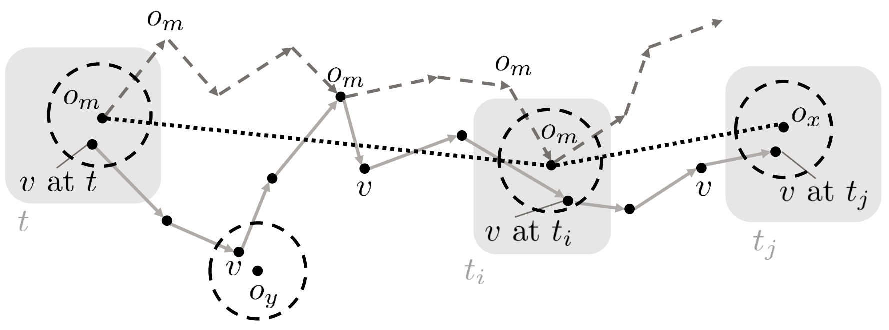

Figure 2.6: The construction of a transition path. The transition path is marked by black dots, while the movement of o_m is depicted by dashed arrows. The movement of v is marked by the gray arrows. Starting at the position of o_m at t, the last time step t_i is identified at which $o_m = o^*$ and $v \in inner(o_m)$. Note, that the role of o^* might change multiple times (e.g. to o_y in the picture) between t and t_i. Afterwards, the path goes to the destination o_x of a short transition.

Definition 2.4.12 — 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 $v \in inner(o_m)$. 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*.

Figure 2.6 illustrates one of the recursion steps of the definition for the transition path.

The distance traveled by $\hat{o}$ during time steps t_1 to t_3 is bounded by the distance of $\hat{o}$ to o^* at time t_1 plus the length of the (t_1,t_3)-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: 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 2.4.9 we may upper bound the length of an edge by $6.001 \cdot \frac{\delta^2}{48960k} \cdot d_{t'}(o^*,o^{*a}) + 8.001m_c$, 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 $t'_1,\dots,t'_k$ be the points in time where the short transitions inducing the second type edges begin. We can upper bound their total length as

$$\sum_{i=1}^{k}(6.001 \cdot \frac{\delta^2}{48960k} \cdot d_{t'_i}(o^*,o^{*a}) + 8.001m_c).$$

Assuming the highest possible distance for each of the $d_{t'_i}(o^*,o^{*a})$, we get for the first transition the total distance of the movement during the sequence added to the original length, which is $d_{t_1}(o^*,o^{*a}) + \hat{t} \cdot (2+\delta)m_s$. 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 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

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

The total path length (including the distance from $\hat{o}$ to o^* in t_1) may hence be 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 a time step t where $d_t(o_j,o^*) > \frac{\delta}{144}\cdot d_t(o^*,o^{*a})$. 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 $v \in inner_{t_2}(o_\ell)$, $v \in inner_{t_3}(o_j)$ and $t^* := t_3 - t_2 \le \frac{inner_{t_2}(o_\ell)}{m_c} + 2$. We have $d_{t_2}(o_j,o_\ell) \le t^*\cdot(m_c + m_s + \frac{\delta}{16}m_s) + inner_{t_2}(o_\ell) + inner_{t_2}(o_j)$ (see Figure 2.5 and the proof of Lemma 2.4.9).

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

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

Comparing the distances at t_1 and t_2, we conclude that

$$d_{t_1}(o_j,o^*) - d_{t_2}(o_j,o^*) \ge \frac{\delta}{144}\cdot d_{t_1}(o^*,o^{*a}) - 4.001\cdot\frac{\delta^2}{48960k}\cdot d_{t_2}(o^*,o^{*a}) - 4.001m_c.$$

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 $t'_1,\dots,t'_k$, we get the upper bound similar to before:

$$\begin{array}{rcl} d_{t_2}(o^*,o^{*a}) &\le& d_{t_1}(o^*,o^{*a}) + \hat{t}\cdot(2+\delta)m_s + \sum_{i=1}^{k}(6.001\cdot\frac{\delta^2}{48960k}\cdot d_{t'_i}(o^*,o^{*a}) + 8.001m_c) \\ &\le& d_{t_1}(o^*,o^{*a}) + \hat{t}\cdot(2+\delta)m_s + 0.0002\delta^2\cdot(d_{t_1}(o^*,o^{*a}) + \hat{t}\cdot(2+\delta)m_s) \\ && +8.002km_c. \end{array}$$

Continuing from above, we have

$$\begin{array}{rcl} d_{t_1}(o_j,o^*) - d_{t_2}(o_j,o^*) &\ge& \frac{\delta}{144}\cdot d_{t_1}(o^*,o^{*a}) - 4.001\cdot\frac{\delta^2}{48960k}\cdot d_{t_2}(o^*,o^{*a}) - 4.001m_c \\ &\ge& \frac{\delta}{144}\cdot d_{t_1}(o^*,o^{*a}) - \frac{4.001\delta^2}{48960k}\cdot(1.0002\cdot(d_{t_1}(o^*,o^{*a}) + \hat{t}\cdot(2+\delta)m_s) \\ && +8.002km_c) - 4.001m_c. \end{array}$$

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 2.4.9 gives us that the distance of o_j to any server decreases by at most $6.001\cdot\frac{\delta^2}{48960k}\cdot d_t(o^*,o^{*a})+8.001m_c$. This decrease is maximized the same as above, i.e., $0.0002\delta^2\cdot(d_{t_1}(o^*,o^{*a})+\hat{t}\cdot 2m_s)+8.002km_c$.

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

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

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 $\hat{o}$:

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

■

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 $\hat{o}=v$.

Lemma 2.4.13 During the execution of step 1 or 2 of the algorithm, $d(\hat{a},\hat{o})\leq 2\cdot d(o^*,o^{*a})+d(a^*,v)$.

Proof. We argue that $\hat{o}\in outer(o^*)$ or $\hat{o}=v$. 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, $v\in inner_{t_1}(o^*)$ and $\hat{o}\in outer_{t_1}(o^*)$. We have already shown that for $t^*\geq\frac{inner_{t_1}(o^*)}{m_c}$, $\hat{o}$ catches up to v within the time t^* in the proof of Lemma 2.4.8. Expressed in distance, $\hat{o}$ catches up to v when v is a distance of $inner_{t_1}(o^*)$ outside the inner circle of o^*. We show that at this time, v is still in $outer(o^*)$.

Let $\hat{t} = \lceil inner_{t_1}(o^*)/m_c \rceil$. We have

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

With

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

we get $d_{t_1+\hat{t}}(v,o^*) \leq \frac{6\delta^2}{48960k} \cdot d_{t_1+\hat{t}}(o^*,o^{*a}) + 3m_c \leq \frac{\delta}{48} \cdot d_{t_1+\hat{t}}(o^*,o^{*a})$ 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 $v = \hat{o}$.

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 $\hat{o} = v$, then $\hat{a} = a^*$ and therefore $d(\hat{a},\hat{o}) = d(a^*,v)$. ■

So far we have shown that all claims of Proposition 2.4.7 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 2.4.14 After the execution of step 3 it holds $\hat{o} = v$.
Furthermore, $d(\hat{a},\hat{o}) \leq 2 \cdot d(o^*,o^{*a}) + d(a^*,v)$ 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., $t_1 < t_2$ where t_1 is chosen maximal and t_2 is 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, $t_2 - t_1 \geq 25741 \cdot \frac{1}{\delta^2}$.

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

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

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 obtained from Proposition 2.4.6. We use a potential composed of two major parts which balance the main ideas of our algorithm against each other: ϕ will measure the costs of the greedy strategy, while ψ will cover the matching to the simulated k-Server algorithm.

Let $\hat{o}$ be an offline server that fulfills the properties stated in Proposition 2.4.7. 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

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

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

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 ϕ 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 version of Lemma 2.3.6 which can be derived by substituting δ with $\delta/2$.

Lemma 2.4.15 Let s be some server with $d(s', v') \leq \frac{\sqrt{\delta}}{4} \cdot d(a_i', v')$ and a_i moves towards v' 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 2.4.4.

Lemma 2.4.16 $\Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} \cdot C_{\mathscr{K}} - \sum_{i=1}^{k} d(a_i, a_i')$.

Proof. Assume that $\tilde{a} = a_1$. 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 $\mathscr{K}$ serves the request with c_1, i.e., $c_1' = v'$, 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, $\mathscr{K}$ serves the request with another server (assume c_2). Since a_2 was not chosen as $\tilde{a}$, it moves the full distance of $(1+\delta)m_s$ and hence

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

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

Lemma 2.4.17 If $d(a^{*'},v')>0$, then $\Delta\psi \le Y\cdot\frac{m_c}{\delta m_s}C_{\mathscr{K}} - \sum_{i=1}^{k} d(a_i,a_i') - \frac{Y-4}{2}m_c$.

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

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

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

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

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

Now consider the case that $v'\notin inner(o^{*'})$. We have $d(a^{*'},v')\le d(o^{*a'},v')\le d(o^{*'},o^{*a'})+d(o^{*'},v')\le(\frac{48960k}{\delta^2}+1)\cdot d(o^{*'},v')$. The movement costs are canceled by $\Delta\psi$ as in Lemma 2.4.16. It only remains to bound the possible increase of ϕ. We use $d(\hat{a}',\hat{o}')-d(\hat{a},\hat{o})\le(3+\frac{1020k}{\delta})\cdot m_c$.

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

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

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

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

In all of the above, the competitive ratio is bounded by $\mathscr{O}(\frac{k^2}{\delta^3})\cdot\frac{m_c}{\delta m_s}+Y\cdot\frac{m_c}{\delta m_s}\cdot c(\mathscr{K})$.

Finally, we consider the case $v'\in inner(o^{*'})$. Whenever $d(a^*,v')>102970\frac{km_c}{\delta^2}$, we use Lemma 2.4.15 to obtain the following:

Lemma 2.4.18 If $d(a^*,v')>102970\frac{km_c}{\delta^2}$ and $v'\in inner(o^{*'})$, then $d(a_i',\hat{o}')-d(a_i,\hat{o})\le-\frac{\delta}{8}m_s$.

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

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

Hence

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

and with Lemma 2.4.15, we get $d(a_i', \hat{o}') - d(a_i, \hat{o}') \le -\frac{1+\frac{1}{4}\delta}{1+\frac{1}{2}\delta} d(\hat{a}, \hat{a}')$.

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

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

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

From Proposition 2.4.7 we get $d(\hat{o}, \hat{o}') \le (1 + \frac{\delta}{8}) m_s$ and therefore

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

■

With this lemma, ϕ can be used to cancel the costs of the algorithm in case of a high distance to the request.

Lemma 2.4.19 If $v' \in inner(o^{*'})$, then $C_{Alg} + \Delta\phi + \Delta\psi \le Y \cdot \frac{m_c}{\delta m_s} \cdot C_{\mathcal{K}} + 2 \cdot d(o^{*'}, v')$.

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

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

to get $C_{Alg} + \Delta\phi \le 6 \cdot d(\hat{a}', \hat{o}') + \sum_{i=1}^{k} d(a_i, a_i')$. Furthermore, $\Delta\psi \le Y \cdot \frac{m_c}{\delta m_s} C_{\mathcal{K}} - \sum\limits_{i=1}^{k} d(a_i, a_i') - \frac{Y-4}{2} m_c$ due to Lemma 2.4.17. In total, $C_{Alg} + \Delta\phi + \Delta\psi \le Y \cdot \frac{m_c}{\delta m_s} \cdot C_{\mathcal{K}}$ with $Y \ge \Omega(\frac{k}{\delta^2})$.

2. $107548 \cdot \frac{km_c}{\delta^2} < d(\hat{a}', \hat{o}')$: We show that the condition of Lemma 2.4.18 applies:

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

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

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

and $\Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} \cdot C_{\mathscr{K}} - \sum_{i=1}^k d(a_i,a_i')$ due to Lemma 2.4.16. In total, we get $C_{Alg} + \Delta\phi + \Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} \cdot C_{\mathscr{K}} + 2 \cdot d(o^{*'},v')$.

■

The resulting competitive ratio of $Y \cdot \frac{m_c}{\delta m_s} \cdot c(\mathscr{K}) + 2$ is less than the $\mathscr{O}(\frac{k^2}{\delta^3}) \cdot \frac{m_c}{\delta m_s} + Y \cdot \frac{m_c}{\delta m_s} \cdot c(\mathscr{K})$ 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 2.4.20 If $m_c \geq (1+\delta)m_s$, the algorithm UMS is $\mathscr{O}(\frac{1}{\delta^4} \cdot k^2 \cdot \frac{m_c}{m_s} + \frac{1}{\delta^3} \cdot k^2 \cdot \frac{m_c}{m_s} \cdot c(\mathscr{K}))$-competitive, where $c(\mathscr{K})$ is the competitive ratio of the simulated k-Server algorithm $\mathscr{K}$.

2.4.3 Extension to the Weighted Problem

In this section we extend the previous algorithm and analysis to the case 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 cost increases 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 is not forced to always have one page at the point of the request. In case of small distances to v, 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.

Algorithm 3 — Weighted-Mobile Servers (WMS). Take any k-Page Migration algorithm $\mathscr{K}$. Upon receiving the next request v', simulate the next step of $\mathscr{K}$. Calculate a minimum weight matching (with the distances as weights) between the servers $a_1,\dots,a_k$ of the online algorithm and the pages $c_1',\dots,c_k'$ of $\mathscr{K}$. Select the closest server $\tilde{a}$ to v' and move it to v' at most a distance $\min\{m_c, \frac{1}{D}(1-\varepsilon) \cdot d(\tilde{a},v')\}$ in case $m_c \leq (1+\delta-\varepsilon)m_s$ and at most $\min\{(1+\frac{\delta}{2})m_s, \frac{1}{D}(1-\frac{\delta}{2}) \cdot d(\tilde{a},v')\}$ 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 $\min\{(1+\delta)m_s, \frac{1}{D} \cdot d(\tilde{a},v')\}$. 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 2.4.2.

Fast Resource Movement

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 2.4.21 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 algorithm $\mathcal{K}$.

Proof. We assume the servers adapt their ordering $a_1,\dots,a_k$ according to the minimum matching in each time step. On the basis of 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 $d(a^*,v) \leq \frac{D}{1-\varepsilon}\cdot m_c \leq 2Dm_c$. 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 v'. Then

$$\Delta\psi \leq \sqrt{2}\cdot\frac{4D}{\varepsilon}\sum_{i=1}^{k} d(c_i,c_i') - \sqrt{2}\cdot\frac{4D}{\varepsilon}\sum_{i=1}^{k} d(a_i,a_i')$$

and

$$C_{Alg} = \sum_{i=1}^{k} d(a_i,a_i') + d(a^{*'},v') \leq \sum_{i=1}^{k} d(a_i,a_i') + 2Dm_c.$$

Consider the server that is matched to $c^{*'}$: Either it reaches $c^{*'}$ or it moves a distance of m_s. In the first case $d(a^{*'},v') \leq d(c^{*'},v')$ which gives a competitive ratio of $\sqrt{2}\cdot\frac{4}{\varepsilon}\cdot c(\mathcal{K})$ immediately. In the latter case, there is a server a_j such that $d(a_j,a_j') = m_s$ and hence

$$\Delta\psi \leq \sqrt{2}\cdot\frac{4D}{\varepsilon}\sum_{i=1}^{k} d(c_i,c_i') - \sqrt{2}\cdot\frac{D}{\varepsilon}\sum_{i=1}^{k} d(a_i,a_i') - \sqrt{2}\cdot\frac{3D}{\varepsilon}m_s,$$

which implies a competitive ratio of at most $\sqrt{2}\cdot\frac{4}{\varepsilon}\cdot c(\mathcal{K})$ as well.

Now assume $\tilde{a} = a_1$ moves towards v' and hence $a^{*'} = a_1'$. We have

$$d(a_1',c_1') - d(a_1,c_1') \leq \min\{m_c, \frac{1}{D}(1-\varepsilon)\cdot d(\tilde{a},v')\}.$$

In all of the following cases, we make use of

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

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

1. $d(a^{*'},v') \leq d(c^{*'},v')$:
   Since we assume $D \geq 2$, we have

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

   It follows

$$C_{Alg} \leq 3\cdot d(c^{*'},v') + D\cdot\sum_{i=2}^{k} d(a_i,a_i')$$

   and

$$\Delta\psi \leq \sqrt{2}\cdot\frac{4D}{\varepsilon}\sum_{i=1}^{k} d(c_i,c_i') + \sqrt{2}\cdot\frac{8}{\varepsilon}\cdot d(c^{*'},v') - D\cdot\sum_{i=2}^{k} d(a_i,a_i').$$

2. $d(a^{*'},v') > d(c^{*'},v')$ and $c^{*'} = c'_1$:
   We know that
   $$d(a'_1,c'_1) - d(a_1,c'_1) \le -\frac{1}{\sqrt{2}} \cdot d(a_1,a'_1) = -\frac{1}{\sqrt{2}} \cdot \min\{m_c, \frac{1}{D}(1-\varepsilon) \cdot d(\tilde{a},v')\}$$
   and hence
   $$\Delta\psi \le \sqrt{2} \cdot \frac{4D}{\varepsilon} \sum_{i=1}^{k} d(c_i,c'_i) - \frac{4D}{\varepsilon} \cdot \min\{m_c, \frac{1}{D}(1-\varepsilon) \cdot d(\tilde{a},v')\} - D \cdot \sum_{i=2}^{k} d(a_i,a'_i).$$
   If $d(a_1,a'_1) = m_c$ then $C_{Alg} \le 3Dm_c + D \cdot \sum_{i=2}^{k} d(a_i,a'_i)$, otherwise $C_{Alg} \le 2 \cdot d(\tilde{a},v') + D \cdot \sum_{i=2}^{k} d(a_i,a'_i)$.

3. $d(a^{*'},v') > d(c^{*'},v')$ and $c^{*'} \ne c'_1$:
   We assume $c^{*'} = c'_2$. It must hold $a'_2 \ne c'_2$ and hence
   $$d(c'_2,a'_2) - d(c'_2,a_2) \le -\min\{m_s, \frac{1}{D} \cdot d(\tilde{a},v')\}.$$
   In the case $d(a_2,a'_2) = \frac{1}{D} \cdot d(\tilde{a},v')$,
   $$d(a'_1,c'_1) - d(a_1,c'_1) + d(a'_2,c'_2) - d(a_2,c'_2) \le -\frac{\varepsilon}{D} \cdot d(\tilde{a},v').$$
   This gives us
   $$\Delta\psi \le \sqrt{2} \cdot \frac{4D}{\varepsilon} (\sum_{i=1}^{k} d(c_i,c'_i) - \frac{\varepsilon}{D} \cdot d(\tilde{a},v') - \sum_{i=3}^{k} d(a_i,a'_i)).$$
   With
   $$C_{Alg} = d(a'_1,v') + D \cdot \sum_{i=1}^{k} d(a_i,a'_i) \le 3 \cdot d(\tilde{a},v') + D \cdot \sum_{i=3}^{k} d(a_i,a'_i)$$
   the bound follows.
   In case $d(a_2,a'_2) = m_s$,
   $$d(a'_1,c'_1) - d(a_1,c'_1) + d(a'_2,c'_2) - d(a_2,c'_2) \le m_c - m_s \le -\varepsilon m_s.$$
   Similar as before,
   $$\Delta\psi \le \sqrt{2} \cdot \frac{4D}{\varepsilon} (\sum_{i=1}^{k} d(c_i,c'_i) - \varepsilon m_s - \sum_{i=3}^{k} d(a_i,a'_i))$$
   and
   $$C_{Alg} \le 4Dm_s + D \cdot \sum_{i=3}^{k} d(a_i,a'_i).$$

■

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 $m_c \le (1+\delta-\varepsilon) \cdot m_s$, then we get the following result:

Corollary 2.4.22 The algorithm is $\frac{\sqrt{2}\cdot 11\cdot(1+\delta)}{\varepsilon}\cdot c(\mathscr{K})$-competitive, where $c(\mathscr{K})$ is the competitive ratio of the k-Page Migration algorithm $\mathscr{K}$.

Slow Resource Movement

Similar to the k-Server projection discussed in Section 2.4.2, we obtain the following result which gives us a new k-Page Migration algorithm needed for the case $m_c \geq (1+\delta)m_s$.

Proposition 2.4.23 Let $\mathscr{K}$ be an online algorithm for the k-Page Migration problem. There exists an online algorithm $\hat{\mathscr{K}}$ for the k-Page Migration problem with pages $\hat{c}_1,\dots,\hat{c}_k$ such that it holds $d(\hat{c}_i,v) \leq (32kD+1)\cdot m_c$ during the whole execution. The costs of $\hat{\mathscr{K}}$ are at most $\mathscr{O}(k)$ times the costs of $\mathscr{K}$.

Proof. The servers of $\mathscr{K}$ are denoted as $c_1,\dots,c_k$ and the servers of $\hat{\mathscr{K}}$ as $\hat{c}_1,\dots,\hat{c}_k$.

We define two circles around v:

- The inner circle $inner(v)$ consists of all points p with $d(p,v) \leq 16kD\cdot m_c$.
- The outer circle $outer(v)$ consists of all points p with $d(p,v) \leq (32kD+1)\cdot m_c$.

We will maintain $\hat{c}_i \in outer(v)$ for the entirety of the execution. The time is divided into phases, where each phase ends when there is a distance of $16kD\cdot m_c$ between the current position and the position at the beginning of the phase of v. During a phase the simulated servers move to preserve the following:

- If $c_i \in inner(v)$, then $\hat{c}_i = c_i$.

At the end of the phase the servers move such that the following holds:

- If $c_i \in inner(v)$, then $\hat{c}_i = c_i$.
- If $c_i \notin inner(v)$, then $\hat{c}_i$ is on the boundary of $inner(v)$ such that $d(c_i,\hat{c}_i)$ is minimized.

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 distance $\hat{c}_i$ moves. Each time c_i moves, ϕ increases by at most D times the distance that c_i moves. In case for the closest server of $\mathscr{K}$ to v, which is $c^* = c_i$, to hold $c_i \in inner(v)$, then $\hat{c}_i = c_i$ and hence the serving costs of the algorithms are the same. Otherwise, $c_i \notin inner(v)$, $\hat{c}_i \in outer(v)$ and hence the serving costs differ at most by a factor of 3.

We show that during each phase, $\mathscr{K}$ has costs of at least $\Omega(1)\cdot kD^2\cdot m_c$. Consider the movement of the request from its starting point v to the final point v'. We know that $d(v,v') \geq 16kD\cdot m_c$. Imagine drawing a straight line between v and v' 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 $\mathscr{K}$ has two segments adjacent to its own. We call the segments that do not contain a server of $\mathscr{K}$ and are not adjacent to a segment containing such a server *unoccupied segments*. Since there are $16kD$ segments in total and k servers of $\mathscr{K}$, there are at least $11kD$ 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 $\mathscr{K}$ 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 $\frac{x}{2}-y$ 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 costs of at least $\frac{x}{2}Dm_c - \frac{D^2}{2}m_c$. No matter how the $11kD$ 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 phase: The argument when $c_i \in inner(v)$ is the same as before. Otherwise, ϕ increases by at most $D \cdot d(\hat{c}_i, \hat{c}_i') \leq 32kD^2 \cdot m_c$. This yields $C_{\mathcal{K}'} \leq \mathcal{O}(k) \cdot C_{\mathcal{K}}$. ■

From here on we assume $\mathcal{K}$ to be a k-Page Migration algorithm obtained from the transformation in Proposition 2.4.23. The offline helper and its invariants as stated in Proposition 2.4.7 do not depend on the simulated algorithm and therefore all insights gained from the corresponding section 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 that fulfills the invariants stated in Proposition 2.4.7. The first part of the potential is then defined as

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

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

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 ψ, reusing ideas from the proof of Theorem 2.4.21.

Lemma 2.4.24 $\Delta\psi \leq \mathcal{O}(1) \cdot Y \cdot \frac{m_c}{\delta m_s} \cdot C_{\mathcal{K}} - D \cdot \sum_{i=1}^{k} d(a_i, a_i')$.

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

1. $d(a^{*'}, v') \leq d(c^{*'}, v')$:
   Since we assume $D \geq 2$, we have

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

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

2. $d(a^{*'},v') > d(c^{*'},v')$ 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
   $\Delta\psi \leq \sqrt{2} \cdot \frac{4D}{\varepsilon} \sum_{i=1}^{k} d(c_i,c'_i) - \frac{4D}{\varepsilon} \cdot d(a_1,a'_1) - D \cdot \sum_{i=2}^{k} d(a_i,a'_i)$.

3. $d(a^{*'},v') > d(c^{*'},v')$ and $c^{*'} \neq c'_1$:
   We assume $c^{*'} = c'_2$. It must hold $a'_2 \neq c'_2$ and hence

$$d(c'_2,a'_2) - d(c'_2,a_2) \leq -\min\{m_s, \frac{1}{D} \cdot d(\tilde{a},v')\}.$$

   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

$$\begin{aligned}\Delta\psi &\leq \sqrt{2} \cdot \tfrac{4D}{\varepsilon} (\textstyle\sum_{i=1}^{k} d(c_i,c'_i) - \varepsilon \cdot d(a_2,a'_2) - \sum_{i=3}^{k} d(a_i,a'_i)) \\ &\leq \sqrt{2} \cdot \tfrac{4D}{\varepsilon} \textstyle\sum_{i=1}^{k} d(c_i,c'_i) - D \cdot \sum_{i=1}^{k} d(a_i,a'_i).\end{aligned}$$

■

Lemma 2.4.25 If $d(a^{*'},v') > d(c^{*'},v')$, then
$\Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} C_{\mathcal{K}} - D \cdot \sum\limits_{i=1}^{k} d(a_i,a'_i) - \frac{Y-4}{2} D \frac{m_c}{\delta m_s} \cdot \min\{m_s, \frac{1}{D} \cdot d(\tilde{a},v')\}$.

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 v'. Then

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

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

Now assume $\tilde{a}$ moves towards v'. 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 $d(a_1,a'_1) = \min\{(1+\frac{\delta}{2})m_s, \frac{1}{D}(1-\frac{\delta}{2}) \cdot d(\tilde{a},v')\}$.

Otherwise, we assume $c^{*'} = c'_2$. It must hold $a'_2 \neq c'_2$ and hence $d(c'_2,a'_2) - d(c'_2,a_2) \leq -\min\{m_s, \frac{1}{D} \cdot d(\tilde{a},v')\}$. 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

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

■

Now consider the case that $v' \notin inner(o^{*'})$. We have $d(a^{*'},v') \leq d(o^{*a'},v') \leq d(o^{*'},o^{*a'}) + d(o^{*'},v') \leq (\frac{48960k}{\delta^2} + 1) \cdot d(o^{*'},v')$. The movement costs are canceled by $\Delta\psi$ as in Lemma 2.4.24. It only remains to bound the possible increase of ϕ. We use $d(\hat{a}',\hat{o}') - d(\hat{a},\hat{o}) \leq (3 + \frac{1020k}{\delta}) \cdot m_c$.

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

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

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

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

In all of the above, the competitive ratio is bounded by $\mathscr{O}(\frac{k^2}{\delta^3}) \cdot \frac{m_c}{\delta m_s} + Y \cdot \frac{m_c}{\delta m_s} \cdot c(\mathscr{K})$.

Finally, we consider the case $v' \in inner(o^{*'})$. As in the previous section, whenever $d(a^*,v') > 102970D\frac{km_c}{\delta^2}$, we make use of Lemma 2.4.15 to obtain the following result, which then helps us bound $\Delta\phi$:

Lemma 2.4.26 If $d(a^*,v') > 102970D\frac{km_c}{\delta^2}$ and $v' \in inner(o^{*'})$, then $d(a_i',\hat{o}') - d(a_i,\hat{o}') \leq -\frac{\delta}{8}m_s$.

Proof. By our construction of the simulated k-Server algorithm, we have $d(c_i',v') \leq 33Dkm_c \leq \frac{\delta^2}{2652} \cdot d(a^{*'},v')$. Furthermore,

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

Hence

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

and with Lemma 2.4.15, $d(a_i',\hat{o}') - d(a_i,\hat{o}') \leq -\frac{1+\frac{1}{2}\delta}{1+\delta} d(\hat{a},\hat{a}')$ follows.

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

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

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

From Theorem 2.4.7 we get $d(\hat{o},\hat{o}') \leq (1 + \frac{\delta}{8})m_s$ and therefore

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

■

Lemma 2.4.27 If $v' \in inner(o^{*'})$, then $C_{Alg} + \Delta\phi + \Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} \cdot C_{\mathscr{K}} + 2 \cdot d(o^{*'}, v')$.

Proof. 1. $d(\hat{a}', \hat{o}') \leq 107548D \cdot \frac{km_c}{\delta^2}$: First consider the case $d(a^{*'}, v') \leq d(c^{*'}, v')$. With Lemma 2.4.24 we can bound the movement costs of the algorithm. Furthermore, we use

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

to get

$$\begin{array}{rcl} d(\hat{o}', \hat{a}') & \leq & d(\hat{o}', a^{*'}) \\ & \leq & d(a^{*'}, v') + d(r', o^{*'}) + d(o^{*'}, \hat{o}') \\ & \leq & d(a^{*'}, v') + 2 \cdot \frac{\delta}{48} \cdot d(o^{*'}, o^{*a'}) \\ & \leq & d(a^{*'}, v') + \frac{2\delta}{47} \cdot d(\hat{o}', \hat{a}') \\ \Rightarrow \quad d(\hat{o}', \hat{a}') & \leq & 2 \cdot d(a^{*'}, v'). \end{array}$$

Hence $\Delta\phi \leq 4 \cdot d(\hat{a}', \hat{o}') \leq 8 \cdot d(c^{*'}, v')$. In total, $C_{Alg} + \Delta\phi + \Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} \cdot C_{\mathscr{K}}$ with $Y \geq 9$. Otherwise, Lemma 2.4.25 applies which gives us

$$\Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} C_{\mathscr{K}} - D \cdot \sum_{i=1}^{k} d(a_i, a_i') - \frac{Y-4}{2} D \frac{m_c}{\delta m_s} \cdot \min\{m_s, \frac{1}{D} \cdot d(\tilde{a}, v')\}.$$

We may either use $C_{Alg} + \Delta\phi \leq 9 \cdot d(\tilde{a}, v') + D \cdot \sum_{i=1}^{k} d(a_i, a_i')$ or

$$\begin{array}{rcl} d(a^{*'}, v') & \leq & d(\hat{a}', v') \\ & \leq & d(\hat{a}', \hat{o}') + d(\hat{o}', v') \\ & \leq & d(\hat{a}', \hat{o}') + 2 \cdot \frac{\delta}{48} \cdot d(o^{*'}, o^{*a'}) \\ & \leq & (1 + \frac{2\delta}{47}) \cdot d(\hat{a}', \hat{o}') \end{array}$$

gives us

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

In any case, $C_{Alg} + \Delta\phi + \Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} \cdot C_{\mathscr{K}}$ 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 2.4.26 applies:

$$\begin{array}{rcl} d(\hat{a}', \hat{o}') & \leq & d(a^{*'}, \hat{o}') \\ & \leq & d(a^{*'}, a^*) + d(a^*, v') + d(v', \hat{o}') \\ & \leq & m_c + d(a^*, v') + 2 \cdot \frac{\delta}{48} \cdot d(o^{*'}, o^{*a'}) \\ & \leq & m_c + d(a^*, v') + \frac{2}{47} \cdot d(\hat{a}', \hat{o}') \\ \Leftrightarrow \quad \frac{45}{47} \cdot d(\hat{a}', \hat{o}') - m_c & \leq & d(a^*, v') \\ \Rightarrow \quad 102970D\frac{km_c}{\delta^2} & \leq & d(a^*, v'). \end{array}$$

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

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

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 2.4.25. In total, we get $C_{Alg} + \Delta\phi + \Delta\psi \leq Y \cdot \frac{m_c}{\delta m_s} \cdot C_{\mathcal{K}} + 2 \cdot d(o^{*'},v')$.

■

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 result:

Theorem 2.4.28 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 algorithm $\mathcal{K}$.

3. Online Facility Location with Mobile Facilities

This chapter deals with a variant of the Facility Location problem, which in our general setting highlights the aspect of placing new resources into the system to improve access for the clients, measured by their distance to the nearest facility. This improvement comes at a cost as every new placed resource induces setup costs (or opening costs in the terminology of Facility Location).

In the Metric Facility Location problem, we are given a set of clients and a set of possible facility locations which both are part of the same metric space. For each of the possible facility locations, we are also given an opening cost. The goal is to select a subset of the facility locations such that the sum of the implied opening costs plus the sum of the distances of each client to the closest facility is minimized. This basic model was extended to the Online Facility Location problem [63] in which the clients are not given in advance but arrive over time. In each time step, new clients must be irrevocably assigned to a facility.

Meyerson showed that it is not possible to achieve a competitive ratio for the online version that is independent of the number of clients (and therefore time) [63]. This lower bound was later improved by Fotakis to $\Omega(\frac{\log n}{\log\log n})$, where n is the number of clients [41]. In the same work, it was shown that this bound is tight for both deterministic and randomized algorithms. Furthermore, the bound holds even for simple metrics such as the 1-dimensional Euclidean space.

The lower bound constructions of Meyerson and Fotakis are both based on a sequence of clients which converges towards a point unknown to the online algorithm, but which is the location of the optimal facility for the offline case. This sequence naturally relies on the order in which the clients are revealed to the online algorithm to maximize its cost. In fact, Meyerson showed that if the clients arrive in a uniform random order, there exists a randomized online algorithm with an expected constant competitive ratio [63].

In our model, we tackle the problem of the specific worst-case sequences from a different angle: Since the high cost of the online algorithm stems from the placement of facilities on points that are effectively rendered useless over time, it is only natural to ask whether the performance of an online algorithm improves asymptotically when it is allowed to make corrections to the position of its placed facilities. Divékih and Imreh [32] considered such a variant in which the online algorithm may move the facilities to an arbitrary new position in each time step, without incurring any additional cost for moving. Their algorithm computes an offline approximation of the optimal solution in each time step and moves the facilities accordingly. While they achieve a constant competitive ratio in this scenario, their model also has a major drawback: Since movement of facilities comes for free, a misplaced facility can be moved to an optimal point without any disadvantage for the algorithm from the initial misplacement.

We propose a model where the online algorithm may make corrections to the positions of its facilities, but instead of this being free, it incurs a cost proportional to the distance a facility is moved. In this way, bigger changes of the configuration of the algorithm are more costly and may therefore be less advantageous than smaller corrections to the positions. Furthermore, we consider a variant in which the facilities may not be moved to arbitrary positions but can only be moved a constant maximum distance in each time step. This allows to measure the improvement of the solution of an online algorithm dependent on the amount of correction it makes in every step, thus getting a more refined view on the complexity of the problem.

Another difference to the model by Divékih and Imreh is that requests have to be served instantly as opposed to only counting the connection cost at the end. Our variant therefore also puts more emphasis on the online aspect of the model, since being able to assign all clients at the end of the computation seems to contradict the idea that the solution has to be created online.

In addition to the theoretical interest of movement as a refined measure for the complexity, the extension of the Facility Location problem by movement might also be of independent interest for some practical applications as mentioned in the introduction. In these scenarios we consider resources that are mobile in the sense that they can be reconfigured to better fit incoming requests or are actually mobile in a physical sense. An example for the latter case is the usage of mobile relay stations to provide network access in certain disaster scenarios, where the commonly used infrastructure in the area has collapsed. For theses cases, our algorithm presents a solution on how to best approximate an optimal, static configuration.

Chapter Basis

The model, algorithms and analysis in this chapter are based on the following publication:

> *Björn Feldkord and Friedhelm Meyer auf der Heide. Online Facility Location with Mobile Facilities. In: Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures (SPAA), pages 373–381, 2018. Cf. [36].*

3.1 Formal Model

We first define the Facility Location problem in the Euclidean space (of arbitrary dimension). The input is a sequence of requests $(v_1,\dots,v_T)$ that are represented as points in the Euclidean space. A feasible solution is a non-empty subset F of facilities. For a facility $f \in F$, $p(f)$ denotes the position of f in the Euclidean space. The cost of the solution F is $c(F) = |F| \cdot c_f + \sum_{t=1}^{T} \min_{f\in F} d(p(f), r_t)$, where $d(x,y)$ denotes the Euclidean distance between two points x and y, and c_f is a constant called the opening cost of a facility. Our goal is to find a set F minimizing $c(F)$.

In the Online Facility Location problem, the requests $(v_1,\dots,v_T)$ arrive over time and each request must be served by connecting it to an open facility in that time step: i.e., if F_t represents the set of open facilities in step t, the cost for serving the request v_t is $\min_{f\in F_t} d(p(f), v_t)$. The request v_{t+1} is revealed to the online algorithm only once F_t is fixed.

In our augmented variant of the Online Facility Location problem, an online algorithm may move currently open facilities to a different position in each time step. In the unlimited version, the new position of a facility may be arbitrary, while in the limited version, the new position may only have a distance from the old position of at most a constant m. A solution now formally consists of sets $F_1 \subseteq F_2 \subseteq \dots \subseteq F_T$ and positions $p_t(f)$ for all $f \in F_t$, where $d(p_t(f), p_{t+1}(f)) \le m$ for all t with $f \in F_t$ in the limited movement variant. The total cost of a solution can be calculated as follows:

$$c(F_1,\dots,F_T,p_1,\dots,p_T) = |F_T| \cdot c_f + \sum_{t=1}^{T} \min_{f\in F_t} d(p_t(f), v_t) + \sum_{t=1}^{T-1} \sum_{f\in F_t} D \cdot d(p_t(f), p_{t+1}(f))$$

for some constant $D \ge 1$. The competitive ratio of an online algorithm in this resource augmentation setting is defined as the worst-case ratio of its cost against a static offline solution, i.e., the adversary may not utilize the movement of facilities.

3.2 Summary of Results

We present randomized online algorithms for the two variants of the problem and analyze their competitive ratios against an oblivious adversary. Our results for the real line as a metric are as follows: For the case of unlimited movement, we achieve an expected competitive ratio of $\mathcal{O}(\frac{\log D}{\log\log D})$. In the case of limiting the movement of a facility to some constant m in each time step, we achieve an expected competitive ratio of $\mathcal{O}(\max\{\frac{\log(c_f/m)}{\log\log(c_f/m)}, \frac{\log D}{\log\log D}\})$. We extend our result to the Euclidean space of arbitrary dimension, where we achieve a competitive ratio with an additional additive $\mathcal{O}(\sqrt{k})$ term, where k is the number of facilities in the optimal solution. The algorithms are the same for both the real line and the Euclidean space of higher dimensions, but the analysis is different in a key point. It explicitly uses the geometry of the line to achieve the better result for that case, while the analysis for the general dimension approaches the problem from a more combinatorial aspect.

Finally, we give lower bounds for both the unlimited and limited movement case, which shows that our algorithms are asymptotically optimal for the line. As in the original Online Facility Location problem, these lower bounds hold even in the case if there is only a single facility in the optimal solution.

3.3 Algorithms

In this section, we present randomized algorithms for our augmented Online Facility Location problem. We first give a technical overview explaining the main ideas of our algorithms and how they are reflected in the analysis. Afterwards, we present an algorithm for the case of unlimited movement on the real line and its analysis in detail. We extend this algorithm to the case of limited movement and to the Euclidean space of higher dimensions in the subsequent sections, reusing many ideas from the first case for the analysis.

Note that due to the cost function for moving facilities, the algorithm by Divékih and Imreh [32] does not achieve an asymptotically optimal competitive ratio if D is large. Their algorithm, called OptFollow (OFW), computes the optimal solution in every time step and moves its facilities accordingly if it has as many or less facilities than the optimum. If the algorithm has already more facilities than the optimal solution, it moves them to the optimal solution of the respective k-median problem, where k is the number of currently owned facilities. We show that this strategy is $\Omega(D)$-competitive even on the line.

Theorem 3.3.1 The OFW algorithm cannot achieve a better competitive ratio than $\frac{1}{2}(1+\frac{D}{4})$ on the real line.

Proof. We construct the following input sequence by identifying points on the line with real numbers: We create a total of T requests where the i'th request appears at position $\frac{c_f}{2^i}$. The optimal solution has a cost less than $2c_f$ which can be achieved by placing a single facility at point 0.

The OFW algorithm places a facility at point $\frac{c_f}{2}$ in the first time step, which induces cost c_f. Since the optimal cost is less than $2c_f$, OFW will never construct a second facility. For $T \geq 4$ the optimal solution w.r.t. the input so far, after the T'th request, will place a single facility at a point identified with a real number at most $\frac{c_f}{4}$. Hence the OFW algorithm will move its facility by a distance of at least $\frac{c_f}{4}$, inducing costs $D \cdot \frac{c_f}{4}$. The lower bound directly follows from comparing the costs of the two solutions. ■

This lower bound illustrates that the high movement costs make the approach of always moving the current optimal solution is infeasible. In the upcoming section we present our approach which solves this problem and in addition only makes local changes to a solution in every time step. Hence, it does not require the computation of an offline solution in every time step.

3.3.1 Technical Overview

When considering the lower bound constructions for the classical Online Facility Location problem in [63, 41], we observe that they rely on an input sequence that converges towards a (previously unknown) point which is optimal for the offline algorithm (similar to the sequence used in the proof of Theorem 3.3.1). The online algorithm cannot place facilities in points "ahead" in the sequence (since subsequent points are randomly determined) and therefore incurs significantly higher costs by requiring the placement of many facilities along the convergence path.

Our first main idea to circumvent the lower bound constructions is to move the facilities along the convergence path rather than placing new ones. In accordance with our cost function for moving facilities, we regard the requests associated with one facility as an instance of the Mobile Server problem. In order to create those instances, we divide the space as follows: Whenever we place a new facility, we remember its original location (called *facility origin*) and divide the given space in such a way that each point is associated to the closest facility origin. A facility will serve all requests on points associated to its origin by moving a $1/D$ fraction towards them. This resembles the algorithm for the Mobile Server problem presented in the previous chapter.

Assume for now that we treat all requests in the described way. The problem of our division of the space is that it might not be the same as in the optimal solution and therefore requests that the online algorithm regards as part of the same Mobile Server problem instance are served by different facilities in the optimal solution. Hence when referring to the optimal solution in our virtual Mobile Server problem, we would have an offline solution that switches positions without incurring any costs. It is clear that no online algorithm can achieve a bounded competitiveness against such an adversary.

Although guessing the optimal division of the space seems hard at first, we show that we can approximate the proper division during the execution fast enough by randomly placing new facilities at current requests where the probability to place a new facility is proportional to the costs that would otherwise occur by serving this request. We analyze this placement strategy by defining so-called *prohibited areas* for pairs of facilities. For a pair of two facilities o and o' in the optimal solution, the prohibited area of o' to o represents all points that are much closer to o' than o. These points should not be served by a facility of the online algorithm for which the origin is placed near o, since in this case the serving costs for the online algorithm would be much higher than for the optimal solution. We show that a facility of the online algorithm associated to o will, after a small amount of requests, no longer serve requests within the prohibited area of o' since a new facility is then placed within that area. The total costs for this separation of optimal facilities are bound by explicitly utilizing the structure of the solution in the 1-dimensional case. For the Euclidean space of higher dimensions, we need a more complex, combinatorial argument to achieve a better bound than the trivial bound that can be achieved by simply counting the maximum costs for the separation for each pair of optimal facilities. This simple counting would only result in a bound of $\mathcal{O}(k)$, where k denotes the number of optimal facilities.

Now assume we have properly separated the optimal facilities from one another. If we use the ideas described so far for serving the requests as part of a virtual Mobile Server problem instance, we would obtain an $\mathcal{O}(D)$-competitive algorithm where the worst case essentially occurs when a facility is placed far away from an optimal facility and has to be moved all the way towards the optimal point (cf. the lower bound for the OFW algorithm, Theorem 3.3.1). To further improve the algorithm, we want to utilize the movement only for requests that are close to the optimal facility. We achieve this by dividing the requests into two categories called *large* and *short*, which is determined by their distance to the closest facility origin of the online algorithm. Short requests, i.e., requests that are close to a facility origin, are treated as part of a Mobile Server problem instance and are therefore served by a mobile facility that moves towards the request as described above. Large requests are served by an additional static facility that never moves and is located at the corresponding facility origin. With this categorization of the requests we reduce the amount of movement done by facilities and obtain a competitive ratio of $\mathcal{O}(\frac{\log D}{\log\log D})$ for the real line. In Section 3.4, we show that this is asymptotically the best possible ratio achievable by any algorithm.

3.3.2 A Randomized Algorithm for Unlimited Movement

We present a randomized online algorithm for the real line that uses unlimited movement of the facilities. Both clients and facilities will be denoted as real numbers corresponding to a point on the line. For two points $x \neq y$, we say that x is to the left of y if $x < y$, otherwise x is to the right of y.

During the execution, our algorithm maintains a set of *facility origins* $\mathscr{F} \subseteq \mathbb{R}$. Whenever a new facility is placed at a point f by the algorithm, f will be added to $\mathscr{F}$. We say a point $x \in \mathbb{R}$ *belongs to* $f \in \mathscr{F}$ if $d(f,x) \leq d(f',x)$ for all $f' \in \mathscr{F}$.

In the further description of the algorithm, we refer to placing a new facility origin at point f. In these cases, the following happens: The algorithm places one *static facility* at f, which will never be moved, and a *mobile facility* at f, which is assigned to the points that belong to f. Hence for every $f \in \mathscr{F}$ there exist 2 facilities in the solution of the online algorithm.

Algorithm 4 — Unlimited Algorithm. Let v be a request and $f \in \mathscr{F}$ such that v belongs to f. We say v is *short* if $d(f,v) \leq 2\frac{c_f}{D}$, otherwise we say that v is *large*.
Upon arrival of a request v, the online algorithm determines $f \in \mathscr{F}$ such that v belongs to f. If v is a large request, we set the *servicing facility* to the static facility located at f. If v is short, the servicing facility is the mobile facility assigned to the area of f. Let a be the position of the servicing facility. Place a new facility origin with probability $\frac{d(a,v)}{17c_f}$ on v. If no new facility was placed on v, then either service v directly if a is static (v is large) or move a a distance of $\frac{d(a,v)}{D}$ towards v and then service v if a is mobile (v is short).

In the following, we analyze the competitive ratio of the online algorithm on the basis of the ideas from the previous section. The costs for short and large requests can by analyzed independently of one another.

Lemma 3.3.2 The cost of the online algorithm for large requests are at most $\mathcal{O}(\frac{\log D}{\log\log D})$ times the cost of the optimal solution.

Proof. Consider a facility o placed by the optimal offline algorithm. We refer to the interval of o as all points for which o is the closest facility of the optimal solution. Define the inner zone of o as all points in its interval that have a distance less than $\frac{c_f}{D}$ to o. The set of outer zones of o consists of zones $1,\dots,h$ where each zone j contains the points in the interval of o that are at a distance in $[\frac{c_f}{x^j}, \frac{c_f}{x^{j-1}})$ from o. We set $x := \frac{\log D}{\log\log D}$ and $h := \Theta(\frac{\log D}{\log\log D})$ such that $x^h = D$. Note that each point in the interval of o now either belongs to the inner zone or one of the outer zones of o.

For each of the zones of o, there exists a facility origin of the online algorithm in the respective zone after expected cost at most $17c_f$ incurred by requests within that zone. After a facility origin is placed within the inner zone of o, large requests can only appear in outer zones since the origin has a distance of at most $\frac{c_f}{D}$ to o and a large request must have a distance of at least $2\frac{c_f}{D}$ to every origin. For the outer zones of o where a facility origin is already placed, the distance from a request v to the origin $f \in \mathscr{F}$ is at most $2 \cdot d(o,v) + x \cdot d(o,v) = \mathcal{O}(\frac{\log D}{\log\log D}) \cdot d(o,v)$. Combined with the fact that there are at most $h+1$ zones for each optimal facility this implies a competitive ratio of $\mathcal{O}(\frac{\log D}{\log\log D})$ for the large requests. ∎

We further divide the short requests by their distance to their respective optimal facility. The competitive ratio on short requests that are far from the optimal facilities is easy to obtain. Note that the movement in these cases would not be necessary, since the movement only improves the performance when the facility origin is already reasonably close to the optimal facility (i.e., the convergence point in the lower bound sequences), but also does not increase the cost of any request by more than a constant factor.

Lemma 3.3.3 Consider the short requests served by a mobile facility of the online algorithm belonging to a facility origin with a distance of at least $4\frac{c_f}{D}$ to the nearest optimal facility. The cost for serving these requests is at most $\mathcal{O}(1)$ times the cost of the optimal solution.

Proof. Consider a facility origin f with $d(o,f) \geq 4\frac{c_f}{D}$, where o is the closest facility in the optimal solution to f. For all short requests served by a mobile facility assigned to this origin, the optimal costs will be at least $2\frac{c_f}{D}$. On the other hand, the expected costs of the online algorithm will be no more than $\mathcal{O}(\frac{c_f}{D})$ for each of those requests by definition. Note that this is also true for short requests that are served by a different optimal facility o' than o since the space is divided equally between o and o' and hence $d(o',f) \geq d(o,f)$. ■

We finally turn our attention to short requests that occur in close proximity to a facility of the optimal solution. As explained in the previous section, those requests should not be served by a facility of the online algorithm that originates in the interval of a different optimal facility, since in that case the costs of the online algorithm are much higher than the costs of the optimal solution. We formally classify this type of requests as follows: For optimal facilities o and o', we define the prohibited area of o' to o to contain all points v for which $d(o',v) \leq \frac{1}{8} \cdot d(o,o')$ holds. It is easy to see that if there is a facility origin within the prohibited area of o' to o, no facility with origin in the interval of o will serve a request in the prohibited area of o' to o. If this is the case, we say that o' is separated from o.

Consider a facility origin f in the interval of o. We call a request v in the prohibited area of o' to o for which f is the closest facility origin a prohibited request. The concepts of prohibited areas and requests are illustrated in Figure 3.1.

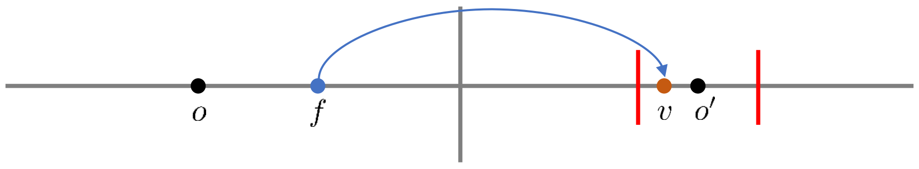

Figure 3.1: The marked interval on the right contains the prohibited area of o' to o. If the request v is served by a facility with origin f (indicated by the arrow), then v is a prohibited request.

Lemma 3.3.4 Consider the short requests served by a mobile facility of the online algorithm belonging to a facility origin with a distance less than $4\frac{c_f}{D}$ to the nearest optimal facility. The cost for serving these requests are at most $\mathcal{O}(1)$ times the cost of the optimal solution excluding prohibited requests.

Proof. For each mobile facility, we define an instance of the Mobile Server problem consisting of all requests served by that facility. Consider a mobile facility with origin in the interval of o. We may assume that all requests, which are not in the prohibited area of a different facility o' to o, are served by o with only a constant factor loss, since for a request v served by o' it holds $d(o,v) \leq d(o,o') + d(o',v) \leq 9d(o',v)$. Let a be the current position of the mobile facility of the algorithm for this instance. We define the potential associated with the instance to be $\phi = 4D \cdot d(o,a)$.

When a mobile facility is placed along with a new facility origin on a point v, the potential ϕ is initialized with $4D \cdot d(o,v)$. Since $d(o,v) \leq 4\frac{c_f}{D}$, the expected cost for placing a new facility when v is served by a facility at a is now $\frac{d(a,v)}{17c_f} \cdot (c_f + 4D \cdot d(o,v)) \leq d(a,v)$. In order to apply the potential argument properly, the change in potential must be accounted for every time the respective mobile facility is moved, including the times when it does so to serve prohibited requests. We show that including the potential in these costs does not increase them by more than a constant factor. When a mobile facility is moved from a point a to a' to serve any request v, we get $\Delta\phi = 4D \cdot (d(o,a') - d(o,a)) \leq 4D \cdot d(a,a') \leq 4d(a,v)$. Hence, including the potential argument does not increase the costs of any request by more than a constant factor.

We now consider the requests that are part of one virtual Mobile Server problem instance but not prohibited. When the algorithm serves a request v with a facility that is moved from a to a', the expected costs are $C_{Alg} \leq D \cdot d(a,a') + d(a',v) + d(a,v)$, while $C_{OPT} = d(o,v)$. First consider the case that $d(a',v) > d(o,v)$. The potential difference is $\Delta\phi = 4D \cdot (d(o,a') - d(o,a)) \leq -4d(a,v)$ since a moves towards v on a line and hence o either has to be located between a' and v or v is located between a' and o. The costs of the algorithm can be estimated by $C_{Alg} \leq D \cdot \frac{1}{D} d(a,v) + d(a',v) + d(a,v) \leq 3 \cdot d(a,v)$ and hence $C_{Alg} + \Delta\phi \leq 0$.

If $d(a',v) \leq d(o,v)$, we distinguish the following cases:

If $d(o,a') - d(o,a) \leq -\frac{1}{2} d(o,a)$, then $\Delta\phi \leq -2D \cdot d(o,a)$ and $C_{Alg} \leq 2d(a,o) + 3d(o,v)$, hence $C_{Alg} + \Delta\phi \leq 3C_{Opt}$.

Else, $d(o,a) < 2d(o,a')$ holds which implies $\Delta\phi \leq 4D \cdot d(a,a') \leq 4(d(a,o) + d(o,v)) \leq 20d(o,v)$ and $C_{Alg} \leq 5d(o,v)$. ■

So far, we have bounded all costs either incurred by large requests or by short requests not part of the prohibited area of an optimal facility o' to the optimal facility o in whose interval the servicing facility is originating. For short requests that remain to be analyzed, i.e., the prohibited requests, the cost of the online algorithm may be much higher than the optimal cost for the same requests. However, we show that when considering all such requests, the total cost is bounded within a constant factor of the costs for facilities in the optimal solution.

Lemma 3.3.5 The expected costs for prohibited requests is bounded by $\mathcal{O}(k \cdot c_f)$ on the real line, where k is the number of facilities in the optimal solution.

Proof. Consider two optimal facilities o and o'. W.l.o.g. assume that o' is located to the right of o. After an expected cost of at most $\mathcal{O}(c_f)$ for requests in the prohibited area of o' to o a facility origin is placed in that area, preventing any facility originating in the interval of o or to the left of it from entering the prohibited area of o' to o again.

Now number the optimal facilities $o_1, \ldots, o_k$ from left to right: i.e., if o_i refers to a position expressed as a real number we have $o_1 < o_2 < \ldots < o_k$. Consider facilities o_i, o_j with $i < j$ for which the algorithm has a facility origin placed in the intervals of those facilities but no facility of the algorithm originates in the intervals of $o_{i+1}, \ldots, o_{j-1}$. We define the following type of events:

(a) The algorithm places a facility origin in the interval of a facility o_ℓ, $\ell \in \{i+1, \ldots, j-1\}$.

(b) The algorithm places a facility origin in the prohibited area of o_i to o_j or vice versa.

Note that we can view these events as being charged by requests in the respective intervals or prohibited areas. After charging the event with an expected cost of $\mathcal{O}(c_f)$, it then actually occurs.

For events of type (a), we observe that after an expected cost of $\mathcal{O}(c_f)$ in the interval of o_ℓ overall, a facility is placed in this interval. When that is the case, facilities originating from intervals of $o > o_\ell$ no longer serve requests in intervals of facilities $o' < o_\ell$. For events of type (b), there are two of them in such a sequence and only occur once as argued above. Furthermore, any event of type (a) stops the events from happening. Hence, after an event of type (a) occurs, we can split the sequence of facilities in two parts $o_i, \ldots, o_\ell$ and $o_\ell, \ldots, o_j$ and apply the same arguments.

It follows that regarding the complete sequence $o_1, \dots, o_k$, event (a) occurs at most k times and between each of such events at most 2 events of type (b) occur. Hence the costs for the events of type (a) and (b) on expectation sum up to at most $3k \cdot \mathcal{O}(c_f)$. ■

Summarizing all lemmas in this section, we have bounded the costs of the online algorithm for all possible types of requests independently and can hence conclude the analysis.

Theorem 3.3.6 The online algorithm stated above is $\mathcal{O}(\frac{\log D}{\log\log D})$-competitive on the real line.

It is noteworthy that our algorithm is in fact $\mathcal{O}(1)$-competitive on short requests, implying that we have an overall $\mathcal{O}(1)$-competitive algorithm if all requests are at a distance of at most $\mathcal{O}(\frac{c_f}{D})$ from the nearest optimal facility. In contrast, the lower bound for the classical Online Facility Location problem would still hold in that scenario for a sufficiently large number of clients T. In fact, the distances in the lower bound decrease with growing T.

3.3.3 Extension to Limited Movement

We extend the algorithm from the previous section to utilize only a maximum moving distance of m in each time step. The algorithm considers two major cases: If D is large, then the algorithm works as in the unlimited case. Else, we need to alter the definition of small and large requests to fit the restriction of the movement for handling the short requests.

Algorithm 5 — Limited Algorithm. We classify a request v by its distance to the closest facility origin $f \in \mathcal{F}$. The classification differs based on the parameters: If $D \geq {}^{c_f}/_m$ we use the same classification as in the unlimited case, namely v is short if $d(f,v) \leq 2\frac{c_f}{D}$ and large otherwise. If $D < {}^{c_f}/_m$ we classify v as short if $d(f,v) \leq 2m$ and large otherwise. The servicing facility of v is either the static facility at f if v is large or the mobile facility assigned to the area in which v lies if v is short. Let a be the position of the servicing facility. Place a new facility origin with probability $\frac{d(a,v)}{17c_f}$ on v. When servicing a short request v from a mobile facility at point a, we move the facility by a distance of $\frac{d(a,v)}{2D}$ towards v and then service the request. Large requests simply get served by the static facility without any movement.

Note that the movement of the algorithm is possible in all cases under the restriction of a maximum movement distance m since v is short:
If $D \geq {}^{c_f}/_m$ then $\frac{d(a,v)}{2D} \leq \frac{2c_f}{2D^2} \leq m$.
If $D < {}^{c_f}/_m$ then $\frac{d(a,v)}{2D} \leq \frac{2m}{2D} \leq m$ (since $D \geq 1$).

Theorem 3.3.7 The online algorithm with limited movement stated above is $\mathcal{O}(\max\{\frac{\log({}^{c_f}/_m)}{\log\log({}^{c_f}/_m)}, \frac{\log D}{\log\log D}\})$-competitive on the real line.

Proof. The structure of the proof follows the analysis of the previous section. We consider only the case $D < {}^{c_f}/_m$ for which we need to show that the algorithm is $\mathcal{O}(\frac{\log({}^{c_f}/_m)}{\log\log({}^{c_f}/_m)})$-competitive. In case $D \geq {}^{c_f}/_m$, the algorithm works the same as in the unlimited case and hence Theorem 3.3.6 directly gives the desired bound.

We begin by analyzing only large requests, i.e., requests with a distance greater than $2m$ to their closest facility origin. The analysis works essentially the same as in Lemma 3.3.2. Consider an optimal facility o. Requests with a distance of at most m to o incur expected costs of at most $\mathcal{O}(c_f)$ before a facility origin is placed within that area and hence no further large requests appear. The other points can be divided into $h = \Theta(\frac{\log({}^{c_f}/_m)}{\log\log({}^{c_f}/_m)})$ outer zones, where the i'th zone contains all points within a distance $[\frac{c_f}{x^i}, \frac{c_f}{x^{i-1}})$ from o where $x = \frac{\log({}^{c_f}/_m)}{\log\log({}^{c_f}/_m)}$. After placing a facility origin in an outer zone of o, it is clear that all large requests in the same outer zone can be served for at most $3x$ times the optimal cost. From this, the bound for large requests follows.

We partition short requests into multiple instances of the Mobile Server problem as before: i.e., all requests served by one mobile facility form an instance. There are two kinds of instances we need to consider: If the mobile facility has its origin at a distance at least $4m$ from the optimal facility o, then the optimal solution has costs of at least $2m$ for each request and the costs of the algorithm are at most $\mathcal{O}(m)$ (cf. Lemma 3.3.3).

For instances where the origin is a distance less than $4m$ from o, we use a potential argument similar as in the proof of Lemma 3.3.4: Let a be the current position of the mobile facility. The potential is defined as $\phi(o,a) = 4D \cdot d(o,a)$. For the initial potential it holds $\phi_0 \leq 4Dm \leq 4c_f$, hence the costs for placing a new facility does not increase more than a constant factor. The analysis for requests that are part of the instance served by the mobile facility outside of prohibited areas of other facilities to o works as in Lemma 3.3.4. The bound on costs for prohibited requests can also be derived exactly as in the former case (Lemma 3.3.5). In total, we have shown that the online algorithm is $\mathcal{O}(1)$-competitive on short requests and $\mathcal{O}(\frac{\log(c_f/m)}{\log\log(c_f/m)})$-competitive on large requests, implying the desired bound. ■

3.3.4 Extension to Higher-Dimensional Spaces

In this section, we turn our attention to the Euclidean space of higher dimensions. We use the same algorithms as before. Note, that most of the arguments presented in Sections 3.3.2 and 3.3.3 still hold in this case. However, it should be obvious that the arguments presented in the proof of Lemma 3.3.5 no longer apply since they make explicit use of the linear structure of the space. Hence we present a new bound of the costs incurred by prohibited requests that relies more on the combinatorial aspects of the separation taking place over time rather than the structure of the underlying space.

Lemma 3.3.8 The expected costs for prohibited requests is bounded by $\mathcal{O}(k^{3/2} \cdot c_f)$, where k is the number of facilities in the optimal solution.

Proof. For two optimal facilities o and o' we write $o \mid o'$ if an online facility origin is located in the prohibited area of o' to o, i.e., o is separated from o'. Recall that this implies that no prohibited requests in the interval of o' will be served by facilities with their origin in the interval of o. We use the same notation with sets of facilities, in which the separation holds pairwise.

Consider the optimal facilities $o_1, \dots, o_k$ sorted by their distance to o_1, such that $d(o_1,o_2) \leq d(o_1,o_3) \leq \dots \leq d(o_1,o_k)$. Now consider a request v which is served by an online facility associated to o_i (meaning the closest optimal facility to its origin is o_i). Let o_1 be the closest optimal facility to v and let v be located in the prohibited area of o_1 to o_i. Furthermore, consider the minimal $j \leq i$ such that v lies within the prohibited area of o_1 to o_j.

Now consider an optimal facility o_ℓ with $j-1 \geq \ell \geq 2$. If $d(o_\ell,v) \geq \frac{1}{8} \cdot d(o_i,o_\ell)$, then $d(o_i,v) \leq 9d(o_\ell,v) \leq 9(d(o_\ell,o_1)+d(o_1,v)) \leq 9^2 \cdot d(o_1,v)$. Hence we may still assume that v is served by o_i in the optimal solution with the loss of only a constant factor. If on the other hand no such ℓ exists, then $d(o_\ell,v) < \frac{1}{8} \cdot d(o_i,o_\ell)$ and hence if a facility origin is placed on v, online facilities associated to o_i will no longer serve requests located in the prohibited area of o_ℓ to o_i for all $j-1 \geq \ell \geq 2$. We assume this to be the case from here on.

We collect the impacts of a new facility origin being placed on v. Most obviously, all o_ℓ with $\ell \geq j$ are now separated from o_1, i.e.,

$$\{o_j, o_{j+1}, \dots, o_i, \dots, o_k\} \mid o_1. \tag{3.1}$$

Additionally, as argued above, we get

$$o_i \mid \{o_1, \dots, o_{j-1}\}. \tag{3.2}$$

The position of v in relation to the facilities is depicted in Figure 3.2.

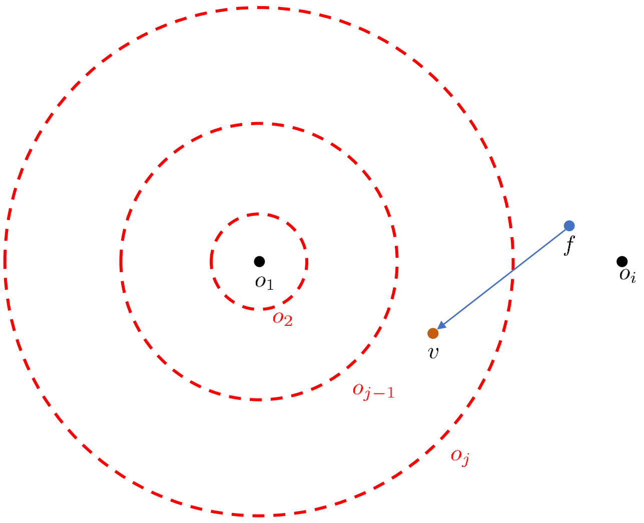

Figure 3.2: A depiction of an event as described in (3.1) and (3.2). The circles mark the prohibited areas of o_1 to the other facilities indicated by their labels.

We refer to the placement of such a facility origin as an event, and each request that has the potential to trigger such an event due to its location is referred to as contributing to an event. After expected costs of $\mathcal{O}(c_f)$ incurred by requests contributing to any of the events characterized above, one the events occurs.

We now collect these type of events in a $k \times k$ matrix B, where each entry b_{ij} represents an event in the area of o_j served from a facility originating in the area of o_i. Note that the implied ordering of the facilities within the matrix does not correspond to the ordering used to describe the events as above, since that ordering may be different in each of the events (it depends on the distances to o_j). Each entry $b_{ij} \in \{1, \dots, k-1\}$ represents the number of facilities o_i separates itself from in the respective event, i.e., the number of elements on the right side of (3.2). If the entry is 0 there was no such an event. Since o_i can separate itself from each facility only once, the sum of entries in a row is at most $k-1$. We say that each o_i has a budget of $k-1$ and each element b_{ij} draws from the budget of o_i.

Furthermore, each number in $\{1, \dots, k-1\}$ can only appear once per column. This is since each entry b_{ij} in column j implies that $k - b_{ij}$ facilities are separated from o_j after the respective event (the elements on the left side of (3.1)). Each event, however, must increase the number of facilities separated from o_j by at least 1, since at least o_i separates itself from o_j.

If we consider the budget of all facilities to be $k \cdot (k-1)$ we can derive that one event draws a total budget of b_{ij}. Since each b_{ij} can appear at most k times, the number of non-zero entries is maximized by first choosing k elements to be 1, then k elements to be 2, etc. The total budget is spent after $\mathcal{O}(\sqrt{k})$ rounds of this assignment, yielding a total of $\mathcal{O}(k^{3/2})$ entries that are not 0. ■

Utilizing the new bound on prohibited requests we achieve the following result for our algorithms in the Euclidean space of arbitrary dimension.

Theorem 3.3.9 The limited algorithm is $\mathcal{O}(\frac{\log D}{\log\log D} + \sqrt{k})$-competitive in the unlimited case and for a maximum moving distance of m the limited algorithm is $\mathcal{O}(\max\{\frac{\log(c_f/m)}{\log\log(c_f/m)}, \frac{\log D}{\log\log D}\} + \sqrt{k})$-competitive in the Euclidean space of arbitrary dimension, where k denotes the number of optimal facilities.

Proof. We sort the requests into categories as before and collect the results for each of them. Large requests can be analyzed as in Lemma 3.3.2 for the unlimited case and as in Theorem 3.3.7 for the limited case. For short requests that are served by a facility with origin more than a distance $4\frac{c_f}{D}$ (or $4m$ in the limited case) from the nearest optimal facility we may apply the analysis of Lemma 3.3.3 and for short requests that are not prohibited we use Lemma 3.3.4. Finally, for prohibited requests we bound the costs via Lemma 3.3.8. ■

3.4 Lower Bounds

In this section, we provide lower bounds on the competitive ratio for randomized algorithms against oblivious adversaries. The results illustrate that our algorithms from the previous section achieve an asymptotically tight competitive ratio in both the unlimited and limited movement versions on the real line.

Both lower bounds closely follow the structure of the original lower bound by Fotakis [41] and utilize Yao's Min-Max Principle [77]: i.e., we construct a randomized sequence of demands independently of a deterministic online algorithm processing the input. We construct complete binary trees as an underlying metric where the distances are chosen in such a way that utilizing the movement becomes essentially useless for the online algorithm. We show that such trees can be embedded onto the real line with all necessary properties staying intact.

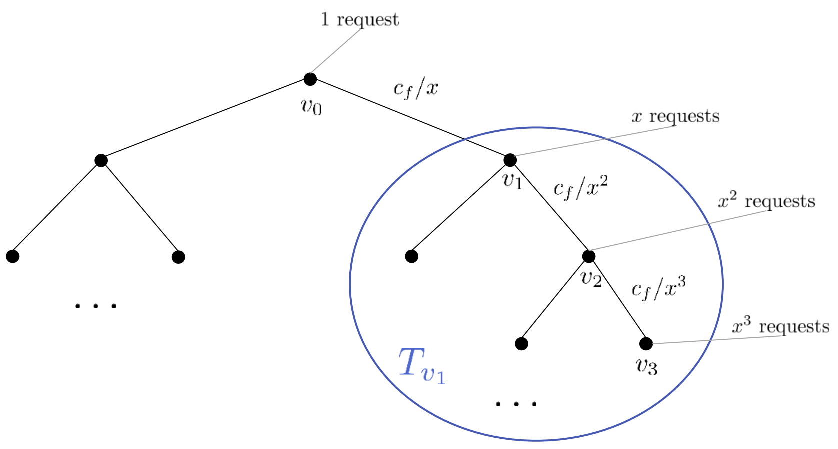

Figure 3.3: An illustration of the tree used in Theorem 3.4.1 and an example sequence. The construction is similar to the lower bound in [41].

Theorem 3.4.1 No randomized algorithm whose competitive ratio does not depend on T can be better than $\Omega(\frac{\log D}{\log\log D})$-competitive even on the real line and against an oblivious adversary.

Proof. We construct a complete binary tree with depth $x := \lfloor \frac{\log D}{\log\log D} \rfloor$. The distance between the root and its children is ${c_f}/{x}$ and the distance between nodes at level $i-1$ and nodes at level i is ${c_f}/{x^i}$. We choose a simple path $v_0, \dots, v_{x-1}$ from the root v_0 to a leaf v_{x-1} by succinctly choosing a child v_{i+1} of v_i uniformly at random. The input sequence is divided into x phases, where in phase $i \in \{0, \dots, x-1\}$ there are x^i requests on v_i. The total number of requests is hence $\sum_{i=0}^{x-1} x^i \le x^x \le D$. The optimal solution can place a facility at v_{x-1} and pays at most $c_f + \sum_{i=0}^{x-2} x^i \cdot \sum_{j=i+1}^{x-1} \frac{c_f}{x^j} \le 3c_f$. Figure 3.3 shows the constructed tree and an example for the input sequence.

Let T_v be the sub-tree rooted at node v. Consider an online algorithm at the end of phase $i \in \{0, \dots, x-2\}$. If the algorithm has not placed a facility in T_{v_i} or on the edge (v_{i-1}, v_i), the costs are at least $x^i \cdot \frac{c_f}{x^i} = c_f$. Otherwise, if the algorithm placed a new facility within T_{v_i}, it will not be part of $T_{v_{i+1}}$ with a probability of at least $1/2$. Hence, the costs for facilities not in $T_{v_{i+1}}$ are at least ${c_f}/{2}$. Finally, if the algorithm has moved a facility by a distance δ, it holds for the costs $D \cdot \delta \ge x^x \cdot \delta$. Hence moving a facility in order to reduce the costs for the requests does not pay off even if it reduces the costs of all requests by the amount the facility is moved. We conclude that for every i, the online algorithm pays $\Omega(c_f)$ for requests and facilities not in $T_{v_{i+1}}$. It follows that the costs of the online algorithm are $\Omega(x \cdot c_f)$ which implies the competitive ratio on the given tree metric.

It remains to describe how to embed the tree onto the real line such that all necessary properties for the lower bound are preserved. We choose the same embedding as in [41] which puts the tree root at point 0, and succinctly maps the children of a node v at level i which is located at position p_v to positions $p_v + \frac{c_f}{x^{i+1}}$ and $p_v - \frac{c_f}{x^{i+1}}$. With this embedding it was shown in [41] that no pairwise distance increases and hence the value of the optimal solution does not increase. Furthermore the relevant distances from the algorithm's perspective are maintained in the classical setting, i.e., without the option to move a facility. Since the number of requests is the same, moving a facility any distance only increases the overall cost. It follows that the arguments from above still hold in the embedding. ■

Theorem 3.4.2 No randomized algorithm whose competitive ratio does not depend on n and whose movement is restricted to m per time step can be better than $\Omega(\frac{\log({c_f}/{m})}{\log\log({c_f}/{m})})$-competitive.

Proof. We construct a similar binary tree as in the proof of Theorem 3.4.1 where the distance between nodes at level $i-1$ and i is $2\frac{c_f}{x^i}$ with $x := \lfloor \frac{\log \sqrt{{c_f}/{m}}}{\log\log \sqrt{{c_f}/{m}}} \rfloor$. The input sequence is again constructed by choosing a simple path $v_0, \dots, v_{x-1}$ from the root v_0 to a leaf v_{x-1} by succinctly choosing a child v_{i+1} of v_i uniformly at random. The input sequence is divided into x phases where in phase $i \in \{0, \dots, x-1\}$ there are x^i requests on v_i. The optimal solution can place a facility at v_{x-1} and pays at most $6c_f$.

Let T_v be the sub-tree rooted at node v. Consider an online algorithm at the end of phase $i \in \{0, \dots, x-2\}$. If the algorithm has not placed a facility in T_{v_i} or on the edge (v_{i-1}, v_i), the costs are at least $x^i \cdot 2\frac{c_f}{x_i} = 2c_f$. Otherwise, if the algorithm placed a new facility within T_{v_i}, it will not be part of $T_{v_{i+1}}$ with a probability of at least $1/2$. Hence, the expected costs for requests and facilities not in $T_{v_{i+1}}$ are at least ${c_f}/{2}$.

Finally, we have to argue that movement does not help the algorithm. Consider the shortest edge, which has length $2\frac{c_f}{x^{x-1}} \ge 2\sqrt{m \cdot c_f}$. On the other hand, there are at most x^x requests in total and $x^x \cdot m \le \sqrt{m \cdot c_f}$. Hence, over the course of the whole instance, a facility cannot even traverse half of the shortest edge in the tree. Movement can therefore reduce costs for the algorithm by at most a factor 2. The embedding onto the real line works the same as in Theorem 3.4.1 and the lower bound holds using the same arguments. ■

Note, that the lower bound in Theorem 3.4.1 also holds for the limited movement case. Hence, the lower bound for online algorithms becomes the maximum of both presented bounds.

4. Conclusion & Open Problems

We will first recap the results from the previous chapters and comment on potential direct improvements to these results. We will also mention direct extensions to the models discussed in this thesis. Afterwards, we will mention some more general extensions to the models that would cover a broader range of aspects of our resource allocation scenario.

For the Mobile Server problem, we have seen that a simple, deterministic algorithm is sufficient to get an almost optimal competitive ratio in the case of $k = 1$. For the Euclidean space of dimension 1, we have an asymptotically optimal competitive ratio that cannot be beaten even by a randomized algorithm against an oblivious adversary. For the Euclidean space of higher dimensions, we miss the optimal competitive ratio by only a factor of $1/\sqrt{\delta}$. 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 improved online algorithm or provide a better analysis to achieve the better competitive ratio.

For the case of $k \geq 2$, we have given deterministic algorithms that transform solutions for the k-Server and k-Page Migration problems into solutions for the Mobile Server problem. While the competitive ratio of our solution depends only on the parameters that also appear in the lower bounds, there still is an asymptotic gap between the upper and lower bounds for the problem. The question in which direction the gap can be closed is 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. On the other hand, if $\Omega(k^2)$ is a lower bound for k-Page Migration, this carries over to our model as well. The fact that for $m_c < m_s$ we achieve a competitive ratio nearly equal to the simulated algorithm suggests that the method of mainly following a simulated algorithm does not inherently lead to an asymptotic loss in the competitive ratio. However, direct solutions could still enable a different analysis, which could lead to improvements in the ratio over algorithms which utilize the simulation. 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, similar to as it is done in the online algorithms for Page Migration.

If we allow randomization, we can get $\mathcal{O}(\text{polylog}(k))$-competitive algorithms against oblivious adversaries for both the k-Server problem [58] and, by the transformation of Bartal et al., for the k-Page Migration problem [11]. As our construction is entirely deterministic, apart from potentially the simulated algorithm, it would be interesting whether randomization in our construction can be used to significantly improve the competitive ratio. The desired result would be an algorithm with a competitive ratio polylogarithmic in k.

Regarding our variant of the Online Facility Location problem, our algorithm achieves a competitive ratio independent of the number of clients, surpassing the lower bound of the Online Facility Location problem without movement. For the real line, the competitive ratio is tight in both the unlimited and limited movement case. For higher dimensions, it is still open whether a competitive ratio independent of the number of facilities in the optimal solution can be achieved. We conjecture that the competitive ratios achieved by our randomized algorithms can also be achieved with deterministic algorithms, although those algorithms may have to use more complex policies with regards to placing new facilities. A good candidate algorithm would be to adapt the deterministic algorithm for the Online Facility Location problem by Fotakis [41], although the algorithm would then lose the property of working essentially without memory of past requests. We focused on the most natural cost function for movement, as the linear cost punishes bigger corrections of the positions and is similar to other resource allocation models. However, it would also be interesting to consider other cost functions, such as an arbitrary polynomial in the distance moved to even further punish big moves. On the other hand, constant cost for movement would encourage the utilization of as few corrections as possible in total. It could also be interesting to globally restrict the total movement possible for one facility instead of an upper bound on the movement per round.

Other interesting problems with regards to Online Facility Location could be derived by changing how the serving costs of the clients are paid for. Right now, the clients pay the distance in the same step as they appear. For applications related to clustering, where the end result is evaluated as a whole, one could imagine to count the cost as in the paper by Divékih and Imreh [32] where the clients are only connected to facilities in the end. Two variants on this are possible: Either the connections work exactly as in [32] and clients can essentially be reassigned to a different facility in each time step, or they need to be assigned to a fixed facility in the same step where they arrive (while still counting the cost as a whole in the end).

We now consider broader extensions to our work. For the Mobile Server problem, it still remains open to investigate the problem with multiple servers and multiple requests in one time step. This problem is also still open for the k-Page Migration problem, i.e., without considering any restrictions on the movement. In Section 1.2 we already mentioned two papers about the Facility Reallocation problem [31, 43] which indicate that the problem would be interesting even when restricted to the real line and with $D = 1$.

In the introduction, we already mentioned the leasing scenario which is a natural extension regarding many applications of the Facility Location problem [65]. In this model, the competitive ratio can be formulated in terms of the length of the longest lease instead of the number of clients [54]. It would be interesting to explore whether the introduction of facility movement to the leasing model would also lead to the removal of that parameter from the competitive ratio. For the real line, this was shown to be possible in a Bachelor's thesis by Marcel Geromel [45].

The leasing framework would also introduce the possibility to temporarily extend the number of resources in problems such as the Mobile Server problem, where one could use the lease of a resource to serve requests far away from the permanent servers. This would lead to situations where the online algorithm has more servers than the offline optimum, a scenario that is also interesting as a more basic model. The related model is the (h,k)-Server problem [55] in which the cost of an online algorithm for the k-Server problem is compared to an optimal solution that only uses $h < k$ servers. Meaningful results for the Euclidean space have not yet been achieved, as the competitive ratio of algorithms for the k-Server problem do not seem to improve their competitive ratio [8, 9] and improvements could only be made on discrete metrics where the competitive ratio depends on the size of the metric [9, 30]. It is reasonable to assume that progress on this basic model is directly linked to possible extensions of our models in this direction, i.e., investigating the (h,k)-Server problem in the Euclidean space would be an important first step. Besides the obvious extension to the Mobile Server problem, the concept would also be interesting for an extension of the Online Facility Location problem if we also allow the adversary to move its facilities for the same cost as the online algorithm. Here, the online algorithm would have to balance buying potentially more facilities than the offline algorithm for higher costs against the benefits of more resources for the underlying movement problem.

The above mentioned problem would be a first idea for how to fuse the two problems of moving a fixed number of resources (Mobile Server problem etc.) and placing an arbitrary number of resources to fixed locations (Online Facility Location). Both aspects can be further extended by making distinctions between read and write requests as in the File Allocation problem [6] or by making the resources contain heterogeneous data or services. The latter was introduced to the Facility Location problem [67], but no online algorithms have been given so far.

The open problems described here are obviously only from a small selection of potentially interesting problems, as the amount of literature surrounding just the basic problems of k-Server and Facility Location demonstrates. We hope our results will serve as a starting point for future work in this area which will hopefully lead to a strong theoretical basis of solving problems in the area of resource allocation in mobile environments.

Bibliography

[1] Sebastian Abshoff, Peter Kling, Christine Markarian, Friedhelm Meyer auf der Heide, and Peter Pietrzyk. Towards the price of leasing online. *J. Comb. Optim.*, 32(4):1197–1216, 2016.

[2] Susanne Albers and Hisashi Koga. Page migration with limited local memory capacity. In *Proceedings of the 4th International Workshop on Algorithms and Data Structures (WADS)*, pages 147–158, 1995.

[3] Aris Anagnostopoulos, Russell Bent, Eli Upfal, and Pascal Van Hentenryck. A simple and deterministic competitive algorithm for online facility location. *Inf. Comput.*, 194(2):175–202, 2004.

[4] Barbara M. Anthony and Anupam Gupta. Infrastructure leasing problems. In *Proceedings of the 12th International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 424–438, 2007.

[5] Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for euclidean k-medians and related problems. In *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 106–113, 1998.

[6] Baruch Awerbuch, Yair Bartal, and Amos Fiat. Competitive distributed file allocation. *Inf. Comput.*, 185(1):1–40, 2003.

[7] Nikhil Bansal, Niv Buchbinder, Aleksander Madry, and Joseph Naor. A polylogarithmic-competitive algorithm for the k-server problem. *J. ACM*, 62(5):40:1–40:49, 2015.

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

[9] Nikhil Bansal, Marek Eliés, Lukasz Jez, and Grigorios Koumoutsos. The (h, k)-server problem on bounded depth trees. *ACM Trans. Algorithms*, 15(2):28:1–28:26, 2019.

[10] Yair Bartal, Avrim Blum, Carl Burch, and Andrew Tomkins. A polylog(n)-competitive algorithm for metrical task systems. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 711–719, 1997.

[11] Yair Bartal, Moses Charikar, and Piotr Indyk. On page migration and other relaxed task systems. *Theor. Comput. Sci.*, 268(1):43–66, 2001.

[12] Yair Bartal, Amos Fiat, and Yuval Rabani. Competitive algorithms for distributed data management. *J. Comput. Syst. Sci.*, 51(3):341–358, 1995.

[13] Yair Bartal and Elias Koutsoupias. On the competitive ratio of the work function algorithm for the k-server problem. *Theor. Comput. Sci.*, 324(2-3):337–345, 2004.

[14] Luca Becchetti, Stefano Leonardi, Alberto Marchetti-Spaccamela, Guido Schäfer, and Tjark Vredeveld. Average-case and smoothed competitive analysis of the multilevel feedback algorithm. *Math. Oper. Res.*, 31(1):85–108, 2006.

[15] Wolfgang W. Bein, Marek Chrobak, and Lawrence L. Larmore. The 3-server problem in the plane. *Theor. Comput. Sci.*, 289(1):335–354, 2002.

[16] Shai Ben-David, Allan Borodin, Richard M. Karp, Gábor Tardos, and Avi Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11(1):2–14, 1994.

[17] Sergei Bespamyatnikh, Binay K. Bhattacharya, David G. Kirkpatrick, and Michael Segal. Mobile facility location. In *Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, pages 46–53, 2000.

[18] Marcin Bienkowski, Jaroslaw Byrka, Miroslaw Korzeniowski, and Friedhelm Meyer auf der Heide. Optimal algorithms for page migration in dynamic networks. *J. Discrete Algorithms*, 7(4):545–569, 2009.

[19] Marcin Bienkowski, Jaroslaw Byrka, and Marcin Mucha. Dynamic beats fixed: On phase-based algorithms for file migration. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 13:1–13:14, 2017.

[20] Marcin Bienkowski and Jaroslaw Kutylowski. The k-resource problem in uniform metric spaces. *Theor. Comput. Sci.*, 459:42–54, 2012.

[21] E. Bittner, Csanád Imreh, and Judit Nagy-György. The online k-server problem with rejection. *Discrete Optimization*, 13:1–15, 2014.

[22] David L. Black and Daniel D. Sleator. Competitive algorithms for replication and migration problems. Technical Report CMU-CS-89-201, Department of Computer Science, Carnegie-Mellon University, 1989.

[23] Allan Borodin, Nathan Linial, and Michael E. Saks. An optimal online algorithm for metrical task systems. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 373–382, 1987.

[24] Sébastien Bubeck, Michael B. Cohen, James R. Lee, and Yin Tat Lee. Metrical task systems on trees via mirror descent and unfair gluing. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 89–97, 2019.

[25] Sébastien Bubeck, Michael B. Cohen, Yin Tat Lee, James R. Lee, and Aleksander Madry. k-server via multiscale entropic regularization. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 3–16, 2018.

[26] Niv Buchbinder, Anupam Gupta, Marco Molinaro, and Joseph (Seffi) Naor. k-servers with a smile: Online algorithms via projections. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 98–116, 2019.

[27] Marek Chrobak, Howard J. Karloff, T. H. Payne, and Sundar Vishwanathan. New results on server problems. *SIAM J. Discrete Math.*, 4(2):172–181, 1991.

[28] Marek Chrobak and Lawrence L. Larmore. An optimal on-line algorithm for k-servers on trees. *SIAM J. Comput.*, 20(1):144–148, 1991.

[29] Marek Chrobak, Lawrence L. Larmore, Nick Reingold, and Jeffery R. Westbrook. Page migration algorithms using work functions. *J. Algorithms*, 24(1):124–157, 1997.

[30] Christian Coester, Elias Koutsoupias, and Philip Lazos. The infinite server problem. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 14:1–14:14, 2017.

[31] Bart de Keijzer and Dominik Wojtczak. Facility reallocation on the line. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 188–194, 2018.

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

[33] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004.

[34] Björn Feldkord, Till Knollmann, Manuel Malatyali, and Friedhelm Meyer auf der Heide. Managing multiple mobile resources. Accepted for publication in: Proceedings of the 17th Workshop on Approximation and Online Algorithms (WAOA), 2019.

[35] Björn Feldkord and Friedhelm Meyer auf der Heide. The mobile server problem. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 313–319, 2017.

[36] Björn Feldkord and Friedhelm Meyer auf der Heide. Online facility location with mobile facilities. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 373–381, 2018.

[37] Björn Feldkord and Friedhelm Meyer auf der Heide. The mobile server problem. *TOPC*, 6(2):14:1–14:17, 2019.

[38] Amos Fiat, Richard M. Karp, Michael Luby, Lyle A. McGeoch, Daniel Dominic Sleator, and Neal E. Young. Competitive paging algorithms. *J. Algorithms*, 12(4):685–699, 1991.

[39] Dimitris Fotakis. Incremental algorithms for facility location and *k*-median. *Theor. Comput. Sci.*, 361(2-3):275–313, 2006.

[40] Dimitris Fotakis. A primal-dual algorithm for online non-uniform facility location. *J. Discrete Algorithms*, 5(1):141–148, 2007.

[41] Dimitris Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, 2008.

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

[43] Dimitris Fotakis, Loukas Kavouras, Panagiotis Kostopanagiotis, Philip Lazos, Stratis Skoulakis, and Nikolas Zarifis. Reallocating multiple facilities on the line. *CoRR*, abs/1905.12379, 2019.

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

[45] Marcel Geromel. *Mobile Facility Leasing*. Bachelor's thesis, Paderborn University, 2018.

[46] Abdolhamid Ghodselahi and Fabian Kuhn. Serving online requests with mobile servers. In *Proceedings of the 26th International Symposium on Algorithms and Computation (ISAAC)*, pages 740–751, 2015.

[47] Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *J. Algorithms*, 31(1):228–248, 1999.

[48] Sudipto Guha and Andrew McGregor. Approximate quantiles and the order of the stream. In *Proceedings of the 25th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 273–279, 2006.

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

[50] Klaus Jansen, Marten Maack, and Malin Rau. Approximation schemes for machine scheduling with resource (in-)dependent processing times. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1526–1542, 2016.

[51] Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. *J. ACM*, 47(4):617–643, 2000.

[52] Amanj Khorramian and Akira Matsubayashi. Uniform page migration problem in euclidean space. *Algorithms*, 9(3):57, 2016.

[53] Peter Kling, Alexander Mäcker, Sören Riechers, and Alexander Skopalik. Sharing is caring: Multiprocessor scheduling with a sharable resource. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 123–132, 2017.

[54] Peter Kling, Friedhelm Meyer auf der Heide, and Peter Pietrzyk. An algorithm for online facility leasing. In *Proceedings of the 19th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 61–72, 2012.

[55] Elias Koutsoupias. Weak adversaries for the k-server problem. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 444–449, 1999.

[56] Elias Koutsoupias and Christos H. Papadimitriou. On the k-server conjecture. *J. ACM*, 42(5):971–983, 1995.

[57] Harry Lang. Online facility location against a t-bounded adversary. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1002–1014, 2018.

[58] James R. Lee. Fusible HSTs and the randomized k-server conjecture. In *Proceedings of the 59th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 438–449, 2018.

[59] Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Inf. Comput.*, 222:45–58, 2013.

[60] Mark S. Manasse, Lyle A. McGeoch, and Daniel Dominic Sleator. Competitive algorithms for server problems. *J. Algorithms*, 11(2):208–230, 1990.

[61] Akira Matsubayashi. A 3+omega(1) lower bound for page migration. In *Proceedings of the 3rd International Symposium on Computing and Networking (CANDAR)*, pages 314–320, 2015.

[62] Akira Matsubayashi. Asymptotically optimal online page migration on three points. *Algorithmica*, 71(4):1035–1064, 2015.

[63] Adam Meyerson. Online facility location. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 426–431, 2001.

[64] Adam Meyerson. The parking permit problem. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 274–284, 2005.

[65] Chandrashekhar Nagarajan and David P. Williamson. Offline and online facility leasing. *Discrete Optimization*, 10(4):361–370, 2013.

[66] Jianli Pan and James McElhannon. Future edge cloud and edge computing for internet of things applications. *IEEE Internet of Things Journal*, 5(1):439–449, 2018.

[67] R. Ravi and Amitabh Sinha. Approximation algorithms for multicommodity facility location problems. *SIAM J. Discrete Math.*, 24(2):538–551, 2010.

[68] Tomislav Rudec, Alfonzo Baumgartner, and Robert Manger. A fast work function algorithm for solving the k-server problem. *CEJOR*, 21(1):187–205, 2013.

[69] Tomislav Rudec and Robert Manger. A fast approximate implementation of the work function algorithm for solving the \(k\) -server problem. *CEJOR*, 23(3):699–722, 2015.

[70] Guido Schäfer and Naveen Sivadasan. Topology matters: Smoothed competitiveness of metrical task systems. *Theor. Comput. Sci.*, 341(1-3):216–246, 2005.

[71] Weisong Shi and Schahram Dustdar. The promise of edge computing. *IEEE Computer*, 49(5):78–81, 2016.

[72] David B. Shmoys. Approximation algorithms for facility location problems. In *Proceedings of the 3rd International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, pages 27–33, 2000.

[73] Daniel Dominic Sleator and Robert Endre Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, 1985.

[74] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463, 2004.

[75] Jeffery R. Westbrook. Randomized algorithms for multiprocessor page migration. *SIAM J. Comput.*, 23(5):951–965, 1994.

[76] Yinfeng Xu, Hongmei Li, Changzheng He, and Li Luo. The online k-server problem with max-distance objective. *J. Comb. Optim.*, 29(4):836–846, 2015.

[77] Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 222–227, 1977.

[78] Shanhe Yi, Cheng Li, and Qun Li. A survey of fog computing: Concepts, applications and issues. In *Proceedings of the 2015 Workshop on Mobile Big Data, Mobidata@MobiHoc*, pages 37–42, 2015.