Optimal Algorithms for Page Migration in Dynamic Networks

Marcin Bienkowski [a,*], Jaroslaw Byrka [b],
Miroslaw Korzeniowski [c], Friedhelm Meyer auf der Heide [d]

[a] *Institute of Computer Science, University of Wroclaw, ul. Joliot-Curie 15, 50-383 Wroclaw, Poland*

[b] *CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands, and Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands*

[c] *Institute of Mathematics and Computer Science, Wroclaw University of Technology, ul. Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland*

[d] *Heinz Nixdorf Institute and Computer Science Department, University of Paderborn, Fürstenallee 11, Paderborn, Germany*

Abstract

We present an extension of a classical data management subproblem, the page migration. The problem is investigated in dynamic networks, where costs of communication between different nodes may change with time. We construct asymptotically optimal online algorithms for this problem, both in deterministic and randomized scenarios.

Key words: online algorithms, randomized algorithms, page migration, data management, dynamic networks

1 Introduction

One of the most crucial services used in every distributed program is a transparent access to variables, databases, memory pages, or files, which are shared by the program instances running at nodes of the network. An implementation of the variable sharing is essential to the performance of such distributed applications. However, the traditional approach of storing the shared data in

* Corresponding author,
email: mbi@ii.uni.wroc.pl, tel.: +48 71 375 7838, fax: +48 71 375 7801.

one or a few central repositories does not scale up well with the increase of the network size and is therefore inherently inefficient. One of the most straightforward, yet imprecise solution, is to abandon these central storage systems and use local memories of the nodes to store the shared objects.

In this paper, we investigate data management strategies that try to exploit *topological locality*, i.e., try to migrate the shared data in the network in such a way that a node accessing a data item finds it "nearby" in the network. Accesses to the shared data can be modeled as an online problem. In this paper, we deal with the classical, basic subproblem called *Page Migration*.

In contrast to previous works on data management in networks, we focus on the page migration in a *dynamic* setting. We assume that the network is no longer static, but is subject to change, and the costs of communication between nodes may change with time. Such a situation is typical in mobile ad-hoc networks, but occurs also in large distributed systems, which are used concurrently by many applications and users. Thus, we have to deal with two sources of online events, namely the requests from nodes to data items and the changes in the network.

1.1 Static Networks

In this subsection, we describe the original data management and page migration problems. They are defined in a static network, i.e., the network in which costs of communication between nodes do not change in time.

In many applications, access patterns to a shared object change frequently. This is common, for example, in parallel pipelined data processing, where the set of processors accessing shared variables changes in the runtime. In these cases, any static placement of the object copies is inefficient. Moreover, the knowledge of the future accesses to the objects is in reality either partial or completely non-existing, which renders any solution based on static placement infeasible. Instead, a data management strategy should migrate the copies to further exploit the locality of accesses. This poses an algorithmic problem, central to this paper.

> *Without knowledge of the future accesses to the shared objects, decide, whether it is worth to change the positions of their copies.*

To keep the bookkeeping overhead small, it is often required that only *one copy* of each object is stored in the system. Additionally, shared objects are usually bigger than the part of their data that is being accessed at one time. Usually, processors want to read or change only one single unit of data from the object, or one record from a database. On the other hand, the data of one

object should be kept in one place to reduce the maintenance overhead. This leads to a so-called *non-uniform model*, where migrating or copying the whole object is much more expensive than accessing one unit of data from it.

This traditional paradigm, called *Page Migration* (PM) was introduced by Black and Sleator [16]. It models an underlying network as a connected, undirected graph, where each edge e has an associated cost $c(e)$ of sending one unit of data over the corresponding communication channel. In case of wired networks, this cost might represent the load induced by sending data through this communication link. The cost of sending one unit of data between two nodes v_a and v_b is defined as the sum of costs of edges on the cheapest path between v_a and v_b. There is only *one copy* of *one single object* of size D, which is further called a *(memory) page*, stored initially at one fixed node in the network.

A PM problem instance is a sequence of nodes $(\sigma_t)_t$, which want to access (read or write) one unit of data from the page. In one step t, one node σ_t issues a request to the node holding the page and appropriate data is sent back. For such a request, an algorithm for PM is charged a cost of sending one unit of data between σ_t and the node holding the page. At the end of each time step, the algorithm may move the page to an arbitrary node. Such a transaction incurs a cost which is D times greater than the cost of sending one unit of data between these two nodes.

The goal is to compute a schedule of page movements which minimizes the total cost. Computing an optimal schedule *offline*, i.e., on the basis of the *whole* input sequence $\mathcal{I} = (\sigma_t)_t$, is an easy task, which can be performed in polynomial time. Thus, the main effort was placed on constructing online algorithms, i.e., ones which have to make decision in time step t solely on the part of the input up to step t.

1.2 Dynamic Networks

In the past, an application executed on a parallel machine was running in a virtually static and invariable environment and one could safely assume that the interconnecting network is predictable and reliable. Such assumptions, which substantially reduced the complexity of the basic services design, ceased to hold when applications started to run in open and unknown networks.

First of all, networks are prone to link failures or bandwidth shortages. Second, other applications running in the network might behave completely unpredictably or even antagonistically, creating high loads on particular links, e.g., by flooding them with messages. Third, if the network consists of mobile stations, its topology may be changed due to nodes mobility.

In our considerations we do not take into account the dynamics induced by nodes joining and leaving the network. In fact, a model where nodes may become active and inactive was already investigated by Awerbuch, Bartal, and Fiat [4] in the context of a data management subproblem, a file allocation.

Basic services for mobile wireless networks and dynamically changing wired networks are a relatively new research subject. Some effort was placed on creating algorithms for topology control and routing in wireless networks (see e.g. [27]) or routing algorithms in faulty wired networks (see e.g. [28]). In comparison, basic services related to data management problems in dynamically changing networks are still in their infancy. Till recently, no theoretical analysis or even experimental evaluation was present in this area, which might have been influenced by the fact that no reasonable model of network changes was proposed. In particular, any model similar to the one described in [28], where the adversary can destroy links between nodes, would be too strong for any data management scheme. This follows from the observation that it is relatively easy to construct a sequence of accesses to a shared object, which eventually forces any competitive (even randomized) algorithm to move all the copies of this object to one node. Afterwards, the link failures may disconnect this node from the rest of the network, leaving the algorithm no chance to access or migrate the data in the future.

Hence, for theoretical modeling of network dynamics, we assume that an adversary may modify the costs of point-to-point communication arbitrarily, as long as the pace of these changes is restricted by, say, an additive constant per step. Intuitively, this gives the data management algorithm time to react to the changes. The model of slow changes in the communication costs, formally defined in the next section, tries also to capture slow changes in bandwidth available in wired networks, which are inherently induced by other programs running or users using (not abusing) the network.

1.3 Our Model

To model the Page Migration problem in dynamic networks we make the following assumptions. The network consists of n mobile nodes (processors) labeled $v_0, v_1, \ldots, v_{n-1}$. For succinctness, we define $[n] = \{0, 1, \ldots, n-1\}$. These nodes are placed in a metric space $(\mathcal{X}, d)$, where the distance between any pair of points from $\mathcal{X}$ is given by the metric d. The metric space is chosen in any fashion by the adversary. In particular, we make no assumptions about the finiteness of the space or about its diameter, i.e., the maximum distance between a pair of points from $\mathcal{X}$.

Time is discrete and slotted into time steps $t = 0, 1, 2, \ldots$. To model dynamics,

we assume that the position of each node is a function of t, i.e., $p_t(v)$ denotes the position of v in time step t. As a natural consequence, the distance between a pair of nodes may also change with time. The distance between any pair of nodes v_a and v_b in time step t is denoted by

$$d_t(v_a, v_b) := d(p_t(v_a), p_t(v_b)) \ . \tag{1}$$

Note that such a distance can be equal to zero in two different cases. The first one occurs if v_a and v_b are different nodes occupying the same position in $\mathcal{X}$. The second one occurs when $a = b$, in which case we are dealing with a single node (and we write $v_a \equiv v_b$).

A tuple describing the positions of all the nodes in time step t is called *configuration* in step t, and is denoted by $\mathcal{C}_t$. A configuration sequence $(\mathcal{C}_t)_{t=0}^{T}$ contains the configurations in the first $T + 1$ time steps, beginning with the *initial configuration* $\mathcal{C}_0$. The actual representation of these positions in complicated metric spaces is not relevant for us. The only requirement is that the distances between any pair of nodes in time t are computable on the basis of $\mathcal{C}_t$.

The changes in nodes' positions over time are arbitrary, as long as the nodes move with a *bounded speed*, as mentioned in the previous section. Formally, for any node v_i, its positions in two consecutive time steps t and $t+1$ cannot be too far apart, i.e.,

$$d(p_t(v_i), p_{t+1}(v_i)) \leq \delta \ , \tag{2}$$

for some fixed δ. An adversarial entity creating sequence of configurations is called *δ-restricted*, if it obeys the inequality above.

Any two nodes are able to communicate directly with each other. Essentially, the communication cost is proportional to the distance between these two nodes, plus a constant overhead. This overhead represents the startup cost for establishing connection. Precisely, the cost of sending a unit of data from node v_a to v_b at time step t is defined by a *cost function* $c_t(v_a, v_b)$, defined as

$$c_t(v_a, v_b) := d_t(v_a, v_b) + 1 \ , \tag{3}$$

if v_a and v_b are different nodes. The communication within one node is free, i.e., $c_t(v_a, v_a) = 0$.

Naturally, the changes in the network (described by the $(\mathcal{C}_t)_{t=0}^{T}$ sequence) do not constitute a problem of its own. According to the described Page Migration model, a copy of memory page of size D is stored at one of the network's nodes, initially at v_0. In each time step $t \geq 1$, exactly one node, denoted σ_t, tries to access one unit of data from the page. Since the model assumes that there is only one copy of the object stored in the system, there is no need of making

distinction between read and write accesses. We refer to them as *requests* and we call σ_t the *requesting node*. The requests create the sequence $(\sigma_t)_{t=1}^{T}$, complementary to the configuration sequence $(\mathcal{C}_t)_{t=0}^{T}$. Note that nodes issue requests from the first step; the initial configuration in time step 0 is introduced to simplify the notation only.

In each step, an algorithm for the Page Migration in dynamic networks has to serve the request, and then to decide, whether it wants to migrate the page to some other node. Precisely, for any algorithm ALG the following stages happen in time step $t \geq 1$.

(1) The positions of the nodes in the current step are defined by $\mathcal{C}_t$.
(2) A node σ_t wants to access one single unit of data from the page. It sends a write or a read request to $P_{\text{ALG}}(t)$, the node holding ALG's page in the current step.
(3) ALG serves this request, i.e., it sends a confirmation in case of write, or a requested unit of data in case of read. This transaction incurs a cost $c_t(P_{\text{ALG}}(t), \sigma_t)$.
(4) ALG optionally moves the page to another node of its choice, called a *jump candidate*. A movement to $P'_{\text{ALG}}(t)$ incurs a cost $D \cdot c_t(P_{\text{ALG}}(t), P'_{\text{ALG}}(t))$.

Sometimes we abuse the notation, writing that ALG is at node v_i when ALG has its page at this node. Analogously, we write that ALG moves or jumps to v_j, when it moves its page there. In fact, the only part which ALG may influence is choosing a new node $P'_{\text{ALG}}(t)$ in the fourth stage. The problem, to which we further refer as *Dynamic Page Migration* (DPM), is to construct a schedule of page movements to minimize the total cost of communication for a given pair of sequences $(\mathcal{C}_t)_t, (\sigma_t)_t$. We will usually abbreviate this notion to $(\mathcal{C}_t, \sigma_t)_t$.

Before we proceed with the considerations on the complexity of the DPM problem, we point out that the DPM model is more general than Page Migration itself. If the network is static, i.e., $\mathcal{C}_t = \mathcal{C}_{t-1}$ for all $t \geq 1$, and we neglect the constant overhead in the cost function definition, then DPM is capable of modeling any situation, in which the cost function satisfies the triangle inequality. Note that even if it is not the case, the page migration algorithm chooses the shortest paths instead of direct connections, and thus the triangle inequality is fulfilled.

It is also straightforward, that the constant overhead may be neglected if the minimum cost of communication in the network is large. For the Page Migration problem in a static network we may assume this property, since without loss of generality, the costs defined by any instance of the problem might be scaled up by any factor.

Like in the Page Migration case, the problem of minimizing the total cost is easy if both $(\mathcal{C}_t)_t$ and $(\sigma_t)_t$ are given in *offline* setting, i.e., if an algorithm may read the whole input beforehand. Using a straightforward dynamic programming approach, it is possible to construct an optimal schedule of page movements for any instance of the DPM problem consisting of T steps, using $\mathcal{O}(T \cdot n^2)$ operations and $\mathcal{O}(T \cdot n)$ additional space.

However, as mentioned earlier, DPM has to be primarily solved in an *online* setting, where an algorithm must make its decisions (where to move the page) in step t, exclusively on the basis of the sequence $\mathcal{C}_0, \mathcal{C}_1, \sigma_1, \mathcal{C}_2, \sigma_2, \ldots, \mathcal{C}_t, \sigma_t$. To measure the performance of online strategies for the DPM problem, we use the competitive analysis (see, e.g., [29,17]). This kind of evaluation, primarily introduced by Sleator and Tarjan [29], compares the cost of an online algorithm to the cost of the optimal offline strategy. In the following, we assume that an optimal algorithm is denoted by OPT, and for any algorithm ALG, $C_{\mathrm{ALG}}(\mathcal{I})$ denotes the cost of this algorithm on input sequence $\mathcal{I} = (\mathcal{C}_t, \sigma_t)_t$.

An online deterministic algorithm ALG is $\mathcal{R}$-*competitive* if there exists a constant α, such that for any input sequence $\mathcal{I}$, it holds that

$$C_{\mathrm{ALG}}(\mathcal{I}) \;\leq\; \mathcal{R} \cdot C_{\mathrm{OPT}}(\mathcal{I}) + \alpha \;. \tag{4}$$

For a randomized algorithm ALG we replace its cost in the definition above by its expectation $\mathbf{E}[C_{\mathrm{ALG}}(\mathcal{I})]$. The expected value is taken over all possible random choices made by ALG.

However, in the randomized case, the power given to the adversary has to be further specified. Following Ben-David et al. [8], we distinguish between three types of adversaries: *oblivious*, *adaptive-online* and *adaptive-offline*. An *oblivious* adversary has to construct the whole input sequence in advance, not taking into account the random bits used by an algorithm. The other two types are adaptive ones; they may decide about the next requests upon seeing the algorithm's current page position. Since they are dependent on the algorithm's random choices, we have to replace $C_{\mathrm{OPT}}(\mathcal{I})$ by its expectation (taken over these random choices). These two adaptive types differ, however, in the way they construct an optimal solution, which is later compared with the solution of ALG. An *adaptive-online* adversary must provide an *answering entity*, which creates an "optimal" solution in parallel to ALG. This solution may not be changed afterwards. An *adaptive-offline* adversary may construct an optimal solution at the end, knowing the whole input sequence.

The power of these adversaries can be related as shown in [8]. Let $\mathcal{R}_{\mathrm{OBL}}$, $\mathcal{R}_{\mathrm{AD-ONL}}$, $\mathcal{R}_{\mathrm{AD-OFF}}$ be the best competitive ratios for randomized algorithms

against oblivious, adaptive-online, and adaptive-offline adversaries, respectively. Let $\mathcal{R}_{\text{DET}}$ be the best possible ratio for deterministic algorithm. Then

$$\mathcal{R}_{\text{OBL}} \leq \mathcal{R}_{\text{AD-ONL}} \leq \mathcal{R}_{\text{AD-OFF}} = \mathcal{R}_{\text{DET}} \ . \tag{5}$$

This relation implies that the randomization does not help against adaptive-offline adversaries. Hence in this paper, we focus on the other three types of adversaries.

1.5 Our Contribution and Outline of the Paper

The main results of this paper are online algorithms which achieve asymptotically optimal competitive ratios against constant-restricted adversaries. The respective ratios are gathered in the following table.

Algorithm	**Competitive ratio**
Deterministic	$\Theta\left(\min\left\{n \cdot \sqrt{D}, D\right\}\right)$
Randomized against adaptive-online adversary	$\Theta\left(\min\left\{n \cdot \sqrt{D}, D\right\}\right)$
Randomized against oblivious adversary	$\Theta\left(\min\left\{\sqrt{D \cdot \log n}, D\right\}\right)$

To prove these results we take the following approach. In Section 2 we present a pair of deterministic algorithms: $\mathcal{O}(n \cdot \sqrt{D})$-competitive algorithm Mark and $O(D)$-competitive algorithm Jump, which combined give an $\mathcal{O}(\min\{n \cdot \sqrt{D}, D\})$-competitive deterministic algorithm. In Section 3.2, we show that a lower bound of $\Omega(\min\{n \cdot \sqrt{D}, D\})$ holds for randomized algorithms fighting against adaptive-online adversaries. By relation (5) this proves the first two entries from the table above. Later in Section 2, we show that a randomization of the Mark algorithm yields an algorithm Ebm, which is $\mathcal{O}(\sqrt{D \cdot \log n})$-competitive against an oblivious adversary. After combining it with the Jump algorithm, we get an upper bound of $\mathcal{O}(\min\{\sqrt{D \cdot \log n}, D\})$ against an oblivious adversary. A matching lower bound is presented in Section 3.3. In our proofs we do not strive for minimizing the constants, but rather at the simplicity of the presentation.

One of the natural extensions is to consider adversaries which are δ-restricted, where δ is no longer a constant, but a parameter. We give partial answers to this problem in Section 4.

Finally, in Section 5, we show that it is not possible to extend our results to the model where object replication is allowed. Specifically, we show that the competitive ratio of any file allocation algorithm in dynamic network is unbounded.

We note that for metric spaces where the maximum distance between two points is at most λ, if we take an $\mathcal{O}(1)$-competitive algorithm for "normal" page migration, then its competitive ratio is at most $\mathcal{O}(\lambda)$. Indeed, the lower bounds stated in this paper do not hold for such limited metric spaces and we show them only for larger spaces. However, it is possible to incorporate the λ term into the competitive ratios presented in the table above; the respective ratios become $\Theta(\min\{n \cdot \sqrt{D}, D, \lambda\})$ and $\Theta(\min\{\sqrt{D \cdot \log n}, D, \lambda\})$. Since the proofs in this case are much longer and more technical, we refrain to present them in the journal paper and we refer the reader to [10].

1.6 Related Work

To our best knowledge, the only work that exists in the area of data management in dynamically changing networks is the paper by Awerbuch, Bartal, and Fiat on distributed paging [4]. However, they consider a setting in which nodes may appear and disappear, which differs much from our model. In particular, their results are inapplicable in our scenario.

On the other hand, the area of data management in static networks has been successfully explored in the past years by numerous researchers. Below, we briefly state some of their results.

1.6.1 Page Migration

The Page Migration problem was thoroughly investigated for different types of adversaries. For a gentle introduction to the algorithms mentioned here, we refer the reader to the survey by Bartal [5].

First randomized solutions presented by Westbrook [30] were a memoryless algorithm which was 3-competitive against an adaptive-online adversary and a phase-based algorithm whose competitive ratio against an oblivious adversary tends to 2.618 as D goes to infinity. The former result was proven to be tight by Bartal, Fiat, and Rabani [7,5]. The lower bound construction was a slight modification of the analogous lower bound for deterministic algorithms by Black and Sleator [16]. On the other hand, the exact competitive ratio against an oblivious adversary is not a completely settled issue. The currently best known lower bound, $2 + \frac{1}{2D}$, is due to Chrobak, Larmore, Reingold, and Westbrook [19]. It is matched only for certain topologies, like trees or uniform networks (see [19] and [26], respectively).

The first deterministic, phase-based, 7-competitive algorithm Move-To-Min was given by Awerbuch, Bartal, and Fiat [2]. The result was subsequently improved by the Move-To-Local-Min algorithm [6] attaining competitive

ratio of 4.086. On the other hand, [19] showed a network with a lower bound of approximately 3.148.

1.6.2 Data Management

There exist many extensions of Page Migration that allow more flexible data management in networks. One of the possible generalizations of PM is allowing more than one copy of an object to exist in the network. This poses new interesting algorithmic questions which have to be resolved by a data management scheme, i.e., how many copies of shared objects should be created and which accesses to shared objects should be handled by which copies. A basic version of this problem (where only one shared object is present in the system), called *file allocation*, was first examined in the framework of competitive analysis by Bartal, Fiat, and Rabani [7]. They presented a randomized strategy that achieves an asymptotically optimal competitive ratio of $\mathcal{O}(\log n)$ against an adaptive-online adversary. Additionally, they showed how to get rid of the central control (which is useful for example for locating the nearest copy of the object) and created $\mathcal{O}(\log^4 n)$-competitive algorithm, which works in a distributed fashion. Awerbuch, Bartal, and Fiat [2] showed that the randomization is not crucial, and constructed deterministic algorithms (centralized and distributed ones) for file allocation problem, attaining asymptotically the same ratios.

For uniform topologies, Bartal, Fiat, and Rabani [7] showed an optimal deterministic 3-competitive algorithm. Lund, Reingold, Westbrook, and Yan [26] gave a 3-competitive algorithm for trees, which is based on *work functions* technique.

If the shared data is read-only, then the file allocation becomes a *page replication* problem. It was also introduced by Black and Sleator [16]. In contrary to the page migration, in general networks one cannot hope for a competitive ratio better than $\Omega(\log n)$. Therefore, the research on page replication conducted by Albers and Koga [1,25] and by Fleisher, Głazek, and Seiden [20–23] concentrated on particular topologies like trees, uniform networks, and rings. For all these topologies $\mathcal{O}(1)$-competitive deterministic and randomized algorithms were given; the ratios for trees and uniform networks are optimal.

If multiple objects are present in the network and the local memory capacity at nodes is limited, then running a file allocation scheme independently for each single object in the network might encounter some problems. Above all, it is not possible to copy an object into a node's memory if it is already full. Possibly, some other object copies have to be dropped, which induces problems if they were the last copies present in the network. This leads to a so called *distributed paging* problem [3,4,7], where file allocation solutions

have to be combined with schemes known from *uni-processor paging* (see for example [29]).

1.6.3 Relaxed Models for Page Migration in Dynamic Networks

The competitive ratios of the best possible algorithms for DPM problem are relatively large, even against the weakest, oblivious adversaries. The poor performance of algorithms is caused by the fact that the part of adversary which changes the network and the part which gives the request patterns may combine and synchronize their efforts. It was therefore proposed that the DPM problem could be analyzed in a scenario where one of these parts is replaced by a stochastic process. This leads to the cases [9,13], in which the competitive ratios can be greatly decreased.

1.7 Bibliographical Notes

Part of the results presented in this paper have been published previously in a preliminary form. The DPM problem was defined in [14]. The deterministic algorithm MARK was first presented in [12,15] and its randomized counterpart EBM in [11]. A lower bound for adaptive adversaries was presented in [14], and a weaker, non-optimal version of the lower bound for oblivious adversaries was given in [12].

2 Algorithms

In this section, we show upper bounds on the competitive ratios: $\mathcal{O}(\min\{n \cdot \sqrt{D}, D\})$ for deterministic algorithms and $\mathcal{O}(\min\{\sqrt{D \cdot \log n}, D\})$ for randomized ones against an oblivious adversary.

To achieve this, first we present a trivial deterministic $\mathcal{O}(D)$-competitive algorithm JUMP. Later we present a whole class of marking based algorithms and demonstrate their properties. Two specific instances from this class are the deterministic algorithm MARK and the randomized algorithm EBM, attaining competitive ratios of $\mathcal{O}(n \cdot \sqrt{D})$ and $\mathcal{O}(\sqrt{D \cdot \log n})$, respectively.

Therefore, if we consider deterministic algorithms, we may — since an algorithm knows D and n — easily achieve the best of both worlds, choosing either JUMP or MARK. The same holds for randomized algorithms where choosing at the beginning between JUMP and EBM guarantees the competitive ratio of $\mathcal{O}(\min\{\sqrt{D \cdot \log n}, D\})$.

2.1 Preliminaries

Since we want all the results to hold for any δ-restricted adversary for constant δ, we show first that all constant-restricted adversaries are up to a constant factor equivalent.

Lemma 1 (Reduction Lemma) *Assume that there exists a (randomized) algorithm A which is k-competitive against an a-restricted adversary. Then A is k-competitive against a b-restricted adversary for $b \leq a$. Additionally, for any $b \geq a$ there exists a (randomized) algorithm B, which is $\frac{b}{a} \cdot k$-competitive against any b-restricted adversary.*

PROOF. If $b \leq a$, then A is k-competitive against a b-restricted adversary, since it was k-competitive against a stronger, a-restricted adversary.

If $b \geq a$, let $(\mathcal{C}_t, \sigma_t)_t$ be any input sequence. Let $\mathcal{C}'_t$ denote the configuration $\mathcal{C}_t$ with all the original distances divided by b/a. Clearly, if $(\mathcal{C}_t)$ was created by a b-restricted adversary, then sequence $(\mathcal{C}'_t)_t$ might be created by an a-restricted adversary. Algorithm B simulates the behavior of A on the sequence $(\mathcal{C}'_t, \sigma_t)_t$, and repeats A's choices on $(\mathcal{C}_t, \sigma_t)_t$. We obtain

$$\begin{aligned} C_B((\mathcal{C}_t, \sigma_t)_t) &\leq \frac{b}{a} \cdot C_A((\mathcal{C}'_t, \sigma_t)_t) \\ &\leq \frac{b}{a} \cdot k \cdot C_{\mathrm{OPT}}((\mathcal{C}'_t, \sigma_t)_t) \\ &\leq \frac{b}{a} \cdot k \cdot C_{\mathrm{OPT}}((\mathcal{C}_t, \sigma_t)_t) \ , \end{aligned}$$

and thus B is $\frac{b}{a} \cdot k$-competitive. For proving the lemma for randomized algorithms, we just replace the algorithm's cost by its expected value. □

Throughout the remaining part of the paper, for constructing algorithms we consider $\frac{1}{2}$-restricted adversaries, since this assures that the distance between any pair of nodes can change only by 1 per time step. For showing lower bounds we consider 1-restricted adversaries.

2.2 Algorithm JUMP

Let JUMP be a deterministic memoryless algorithm which upon receiving a request from the node σ_t serves this request and jumps to σ_t. We prove the following theorem.

Theorem 2 JUMP *is $\mathcal{O}(D)$-competitive for the DPM problem.*

PROOF. We take any input sequence $\mathcal{I}$. Recall that for any algorithm A and any step t, $P_A(t)$ denotes the node holding the page of A at the beginning of step t. By $C_A(t)$ we denote the cost of A in step t.

Recall that both JUMP and the optimal offline algorithm OPT start with their pages at the same node, i.e., $P_{\text{JUMP}}(1) \equiv P_{\text{OPT}}(1)$. First, we observe that in the first step of $\mathcal{I}$, JUMP pays for serving the request at σ_1 and for moving to σ_1, whereas OPT pays at least for serving the request at σ_1. Therefore,

$$C_{\text{JUMP}}(1) = (1+D) \cdot c_t(P_{\text{OPT}}(1), \sigma_1) \leq (1+D) \cdot C_{\text{OPT}}(1) \ . \tag{6}$$

Next, we show that the following relation holds for any step $t > 1$.

$$C_{\text{JUMP}}(t) = \mathcal{O}(D) \cdot (C_{\text{OPT}}(t-1) + C_{\text{OPT}}(t)) \tag{7}$$

If $\sigma_{t-1} \equiv \sigma_t$, then in step t, JUMP is already at node σ_t and its cost is 0. In this case, (7) holds trivially. Otherwise, the costs are as follows:

$$\begin{aligned} C_{\text{JUMP}}(t) &= (1+D) \cdot c_t(\sigma_{t-1}, \sigma_t) \ , \\ C_{\text{OPT}}(t-1) &= c_{t-1}(P_{\text{OPT}}(t-1), \sigma_{t-1}) + D \cdot c_{t-1}(P_{\text{OPT}}(t-1), P_{\text{OPT}}(t)) \ , \\ C_{\text{OPT}}(t) &\geq c_t(P_{\text{OPT}}(t), \sigma_t) \ . \end{aligned}$$

It is easy to observe that the function c_t satisfies the triangle inequality and for any two (not necessarily different) nodes v_a and v_b, it holds that $c_t(v_a, v_b) \leq 2 \cdot c_{t-1}(v_a, v_b)$. Hence,

$$\begin{aligned} C_{\text{OPT}}(t-1) + C_{\text{OPT}}(t) &\geq c_{t-1}(P_{\text{OPT}}(t), \sigma_{t-1}) + c_t(P_{\text{OPT}}(t), \sigma_t) \\ &\geq \frac{1}{2} \cdot c_t(\sigma_{t-1}, \sigma_t) \ , \end{aligned}$$

and (7) follows.

By summing up relations (6) and (7) for all steps t, we get the lemma. □

We note that the analysis is up to a constant factor tight. For example, if the requests are placed alternately at nodes v_0 and v_1, the cost of JUMP is $\Omega(D)$ times worse than the cost of the optimal algorithm.

2.2.1 Last Request Based Algorithms

One may think that by cleverly modifying the JUMP algorithm, i.e., by not moving in each step but every, say, D steps, one may reduce the competitive ratio to $o(D)$. The following argument shows that it is not possible.

We consider a class of algorithms which in any step may decide whether to move the page, but they may move only to the node which issued a request in this step. This class we call *last request based*. All previous randomized algorithms for Page Migration (presented for example in [19,26,30]) as well as our JUMP algorithm fall into this category. Surprisingly, no such deterministic algorithm can have a competitive ratio $o(D)$. The same argument works also for randomized, last request based algorithms against an adaptive-online adversary.

Lemma 3 *Consider any deterministic, last request based, $\mathcal{R}$-competitive algorithm* ALG. *Then $\mathcal{R} = \Omega(D)$.*

PROOF. We consider a three-node network and a 1-restricted adversary. Initially, all nodes occupy the same place in the space. Without loss of generality, we may assume that ALG starts in v_0. The adversary chooses to keep OPT all the time in v_2.

We divide the time into phases. In the first phase, requests are given at v_1, and v_0 is moved apart with the maximum possible speed, i.e., in step t of this phase, the distance between v_0 and v_1 (and between v_0 and v_2) is t. At some point ALG decides to jump. Note that if ALG never jumps, then its competitive ratio is unbounded. As a jump candidate it can only choose v_1. Let X denote the distance between v_0 and v_1 at the moment of jump. In the next X steps, requests are still given at v_1, and nodes are contracted, i.e., v_0 is moved to v_1 and v_2 with the maximum possible speed, till it reaches the initial configuration. After contracting, the phase ends, having lasted $2 \cdot X$ steps. In this phase OPT pays $2 \cdot X$ and ALG pays $D \cdot X$ just for moving the page in the middle of the phase.

We can continue this process for any number of phases. The even phases are symmetric to the first one, i.e., the roles of v_0 and v_1 are reversed and the odd phases follow the same rules as the first one. Thus, we conclude that the competitive ratio of ALG is at least $\mathcal{R} \geq D/2 = \Omega(D)$. □

2.3 Marking

In this section, we present a marking scheme and a whole class of marking-based algorithms. First, we prove some common properties of these algorithms. Later, we pick two algorithms from this class: a deterministic algorithm MARK and a randomized algorithm EBM and we compute their competitive ratios.

For the needs of this subsection, we introduce the following notation. By a *subsequence* we understand any uninterrupted time interval of the input sequence.

To simplify the notation, we also treat subsequences as sets of the corresponding time steps. For any subsequence $\mathcal{S}$ and any algorithm Alg, by $C_{\text{ALG}}(\mathcal{S})$ we denote the cost (of serving requests within $\mathcal{S}$ and moving the page) incurred by $\mathcal{S}$ on Alg. In particular, by Opt we denote the optimal offline algorithm, and by $C_{\text{OPT}}(\mathcal{S})$ its cost in $\mathcal{S}$.

2.3.1 Gravity Centers

Keeping in mind the results for the last request based algorithms, we want to create a deterministic algorithm achieving a competitive ratio of $o(D)$. The class of our marking algorithms is partially inspired by the Move-To-Min algorithm by Awerbuch, Bartal and Fiat [2]. A brief idea of this 7-competitive algorithm is as follows. It divides the input sequence into chunks of length D, in each chunk serves all the requests, and moves only at the end of chunks to a so-called *gravity center*. A gravity center is a node, which would be the best place for the page in the last chunk, i.e., it minimizes the sum of distances to all the requests issued.

Since in our setting the distances can change with time, we have to be careful with defining gravity centers. Consider any subsequence $\mathcal{S}$ of length ℓ steps. We number these steps from 1 to ℓ. Let σ_i be the node which issues a request in the i-th step of $\mathcal{S}$ and $d_i(\cdot)$, $c_i(\cdot)$ be the distance and cost functions, respectively, in the i-th step.

Definition 4 *A gravity center for a subsequence $\mathcal{S}$ of length ℓ is a vertex v, which minimizes the sum $\sum_{i=1}^{\ell} d_\ell(v, \sigma_i)$. We denote it by $\mathcal{G}_\mathcal{S}$. If there is more than one such vertex, then we break ties arbitrarily.*

2.3.2 Marking Scheme

Marking-based algorithms take the chunk-based approach after Move-To-Min. The chosen chunk's length must be large enough to allow amortization of the page movements against the cost incurred by serving requests, and short enough to make the network changes negligible. In this whole section, $K \leq 2 \cdot \sqrt{D}$ denotes the length of the chunk. K is a parameter, which takes different values for different algorithms.

It can be proven that considering only gravity centers as potential jump candidates does not differ much from considering a class of last request based algorithms. Therefore, the algorithm has to consider moving not only to a gravity center, but also to the nodes from a surrounding of this center. We show a completely different approach which — quite surprisingly — also leads to this goal.

To specify this approach, we have to consider the question when it is worth to change the position of the algorithm's page. We already mentioned that we do not move the page inside chunks, i.e., not more frequently than once per K steps. Additionally, it usually makes no sense to move the page if we are close to the gravity center, i.e., if during the last chunk we have paid little. Instead we move the page if in last few chunks the cost of staying at the node reaches some threshold. To formalize it, we introduce the following definition.

Definition 5 *For any subsequence $\mathcal{S}$, a* counter $A_i(\mathcal{S})$ *denotes the cost of serving the requests within $\mathcal{S}$ using a page at node v_i. Equivalently, $A_i(\mathcal{S})$ is the cost of an algorithm, which remains at v_i for the whole $\mathcal{S}$ and does not move.*

Using the counters A_i, we may define a class of marking based algorithms. The description may seem superfluous for the Mark algorithm, but it becomes useful when we define a randomized algorithm Ebm. As mentioned above, a marking algorithm works in chunks of length $K \leq 2 \cdot \sqrt{D}$. Chunks are grouped in epochs, i.e., the partition into chunks is a refinement of the partition into epochs. The first epoch starts with the beginning of the input sequence $\mathcal{I}$.

In each epoch we track counters A_i for the part of the epoch seen so far. If counter A_i exceeds D, then we call v_i *saturated*. If at the end of some chunk all nodes are saturated, then the current epoch ends. Possibly, at the end of the input sequence, there is one epoch which has not ended; such an epoch we call *unfinished*.

We also introduce marks as something between the precise amount given by A_i and a binary saturation indicator: node v_i has $M_i := \lfloor A_i/(\frac{1}{4} \cdot K^2) \rfloor$ marks. It means that each epoch begins with all nodes unmarked. If within a chunk I counter M_i increases, then we say that v_i *was marked in* I, or that I is a *marking chunk* for v_i. This means that if a node has at least $F := 4D/K^2$ marks, it becomes saturated. Note that if $K = 2 \cdot \sqrt{D}$, then marking is equivalent to saturating. At the end of the epoch, the scheme unmarks all nodes. The exact pseudo-code for such an algorithm is presented in Figure 1.

We note that the division into epochs and chunks as well as the marking scheme is independent of the algorithm and depends only on the input sequence and the value of K. For clarity of the presentation, we assume that K is so chosen, that both K and F are integers. This condition can be fulfilled by increasing D by a constant factor. Finally, we assume that $D \geq 4$. Otherwise, we may use the algorithm Jump to achieve constant competitiveness.

We have a trivial lower bound on Opt in any finished epoch.

Lemma 6 *For any finished epoch $\mathcal{E}$, it holds that $C_{\mathrm{OPT}}(\mathcal{E}) \geq D$.*

```
Node v_i has M_i marks, initially M_i := 0 for all v_i
F := 4D/K^2
(I_1, I_2, I_3, ..., I_m) := 𝓘                 /* division into chunks I_j */
𝓔 := ∅                                          /* 𝓔 is the current epoch */
for j = 1 to m do
    𝓔 := 𝓔 ⊎ I_j
    for each v_i ∈ V
        M_i := ⌊A_i(𝓔)/(1/4 · K^2)⌋             /* compute current marks */
        if M_i ≥ F for all v_i then             /* all nodes marked F times? */
            Set M_i := 0 for all v_i            /* unmark all the nodes */
            𝓔 := ∅                              /* a new epoch begins */
```

Fig. 1. Marking scheme for input $\mathcal{I}$ with chunks of length K

PROOF. If Opt moves its page within $\mathcal{E}$, then it pays at least D. Otherwise, it remains for the whole epoch $\mathcal{E}$ in one node v_i, paying $A_i(\mathcal{E})$. Since v_i is marked at least F times within $\mathcal{E}$, $A_i(\mathcal{E}) \geq F \cdot \frac{1}{4} \cdot K^2 = D$. □

2.3.3 Marking-based Algorithms

As we have a lower bound for Opt guaranteed by Lemma 6, the role of our algorithm is to force the adversary to end the epoch, i.e., to have all the nodes marked at least F times, as quick as possible. Note that the algorithm could hardly trigger that marking event if it is at a saturated node. In that case the adversary may issue requests at a node with a low number of marks, deferring this way the end of the current epoch. Therefore, the idea of our algorithm is to remain at a node with a small number of marks, until it gets marked in some chunk I, and then to move to another unsaturated node. On the other hand, to mimic the behavior of the algorithm Move-To-Min, we want the jump candidate to be as close to the gravity center as possible. As we prove later, it appears that choosing a node with small number of marks guarantees that condition. These considerations lead us to the following definition.

Definition 7 *We call an algorithm* MB marking-based *if* MB *moves only at the end of the chunk I, which*

(1) was a marking chunk for P_{MB}, the node holding the page of MB, *or*
(2) was the last chunk in the epoch.

Additionally, if condition (2) is met, then MB *moves to $\mathcal{G}_I$.*

For making our arguments concise, we assume that after condition (1) or (2) of Definition 7 occurs, the algorithm always moves, although in rare cases it may move to the same node it is currently in. We distinguish between choosing a jump candidate *during an epoch* and choosing a jump candidate *at the end*

of the epoch. In the latter case, the jump candidate is always the gravity center of the last epoch's chunk. Choosing a jump candidate inside an epoch will be specified later by a concrete marking-based algorithm that we will analyze.

We call a subsequence between two movements of such an algorithm a *phase*. Alternatively speaking, a phase is a sequence of consecutive chunks, in which the algorithm remains at one node. Recall that chunks, phases, and epochs are all subsequences of input sequence. Moreover, the whole input sequence is partitioned into epochs, each epoch into phases, and each phase into chunks. We already know that the division into epochs and chunks is independent from any algorithm. The division of each epoch into phases depends directly on the choice of jump candidates inside epochs.

2.3.4 A Note about Unfinished Epochs

In many papers on online algorithms in which input is divided into some kind of epochs, it is assumed that the input consists of *finished* epochs only, which usually simplifies the analysis. However, a usual argument claiming that the cost of the algorithm in each phase is bounded by a function, which does not depend on the input sequence (and can be therefore placed in an additive constant of the competitive ratio), does not work in our setting. Indeed, for unbounded metric spaces, the cost of our algorithms in an epoch can be arbitrarily large. Hence, we have to use a more careful argument here.

Assume that $\mathcal{I}$ is an input sequence which has an unfinished epoch $\mathcal{E}$ at the end. We show how to prolong this sequence to a sequence $\mathcal{I}'$ which consists of whole epochs only, so that the optimal cost on such a sequence does not increase much.

We fix any optimal solution Opt for this sequence. Let v_k be the node which holds Opt page at the end of $\mathcal{I}$. After $\mathcal{I}$, the adversary gives all the requests at v_k and moves arbitrary node v_ℓ to the point where v_k lies. Afterwards, the adversary gives the requests alternately at v_k and v_ℓ. This increases counters A_k and A_ℓ by 1 per every second step. There can only be $\mathcal{O}(D)$ such steps before $\mathcal{E}$ ends. We denote the resulting input sequence by $\mathcal{I}'$. There exists an algorithm $\textsc{Opt}'$, which behaves as $\textsc{Opt}(\mathcal{I})$ on $\mathcal{I}$, and then it remains at v_k. Since A_k increases by one every second step, $\textsc{Opt}(\mathcal{I}') \leq \textsc{Opt}'(\mathcal{I}') = \textsc{Opt}(\mathcal{I}) + \mathcal{O}(D)$.

In effect, if the competitive ratio of an algorithm Alg on any sequence of finished epochs is $\mathcal{R}$, then for any (possibly unfinished) sequence $\mathcal{I}$ it holds that

$$\textsc{Alg}(\mathcal{I}) \leq \textsc{Alg}(\mathcal{I}') \leq \mathcal{R} \cdot \textsc{Opt}(\mathcal{I}') + \alpha \leq \mathcal{R} \cdot \textsc{Opt}(\mathcal{I}) + (\alpha + \mathcal{O}(\mathcal{R} \cdot D)) \ , \quad (8)$$

where α is a constant. This proves that Alg is $\mathcal{R}$-competitive on any sequence.

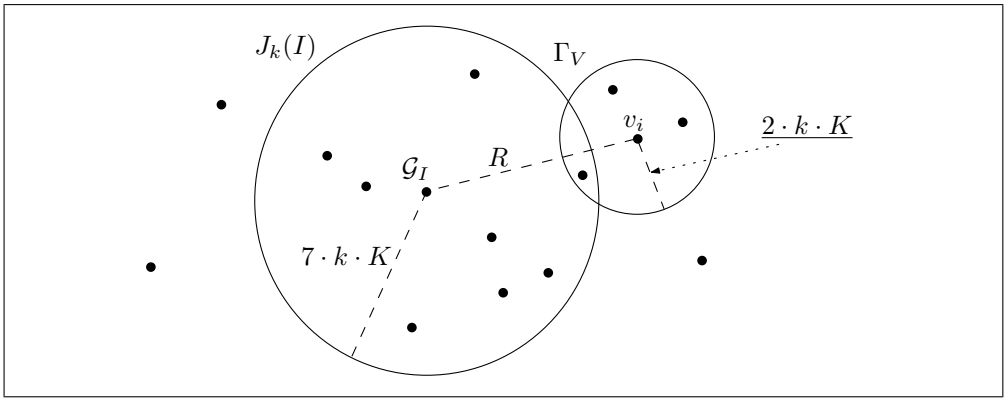

Fig. 2. Jump Set $J_k(I)$

Hence, in the remaining part of this section, we consider finished epochs only.

2.4 Deterministic Algorithm MARK

In this subsection we present a deterministic marking-based algorithm Mark and prove that it is $\mathcal{O}(n \cdot \sqrt{D})$-competitive. It is necessary to provide only two pieces of information: the chunk length K and the way of choosing jump candidates inside an epoch. Let $K = 2 \cdot \sqrt{D}$, which implies that $F = 1$, i.e., an epoch ends when all the nodes are marked at least once. For a jump candidate inside an epoch, Mark chooses any not yet marked node.

Since any epoch begins with all nodes unmarked and ends with all nodes marked, and in each phase at least one node is marked, we immediately get the following lemma.

Lemma 8 *The number of* Mark *phases in any epoch is at most n.*

2.4.1 Jump Sets

First, we prove that if Mark moves after some chunk, then — as a jump candidate — it chooses a node, which is close to the gravity center of this chunk. We make the definition slightly more general than needed for the analysis of the Mark algorithm; we use it later for a randomized version of Mark. Below we concentrate on a single chunk I, and we number its steps from 1 to K.

Definition 9 *For any chunk I and any integer $k \geq 1$, a* k-JumpSet*, which we denote by $J_k(I)$, is the set of all nodes whose distance to $\mathcal{G}_I$, measured in the last step of I, is at most $7 \cdot k \cdot K$, i.e., $J_k(I) = \{v \in V : d_K(v, \mathcal{G}_I) \leq 7 \cdot k \cdot K\}$.*

Intuitively, if an algorithm remains at a node which was far away from the gravity center or outside a jump set, it has to pay much. This is formalized in

the following lemma.

Lemma 10 *For any chunk I of K steps, any node $v_i \in V$, and any $k \geq 1$, if $v_i \notin J_k(I)$ at the end of I, then $A_i(I) \geq \frac{k}{4} \cdot K^2$.*

PROOF. We look at the configuration of nodes in time step K. Let $R := d_K(\mathcal{G}_I, v_i)$. Since $v_i \notin J_k(I)$, $R > 7 \cdot k \cdot K$. By Γ we denote a set of time steps t from chunk I, such that the K-th step distance between σ_t and v_i is at most $2 \cdot k \cdot K$. Formally,

$$\Gamma := \{\, t \in I : d_K(\sigma_t, v_i) \leq 2 \cdot k \cdot K \,\} \ . \tag{9}$$

The situation in time step K is depicted in Figure 2. Γ_V, shown there, is a multi set of nodes induced by Γ, i.e.,

$$\Gamma_V = \{\sigma_t : d_K(\sigma_t, v_i) \leq 2 \cdot k \cdot K\} \ . \tag{10}$$

Intuitively, Γ_V is the set of nodes, which issued requests in I and are now close to v_i.

First, we prove that $|\Gamma| \leq \frac{3}{4} \cdot K$. Assume the contrary, i.e., $|\Gamma| > \frac{3}{4} \cdot K$. Using the triangle inequality, we obtain

$$\begin{aligned}
\sum_{t=1}^{K} d_K(v_i, \sigma_t) &= \sum_{t \in \Gamma} d_K(v_i, \sigma_t) + \sum_{t \notin \Gamma} d_K(v_i, \sigma_t) \\
&\leq |\Gamma| \cdot 2 \cdot k \cdot K + \sum_{t \notin \Gamma} (d_K(v_i, \mathcal{G}_I) + d_K(\mathcal{G}_I, \sigma_t)) \\
&< 2 \cdot k \cdot K^2 + \frac{1}{4} \cdot K \cdot R + \sum_{t \notin \Gamma} d_K(\mathcal{G}_I, \sigma_t) \\
&\leq \frac{3}{4} \cdot K \cdot (R - 2 \cdot k \cdot K) + \sum_{t \notin \Gamma} d_K(\mathcal{G}_I, \sigma_t) \ .
\end{aligned}$$

Since $\frac{3}{4} \cdot K < |\Gamma|$ and in the last step of I the distance between $\mathcal{G}_I$ and any node from Γ is at least $R - 2 \cdot k \cdot K$, we get that

$$\sum_{t=1}^{K} d_K(v_i, \sigma_t) < \sum_{t \in \Gamma} d_K(\mathcal{G}_I, \sigma_t) + \sum_{t \notin \Gamma} d_K(\mathcal{G}_I, \sigma_t) = \sum_{t=1}^{K} d_K(\mathcal{G}_I, \sigma_t) \ .$$

This contradicts that $\mathcal{G}_I$ is a gravity center.

Since $|\Gamma| \leq \frac{3}{4} \cdot K$, at least $\frac{1}{4} \cdot K$ of the requests in chunk I were issued "far away" from v_i. Precisely speaking, since during K steps each distance can be changed at most by an additive term of K, each of these requests was issued at the distance of at least $2 \cdot k \cdot K - K \geq k \cdot K$ from v_i. Therefore, $A_i(I) \geq (K - |\Gamma|) \cdot k \cdot K = \frac{k}{4} \cdot K^2$ and the lemma follows. $\square$

By the definition of the marking scheme, we immediately conclude the following.

Corollary 11 *For any chunk I, if a node v_i is outside $J_k(I)$ at the end of I, then v_i received at least k marks in I.*

This corollary states that by choosing nodes which have small number of marks, we choose nodes which are close to the gravity center. After any chunk I, MARK chooses a jump candidate, which is either a gravity center of I, or a node which is not yet marked. But in the latter case, by Corollary 11, such a node has to belong to the 1-JumpSet of I.

Corollary 12 *If* MARK *moves its page after chunk I, then it always chooses a node belonging to $J_1(I)$ for a jump candidate. $J_1(I)$ denotes the 1-JumpSet of I.*

2.4.2 Amortized Analysis

In this subsection, we show the competitiveness of MARK using the corollary above. In the proof, we use potential function analysis. By an *amortized cost* of an action (e.g., serving requests or moving the page), we understand the actual cost of this action plus the change in the potential this action induced. We show that for any phase, the amortized cost is bounded by a term proportional to C_{OPT}.

Let L denote the distance between P_{MARK} and P_{OPT}. We define a potential as

$$\Phi = 2 \cdot D \cdot L \ . \tag{11}$$

Clearly, at the beginning of an input sequence $\Phi = 0$ and Φ is always non-negative. For any subsequence $\mathcal{S}$, by $\Delta\Phi(\mathcal{S})$ we denote the difference between the potential after $\mathcal{S}$ (after both OPT and MARK moved their pages), and before $\mathcal{S}$ (at the very end of the step preceding $\mathcal{S}$).

In fact, we can extend the definitions above to any marking-based algorithm. Most of the lemmas below hold for any such algorithm. In particular Φ may be the potential function for any marking-based algorithm MB, in which case it is equal to $2 \cdot D$ times the distance between the nodes holding the pages of MB and OPT.

First, we bound the cost of MARK in a single phase P. Let P consist of ℓ chunks, numbered from 1 to ℓ, i.e., $P = (I_1, I_2, \ldots, I_\ell)$. By the definition of a phase, we get that MARK remains at one node during the whole P. In the last step of the phase, it moves to a jump candidate v^*.

Since we want to upper-bound the cost of MARK, we assume that instead of moving directly to v^*, MARK first moves to $\mathcal{G}_{I_\ell}$ and then to v^*. Thus, in order to upper-bound the amortized cost of MARK in P, we divide its cost into two parts which we bound separately:

(1) $C^{\mathrm{A}}_{\mathrm{MARK}}(P)$: the amortized cost of serving all requests in P and moving to $\mathcal{G}_{I_\ell}$;
(2) $C^{\mathrm{B}}_{\mathrm{MARK}}(P)$: the amortized cost of moving from $\mathcal{G}_{I_\ell}$ to v^*.

Note that the second part of the cost is non-existent for the last phase in the epoch, as for such phases $v^* \equiv \mathcal{G}_{I_\ell}$. We can bound these two parts as follows.

Lemma 13 (Phase Lemma) *Let* MB *be any marking-based algorithm and $P = (I_1, \ldots, I_\ell)$ be one of its phases. Let K be the length of chunks I_j and Φ be the potential function of* MB*. Assume that at the end of P,* MB *moves to $\mathcal{G}_{I_\ell}$. Then $C_{\mathrm{MB}}(P) + \Delta\Phi(P) \leq \mathcal{O}(D/K) \cdot C_{\mathrm{OPT}}(P) + \mathcal{O}(D \cdot K)$.*

For clarity, the proof of the Phase Lemma was moved to Section 2.6. Obviously, since MARK is defined as a marking-based algorithm, we may utilize the lemma above for any phase P to get that $C^{\mathrm{A}}_{\mathrm{MARK}}(P) \leq \mathcal{O}(D/K) \cdot C_{\mathrm{OPT}}(P) + \mathcal{O}(D \cdot K)$. The bound on $C^{\mathrm{B}}_{\mathrm{MARK}}$ can be derived easily.

Lemma 14 *For any phase P of* MARK*, it holds that $C^{\mathrm{B}}_{\mathrm{MARK}}(P) \leq \mathcal{O}(D \cdot K)$.*

PROOF. By Corollary 12, a jump candidate v^* lies inside 1-JumpSet of I_ℓ. Thus, the distance between $\mathcal{G}_{I_\ell}$ and v^* is at most $7 \cdot K$. The (non-amortized) cost of moving the page between $\mathcal{G}_{I_\ell}$ and v^* is, therefore, at most $D \cdot (7 \cdot K + 1) = \mathcal{O}(D \cdot K)$. An increase in the potential induced by this movement is at most $2 \cdot D \cdot (7 \cdot K) = \mathcal{O}(D \cdot K)$. Thus, the amortized cost, $C^{\mathrm{B}}_{\mathrm{MARK}}(P) = \mathcal{O}(D \cdot K)$. □

Theorem 15 *The algorithm* MARK *is $\mathcal{O}(n \cdot \sqrt{D})$-competitive for the DPM problem.*

PROOF. Let $\mathcal{I}$ be any input sequence. Assume that it consists of k epochs, i.e., $\mathcal{I} = (\mathcal{E}_1, \mathcal{E}_2 \ldots, \mathcal{E}_k)$. By Lemma 6, $C_{\mathrm{OPT}}(\mathcal{I}) \geq k \cdot D$. On the other hand, by Lemma 8, $\mathcal{I}$ consists of at most $k \cdot n$ phases. Therefore, summing the guarantees provided by the Phase Lemma and Lemma 14 for all the phases, we get that

$$\begin{aligned} C_{\mathrm{MARK}}(\mathcal{I}) &\leq \mathcal{O}(D/K) \cdot C_{\mathrm{OPT}}(\mathcal{I}) + n \cdot k \cdot \mathcal{O}(D \cdot K) \\ &= \mathcal{O}(D/K) \cdot C_{\mathrm{OPT}}(\mathcal{I}) + \mathcal{O}(n \cdot K) \cdot C_{\mathrm{OPT}}(\mathcal{I}) \\ &= \mathcal{O}\left(n \cdot \sqrt{D}\right) \cdot C_{\mathrm{OPT}}(\mathcal{I}) \ , \end{aligned}$$

which finishes the proof. □

In this section, we show how to use randomization with the marking scheme to improve the competitive ratio achieved by algorithm MARK to $\mathcal{O}(\sqrt{D \cdot \log n})$ against an oblivious adversary. Although we do not prove it here, if we simply take the MARK algorithm, but we choose the jump candidate randomly among not yet marked nodes, then the expected number of phases becomes $\mathcal{O}(\log n)$. Since other bounds hold as well, in the proof of Theorem 15 we may simply replace n by $\mathcal{O}(\log n)$. This leads to an upper bound of $\mathcal{O}(D/K + \log n \cdot K) = \mathcal{O}(\sqrt{D} \cdot \log n)$ on the competitiveness of such randomized algorithm (see [12]). We observe that terms D/K and $\log n \cdot K$ become equal if we choose $K = \Theta(\sqrt{D/\log n})$. However, for the analysis to hold, we should guarantee that the cost in each phase can be bounded as before, and that each epoch consists (in expectation) of at most $\mathcal{O}(\log n)$ phases.

2.5.1 Balancing Algorithm EBM

We define an Exponential Balancing Marking algorithm (EBM) as follows. EBM works in chunks of length $K = 2 \cdot \sqrt{D/\log n}$. It follows that each node has to be marked $F = \log n$ times in order for an epoch to end. For choosing jump candidates, we introduce the following definition. If $\mathcal{S}$ is any subsequence of an epoch, then by $M_i(\mathcal{S})$ and $M'_i(\mathcal{S})$ we denote the number of marks v_i has, respectively, before and after $\mathcal{S}$. We also define $\Delta M_i(\mathcal{S}) = M'_i(\mathcal{S}) - M_i(\mathcal{S})$.

Assume that I is a marking chunk for P_{EBM}. Then the probability that v_i becomes a jump candidate is equal to $2^{-M'_i(I)} / \sum_{k \in [n]} 2^{-M'_k(I)}$, i.e., it is inversely proportional to $2^{M'_i(I)}$, whereas the denominator is just a scaling factor. This way, the algorithm prefers to remain at the nodes with low number of marks, but nodes with high number of marks are also taken into consideration.

It appears that we can reasonably bound the number of EBM's jumps within one epoch.

Lemma 16 *The expected number of* EBM *phases in one epoch is* $\mathcal{O}(\log n)$. *The expectation is taken over all random choices made by* EBM.

PROOF. Fix any epoch $\mathcal{E} = (I_1, I_2, \ldots, I_m)$. We define a *value of a node* after any chunk I as $n \cdot 2^{-M'_i(I)}$ and the *total value* after I as $\mathcal{W}_I := \sum_{i \in [n]} n \cdot 2^{-M'_i(I)}$. We make three key observations. First, $\mathcal{W}_I$ is monotonically non-increasing within $\mathcal{E}$. Second, $\mathcal{W}_I \leq n^2$ for any chunk I in $\mathcal{E}$, Third, at the end of chunk I_{m-1} there is at least one node having less than $\log n$ marks, otherwise the epoch would end earlier. Thus, $\mathcal{W}_{I_{m-1}} \geq 2$.

The first phase of an epoch is special, as within this phase the position of the algorithm was not chosen randomly. Starting from the second phase, we may think that a jump candidate is chosen at the very beginning of a phase and it determines where this phase ends. We show that with probability at least $1/2$, a phase reduces the total value $\mathcal{W}$ by a constant factor or ends the whole epoch. We call such a phase *successful*.

Assume that at the beginning of a phase we chose v^* for the jump candidate, i.e., $P_{\text{EBM}} = v^*$ within this phase. Then this phase ends either at the end of the first marking chunk for v^*, or at the end of I_m, if v^* is not marked in the remaining part of $\mathcal{E}$. We call this chunk *stopping* for v^*. We sort the nodes in the order induced by their stopping chunks, obtaining a sorted sequence $v_{i_1}, v_{i_2}, \ldots, v_{i_n}$. Let $p_{i_1}, p_{i_2}, \ldots, p_{i_n}$ be the probabilities of choosing these nodes as jump candidates. Let j be the smallest index for which $\sum_{k=1}^{j} p_{i_k} \geq 1/2$, and I' be the stopping chunk for v_{i_j}. Since j is the smallest index with this property, it follows immediately that, with probability $\sum_{k=j}^{n} p_{i_k} \geq 1/2$, Ebm chooses one of $v_{i_j}, v_{i_{j+1}}, \ldots, v_{i_n}$ for a jump candidate. Any such choice guarantees that the phase lasts at least until the end of I'. If $I' = I_m$ then this phase ends epoch $\mathcal{E}$, and the proof follows. Otherwise, note that between the beginning of the phase and the end of I', nodes $v_{i_1}, v_{i_2}, \ldots, v_{i_j}$ are marked at least once. Since probabilities p_{i_k} are directly proportional to the corresponding values of nodes and $\sum_{k=1}^{j} p_{i_k} \geq 1/2$, these values of nodes constitute at least one half of $\mathcal{W}_I$. By marking them once, one half of their values (and thus at least $1/4$ of the total value) is removed. Thus, $\mathcal{W}_{I'} \leq \frac{3}{4} \cdot \mathcal{W}_I$.

Hence, after the first phase, we need at most $\log_{4/3}(n^2)$ successful phases to end the epoch or reduce the total value from n^2 to 1. In expectation, at most $2 \cdot \log_{4/3}(n^2) = \mathcal{O}(\log n)$ phases suffice to either finish the epoch, or to end after the chunk I_{m-1}. In the latter case, there is at most one additional phase containing only chunk I_m. □

2.5.2 Analysis of EBM Phase

Since Ebm is a marking-based algorithm, the scheme of choosing jump candidates is coherent with the strategy of being close to the gravity center. In particular, we may reformulate Corollary 11 using values ΔM_i.

Corollary 17 *For any chunk I, a node v_i belongs to the $(\Delta M_i(I) + 1)$-JumpSet at the end of I.*

We note that Ebm may choose nodes that already have $\log n$ or more marks. Thus, we cannot bound the cost of transporting the page in the worst case, as we did for Mark algorithm. Instead, we use the observation that even if sometimes Ebm moves to the nodes which are far away from the gravity centers, it moves there only occasionally.

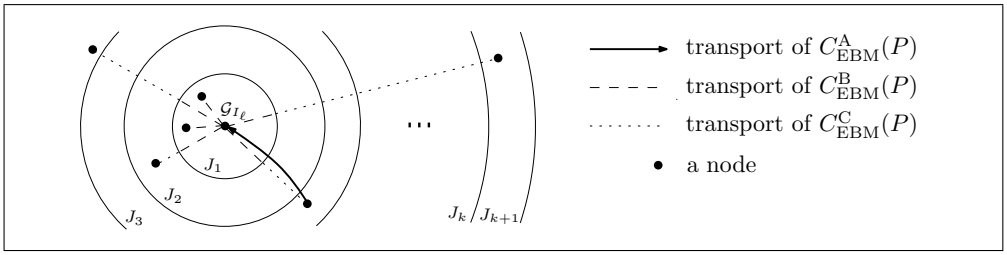

Fig. 3. Transports at the end of phase $P = (I_1, \ldots, I_\ell)$

We define the same potential function Φ as for algorithm Mark. Similarly to the proof of Mark competitiveness, we divide the amortized cost of Ebm in any phase P, consisting of ℓ chunks $(I_1, I_2, \ldots, I_\ell)$, into three parts:

(1) $C^{\mathrm{A}}_{\mathrm{EBM}}(P)$, the amortized cost of serving all requests in P and moving to $\mathcal{G}_{I_\ell}$;
(2) $C^{\mathrm{B}}_{\mathrm{EBM}}(P)$, the amortized cost of moving from $\mathcal{G}_{I_\ell}$ to the boundary of 1-JumpSet;
(3) $C^{\mathrm{C}}_{\mathrm{EBM}}(P)$, the amortized cost of moving from the boundary of 1-JumpSet to a randomly chosen jump candidate v^*.

Obviously, $C_{\mathrm{EBM}}(P) + \Delta\Phi(P) \leq C^{\mathrm{A}}_{\mathrm{EBM}}(P) + C^{\mathrm{B}}_{\mathrm{EBM}}(P) + C^{\mathrm{C}}_{\mathrm{EBM}}(P)$. Note that for the last phase in an epoch, parts $C^{\mathrm{B}}_{\mathrm{EBM}}(P)$ and $C^{\mathrm{C}}_{\mathrm{EBM}}(P)$ do not exist, as in that case Ebm moves only to the gravity center. This conceptually divides the movement of the page to the jump candidate v^* into three parts, called *transports*. These transports are schematically depicted in Figure 3.

For each epoch $\mathcal{E}$, we separately bound the expected values of these three parts. The following bound on $C^{\mathrm{A}}_{\mathrm{EBM}}$ is implied by the Phase Lemma.

Corollary 18 *For any phase P, it holds that $C^{\mathrm{A}}_{\mathrm{EBM}}(P) \leq \mathcal{O}(D/K) \cdot C_{\mathrm{OPT}}(P)$ $+\mathcal{O}(D \cdot K)$.*

On the other hand, since $C^{\mathrm{B}}_{\mathrm{EBM}}$ describes a transport within the first jump set, it can be bounded in the same way as $C^{\mathrm{B}}_{\mathrm{MARK}}$ (see Lemma 14).

Corollary 19 *For any phase P, it holds that $C^{\mathrm{B}}_{\mathrm{EBM}}(P) \leq \mathcal{O}(D \cdot K)$.*

We note that in the two corollaries above we bound the random variables $C^{\mathrm{A}}_{\mathrm{EBM}}(P)$, $C^{\mathrm{B}}_{\mathrm{EBM}}(P)$ in the worst case, not only their expected values. On the other hand, we cannot hope for a reasonable worst case bound on $C^{\mathrm{C}}_{\mathrm{EBM}}(P)$, as Ebm may jump very far away from the gravity center. Moreover, even if we bound the expected value of $C^{\mathrm{C}}_{\mathrm{EBM}}(P)$ for any single phase P, we may not combine it with the logarithmic bound on the expected number of phases in one epoch, as both bounds hold only on expectation and may depend on each other.

Therefore, we aim at constructing a bound for $\mathbf{E}[C^{\mathrm{C}}_{\mathrm{EBM}}(P)]$ that depends on the number of marks at the beginning and at the end of phase P. We show how, for any epoch $\mathcal{E}$, this yields a bound on $\mathbf{E}[C^{\mathrm{C}}_{\mathrm{EBM}}(\mathcal{E})]$ independently of the number of phases epoch $\mathcal{E}$ consists of. We use the following technical claim, proven in the appendix.

Claim 20 *For any two sequences $\{a_i\}_{i=1}^n$, $\{b_i\}_{i=1}^n$, such that $1 \leq a_i \leq b_i$, it holds that*

$$\frac{\sum_i 2^{-b_i} \cdot (b_i - a_i)}{\sum_i 2^{-b_i}} \leq \log \frac{\sum_i 2^{-a_i}}{\sum_i 2^{-b_i}} \ .$$

Lemma 21 *For any epoch $\mathcal{E}$, it holds that $\mathbf{E}[C^{\mathrm{C}}_{\mathrm{EBM}}(\mathcal{E})] = \mathcal{O}(D \cdot K \cdot \log n)$.*

PROOF. First, we bound $C^{\mathrm{C}}_{\mathrm{EBM}}$ in a single phase $P = (I_1, I_2, \ldots, I_\ell)$, where I_j are chunks of P. By Corollary 17, at the end of I_ℓ, each node v_i lies inside $(\Delta M_i(I_\ell)+1)$-JumpSet, and thus inside $(\Delta M_i(P)+1)$-JumpSet. The marking system is coherent with the approach of choosing nodes close to the gravity centers — if a node is far away from the gravity center, it has many marks and the probability that EBM moves to such node is exponentially small.

Formally, if we transport the page to v_i, the $C^{\mathrm{C}}_{\mathrm{EBM}}(P)$ part of the cost reflects only the cost of moving the page from the boundary of 1-JumpSet to a node within $(\Delta M_i(P)+1)$-JumpSet, i.e., the cost at most $D \cdot (7 \cdot K \cdot \Delta M_i(P))$. We do not consider the constant overhead for the communication, since it was already taken into account in the $C^{\mathrm{B}}_{\mathrm{EBM}}(P)$ part of the cost. As the corresponding change in the potential is at most twice this cost, the amortized cost of such a movement is at most $21 \cdot K \cdot D \cdot \Delta M_i(P)$. Thus, the expected amortized cost of moving the page to v^* (taken over all possible random choices of v^*) is

$$\begin{aligned}\mathbf{E}[C^{\mathrm{C}}_{\mathrm{EBM}}(P)] &\leq \sum_{i\in[n]} \frac{2^{-M'_i(P)}}{\sum_{k\in[n]} 2^{-M'_k(P)}} \cdot \Delta M_i(P) \cdot 21 \cdot D \cdot K \\ &\leq 21 \cdot D \cdot K \cdot \log\left(\frac{\sum_{i\in[n]} 2^{-M_i(P)}}{\sum_{i\in[n]} 2^{-M'_i(P)}}\right) \ ,\end{aligned}$$

where the latter inequality follows from Claim 20, by taking $b_i = M'_i(P)$ and $a_i = M_i(P)$.

Now we fix an epoch $\mathcal{E} = (P_1, P_2 \ldots P_p)$. Note that $C^{\mathrm{C}}_{\mathrm{EBM}}(P_p) = 0$. Thus, it is sufficient to prove that $\sum_{j=1}^{p-1} \mathbf{E}[C^{\mathrm{C}}_{\mathrm{EBM}}(P_j)] = \mathcal{O}(K \cdot D \cdot \log n)$. Consider the following bound

$$\frac{\sum_{j=1}^{p-1} \mathbf{E}[C^{\mathrm{C}}_{\mathrm{EBM}}(P_j)]}{21 \cdot K \cdot D} \leq \log\left(\prod_{j=1}^{p-1} \frac{\sum_{i\in[n]} 2^{-M_i(P_j)}}{\sum_{i\in[n]} 2^{-M'_i(P_j)}}\right) = \log\left(\frac{\sum_{i\in[n]} 2^{-M_i(P_1)}}{\sum_{i\in[n]} 2^{-M'_i(P_{p-1})}}\right) \ .$$

Since $M_i(P_1) = 0$ for all i, the numerator in the last term above is equal to n. There exists a node v_i, which has less than $\log n$ marks at the end of P_{p-1}, otherwise epoch $\mathcal{E}$ would be finished earlier. Thus, the corresponding denominator is at least $1/n$, and we get

$$\mathbf{E}[C^{\mathrm{C}}_{\mathrm{EBM}}(\mathcal{E})] = \sum_{j=1}^{p-1} \mathbf{E}[C^{\mathrm{C}}_{\mathrm{EBM}}(P_j)] \;\leq\; 21 \cdot D \cdot K \cdot \log \frac{n}{1/n} = \mathcal{O}(D \cdot K \cdot \log n) \;\; .$$

□

Finally, we may bound the total amortized cost in any epoch.

Theorem 22 *The algorithm* EBM *is* $\mathcal{O}(\sqrt{D \cdot \log n})$*-competitive against an oblivious adversary for the DPM problem.*

PROOF. The proof follows the pattern of the analogous proof for algorithm MARK. Again, $\mathcal{I}$ is an input sequence, consisting of k epochs, i.e., $\mathcal{I} = (\mathcal{E}_1, \mathcal{E}_2 \ldots, \mathcal{E}_k)$.

Fix any epoch $\mathcal{E}_i$. By the definition, we get

$$\mathbf{E}[C_{\mathrm{EBM}}(\mathcal{E}_i) + \Delta\Phi(\mathcal{E}_i)] = \mathbf{E}[C^{\mathrm{A}}_{\mathrm{EBM}}(\mathcal{E}_i)] + \mathbf{E}[C^{\mathrm{B}}_{\mathrm{EBM}}(\mathcal{E}_i)] + \mathbf{E}[C^{\mathrm{C}}_{\mathrm{EBM}}(\mathcal{E}_i)] \;\; .$$

We combine the worst-case bounds on the first two terms with the logarithmic bound on the expected number of phases in $\mathcal{E}_i$, and we apply the bound on $\mathbf{E}[C^{\mathrm{C}}_{\mathrm{EBM}}(\mathcal{E}_i)]$ provided by Lemma 21, obtaining

$$\mathbf{E}[C_{\mathrm{EBM}}(\mathcal{E}_i) + \Delta\Phi(\mathcal{E}_i)] = \mathcal{O}(D/K) \cdot C_{\mathrm{OPT}}(\mathcal{E}_i) + \mathcal{O}(D \cdot K \cdot \log n).$$

Summing the bounds on amortized costs in particular epochs, and using $C_{\mathrm{OPT}}(\mathcal{I}) \geq k \cdot D$, we get

$$\begin{aligned} C_{\mathrm{EBM}}(\mathcal{I}) \;&\leq\; \mathcal{O}(D/K) \cdot C_{\mathrm{OPT}}(\mathcal{I}) + k \cdot \mathcal{O}(D \cdot K \cdot \log n) \\ &=\; \mathcal{O}\left(\sqrt{D \cdot \log n}\right) \cdot C_{\mathrm{OPT}}(\mathcal{I}) \;\; . \end{aligned}$$

Thus, EBM is $\mathcal{O}(\sqrt{D \cdot \log n})$-competitive. □

2.6 Proofs of the Phase Lemma

In this section we prove the Phase Lemma (Lemma 13). Throughout this section we use the following notation. Let MB be any marking-based algorithm working in chunks of length K. Let P be any phase of MB. We assume that

P consists of ℓ chunks $I_1, I_2, \ldots, I_\ell$. Let v_{P} denote the node in which algorithm MB has its page in whole phase P; then A_{P} is the cost of serving requests by MB in P. We assume that at the end of P, MB moves to $\mathcal{G}_{I_\ell}$. We note that v_{P} is marked only in I_ℓ.

We divide the cost of MB in P into two parts: the cost incurred by chunks $(I_1, I_2, \ldots, I_{\ell-1})$ and the cost incurred by I_ℓ. The latter includes the cost of movement to $\mathcal{G}_{I_\ell}$. We bound the amortized cost in each part separately.

2.6.1 Bound for All Chunks but the Last One

Lemma 23 *Let P be any phase consisting of ℓ chunks $(I_1, I_2, \ldots, I_\ell)$ and let P' be the first $\ell - 1$ chunks of P. Then*

$$C_{\mathrm{MB}}(P') + \Delta\Phi(P') \leq \mathcal{O}(D/K) \cdot C_{\mathrm{OPT}}(P') + \mathcal{O}(D \cdot K) \ .$$

PROOF. First, we note that the cost of serving requests within P', $C_{\mathrm{MB}}(P')$, is equal to $A_{\mathrm{P}}(P') < \frac{1}{4} \cdot K^2 = \mathcal{O}(D \cdot K)$, because otherwise v_{P} would be marked within P', and the phase would last shorter. Thus, it remains to bound the change in the potential, $\Delta\Phi(P')$.

We denote the distance between P_{MB} and P_{OPT} by L. Let t_{A} be the last step of P' at the end of which it holds that $L \leq K$. Further, we divide P' into two disjoint parts, P'_{A} ending at step t_{A} and P'_{B} starting at step $t_{\mathrm{A}}+1$. One of these parts may be an empty sequence. The change of the potential within P'_{A} is at most the potential at the end of step t_{A}, and thus $\Delta\Phi(P'_{\mathrm{A}}) \leq 2 \cdot D \cdot K$. In the remaining part of the proof, we concentrate on bounding the value of $\Delta\Phi(P'_{\mathrm{B}})$.

Let $s = |P'_{\mathrm{B}}|$; we number the time steps within P'_{B} from 1 to s. Since the cost of serving requests in P'_{B} is small, the total sum of distances between v_{P} and requests is even smaller, i.e., $\sum_{t=1}^{s} d_t(v_{\mathrm{P}}, \sigma_t) \leq \frac{1}{4} \cdot K^2$. We call a request *close* if it was issued at the distance at most $K/2$ from v_{P}; otherwise we call a request *far*. Clearly, at most $K/2$ requests from P'_{B} are far and at least $s - K/2$ requests are close.

At the end of any step of P'_{B}, $L > K$. Therefore, OPT pays at least $K/2$ for serving any close request. Let J be the sum of lengths of OPT jumps. Then $C_{\mathrm{OPT}}(P'_{\mathrm{B}}) \geq D \cdot J + (s - \frac{K}{2}) \cdot \frac{K}{2}$.

As MB remains at the same node, L is influenced by two factors: the adversarial change to the network and the movement of the optimal algorithm. The total amount of these changes can be bounded by $1 \cdot s$ and J, respectively. Thus, $\Delta\Phi(P'_{\mathrm{B}}) \leq D \cdot s + D \cdot J$.

When we combine the two bounds above, we get $\Delta\Phi(P'_{\mathrm{B}}) \leq 2 \cdot (D/K) \cdot \textsc{Opt}(P_{\mathrm{B}}) + \frac{1}{2} \cdot D \cdot K$, which finishes the proof. $\square$

2.6.2 Bound for the Last Chunk

This subsection, devoted to bounding the cost in the last chunk, I_ℓ, is inspired by the proof of the competitiveness of the Move-To-Min algorithm [2]. However, in our proof the chunk lengths are shorter than D, and additionally we have to take into account the movement of the nodes, which makes the proof more complex.

We number all time steps within I_ℓ from 1 to K. As $D \geq 4$, $K \leq 2 \cdot \sqrt{D} \leq D$.

Before we bound the amortized cost of MB in I_ℓ, we construct a lower bound on $C_{\mathrm{OPT}}(I_\ell)$. By a_{t-1} and a_t we denote the node holding the page of Opt, respectively at the beginning and at the end of the t-th step. In particular, we get $a_0 = P_{\mathrm{OPT}}(1)$ and $a_K = P'_{\mathrm{OPT}}(K)$. In step t, Opt pays $c_t(a_{t-1}, \sigma_t)$ for serving a request and $D \cdot c_t(a_{t-1}, a_t)$ for moving the page. Thus,

$$C_{\mathrm{OPT}}(I_\ell) = \sum_{t=1}^{K} \left(c_t(a_{t-1}, \sigma_t) + D \cdot c_t(a_{t-1}, a_t) \right) \ . \tag{12}$$

The following lemma states that the cost of Opt in one chunk is (up to constant terms) lower-bounded by a cost of an algorithm which remains at one node throughout this whole chunk. For succinctness of proofs, we additionally introduce two distance functions:

$$\underline{d}(v_a, v_b) = \min_{0 \leq t \leq K} d_t(v_a, v_b) \qquad \text{and} \qquad \overline{d}(v_a, v_b) = \max_{0 \leq t \leq K} d_t(v_a, v_b) \ . \tag{13}$$

As the adversary is $\frac{1}{2}$-restricted, the distances given by $\underline{d}$ and $\overline{d}$ differ at most by K.

Lemma 24 *For any T, it holds that $\sum_{t=1}^{K} \overline{d}(a_T, \sigma_t) \leq C_{\mathrm{OPT}}(I_\ell) + \mathcal{O}(K^2)$.*

PROOF. It follows from the triangle inequality that

$$\begin{aligned}
\sum_{t=1}^{K} \overline{d}(a_T, \sigma_t) &\leq \sum_{t=1}^{K} \underline{d}(a_{t-1}, \sigma_t) + \sum_{j=1}^{K} \underline{d}(a_{j-1}, a_T) + \mathcal{O}(K^2) \\
&\leq \sum_{t=1}^{K} \underline{d}(a_{t-1}, \sigma_t) + \sum_{j=1}^{K} \left(\sum_{t=1}^{K} \underline{d}(a_{t-1}, a_t) \right) + \mathcal{O}(K^2) \\
&\leq \sum_{t=1}^{K} c_t(a_{t-1}, \sigma_t) + K \cdot \sum_{t=1}^{K} c_t(a_{t-1}, a_t) + \mathcal{O}(K^2) \\
&\leq \textsc{Opt}(I_\ell) + \mathcal{O}(K^2) \ ,
\end{aligned}$$

□

Using the lower bound on Opt presented above, we can bound the amortized cost of the algorithm in the last chunk I_ℓ. We denote the potential at the beginning of I_ℓ by Φ_{B} and the potential at the end of I_ℓ by Φ_{E}. We split the amortized cost of MB in this chunk, $C_{\mathrm{MB}}(I_\ell) + \Phi_{\mathrm{E}} - \Phi_{\mathrm{B}}$, into two parts, which we bound separately in the two following lemmas.

Lemma 25 *It holds that* $A_{\mathrm{P}}(I_\ell) - \Phi_{\mathrm{B}}/2 \leq C_{\mathrm{OPT}}(I_\ell) + \mathcal{O}(D \cdot K)$.

PROOF. By the definition, $\Phi_{\mathrm{B}} = 2 \cdot D \cdot d_0(v_{\mathrm{P}}, a_0) \geq 2 \cdot D \cdot (\overline{d}(v_{\mathrm{P}}, a_0) - K)$. Utilizing the triangle inequality, Lemma 24, and $K \leq D$, we get

$$\begin{aligned}
A_{\mathrm{P}}(I_\ell) - \Phi_{\mathrm{B}}/2 &= \sum_{t=1}^{K} c_t(v_{\mathrm{P}}, \sigma_t) - D \cdot \overline{d}(v_{\mathrm{P}}, a_0) + \mathcal{O}(D \cdot K) \\
&\leq \sum_{t=1}^{K} (1 + d_t(v_{\mathrm{P}}, a_0) + d_t(a_0, \sigma_t)) - D \cdot \overline{d}(v_{\mathrm{P}}, a_0) + \mathcal{O}(D \cdot K) \\
&\leq \sum_{t=1}^{K} d_t(a_0, \sigma_t) + \mathcal{O}(D \cdot K) \\
&\leq C_{\mathrm{OPT}}(I_\ell) + \mathcal{O}(D \cdot K) \ .
\end{aligned}$$

□

Lemma 26 *It holds that* $D \cdot c_K(v_{\mathrm{P}}, \mathcal{G}_{I_\ell}) + \Phi_{\mathrm{E}} - \Phi_{\mathrm{B}}/2 \leq \mathcal{O}(D/K) \cdot C_{\mathrm{OPT}}(I_\ell) + \mathcal{O}(D \cdot K)$.

PROOF. First, we show how to bound the value of $d_K(a_T, \mathcal{G}_{I_\ell})$ for any $T \in$

$\{0, ..., K\}$. By the triangle inequality,

$$K \cdot d_K(a_T, \mathcal{G}_{I_\ell}) \leq \sum_{t=1}^{K} (d_K(a_T, \sigma_t) + d_K(\sigma_t, \mathcal{G}_{I_\ell})) \quad .$$

Since $\mathcal{G}_{I_\ell}$ is a gravity center of I_ℓ, $\sum_{t=1}^{K} d_K(\mathcal{G}_{I_\ell}, \sigma_t) \leq \sum_{t=1}^{K} d_K(a_T, \sigma_t)$. By combining these inequalities with Lemma 24, we obtain

$$K \cdot d_K(a_T, \mathcal{G}_{I_\ell}) \leq 2 \cdot \sum_{t=1}^{K} d_K(a_T, \sigma_t) \leq 2 \cdot \textsc{Opt}(I_\ell) + \mathcal{O}(K) \quad .$$

Finally, using $\Phi_{\mathrm{B}} = 2 \cdot D \cdot d_0(v_{\mathrm{P}}, a_0) \geq 2 \cdot D \cdot (d_K(v_{\mathrm{P}}, a_0) - K)$, the triangle inequality, and the bound above, we get

$$\begin{aligned} & D \cdot c_K(v_{\mathrm{P}}, \mathcal{G}_{I_\ell}) - \Phi_{\mathrm{B}}/2 + \Phi_{\mathrm{E}} \\ & \qquad \leq D \cdot d_K(v_{\mathrm{P}}, \mathcal{G}_{I_\ell}) - D \cdot d_K(v_{\mathrm{P}}, a_0) + 2 \cdot D \cdot d_K(a_K, \mathcal{G}_{I_\ell}) + \mathcal{O}(D \cdot K) \\ & \qquad \leq D \cdot d_K(a_0, \mathcal{G}_{I_\ell}) + 2 \cdot D \cdot d_K(a_K, \mathcal{G}_{I_\ell}) + \mathcal{O}(D \cdot K) \\ & \qquad \leq \mathcal{O}(D/K) \cdot C_{\mathrm{OPT}}(I_\ell) + \mathcal{O}(D \cdot K) \quad , \end{aligned}$$

which finishes the proof. □

The proof of the Phase Lemma is a straightforward consequence of the bounds above.

PROOF of Lemma 13 (Phase Lemma). We want to bound amortized cost of MB cost in a phase $P = (I_1, \ldots, I_\ell)$. By Lemmas 25 and 26, we get that $C_{\mathrm{MB}}(I_\ell) + \Delta\Phi(I_\ell) = A_{\mathrm{P}}(I_\ell) + D \cdot c_K(v_{\mathrm{P}}, \mathcal{G}_{I_\ell}) + \Phi_{\mathrm{E}} - \Phi_{\mathrm{B}} \leq \mathcal{O}(D/K) \cdot C_{\mathrm{OPT}}(I_\ell) + \mathcal{O}(D \cdot K)$. By Lemma 23, we get a similar bound on the amortized cost in the previous chunks. These bounds, combined, yield the lemma. □

3 Lower Bounds

In this section, we prove asymptotically matching lower bounds on the competitive ratios. In particular, we prove the following two theorems.

Theorem 27 *There exists a metric space $\mathcal{X}$ such that for any randomized, $\mathcal{R}$-competitive algorithm for the DPM problem playing against an adaptive-online adversary, $\mathcal{R} = \Omega(\min\{n \cdot \sqrt{D}, D\})$.*

Theorem 28 *There exists a metric space $\mathcal{X}$ such that for any randomized, $\mathcal{R}$-competitive algorithm for the DPM problem playing against an oblivious adversary, $\mathcal{R} = \Omega(\min\{\sqrt{D \cdot \log n}, D\})$.*

The main ingredient of our solution is pulling the node holding the algorithm's page away from the group of other nodes, while giving the requests at one of the nodes from this group. This intuition is formalized in the following subsections.

As mentioned earlier in the introduction, we do not provide lower bounds for arbitrary metric spaces. For simplicity, throughout this section, we assume that our metric space $\mathcal{X}$ is equal to the real line $\mathbb{R}$ with Euclidean metric. In fact, it is sufficient for our construction when $\mathcal{X}$ contains just an interval of length $\Theta(\mathcal{R})$ of this real line, where $\mathcal{R}$ is the lower bound on the competitive ratio.

We illustrate the construction using positions on a line. Any point of $\mathcal{X}$ has a coordinate which is a real number and for any nodes v_a, v_b and time step t, $p_t(v_a), p_t(v_b) \in \mathbb{R}$ and $d_t(v_a, v_b) = |p_t(v_a) - p_t(v_b)|$. We omit subscript t if it is clear from the context. The point with coordinate 0 is called point zero. When we write *above point* s, we mean points with coordinates greater than s. The opposite notion is *below point* s. We say that a node is *moving with speed* f up or down if it increases or decreases its coordinate by f per time step.

For simplicity of the proofs in this section, we assume that $\sqrt{D}$ is an integer. Otherwise, we could use $\lfloor\sqrt{D}\rfloor$ instead and lose only a constant factor in the analysis. For the same reason, we may assume that if $D \geq \log n$, then D is divisible by $\log n$. We also assume that n is a power of 2. If it is not the case, then the adversary may give requests only at the first $2^{\lfloor \log n \rfloor}$ nodes and put the other nodes exactly at the same point of space $\mathcal{X}$ as v_0. Then, for any algorithm ALG that uses these additional nodes, an algorithm ALG$'$ which uses v_0 instead has a cost not greater than ALG. Thus, we lose at most a constant factor due to such rounding.

3.1 Lower Bound for Deterministic Algorithms

In this section, we present a construction of a lower bound of $\Omega(\min\{n \cdot \sqrt{D}, D\})$ on the competitive ratio of any deterministic algorithm. This proof is redundant, as in the next section we show that the same lower bound holds for randomized algorithms against adaptive-online adversaries. However, it serves as a warm-up and illustrates key concepts, which are reused later.

Fix any deterministic algorithm DET. We show how to adaptively construct a subsequence, called an epoch $\mathcal{E}$, on which the ratio between the costs of DET and OPT is large. Moreover, this construction can be repeated many times, so that the cost incurred on DET is arbitrarily high.

An epoch consists of several phases. Let P_{DET} denote the node holding the

page of DET. At the beginning of a phase, the adversary chooses a node with the smallest index different from P_{DET}, i.e., either v_0 or v_1. All the requests in this phase are given at that node.

Each phase consists of two parts of equal length: an *expanding part* and a *contracting part*. In each step of the expanding part, the adversary increases the distance between P_{DET} and the rest of the nodes, i.e., it moves P_{DET} with speed 1 up, so that in the t-th step of the expanding part $p(P_{\text{DET}}) = t - 1$. The node which is moved away is called *active* in this phase. Other nodes remain at their positions throughout the whole phase. The expanding part continues till the algorithm decides to move its page to a new node. Note that if the algorithm never jumped, the expanding part would last forever and incur infinite cost on DET. Then comes a contracting part of the same length, in which the active node is moved down with speed 1.

If the number of steps in the expanding part is at least $\sqrt{D}$, we call a phase *long*, otherwise we call it *short*. Epoch $\mathcal{E}$ ends at the end of the $(n/2)$-th long phase or at the end of the $(n \cdot \sqrt{D})$-th short phase, whichever occurs first. This guarantees that $\mathcal{E}$ contains at least $\Omega(n \cdot \sqrt{D})$ steps. Note that at the beginning and at the end of $\mathcal{E}$, all nodes are at point zero.

In our construction, we use the counters A_i as described in Definition 5, i.e., A_i is the cost of an algorithm which remains at node v_i. The following technical lemma compares the costs of DET to the values of counters A_i. Later, on the basis of these counters, we show that an offline algorithm which remains at one node throughout the whole epoch performs much better than DET.

Lemma 29 *Fix any deterministic algorithm* DET *and create an epoch $\mathcal{E}$ in the way described above. Let P be any phase of $\mathcal{E}$, X denote the length of the expanding part of P, and a be the index of the node active in P. Then the following properties hold:*

(1) $C_{\text{DET}}(P) \geq D \cdot X$,
(2) if P is a short phase, then $\sum_{i \in [n]} A_i(P) = \mathcal{O}(n + \sqrt{D}) \cdot X$,
(3) if P is a long phase, then $\sum_{i \in [n] \setminus \{a\}} A_i(P) = \mathcal{O}(n) \cdot X$.

PROOF. First, we observe that DET pays at least for moving the page at the end of the expanding part of P, which amounts to $D \cdot X$. Hence, property 1 holds.

Second, we bound the counters A_i. If $i \neq a$, then $A_i(P)$ corresponds to a cost of serving the requests from a node which remains at point zero, where all the requests are issued. In this case $A_i(P) = 2 \cdot X$, which implies property 3. On the other hand, $A_a(P)$ corresponds to the cost of serving the requests from the active node, which amounts to $2 \cdot \sum_{\ell=1}^{X} \ell = \mathcal{O}(X^2)$. As for a short

phase, $X^2 = \mathcal{O}(\sqrt{D}) \cdot X$, we obtain $\sum_{i\in[n]} A_i(P) = \mathcal{O}(n + \sqrt{D}) \cdot X$, and thus property 2 holds. □

Now we consider a set of n simple algorithms for an epoch $\mathcal{E}$. For any $i \in [n]$, strategy B_i is to move the page to v_i at the beginning of $\mathcal{E}$ and remain there till the end of $\mathcal{E}$. As at the beginning of any epoch, all nodes are at point zero, B_i pays D for the initial movement, and thus $C_{\mathrm{B}_i}(\mathcal{E}) = D + A_i(\mathcal{E})$.

We denote the set of all nodes which are not active in any long phase of $\mathcal{E}$ by $\mathcal{V}_\mathcal{E}$. Clearly, $|\mathcal{V}_\mathcal{E}| \geq n/2$. By the technical lemma above, we may infer that a good algorithm should not be in an active node in a long phase, which can be achieved by remaining in a node from $\mathcal{V}_\mathcal{E}$ for the whole epoch. This intuition is formalized below.

Lemma 30 *For any deterministic algorithm* DET*, if we create an epoch $\mathcal{E}$ in the way described above, then $C_{\mathrm{DET}}(\mathcal{E}) = \Omega(\min\{\sqrt{D}, D/n\}) \cdot \sum_{v_i\in\mathcal{V}_\mathcal{E}} C_{\mathrm{B}_i}(\mathcal{E})$.*

PROOF. Fix any algorithm DET and an epoch $\mathcal{E}$. Let $P_1, P_2, \ldots$ be the phases of $\mathcal{E}$ and let X_j denote the length of the expanding part of P_j.

First, property 1 of Lemma 29 implies that $C_{\mathrm{DET}}(\mathcal{E}) \geq D \cdot \sum_{P_j\in\mathcal{E}} X_j$, and thus $C_{\mathrm{DET}}(\mathcal{E}) = \Omega(n \cdot D \cdot \sqrt{D})$.

Second, by the definition of $\mathcal{V}_\mathcal{E}$, Lemma 29 implies that for any phase $P_j \in \mathcal{E}$, $\sum_{v_i\in\mathcal{V}_\mathcal{E}} A_i(P_j) = \mathcal{O}(n + \sqrt{D}) \cdot X_i$. Therefore,

$$\begin{aligned}\sum_{v_i\in\mathcal{V}_\mathcal{E}} C_{\mathrm{B}_i}(\mathcal{E}) &= \sum_{v_i\in\mathcal{V}_\mathcal{E}} (D + A_i(\mathcal{E})) \\ &= \mathcal{O}(n \cdot D) + \mathcal{O}\left(n + \sqrt{D}\right) \cdot \sum_{P_j\in\mathcal{E}} X_j \ .\end{aligned}$$

By comparing $C_{\mathrm{DET}}(\mathcal{E})$ with $\sum_{v_i\in\mathcal{V}_\mathcal{E}} C_{\mathrm{B}_i}(\mathcal{E})$, the lemma follows. □

Finally, we show how the lemma above implies the lower bound on the competitive ratio.

Theorem 31 *There exists a metric space, such that for any deterministic, $\mathcal{R}$-competitive algorithm for the DPM problem, $\mathcal{R} = \Omega(\min\{n \cdot \sqrt{D}, D\})$.*

PROOF. Fix any deterministic algorithm DET. As the construction of epoch can be repeated many times, the cost of DET can be made arbitrarily high. Therefore, it suffices to show that in a single epoch $\mathcal{E}$, the ratio between costs of DET and an offline algorithm is $\Omega(\min\{n \cdot \sqrt{D}, D\})$.

Let Off be the offline algorithm which in epoch $\mathcal{E}$ follows the minimum cost strategy from the set $\{\mathrm{B}_i : i \in \mathcal{V}_\mathcal{E}\}$. By the average argument, $\textsc{Off}(\mathcal{E}) \leq \frac{1}{|\mathcal{V}_\mathcal{E}|} \cdot \sum_{i \in \mathcal{V}_\mathcal{E}} C_{\mathrm{B}_i}(\mathcal{E}) = \mathcal{O}(1/n) \cdot \sum_{i \in \mathcal{V}_\mathcal{E}} C_{\mathrm{B}_i}(\mathcal{E})$, and hence by Lemma 30, $C_{\mathrm{DET}}(\mathcal{E}) = \Omega(\min\{n \cdot \sqrt{D}, D\}) \cdot C_{\mathrm{OFF}}(\mathcal{E})$. □

3.2 Lower Bound for Adaptive-online Adversary

In this section, we adapt the proof of the lower bound for deterministic algorithms to the setting of randomized algorithms fighting against an adaptive-online adversary.

First, we take a look at what happens if we just copy the construction of a single epoch from the previous subsection. Previously, we compared a deterministic algorithm to an *offline* solution Off, which chose its strategy at the end, knowing the final shape of $\mathcal{E}$. Now the solution of the adversary has to be created also in online manner. As the construction of an epoch depends now on the random choices of the algorithm, it is no longer possible for the adversary to fully predict the shape of $\mathcal{E}$ and to choose a strategy optimally at the beginning of $\mathcal{E}$.

In the previous subsection, we showed that there existed a "good" solution Off in the set of predefined strategies $\{\mathrm{B}_i : i \in \mathcal{V}_\mathcal{E}\}$. This time, the adversary does not know $\mathcal{V}_\mathcal{E}$ in advance, but it may try a similar approach: we show that one of the strategies from the set $\{\mathrm{B}_i : i \in [n]\}$ works for the adversary.

We take a closer look at what happens if the solution chosen by the adversary follows a strategy B_i for $i \notin \mathcal{V}_\mathcal{E}$. In that case, the algorithm may try to induce only very long phases, e.g. it may jump after D steps of each expanding part. After $n/2$ such phases, the epoch ends and while the algorithm pays $\Theta(n \cdot D^2)$, the adversary remains at a node which is active in one of these phases, and thus has cost of $\Omega(D^2)$. This would lead to a weaker lower bound of $\Omega(n)$. To alleviate this problem, the adversary has to adapt the generation of an epoch appropriately.

Below, we formalize these intuitions. For creating a single epoch, we consider n strategies of the adversary, $\textsc{Adv}_i$ where $i \in [n]$. To answer requests, $\textsc{Adv}_i$ follows the strategy B_i. The only difference to the previous subsection is the following. For the adversary $\textsc{Adv}_i$, if v_i is active in a phase and its expanding part lasted already for $\sqrt{D}$ steps, then $\textsc{Adv}_i$ starts the contracting part immediately, without waiting for the algorithm to jump. After such a phase, called *shortened*, the epoch ends. We denote the epoch created by $\textsc{Adv}_i$ by $\mathcal{E}_i$. The last step of the expanding part of the shortened phase is called *critical*

for $\textsc{Adv}_i$.

Lemma 32 *Fix any randomized algorithm* Alg. *For any $i \in [n]$, let $\mathcal{E}_i$ be the epoch created by the adversary $\textsc{Adv}_i$ in the way described above. Then it holds that*

$$\sum_{i\in[n]} \mathbf{E}[C_{\mathrm{ALG}}(\mathcal{E}_i)] = \Omega\left(\min\{n\cdot\sqrt{D}, D\}\right) \cdot \sum_{i\in[n]} \mathbf{E}[C_{\mathrm{ADV}_i}(\mathcal{E}_i)] \ ,$$

where the expectation is taken over all random choices made by Alg.

PROOF. We show a stronger result, i.e., we show that the relation above holds not only in expectation, but for any fixed choice of random bits used by Alg. Thus, it suffices to show that for any deterministic algorithm Det, if the adversaries $\textsc{Adv}_i$ create their epochs $\mathcal{E}_i$ for Det, then the following relation holds:

$$\sum_{i\in[n]} C_{\mathrm{DET}}(\mathcal{E}_i) = \Omega\left(\min\{n\cdot\sqrt{D}, D\}\right) \cdot \sum_{i\in[n]} C_{\mathrm{B}_i}(\mathcal{E}_i) \ . \tag{14}$$

Let $\mathcal{E}$ be an epoch that would be created if the adversary followed the strategy from the previous subsection. Each $\textsc{Adv}_i$ follows this strategy up to the critical step. On the other hand, the critical step is the first place where Det may learn anything about the adversary it is fighting against. Det may use this knowledge only in the contracting part of the last, shortened phase; we neglect its behavior there.

In other words, for $i \in \mathcal{V}_\mathcal{E}$, $\mathcal{E}_i = \mathcal{E}$, as the corresponding adversary $\textsc{Adv}_i$ has no reason to end the epoch earlier. On the other hand, for $i \notin \mathcal{V}_\mathcal{E}$, $\mathcal{E}_i$ ends with a shortened phase in which v_i is active. If we neglect this last phase, then the previous phases of $\mathcal{E}_i$ are a prefix of $\mathcal{E}$. An example of this relation is shown in Figure 4.

Let $P_1, P_2, \ldots$ be the phases of $\mathcal{E}$, and X_j denote the length of the expanding part of phase P_j. Note that P_j may not exist in $\mathcal{E}_i$ or $\mathcal{E}_i$ may contain only a shortened version of P_j as its last phase. Otherwise, we write $P_j \in \mathcal{E}_i$.

We observe that for any phase $P_j \in \mathcal{E}$, it holds that

$$\sum_{i:P_j\in\mathcal{E}_i} A_i(P_j) = \mathcal{O}\left(n + \sqrt{D}\right) \cdot X_j \ . \tag{15}$$

This relation follows trivially by property 2 of Lemma 29 if P_j is a short phase. If P_j is a long phase, and if we take any epoch $\mathcal{E}_i$ containing P_j, then by our construction v_i is not active in P_j. In this case, the relation above follows by property 3 of Lemma 29.

For bounding $\sum_{i\in[n]} C_{\mathrm{DET}}(\mathcal{E}_i)$, we forgive Det the cost incurred on epochs $\mathcal{E}_i$ shorter than $\mathcal{E}$. Thus, we obtain $\sum_{i\in[n]} C_{\mathrm{DET}}(\mathcal{E}_i) \geq \sum_{i\in\mathcal{V}_\mathcal{E}} C_{\mathrm{DET}}(\mathcal{E}_i) = |\mathcal{V}_\mathcal{E}| \cdot$

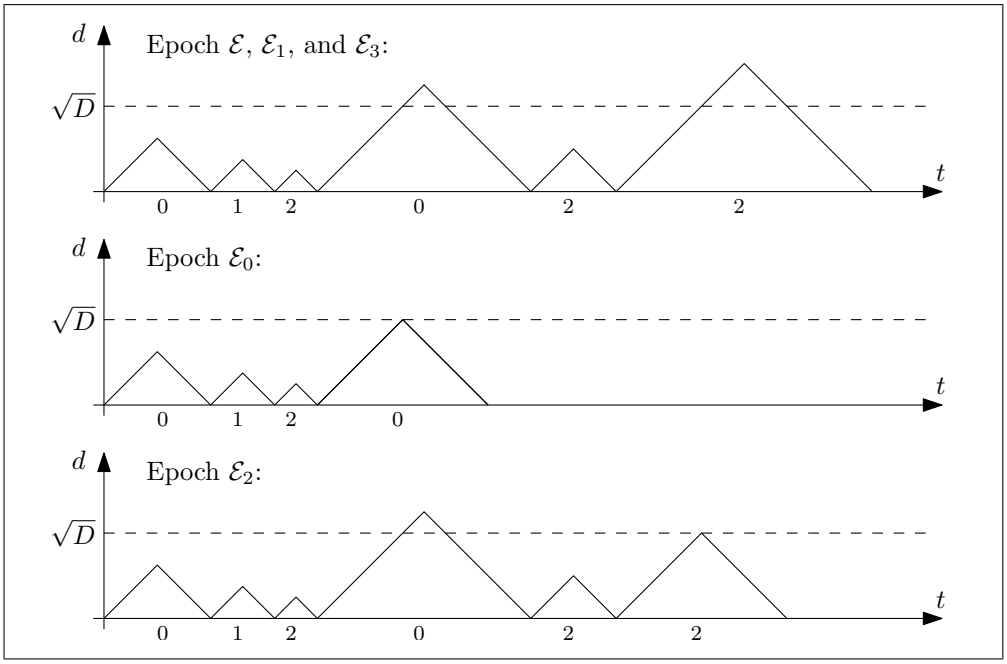

Fig. 4. Example relation (for $n = 4$) between epochs $\mathcal{E}$, $\mathcal{E}_0$, $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$. The distance between a currently active node and remaining nodes is denoted by d. Numbers denote indexes of active nodes.

$C_{\mathrm{DET}}(\mathcal{E})$. As $|\mathcal{V}_{\mathcal{E}}| \geq n/2$, by Lemma 29, we get

$$\sum_{i\in[n]} C_{\mathrm{DET}}(\mathcal{E}_i) = \Omega(n \cdot D) \cdot \sum_{P_j \in \mathcal{E}} X_j \ . \tag{16}$$

On the other hand, we consider the performance of B_i on $\mathcal{E}_i$ for any fixed $i \in [n]$. B_i pays D for the initial movement of the page to v_i. Epoch $\mathcal{E}_i$ contains some phases $P_j \in \mathcal{E}_i$ and optionally a shortened phase at the end. As the length of the shortened phase is $\sqrt{D}$, the cost incurred by serving requests in this phase is $\mathcal{O}(D)$. Thus, $C_{\mathrm{B}_i}(\mathcal{E}_i) = \mathcal{O}(D) + \sum_{P_j \in \mathcal{E}_i} A_i(P_j)$. Summing up over all $i \in [n]$ and applying (15), we obtain

$$\begin{aligned}
\sum_{i\in[n]} C_{\mathrm{B}_i}(\mathcal{E}_i) &= \mathcal{O}(n \cdot D) + \sum_{i\in[n]} \sum_{P_j \in \mathcal{E}_i} A_i(P_j) \\
&= \mathcal{O}(n \cdot D) + \sum_{P_j \in \mathcal{E}} \sum_{i : P_j \in \mathcal{E}_i} A_i(P_j) \\
&= \mathcal{O}(n \cdot D) + \mathcal{O}\left(n + \sqrt{D}\right) \cdot \sum_{P_j \in \mathcal{E}} X_j \ .
\end{aligned} \tag{17}$$

Finally, by combining (16) with (17) and using the fact that $\sum_{P_j \in \mathcal{E}} X_j = \Omega(n \cdot \sqrt{D})$, we obtain (14). □

Now we show how the lemma above implies the lower bound on the competitive ratio.

PROOF of Theorem 27. Fix any randomized algorithm ALG. Again, it is sufficient to show that there exists a strategy of the adversary, which guarantees that the ratio between expected costs of ALG and the adversary is $\Omega(\min\{n \cdot \sqrt{D}, D\})$ in a single epoch.

Let c be a constant hidden in the Ω notation in Lemma 32 and let $L_i = \mathbf{E}\left[C_{\mathrm{ALG}}(\mathcal{E}_i) - c \cdot \min\{n \cdot \sqrt{D}, D\}) \cdot \mathrm{ADV}_i(\mathcal{E}_i)\right]$. Then Lemma 32 states that $\sum_{i \in [n]} L_i \geq 0$. The adversary chooses a strategy index $i^* \in [n]$, which maximizes L_i. By the average argument, $L_{i^*} \geq 0$, which means that in the epoch generated by the adversary ADV_{i^*}, the ratio between expected costs of the algorithm and the adversary is at least $c \cdot \min\{n \cdot \sqrt{D}, D\}$. $\square$

3.3 Lower Bound for Oblivious Adversary

In this section, we prove Theorem 28, i.e., we show an asymptotically optimal lower bound for any algorithm playing against an oblivious adversary. We construct a probability distribution π over inputs (arbitrarily long ones) and prove that each deterministic algorithm (even knowing this distribution) has a high competitive ratio. Then by applying the Yao min-max principle [31], we get that the same lower bound holds for any randomized algorithm against an oblivious adversary. We will use the simple formulation of this principle proved in [18].

Lemma 33 (Yao min-max principle [18]) *Consider any cost minimization problem. Suppose that for arbitrarily large α there exists a probability distribution π over request sequences $\mathcal{I}$, such that for any deterministic algorithm* DET*, it holds that*

(1) $\mathbf{E}_\pi[C_{\mathrm{DET}}(\mathcal{I})] \geq \alpha$ *and*
(2) $\mathbf{E}_\pi[C_{\mathrm{DET}}(\mathcal{I})] \geq \mathcal{R} \cdot \mathbf{E}_\pi[C_{\mathrm{OPT}}(\mathcal{I})]$.

Then no randomized online algorithm is $\mathcal{R}'$-competitive for $\mathcal{R}' < \mathcal{R}$.

Let $K = \min\{\log n, D\}$ and $\gamma = \sqrt{D/K} = \max\{\sqrt{D/\log n}, 1\}$. We show how to randomly create an input of arbitrary length. This will implicitly define a probability distribution π. We divide an input sequence into phases, each of length $2 \cdot K \cdot \gamma + D$ steps. Each phase consists of K *expanding parts* of length γ, a *main part* of length D and a *contracting part* of length $K \cdot \gamma$.

First, we inductively define the behavior of nodes in expanding parts. Let $\mathcal{A}_0$ be the set of all nodes. They remain at point zero at the beginning of any phase. In expanding part i, set $\mathcal{A}_{i-1}$ is divided arbitrarily into two parts of equal size. Nodes from one of these parts, as well as all the nodes above $\mathcal{A}_{i-1}$, are moved up with speed 1, all the other nodes remain at their positions. The

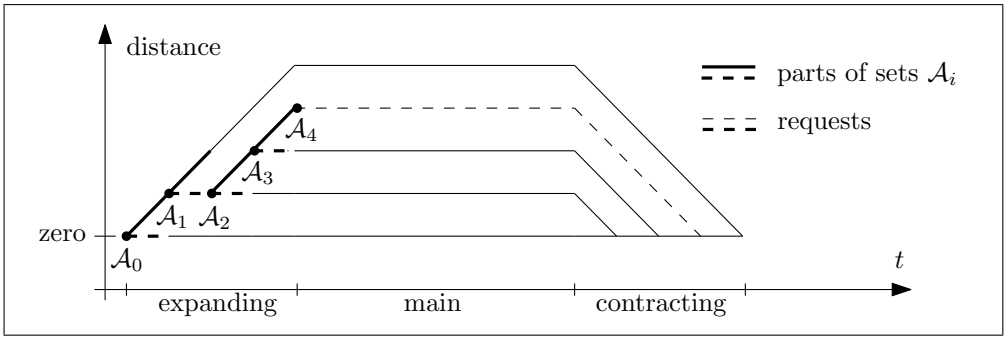

Fig. 5. One phase of a lower bound sequence for $n = 2^4$.

moving part of $\mathcal{A}_{i-1}$ we call an *upper half*, and non-moving one we call a *lower half*. At the end of the expanding part, one of these halves is chosen (each with probability 1/2) and denoted $\mathcal{A}_i$. In the i-th expanding part, all the requests are issued at one of the nodes from the lower half of $\mathcal{A}_{i-1}$. Note that at the end of any expanding part, the coordinate of any node is equal to $k \cdot \gamma$, where k is an integer.

In the main part, nodes do not move. In the contracting part, all the nodes which are above point zero move down with speed 1. In both these parts, requests are given at any node from the set $\mathcal{A}_K$. An example of a phase is presented in Figure 5.

We prove two following lemmas which directly lead to the proof of the lower bound.

Lemma 34 *For any phase P, it holds that $C_{\mathrm{OPT}}(P) = \mathcal{O}(D)$.*

PROOF. Fix any phase P and consider an offline algorithm Off, which moves to an arbitrary node from the set $\mathcal{A}_K$ at the beginning of P. Off pays D for such movement. Note that if $K = \log n$, then $\mathcal{A}_K$ is a singleton set. The distance between P_{OFF} and the requests is 0 in the main part and the contracting part, and at most γ in expanding parts. Therefore, $C_{\mathrm{OPT}}(P) \leq C_{\mathrm{OFF}}(P) \leq D + (D + K \cdot \gamma) + (\gamma + 1) \cdot \gamma \cdot K = \mathcal{O}(D + K \cdot \gamma^2) = \mathcal{O}(D)$. □

Lemma 35 *For any phase P and any deterministic algorithm* Det*, it holds that $\mathbf{E}_\pi[C_{\mathrm{DET}}(P)] = \Omega(D \cdot \min\{\sqrt{D \cdot \log n}, D\})$.*

PROOF. We denote expanding parts of P by $E_1, E_2, \ldots E_K$. First, we prove that at the end of any expanding part j, it holds that

$$\mathbf{E}_\pi[C_{\mathrm{DET}}(E_1, E_2, \ldots, E_j)] + D \cdot \mathbf{E}_\pi[d(\mathcal{A}_j, P_{\mathrm{DET}})] \geq \frac{j \cdot \gamma \cdot D}{2} \ , \qquad (18)$$

where $d(\mathcal{A}_j, P_{\text{DET}})$ is defined as the minimum distance between P_{DET} (the node holding page of DET) and a node from $\mathcal{A}_j$.

The inequality holds trivially for $j = 0$, i.e., at the beginning of the phase. Assume that (18) holds after the j-th expanding part. We separately analyze the change incurred by executing the $(j+1)$-th expanding part and by choosing set A_j at the end of such part.

We forgive DET the cost of serving requests during the expanding parts. Note that if during the $(j+1)$-th expanding part DET does not move, then $d(\mathcal{A}_j, P_{\text{DET}})$ does not change. On the other hand, if it does move along a distance of X, then the cost incurred is $D \cdot (X+1)$ and $d(\mathcal{A}_j, P_{\text{DET}})$ decreases at most by X. In particular, if DET jumps between upper and lower parts of $\mathcal{A}_j$, $d(\mathcal{A}_j, P_{\text{DET}})$ is not changed at all. Thus, the left hand side of (18) can only increase as a consequence of such a movement.

Finally, at the very end of the $(j+1)$-th expanding part, set $\mathcal{A}_{j+1}$ is chosen. For any position of DET's page, $\mathbf{E}_\pi[d(\mathcal{A}_{j+1}, P_{\text{DET}})] = d(\mathcal{A}_j, P_{\text{DET}}) + \frac{1}{2} \cdot \gamma$, and therefore (18) holds.

Now we show how (18) implies a lower bound on DET's cost in the main part of P. Note that within this part, set $\mathcal{A}_K$ occupies one position in the space. Let L be the distance $d(\mathcal{A}_K, P_{\text{DET}})$ at the beginning of the main part. If the distance traveled by DET in the main part is at least $L/2$, then the cost of the corresponding jumps is obviously at least $D \cdot L/2$. Otherwise, DET's distance to $\mathcal{A}_K$ in the whole main part is at least $L/2$, and thus it pays at least $D \cdot L/2$ for serving the requests.

Therefore, $\mathbf{E}_\pi[C_{\text{DET}}(P)] \geq \mathbf{E}_\pi[C_{\text{DET}}(E_1, E_2, \ldots, E_K)] + \frac{1}{2} \cdot D \cdot \mathbf{E}_\pi[L]$. By applying (18), we get that $\mathbf{E}_\pi[C_{\text{DET}}(P)] \geq \frac{1}{4} \cdot K \cdot \gamma \cdot D = \Omega(D \cdot \min\{\sqrt{D \cdot \log n}, D\})$, which finishes the proof. □

PROOF of Theorem 28. Fix any deterministic algorithm DET and any integer ℓ. We choose an input sequence consisting of ℓ phases. By Lemmas 34 and 35, the ratio between expected costs of OPT and DET in one phase is $\Omega(\min\{\sqrt{D \cdot \log n}, D\})$. By the linearity of expectation, the same relation holds for the costs in the whole input sequence. On the other hand, the expected cost of DET can be made arbitrarily large by choosing sufficiently large ℓ.

Hence, if we apply the Yao min-max principle, the lower bound on the competitive ratio, $\Omega(\min\{\sqrt{D \cdot \log n}, D\})$, applies for any randomized algorithm playing against an oblivious adversary. □

4 Faster movement

In this section, we consider the scenario in which the bound on the maximum nodes' speed δ is not a constant, but an additional parameter greater than 1.

Let $\mathcal{R}_{\text{DET}}$, $\mathcal{R}_{\text{AD-ONL}}$, and $\mathcal{R}_{\text{OBL}}$ be the competitive ratios of the best deterministic algorithm, the best randomized algorithm against an adaptive-online adversary, and the best randomized algorithm against an oblivious adversary, respectively.

By Lemma 1 (the Reduction Lemma), we immediately get that our algorithms lose at most factor δ in the competitive ratio if the input sequence is generated by a δ-restricted adversary, i.e., we get that $\mathcal{R}_{\text{AD-ONL}} \leq \mathcal{R}_{\text{DET}} = \mathcal{O}(\delta \cdot \min\{n \cdot \sqrt{D}, D\})$ and $\mathcal{R}_{\text{OBL}} = \mathcal{O}(\delta \cdot \min\{\sqrt{D \cdot \log n}, D\})$.

Obviously, the lower bounds presented in the previous section still work if $\delta \geq 1$, but we may improve them by modifying parameters from the proofs of Section 3. In particular, we show that for $1 \leq \delta \leq D$, it holds that $\mathcal{R}_{\text{DET}} \geq \mathcal{R}_{\text{AD-ONL}} = \Omega(\min\{\sqrt{\delta} \cdot n \cdot \sqrt{D}, \delta \cdot D\})$ and $\mathcal{R}_{\text{OBL}} = \Omega(\min\{\sqrt{\delta} \cdot \sqrt{D \cdot \log n}, D\})$.

An intuition behind the increase of competitive ratio by the factor of $\sqrt{\delta}$ is the following. Our lower bounds charge online algorithms for their jumps and show that the optimal algorithm pays mainly for serving the requests. In our constructions, we increase the distance from 0 to some distance k, which required moving the nodes for k steps. The cost of serving the requests incurred on the adversary was roughly the sum of distances in consecutive steps, i.e., $\sum_{i=0}^{k} i = \Theta(k^2)$. Currently, in $k/\sqrt{\delta}$ steps, the adversary may increase the distance from 0 to $k \cdot \sqrt{\delta}$. Thus the distances increase by the factor of $\sqrt{\delta}$, whereas the cost of serving the requests remains asymptotically the same, i.e., $\sum_{i=0}^{k/\sqrt{\delta}} (\delta \cdot i) = \Theta(k^2)$.

For simplicity, in our proofs, we assume that D is divisible by δ; otherwise, we may round D up and lose at most a constant factor in the analysis.

Theorem 36 *Fix δ, such that $1 \leq \delta \leq D$. For any randomized $\mathcal{R}$-competitive algorithm for the DPM problem playing against a δ-restricted adaptive-online adversary, $\mathcal{R} = \Omega(\min\{\sqrt{\delta} \cdot n \cdot \sqrt{D}, \delta \cdot D\})$.*

PROOF. We focus on showing the lower bound of $\Omega(\min\{\sqrt{\delta} \cdot n \cdot \sqrt{D}, \delta \cdot D\})$ on the competitive ratio of any deterministic algorithm DET. We show this by tuning the parameters in the original proof from Section 3.1.

First, the adversary moves the nodes with speed δ. Second, a phase is called long if the length of its expanding part is at least $\sqrt{D/\delta}$; otherwise we call

it short. Thus, in a short phase the distance between an active node and remaining nodes is at most $\delta \cdot \sqrt{D/\delta} = \sqrt{\delta \cdot D}$. By the changes above, we may restate the properties enumerated in Lemma 29. For any phase P, assuming X is the length of its expanding part, and a is the node active in P, it holds that

(1) $C_{\mathrm{DET}} \geq D \cdot \delta \cdot X$,
(2) if P is a short phase, then $\sum_{i \in [n]} A_i(P) = \mathcal{O}(n + \sqrt{\delta \cdot D}) \cdot X$,
(3) if P is a long phase, then $\sum_{i \in [n] \setminus \{a\}} A_i(P) = \mathcal{O}(n) \cdot X$.

If we plug these new values into the proof of Lemma 30, we get that for any epoch $\mathcal{E}$ consisting of phases $P_1, P_2, \ldots$, it holds that $C_{\mathrm{DET}}(\mathcal{E}) \geq D \cdot \delta \cdot \sum_{P_j \in \mathcal{E}} X = \Omega\left(\sqrt{\delta} \cdot n \cdot D \cdot \sqrt{D}\right)$. On the other hand, we get that

$$\begin{aligned} \sum_{v_i \in \mathcal{V}_\mathcal{E}} C_{\mathrm{B}_i}(\mathcal{E}) &= \sum_{v_i \in \mathcal{V}_\mathcal{E}} (D + A_i(\mathcal{E})) \\ &= \mathcal{O}(n \cdot D) + \mathcal{O}(n + \sqrt{\delta \cdot D}) \cdot \sum_{P_j \in \mathcal{E}} X_j \ . \end{aligned}$$

In effect, by comparing $C_{\mathrm{DET}}(\mathcal{E})$ with $\sum_{v_i \in \mathcal{V}_\mathcal{E}} C_{\mathrm{B}_i}(\mathcal{E})$, we obtain $C_{\mathrm{DET}}(\mathcal{E}) = \Omega(\min\{\sqrt{\delta} \cdot \sqrt{D}, \delta \cdot D/n\}) \cdot \sum_{v_i \in \mathcal{V}_\mathcal{E}} C_{\mathrm{B}_i}(\mathcal{E})$. Finally, by the same average argument as in the original proof, we get the desired bound on DET.

In Section 3.2, we showed how to modify such a lower bound for a deterministic algorithm, so that it works also for randomized algorithms against an adaptive-online adversary. The same modification applies here, yielding the lemma. □

Theorem 37 *Fix δ, such that $1 \leq \delta \leq D$. For any randomized $\mathcal{R}$-competitive algorithm for the DPM problem playing against a δ-restricted oblivious adversary, $\mathcal{R} = \Omega(\min\{\sqrt{\delta} \cdot \sqrt{D \cdot \log n}, D\})$.*

PROOF. Consider the construction of the lower bound from Section 3.3. This time in the construction of a single phase, we set $K = \min\{\log n, D/\delta\}$ and $\gamma = \sqrt{D/(K \cdot \delta)} = \max\{\sqrt{D/(\delta \cdot \log n)}, 1\}$. Again, in each step in which the adversary originally moved a node with speed 1, now it moves it with speed δ. Recall that γ was the number of steps in the expanding part of a phase; the corresponding increase of the distance between group of nodes is therefore $\delta \cdot \gamma$.

Fix any phase P and consider the cost of an offline strategy OFF presented in the original proof of Lemma 34. This time, OFF pays D for the initial movement and $D + K \cdot \gamma$ for serving the requests in the main and contracting parts. Its distance to requests in expanding parts ($K \cdot \gamma$ steps in total) is at most $\delta \cdot \gamma$. Therefore, $C_{\mathrm{OFF}}(P) = \mathcal{O}(D + \delta \cdot K \cdot \gamma^2) = \mathcal{O}(D)$.

When we bounded the cost of the algorithm (see the proof of Lemma 35), we considered only the cost of moving the page and we lower-bounded it by D times the distance along which the page was moved. The terms γ occurring there are now replaced by $\delta \cdot \gamma$. Thus, we get that $\mathbf{E}_\pi[C_{\mathrm{DET}}(P)] = \Omega(K \cdot (\delta \cdot \gamma) \cdot D) = \Omega(D \cdot \sqrt{\delta} \cdot \sqrt{D \cdot K}) = \Omega(D \cdot \min\{\sqrt{\delta} \cdot \sqrt{D \cdot \log n}, D\})$.

By comparing $\mathbf{E}_\pi[C_{\mathrm{DET}}(P)]$ to $C_{\mathrm{OFF}}(P)$, we get the lemma. □

5 File Allocation in Dynamic Networks

In this section, we prove that the competitive ratio of the file allocation (FA) problem [7] in our model of dynamic networks is infinite. In short, file allocation is an extension of page migration, where we are no longer limited to having just one copy of the page in the system. New copies may be created and some copies may be discarded, but at least one copy has to be present in the network. We distinguish between write and read requests. In the case of a read, the requesting processor contacts the nearest node holding the copy, and in case of a write, all the copies have to be updated.

We define the exact costs for a two-node case, as even in such scenario we are able to prove that no online algorithm is able to achieve a finite competitive ratio. Whereas the request sequence in case of the page migration problem was just the sequence of node numbers, now it consists of four possible requests: READ(0), READ(1), WRITE(0), WRITE(1), denoting respectively that nodes v_0 and v_1 want to read from the shared object or write to it. At one step t, exactly one such request appears. Again, local updates are free, and therefore a read request at step t incurs a cost $c_t(v_0, v_1)$ if it is issued at a node not holding the copy and 0 otherwise. A write request incurs a cost $c_t(v_0, v_1)$ if there is a copy at the other node and 0 otherwise. After serving the request, the algorithm may replicate the page to the second node; such transaction incurs a cost $D \cdot c_t(v_0, v_1)$. Algorithm may also remove the copy from a node without paying anything.

We show the lower bound in the strongest sense, i.e., for randomized algorithms against an oblivious adversary. For the following theorem to hold, we assume that our metric space is a real line $\mathbb{R}$ with Euclidean metric. We also use the notation from the previous section.

Theorem 38 *There exists a metric space, such that no randomized algorithm can be competitive for the FA problem in dynamic networks (even against an oblivious adversary).*

PROOF. As mentioned above, we prove the theorem for network consisting of just two nodes on an infinite line. Assume that there exists a k-competitive randomized algorithm ALG for the file allocation problem. We show how to create a nemesis input sequence for ALG. Since the adversary is oblivious, it does not know the exact configuration of ALG, but it may compute the probability that ALG has its copy in a certain node. We denote these probabilities by μ_0 and μ_1, respectively.

The adversary creates an input sequence, divided into phases. We show that there exists an offline solution OFF to this problem, such that for each phase P, it holds that $\mathbf{E}[C_{\text{ALG}}(P)] > k \cdot C_{\text{OFF}}(P)$. Additionally, $\mathbf{E}[C_{\text{ALG}}(P)] = \Omega(D)$ for each phase P. Thus, for any such created input sequence $\mathcal{I}$, we get that $\mathbf{E}[C_{\text{ALG}}(\mathcal{I})] > k \cdot C_{\text{OFF}}(\mathcal{I})$ and the expected cost of ALG can be made arbitrarily large. This would imply that the competitive ratio is greater than k, which contradicts our assumption.

One phase P is constructed in the following way. Let $k' = 4 \cdot k$. At the beginning of P, nodes occupy the same point of the space. The phase consists of a *forcing part*, an *expanding part*, a *main part*, and a *contracting part*. These parts last for D, k', D, and k' steps, respectively. In the forcing and main part, nodes do not move. In the expanding part, v_1 moves up with speed 1; in the contracting part it moves down with the same speed. Thus, the distance between v_0 and v_1 is k' in the main part and 0 in the forcing part.

All the requests in the forcing part are WRITE(0) and all the requests in the expanding part are READ(0). The remaining requests are decided as follows. If throughout the forcing part $\mu_1 \geq \frac{1}{4}$ or at the end of the expanding part $\mu_1 \geq \frac{1}{2}$, then the requests in the main and contracting parts are WRITE(0). Otherwise they are all equal to READ(1).

The intuition behind this construction is as follows. The forcing part forces any reasonable algorithm to move the page to v_0 and discard a copy from v_1. Afterwards, in the expanding part, the algorithm has no information whether it is better to have a copy only at v_0 or to have a copy at each node. This is exploited in the main part.

The algorithm OFF starts and ends each phase with a copy only at node v_0. In the following, we show that the relation $C_{\text{ALG}}(P) > k \cdot C_{\text{OFF}}(P)$ holds for any phase P. We consider three cases.

(1) If throughout the forcing part $\mu_1 \geq \frac{1}{4}$, then within P all the requests (both READ and WRITE) are given at v_0. Thus, if OFF does not move, it pays 0. On the other hand, in expectation, ALG pays at least $\frac{1}{4} \cdot D$ for serving the requests in the forcing part.
(2) If there exists a step in the forcing part with $\mu_1 < \frac{1}{4}$ and at the end of the expanding part $\mu_1 \geq \frac{1}{2}$, then again all the requests are given at v_0,

and $C_{\text{OFF}}(P) = 0$. This time the algorithm has to pay in expectation at least $\frac{1}{4} \cdot D$ for increasing the probability μ_1 from $\frac{1}{4}$ to $\frac{1}{2}$.

(3) If at the end of the expanding part $\mu_1 < \frac{1}{2}$, all the remaining requests are READ(1). In this case, OFF replicates the page to v_1 at the very end of the forcing part and removes this copy at the end of the contracting part. Such a move costs D and ensures that the cost of serving requests is zero. On the other hand, the cost of ALG in the main part can be lower-bounded as follows. With probability $1 - \mu_1 > \frac{1}{2}$, ALG has no copy at v_1. If at some time step of the main part this probability drops below $\frac{1}{4}$, it means that the algorithm had to pay in expectation at least $(\frac{1}{2} - \frac{1}{4}) \cdot (k'+1) \cdot D$ for replicating the page to v_1. Otherwise, the algorithm has to pay in expectation at least $\frac{1}{4} \cdot (k' + 1)$ for each of the READ(1) requests in the main part. Thus, in either case the expected cost of ALG in the main part is at least $\frac{1}{4} \cdot (k' + 1) \cdot D > k \cdot D$.

Therefore, we get that for any phase P, $C_{\text{ALG}}(P) > k \cdot C_{\text{OFF}}(P) \geq k \cdot C_{\text{OPT}}(P)$. This finishes the proof. □

We note that if the metric space $\mathcal{X}$ is bounded, but contains a copy of an interval of length λ, then the construction above implies that the lower bound for file allocation problem is at least $\Omega(\lambda)$. On the other hand, if the maximum distance between the nodes is bounded by a constant λ, then we may apply the $\mathcal{O}(\log n)$-competitive file allocation strategies for static networks [2,7], losing at most an additional factor of λ.

6 Conclusions

This paper aims to bring the dynamic behavior to the world of data management problems in networks. We considered the most basic of these problems, called the Page Migration. By dynamics we mean that the network is subject to small continuous changes, like changes in bandwidth capacity, or the changes in the topology induced by node mobility. These network alterations induce the changes in the costs of communication between pairs of nodes. This paper is a summary of the first papers concerning the *analytic* treatment of this problem. While our model is rather simple, it covers quite a lot of common cases.

Our algorithms exploit topological localities of requests, i.e., they try to adapt to the changing patterns of accesses to the shared object by moving the object "near" the requesting nodes. Our main concern was to construct algorithms which are robust to the network changes. We considered several scenarios, which differed in the way of how the input sequence was created. Using the

competitive analysis, we rigorously analyzed algorithms for each of them, proving their optimality.

A remaining open problem is to investigate further the case where the maximum bound on the nodes' speed is not a constant but a parameter. In particular, it would be interesting to make a smooth transition between static networks (where the speed is zero) and dynamic networks where nodes move with infitesimally small speed.

7 Acknowledgements

This work was partially supported by MNiSW grant number N206 001 31/0436, 2006-2008, MNiSW grant number PBZ/MNiSW/07/2006/46, the EU Marie Curie Research Training Network ADONET, contract no. MRTN-CT-2003-504438, and by the EU within the 6th Framework Programme under Contract 001907 "Dynamically Evolving large Scale Information Systems (DELIS)".

Part of this work was done when Marcin Bienkowski and Miroslaw Korzeniowski were members of the International Graduate School "Dynamic Intelligent systems" at the University of Paderborn, Germany.

A Proofs of Technical Claims

PROOF of Claim 20. Let $f(x) := x \cdot \log \frac{1}{x}$. It is easy to check that f is continuous and concave for all $x > 0$. Therefore, we can apply Jensen's Inequality (see [24]) to get $f\left(\sum_i p_i \cdot x_i\right) \geq \sum_i p_i \cdot f(x_i)$ for any $x_i > 0$ and for $0 \leq p_i \leq 1$, such that $\sum_i p_i = 1$.

Let $p_i = 2^{-a_i} / \sum_k 2^{-a_k}$ and $x_i = 2^{-b_i + a_i}$. Then, $\sum_i p_i \cdot x_i = \sum_i 2^{-b_i} / \sum_i 2^{-a_i}$, and therefore

$$\begin{aligned}\frac{\sum_i 2^{-b_i}}{\sum_i 2^{-a_i}} \cdot \log \frac{\sum_i 2^{-a_i}}{\sum_i 2^{-b_i}} &\geq \sum_i \frac{2^{-a_i}}{\sum_k 2^{-a_k}} \cdot 2^{-b_i + a_i} \cdot \log\left(\frac{1}{2^{-b_i + a_i}}\right) \\ &= \frac{\sum_i 2^{-b_i}}{\sum_k 2^{-a_k}} \cdot (b_i - a_i) \ .\end{aligned}$$

By multiplying both sides by $\sum_i 2^{-a_i} / \sum_i 2^{-b_i}$, we get the claim. □

References

[1] S. Albers, H. Koga, New on-line algorithms for the page replication problem, Journal of Algorithms 27 (1) (1998) 75–96, also appeared in *Proc. of the 4th SWAT*, pages 25–36, 1994.

[2] B. Awerbuch, Y. Bartal, A. Fiat, Competitive distributed file allocation, in: Proc. of the 25th ACM Symp. on Theory of Computing (STOC), 1993.

[3] B. Awerbuch, Y. Bartal, A. Fiat, Heat & dump: Competitive distributed paging, in: Proc. of the 34th IEEE Symp. on Foundations of Computer Science (FOCS), 1993.

[4] B. Awerbuch, Y. Bartal, A. Fiat, Distributed paging for general networks, Journal of Algorithms 28 (1) (1998) 67–104, also appeared in *Proc. of the 7th SODA*, pages 574–583, 1996.

[5] Y. Bartal, Distributed paging, in: Dagstul Workshop on On-line Algorithms, 1996.

[6] Y. Bartal, M. Charikar, P. Indyk, On page migration and other relaxed task systems, Theoretical Computer Science 268 (1) (2001) 43–66, also appeared in *Proc. of the 8th SODA*, pages 43–52, 1997.

[7] Y. Bartal, A. Fiat, Y. Rabani, Competitive algorithms for distributed data management, Journal of Computer and System Sciences 51 (3) (1995) 341–358, also appeared in *Proc. of the 24nd STOC*, pages 39–50, 1992.

[8] S. Ben-David, A. Borodin, R. M. Karp, G. Tardos, A. Wigderson, On the power of randomization in online algorithms, in: Proc. of the 22nd ACM Symp. on Theory of Computing (STOC), 1990.

[9] M. Bienkowski, Dynamic page migration with stochastic requests, in: Proc. of the 17th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA), 2005.

[10] M. Bienkowski, Page migration in dynamic networks, Ph.D. thesis, Universität Paderborn (2005).

[11] M. Bienkowski, J. Byrka, Bucket game with applications to set multicover and dynamic page migration, in: Proc. of the 13th European Symp. on Algorithms (ESA), 2005.

[12] M. Bienkowski, M. Dynia, M. Korzeniowski, Improved algorithms for dynamic page migration, in: Proc. of the 22nd Symp. on Theoretical Aspects of Computer Science (STACS), 2005.

[13] M. Bienkowski, M. Korzeniowski, Dynamic page migration under brownian motion, in: Proc. of the European Conf. in Parallel Processing (Euro-Par), 2005.

[14] M. Bienkowski, M. Korzeniowski, F. Meyer auf der Heide, Fighting against two adversaries: Page migration in dynamic networks, in: Proc. of the 16th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA), 2004.

[15] M. Bienkowski, F. Meyer auf der Heide, Page migration in dynamic networks, in: Proc. of the 30th Int. Symp. on Mathematical Foundations of Computer Science (MFCS), 2005, invited paper.

[16] D. L. Black, D. D. Sleator, Competitive algorithms for replication and migration problems, Tech. Rep. CMU-CS-89-201, Department of Computer Science, Carnegie-Mellon University (1989).

[17] A. Borodin, R. El-Yaniv, Online Computation and Competitive Analysis, Cambridge University Press, 1998.

[18] M. Chrobak, L. L. Larmore, C. Lund, N. Reingold, A better lower bound on the competitive ratio of the randomized 2-server problem, Information Processing Letters 63 (2) (1997) 79–83.

[19] M. Chrobak, L. L. Larmore, N. Reingold, J. Westbrook, Page migration algorithms using work functions, Journal of Algorithms 24 (1) (1997) 124–157, also appeared in *Proc. of the 4th ISAAC*, pages 406–415, 1993.

[20] R. Fleischer, W. Głazek, S. S. Seiden, New results for online page replication, Theoretical Computer Science 324 (2–3) (2004) 219–251.

[21] R. Fleischer, S. S. Seiden, New results for online page replication, in: Proc. of the 3rd Int. Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX), 2000.

[22] W. Głazek, Lower and upper bounds for the problem of page replication in ring networks, in: Proc. of the 24th Int. Symp. on Mathematical Foundations of Computer Science (MFCS), 1999.

[23] W. Głazek, Online algorithms for page replication in rings, Theoretical Computer Science 268 (1) (2001) 107–117.

[24] G. H. Hardy, J. E. Littlewood, G. Pólya, Inequalities, 2nd ed., Cambridge University Press, 1988.

[25] H. Koga, Randomized on-line algorithms for the page replication problem, in: Proc. of the 4th Int. Symp. on Algorithms and Computation (ISAAC), 1993.

[26] C. Lund, N. Reingold, J. Westbrook, D. C. K. Yan, Competitive on-line algorithms for distributed data management, SIAM Journal on Computing 28 (3) (1999) 1086–1111, also appeared as On-Line Distributed Data Management in *Proc. of the 2nd ESA*, pages 202–214, 1994.

[27] R. Rajaraman, Topology control and routing in ad hoc networks: a survey, SIGACT News 33 (2) (2002) 60–73.

[28] C. Scheideler, Models and techniques for communication in dynamic networks, in: Proc. of the 19th Symp. on Theoretical Aspects of Computer Science (STACS), 2002.

[29] D. D. Sleator, R. E. Tarjan, Amortized efficiency of list update and paging rules, Communications of the ACM 28 (2) (1985) 202–208.

[30] J. Westbrook, Randomized algorithms for the multiprocessor page migration, SIAM Journal on Computing 23 (1994) 951–965, also appeared in *Proc. of the DIMACS Workshop on On-Line Algorithms*, pages 135–149, 1992.

[31] A. C.-C. Yao, Probabilistic computation: towards a uniform measure of complexity, in: Proc. of the 18th IEEE Symp. on Foundations of Computer Science (FOCS), 1977.