

Fighting Against Two Adversaries: Page Migration in Dynamic Networks

(full version of the paper)*

Marcin Bienkowski[†]
International Graduate School
of Dynamic Intelligent Systems
University of Paderborn,
Germany
young@upb.de

Miroslaw Korzeniowski[†]
International Graduate School
of Dynamic Intelligent Systems
University of Paderborn,
Germany
rudu@upb.de

Friedhelm Meyer auf der Heide[†]
Heinz Nixdorf Institute and
Computer Science Department
University of Paderborn,
Germany
fmadh@upb.de

ABSTRACT

Page migration is one of the fundamental subproblems in the framework of data management in networks. It occurs in a distributed network of processors sharing one indivisible memory page of size D , which is stored in one of the processors. During runtime, processors access unit size data items from the page, and the system is allowed to move the page from one processor to another in order to minimize the total communication cost.

This problem was considered in the online setting numerous times by many researchers, and some online algorithms were proven to achieve a cost within a constant factor of the optimal *offline* solution. However, all results were achieved under the assumption that the communication costs between processors were fixed during the execution of the whole process.

In this paper we consider a model in which the communication costs can change in each time step, but the pace of the changes is restricted. This is typical in mobile networks, and also models the dynamics of networks that are not exclusively dedicated to the page migration.

If both changes of the network and the request sequence are given by some adversarial entity, we prove a tight bound on the competitive ratio of the problem. However, the size of this ratio motivates us to assume that the changes of communication costs are modeled by some stochastic process, and an adversary dictates only which processor issues a request. To analyze such a hybrid case, we introduce the notion of expected competitive ratio and prove that, for the case where constant number of processors perform a random walk on a torus or on a mesh of diameter \sqrt{D} , it is $\mathcal{O}(\log^2 D)$.

Categories and Subject Descriptors

E.1 [Data Structures]: Graphs and networks; F.2.2 [Non-numerical Algorithms and Problems]: Computations on discrete structures; G.2.2 [Graph Theory]: Network problems; G.3 [Probability and Statistics]: Markov processes, Probabilistic algorithms, Stochastic processes

General Terms

Algorithms, Theory

Keywords

online algorithms, data management, page migration

1. INTRODUCTION

Data Management problems arise in a distributed network of processors which share some global memory. Copies of shared variables are stored in some of these processors. In this context a variable can be a single program variable, a system database, a conventional file, a memory page, etc. When a processor wants to access (read or write) a single data item of the file, it has to send a request to the processor holding a copy and appropriate data is sent back to it. The cost incurred by such an operation is a function of the distance between these two processors. Copies of the variable can be replicated and deleted to reduce communication costs. Such transactions incur a cost which is equal to the cost of sending one data item times the size of the variable. In any case we have to preserve the consistency between the copies. The algorithm for the data management decides where and when the copies should be moved or replicated to, or deleted.

The file allocation problem considered in the online setting by Bartal *et al* in [4] and Awerbuch *et al* in [1] is a fundamental case of data management, where we have a single file (and its copies) in the system. One basic subproblem of the file allocation is *page migration* introduced by Black and Sleator in [7], where only one copy of the file can be kept in the network. While considering this problem, we do not have to be concerned about inconsistencies between the copies, and, as proven in [1, Thm 4.2], the solution to the page migration problem applies to the file allocation problem, where only write requests are allowed.

*The extended abstract of this paper appeared in the proceedings of 16th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)

[†]Partially supported by DFG-Sonderforschungsbereich 376 “Massive Parallelität: Algorithmen Entwurfsmethoden Anwendungen”, and by the Future and Emerging Technologies programme of the EU under EU Contract 001907 DELIS “Dynamically Evolving, Large Scale Information Systems”.

In this paper we extend the page migration problem by allowing nodes to move. This is typical in mobile networks, and also models the dynamics of networks that are not exclusively dedicated to the page migration. We assume that processors are nodes with some position in a metric space (\mathcal{X}, d) . This induces a complete graph of all nodes, with the distances given by the metric d . In the original problem, to which we will refer as *static page migration*, the positions of the points are fixed and the cost of sending one data item between two nodes is equal to the distance between these two nodes. In each round, exactly one processor issues a request for accessing a data item from the page. We allow movement of the nodes. However, we restrict the speed of the network changes, namely in each round the new position of a node is chosen within the ball of a constant diameter centered around its current position. We define the cost of accessing a data item as the distance between the requesting node and the node holding the page plus a constant overhead for communication. Moving the page has a cost equal to the distance between the old and the new place times the size of the page plus a constant overhead for communication.

1.1 Our model

We define the *dynamic page migration* problem as follows. The network is modeled as a set V of n nodes (processors) numbered from 0 to $n - 1$ in some metric space (\mathcal{X}, d) . The distances between the nodes are given by the metric d , however, we extend the notion of the distance between two nodes in the following way. If v_i and v_j are the same node, then we denote it by $v_i \equiv v_j$ and we denote the distance between them by 0_E . Note, that this is different from the case where they just occupy the same point in the space, in which case we write $v_i = v_j$ and $d(v_i, v_j) = 0$. The positions of all nodes in a time step t describe the *configuration at time t* .

The cost of sending a data item from node v_x to node v_y is defined by a cost function $c(\cdot)$. $c(v_x, v_y) = 0$ if $v_x \equiv v_y$. Otherwise $c(v_x, v_y) = d(v_x, v_y) + 1$. We have one shared, indivisible memory page of size D , initially stored at the node $v_0 \in V$. The cost of moving the whole page from v_x to v_y is equal to $D \cdot c(v_x, v_y)$.

We assume discrete time steps $t = 0, 1, \dots$. An input consists of a *request sequence* (σ_t) and a *configuration sequence* (C_t) . σ_t denotes the node that issues a request at time t , and C_t is the configuration at time t . We will call C_0 the *initial configuration*. We impose restrictions on how much two consecutive configurations can differ. For any node v_x , its positions x_t and x_{t+1} in two consecutive configurations C_t, C_{t+1} cannot be too far apart, we demand that $d(x_j, x_{j+1}) \leq 1$. In time step $t \geq 0$, the following happens

1. The positions of the nodes for this time step are defined by C_t .
2. Node σ_t issues a request.

The node holding the page at the beginning of time step t is denoted by $P_{\text{ALG}}(t)$. In each time step, the algorithm for dynamic page migration has to serve the request paying the cost $c(P_{\text{ALG}}(t), \sigma_t)$. Then it can optionally move the page to a new position $P'_{\text{ALG}}(t)$ paying the cost $D \cdot c(P_{\text{ALG}}(t), P'_{\text{ALG}}(t))$. Sometimes, we will abuse notation by writing that an algorithm is at v_i or jumps to v_j , meaning that this algorithm has its page at v_i or moves its page to v_j .

The optimal cost of the offline version of this problem (where the algorithm is allowed to read the whole input at once) can be computed in the time polynomial in n and T , where T is the number of time steps. In fact, a simple algorithm based on dynamic programming has the running time of $\mathcal{O}(n^2 \cdot T)$.

In this paper we will focus on the online version of the problem, i.e. the algorithm has to base its decision what to do in step t solely on the initial parts of the two input sequences up to step t .

The input for the online problem is generated by two adversaries. The *configuration adversary* creates a configuration sequence (C_t) , deciding, for each time step t and for each node, to which position (within distance one) it moves. The *request adversary* creates a request sequence (σ_t) specifying, for each time step t , the node which issues a request. To make the description of the online problem complete, we have to specify the capabilities of the adversaries, how and when they can generate their sequences, and if they are allowed to cooperate and combine their efforts. The strongest type of adversary considered is *adaptive-online*, i.e. the one which can choose the elements of the sequence during the run of the algorithm, depending on the algorithm's past behavior. The adaptive-online adversary has to provide an *answering part* which gives answers online to the requests generated by the adversary. A weaker adversary is an *oblivious* one, which has to generate its whole input sequence in advance. We will also consider scenarios where we replace one of the adversaries by a stochastic process, which generates a sequence according to some probability distribution. We will refer to the last one as *randomized adversary*. In this paper we consider three scenarios

1. **Adaptive cooperative.** Both configuration and request adversaries are adaptive-online and they cooperate.
2. **Oblivious cooperative.** Both configuration and request adversaries are oblivious and they cooperate.
3. **Hybrid non-cooperative.** The configuration adversary is randomized, the request adversary is oblivious and they do not cooperate, i.e. the request adversary chooses its sequence before the configuration sequence is generated.

The motivation for the third scenario is the insight that it seems reasonable to assume that the network itself does not play against our algorithm (by changing distances between nodes).

1.2 Performance measures

In order to analyze the performance in the two first scenarios we use the classical competitive analysis [14, 8]. In case of oblivious adversaries the competitive ratio is defined as the maximum over all allowed combinations of request and configuration sequences, of the ratio of the online algorithm's cost to the optimal offline cost. In case of adaptive-online adversaries we consider the ratio of the cost of the online algorithm to the cost of the answering part of the adversary. We denote them by the cost of the algorithm and the cost of the adversary, respectively. It is straightforward (see e.g. [6]) that the adaptive, cooperative adversary is more powerful than the oblivious, cooperative one.

For the hybrid non-cooperative scenario we introduce a notion of *expected competitive ratio*. Let $C_{\text{ALG}}(\sigma_t, C_t)$ and $C_{\text{OPT}}(\sigma_t, C_t)$ denote the costs of a deterministic online algorithm ALG and the optimal offline algorithm, respectively, both computed on input sequence (σ_t, C_t) . We define the expected competitive ratio as the maximum over all sequences (σ_t) generated by the request adversary, of the expected value of $C_{\text{ALG}}(\sigma_t, C_t)$ -to- $C_{\text{OPT}}(\sigma_t, C_t)$ -ratio. The expectation is taken over all possible random choices of the randomized configuration adversary. We extend this notion slightly and allow a constant additive term in the cost of the online algorithm. Formally, we say that ALG achieves the expected competitive ratio \mathcal{R}_{ALG} , if there exists a constant $c \geq 0$ such that, for all choices of (σ_t) , the following holds.

$$\mathbf{E}_{(C_t)} \left[\frac{C_{\text{ALG}}(\sigma_t, C_t) - c}{C_{\text{OPT}}(\sigma_t, C_t)} \right] \leq \mathcal{R}_{\text{ALG}}$$

We note that this notion is much stronger than the ratio of the expected cost of the algorithm to the expected cost of the adversary.

1.3 Contribution of this paper

In this paper we describe algorithms for dynamic page migration and analyze them with respect to the three scenarios defined above. In Section 2 we present an algorithm for the *adaptive cooperative* scenario. Its competitive ratio is $\mathcal{O}(\min\{n \cdot \sqrt{D}, D, \lambda\})$, where n is the number of nodes, D is the size of the page and λ is the maximum distance between any pair of nodes in the whole history of requests. We further prove a matching lower bound in this model, showing that our algorithm is asymptotically optimal.

In Section 3 we consider the *oblivious cooperative* scenario. It is clear that the above upper bound also holds here. We prove a lower bound of $\Omega(\min\{\sqrt{D}, \lambda\})$ for the competitive ratio in this scenario.

For the *hybrid non-cooperative* scenario, considered in Section 4, we can prove that, for a constant number of nodes, if each node performs a random walk on a d -dimensional torus of diameter \sqrt{D} , then there exists an algorithm achieving expected competitive ratio of $\mathcal{O}(d \cdot \log^2 D)$. We extend this result to a d -dimensional mesh of diameter $\Theta(\sqrt{D})$, which models mobile nodes moving on a bounded territory.

This work is one of few papers which combine the competitive analysis of an online problem with the average case analysis for the stochastic process. The problem considered is one of few natural problems which use two distinct and independent resources (in our case network distances and nodes issuing requests), so that the input can be divided into two parts in a natural manner. We believe that the expected competitive ratio constitutes a reasonable performance criterion for an online algorithm and can be applied also to other problems where some part of the input is given by an adversary and some is generated stochastically.

1.4 Related work

Bartal, Fiat and Rabani [4] and Awerbuch, Bartal and Fiat [1] gave tight bounds for the file allocation problem. Particularly, they proved that the competitive ratio of this problem is $\Theta(\log n)$, both for randomized and deterministic algorithms.

Westbrook [15] gave first $\mathcal{O}(1)$ -competitive algorithms for the randomized page migration problem, both against oblivious and adaptive adversaries. This result was improved

for some network topologies like trees or uniform graphs by Chrobak *et al* [9]. The first constant competitive deterministic algorithm was given by Awerbuch *et al* in [1] and the competitive ratio was subsequently improved by Bartal, Charikar and Indyk [3]. To the best of our knowledge no page migration related problem was considered before in non-static networks.

Scharbrodt, Schickinger and Steger [13] also consider expected competitive ratio, however, they do it in relation to a scheduling problem, where the whole input sequence is generated randomly. More similar to our approach is a paper by Becchetti *et al* [5], where the input sequence is generated by an adversary, but then a random distortion is added to this sequence.

2. ADAPTIVE COOPERATIVE

In this section we present an online algorithm for the adaptive cooperative scenario and prove that its competitive ratio is asymptotically optimal.

2.1 Upper bound

In this section we will prove that the competitive ratio of the dynamic page migration problem against adaptive-online cooperating adversaries is $\mathcal{O}(\min\{n \cdot \sqrt{D}, D, \lambda\})$, where λ is the maximal allowed distance in the graph. We begin with the algorithm which achieves a competitive ratio of $\mathcal{O}(n \cdot \sqrt{D})$.

All known competitive randomized algorithms for static page migration (an excellent overview of file allocation related problems can be found in Bartal survey [2]) were based on the idea of moving the page with a certain probability towards the node which issued the request. We could develop this idea as follows. If the algorithm keeps a page in a node P_{ALG} , different from node P_{R} issuing the request, then the algorithm moves the page towards P_{R} with a probability which depends only on the distance between P_{ALG} and P_{R} . However, it appears to be not sufficient, i.e. no algorithm which considers moving its page only to the node issuing the request can be better than $\Omega(\min\{D, \lambda\})$ competitive. The proof of this fact is partially covered in the proof of the Theorem 6 and we omit it here. One of the remedies to that, is to move the page to a node chosen randomly from the nodes contained in a surroundings of P_{R} . These intuitions are formalized in the algorithm ALG_{DIST} depicted in Figure 1.

The algorithm ALG_{DIST} works as follows. After serving a request, it chooses randomly a new node P_{ALG}' for moving its page. Let X be the distance between P_{ALG} and P_{R} . First, the algorithm computes the probability $B(X)$, that the page will be moved at all. With the probability $B(X)$ it proceeds, and picks (uniformly at random) P_{ALG}' as a node contained in a ball of diameter $\frac{1}{4} \cdot X$ centered at the node P_{R} , and then jumps to P_{ALG}' . We will call $\mathcal{B}(x)$ a *jump function*. It is straightforward that $\mathcal{B}(x)$ is continuous and bounded by $b \cdot \frac{1}{D} \leq \mathcal{B}(x) \leq b \cdot \frac{1}{D} \cdot \sqrt{D}$, where b is a constant which will be specified later. It is possible to prove that ALG_{DIST} is competitive.

THEOREM 1. *There exists a constant b such that algorithm ALG_{DIST} achieves a competitive ratio of $\mathcal{O}(n\sqrt{D})$ in the adaptive cooperative variant of the dynamic page migration problem.*

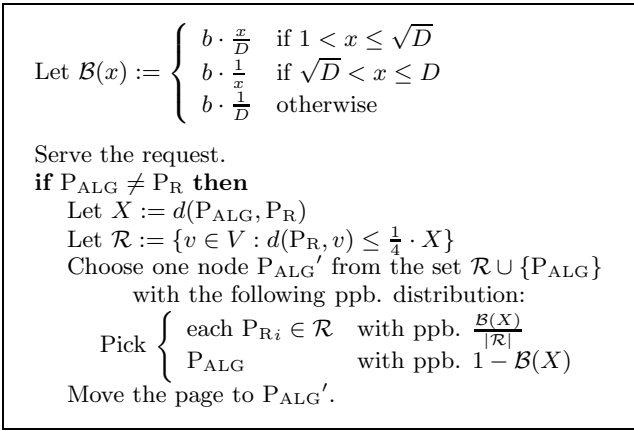


Figure 1: Algorithm ALG_{DIST}

PROOF. In the analysis of algorithm ALG_{DIST} we are not aiming at minimizing the constants but rather at the simplicity of the proof. For proving competitiveness of this algorithm we will use an amortized analysis and a potential function. The potential function is 0 if the algorithm and the adversary keep their pages at the same node ($P_{\text{ALG}} \equiv P_{\text{OPT}}$), otherwise it depends solely on the distance between these nodes. For any time step t we will denote these nodes by $P_{\text{ALG}}(t)$ and $P_{\text{OPT}}(t)$, respectively. Formally, let the potential in the time step t be defined as $\Phi_t = \mathcal{F}(d(P_{\text{ALG}}(t), P_{\text{OPT}}(t)))$ and

$$\mathcal{F}(x) = \begin{cases} 0 & \text{if } x = 0_E \\ \mathcal{F}_B + \mathcal{F}'(x) & \text{otherwise} \end{cases}$$

where

$$\mathcal{F}_B = f \cdot n \cdot D \cdot \sqrt{D},$$

$$\mathcal{F}'(x) = f \cdot x \cdot \sqrt{D} \cdot \min\{x, \sqrt{D}\}.$$

and f is some constant which will be fixed later. It is clear that $\Phi_0 = \mathcal{F}(0_E) = 0$, because the adversary and the algorithm start from the same node, and that for any t , $\Phi_t \geq 0$. Let $C_{\text{ALG}}(t)$ and $C_{\text{OPT}}(t)$ be the cost of the algorithm and the adversary, respectively, in this time step. We prove that in each round the amortized cost of algorithm ALG_{DIST} is smaller than the cost of the adversary times $\mathcal{O}(n \cdot \sqrt{D})$. For clarity, in the following we will omit t indexes in C_{ALG} and C_{OPT} . Furthermore, we denote $\Phi_t - \Phi_{t-1}$ by $\Delta\Phi$. Formally, we show that for $b = 40$ and $f = 22$, for any time step, the following inequality holds

$$\mathbf{E}[C_{\text{ALG}} + \Delta\Phi] = \mathcal{O}(n \cdot \sqrt{D}) \cdot C_{\text{OPT}}. \quad (1)$$

where the expected value is taken over the random choices of the algorithm. After summing this inequality over all time steps we get that the algorithm is $\mathcal{O}(n \cdot \sqrt{D})$ competitive.

To assure that the probability given by the jump function $\mathcal{B}(x)$ is valid, i.e. not greater than 1, we assume that $\sqrt{D} \geq b$. Otherwise, we easily get constant competitiveness of the problem. We divide each time step into two parts:

1. The algorithm and the adversary serve the request, the algorithm optionally moves the page
2. The adversary optionally moves the page.

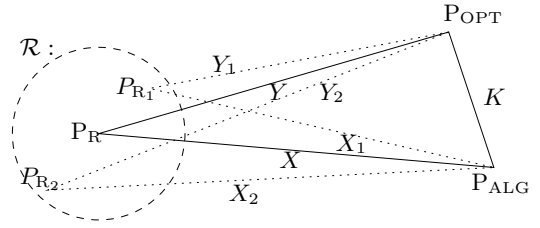


Figure 2: Illustration of algorithm ALG_{DIST}

For each part we prove separately that the Inequality 1 holds. We begin with the first one, i.e. we assume that the adversary does not move its page. Let P_{ALG} and P_{OPT} denote the nodes where the algorithm and the adversary keep their pages and P_{R} be the node which issues a request. Let X , Y and K be the distances between P_{ALG} and P_{R} , P_{OPT} and P_{R} , and P_{ALG} and P_{OPT} , respectively. Additionally K_0 denotes the distance between P_{ALG} and P_{OPT} at the beginning of the time step (before the adversary moves the nodes).

Let \mathcal{R} be defined as in Figure 1, i.e. as the set of all points whose distance to P_{R} is not larger than $X/4$. For each point $P_{\text{R}_i} \in \mathcal{R}$ we define X_i and Y_i as the distances of these points to P_{ALG} and P_{OPT} , respectively, i.e. $X_i = d(P_{\text{R}_i}, P_{\text{ALG}})$ and $Y_i = d(P_{\text{R}_i}, P_{\text{OPT}})$. An example is given in Figure 2. We also denote $|\mathcal{R}|$ by m . Since $P_{\text{ALG}} \notin \mathcal{R}$, $m \leq n-1$. It follows from the triangle inequality that $X_i \leq X + X/4 = \frac{5}{4}X$. Then we have the following costs:

$$\begin{aligned} C_{\text{OPT}} &= c(Y) \\ \mathbf{E}[C_{\text{ALG}}] &= c(X) + \sum_{P_{\text{R}_i} \in \mathcal{R}} \frac{\mathcal{B}(X)}{m} \cdot D \cdot c(X_i) \\ \mathbf{E}[\Delta\Phi] &= \mathcal{F}(K) - \mathcal{F}(K_0) \\ &\quad + \sum_{P_{\text{R}_i} \in \mathcal{R}} \frac{\mathcal{B}(X)}{m} \cdot [\mathcal{F}(Y_i) - \mathcal{F}(K)] \end{aligned} \quad (2)$$

$\mathcal{F}(K) - \mathcal{F}(K_0)$ is the change in the potential induced when the adversary changes the cost of the edge connecting P_{ALG} with P_{OPT} . The rest of the Φ change is caused by the algorithm moving the page with probability $\mathcal{B}(X)/m$ to each of the points in \mathcal{R} .

First, we note that Inequality 1 is trivially fulfilled if $P_{\text{ALG}} \equiv P_{\text{R}}$, i.e. the request is at the same node as the algorithm. In that case the algorithm pays nothing for serving the request, it does not jump to other nodes, there is no change in the potential, and thus we have $\mathbf{E}[C_{\text{ALG}}] + \mathbf{E}[\Delta\Phi] = 0 \leq C_{\text{OPT}}$. Therefore, for the remaining part we assume that $P_{\text{ALG}} \neq P_{\text{R}}$. The two cases, considered in the lemmas below, are proven in the appendix.

LEMMA 2. *If $P_{\text{ALG}} \equiv P_{\text{OPT}}$, then it holds $\mathbf{E}[C_{\text{ALG}}] + \mathbf{E}[\Delta\Phi] = \mathcal{O}(n \cdot \sqrt{D}) \cdot C_{\text{OPT}}$.*

LEMMA 3. *If $P_{\text{ALG}} \neq P_{\text{OPT}}$, then it holds $\mathbf{E}[C_{\text{ALG}}] + \mathbf{E}[\Delta\Phi] = \mathcal{O}(\sqrt{D}) \cdot C_{\text{OPT}}$.*

Now we prove that the Inequality 1 holds also in the second part of each time step, in which the adversary optionally moves the page. In this part $C_{\text{ALG}} = 0$ and we have already taken into account the changes in the potential due to the changing distance between P_{ALG} and P_{OPT} . Assume that the adversary moves its page to a node $P_{\text{OPT}'}$. We reuse

notation from the previous proof, i.e. let K , X and Y denote the distances between P_{ALG} and P_{OPT} , P_{ALG} and $P_{\text{OPT}'}$ and P_{OPT} and $P_{\text{OPT}'}$, respectively. Since the adversary does not jump to the same node it is already in, $Y \neq 0_E$. Then $C_{\text{OPT}} = D \cdot c(Y) = D \cdot (1 + Y)$ and $\Delta\Phi = \mathcal{F}(X) - \mathcal{F}(K)$. Obviously $C_{\text{ALG}} = 0$ and therefore it is sufficient to prove that

$$(*) := \mathcal{F}(X) - \mathcal{F}(K) \leq \mathcal{O}(n\sqrt{D}) \cdot D \cdot (1 + Y) .$$

We assume that $X \geq K$, otherwise the inequality above is trivially fulfilled. We consider three cases.

1. If $K = 0_E$ or $K = 0$, then $Y = X$ and we have

$$(*) = \mathcal{F}(X) \leq fnD \cdot \sqrt{D} + fDY \leq f \cdot n\sqrt{D} \cdot D \cdot (1 + Y).$$

2. If $0 < K \leq \sqrt{D}$, then from the triangle inequality follows that $X \leq Y + K \leq Y + \sqrt{D}$ and using the fact that $\mathcal{F}'(x) \leq fD \cdot x$ we have

$$(*) \leq \mathcal{F}'(X) \leq \mathcal{F}'(Y + \sqrt{D}) = fD \cdot (Y + \sqrt{D}) .$$

If $Y < \sqrt{D}$, then $(*) \leq fD \cdot (Y + \sqrt{D}) < 2f\sqrt{D} \cdot D$. Otherwise, $(*) \leq fD \cdot (Y + \sqrt{D}) \leq 2f \cdot D \cdot Y$.

3. If $K > \sqrt{D}$, then $\mathcal{F}(X) - \mathcal{F}(K) = fD(X - K) \leq fDY$.

In either case it holds $\mathcal{F}(X) - \mathcal{F}(K) \leq \mathcal{O}(n\sqrt{D}) \cdot D \cdot (1 + Y)$ which finishes the proof of Theorem 1. \square

We can also construct an algorithm which achieves competitive ratio of $\mathcal{O}(D)$. Let ALG_{JUMP} be an algorithm which upon receiving a request from the node P_{R} serves this request and jumps to P_{R} . Then we have the following theorem, which is proven in the appendix.

THEOREM 4. ALG_{JUMP} achieves competitive ratio $\mathcal{O}(D)$.

For proving the existence of the algorithms achieving a trivial competitive ratio of $\mathcal{O}(\lambda)$ we take any algorithm which achieves a constant competitive ratio on static networks (for example the 3-competitive algorithm against adaptive-online adversaries shown in [15]) and we run it with the given request sequence on the uniform graph with all costs of communication between nodes equal to 1. The outcome of this algorithm, i.e. the decisions where and when to move the page, we apply directly in the original dynamic graph. It is straightforward that the competitive ratio of such algorithm is at most $3\lambda = \mathcal{O}(\lambda)$. Moreover, from the analysis of the potential function used in [15] we can conclude that it is possible to combine these algorithms. Precisely, it is possible to choose such constants in the potential functions, so that if we run $\mathcal{O}(\lambda)$ -competitive algorithm till the maximum distance in the graph exceeds D (or $n \cdot \sqrt{D}$, whichever is smaller) and then switch to algorithm ALG_{DIST} (or respectively ALG_{JUMP}) then the change of the potential during switching is not greater than 0. Thus, we proved the following theorem.

THEOREM 5. *There exists a randomized algorithm for the page migration problem which, upon knowing the constant approximation of n , achieves a competitive ratio of $\mathcal{O}(\min\{n\sqrt{D}, D, \lambda\})$.*

2.2 Lower bound

We can also prove the matching lower bound for the adaptive cooperative scenario of the dynamic page migration problem.

THEOREM 6. *For any c -competitive algorithm for the dynamic page migration problem playing against cooperating adaptive-online configuration and sequence adversaries, $c = \Omega(\min\{n \cdot \sqrt{D}, D, \lambda\})$.*

PROOF. We assume that $D \geq 16$, otherwise the theorem trivially holds. We also assume that \sqrt{D} is an integer, otherwise we could use $\lfloor \sqrt{D} \rfloor$ and lose only a constant factor in the analysis. Furthermore, we assume that $n \geq 3$ and $\lambda \geq \sqrt{D}$. For the case where $n = 2$ or $\lambda < \sqrt{D}$ we can use Theorem 8 and $\Omega(\min\{\sqrt{D}, \lambda\})$ bound for the competitive ratio against oblivious adversary.

The core of this proof is to show that there exists a class of adversaries \mathcal{C} , such that for any online deterministic algorithm ALG , most of the adversaries from this class incur a high competitive ratio on ALG . Then we use a standard argument to show (non-constructively) that for any online randomized algorithm ALG_{RAND} , there exists an adaptive-online adversary which incurs a high competitive ratio on ALG_{RAND} .

The class \mathcal{C} is constructed as follows. Let us fix a finite sequence r of length S , where S is a parameter which will be fixed later. The sequence consists of S integers from the interval $[0, \dots, n - 1]$, i.e. $r = (r_1, r_2, \dots, r_S)$ and $0 \leq r_i \leq n - 1$. We define an adaptive-online adversary ADV_r as follows. In the beginning all nodes are in the same point of \mathcal{X} and the page of the adversary as well as the page of the algorithm is at the node v_0 . In the first step the adversary issues a request in v_0 and moves to v_{r_0} . Next steps are divided in phases. Each phase consists of two parts: an *expanding part* and a *contracting part*.

Let P_{ALG} and P_{OPT} be the vertices at which the algorithm and the adversary, respectively, have their pages. During the whole request sequence, the request is issued at v_0 , unless $P_{\text{ALG}} \equiv v_0$, in which case the request is issued at v_1 .

In each step of the expanding part the adversary increases the distance between P_{ALG} and the rest of the nodes, i.e. in the i -th step of the expanding part we have $d(P_{\text{ALG}}, v_j) = \min\{i - 1, \lambda\}$, for any vertex $v_j \neq P_{\text{ALG}}$. The distances between any other pairs of nodes remain equal to 0.

If $P_{\text{ALG}} \neq P_{\text{OPT}}$ at the beginning of the phase, then the expanding part continues till the algorithm decides to move its page to a new vertex, say from v_a to v_b .¹ Let X_k be the duration of the expanding part in the k -th phase; then $K_k := \min\{X_k - 1, \lambda\}$ is the distance to which algorithm's page at the node v_a was moved away from the rest of the nodes. Then a contracting part comes, in which the node v_a (in which the algorithm had its page during the expanding part) is moved closer to the other nodes. Formally, the contracting part of phase k takes K_k time steps and in the i -th time step of this part $d(v_a, v_j) = K_k - i + 1$, for all $v_j \neq v_a$. The distances between any other pairs of nodes remain equal to

¹Note that if the algorithm never jumps, the expanding part would last forever and after some point the algorithm would pay at least $\lambda + 1$ per round, while the adversary cost would still be 1 per round. This would immediately result in $C_{\text{ALG}} \geq \Omega(\lambda) \cdot C_{\text{ADV}_r}$.

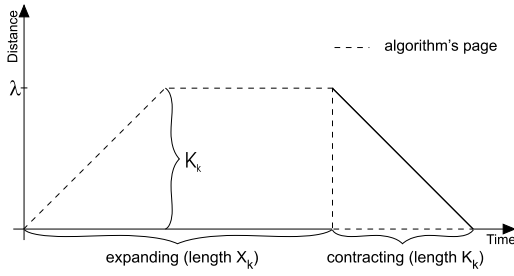


Figure 3: k -th (long) phase

0. Illustration of such a phase is given in Figure 3, where *distance* denotes the distance between v_a (which is equal to P_{ALG} during the expanding part) and the rest of the nodes.

However, if at the beginning of the phase $P_{\text{ALG}} \equiv P_{\text{OPT}}$ then the expanding part lasts at most \sqrt{D} steps. If the algorithm did not move its page in \sqrt{D} steps, then the adversary starts the contracting part by itself. If it happens, the algorithm can be sure that the adversary has just been at the same node as the algorithm has and we say that the algorithm has *detected the adversary's position*. In either case, the contracting part takes the same number of time steps as the expanding part did. After the contracting part, if the adversary's position has been detected, the adversary moves its page to the next place from the sequence r .

Additionally we call a phase *long* if its expanding part is longer than \sqrt{D} and *short* otherwise. The adversary maintains a counter of the long phases since it has moved its page last time and if the number of long phases exceeds $\lfloor n/2 \rfloor$, the adversary moves its page to the next place from the sequence r , immediately after the $\lfloor n/2 \rfloor$ -th long phase.

Moving a page by the adversary at the end of some phases partitions the sequence of phases into *metaphases*. Formally, a metaphase contains all the phases between two consecutive movements of the adversary's page. Therefore in metaphase i the position of the adversary is fixed and equal to r_i . The game between the adversary ADV_r and the algorithm ends after S metaphases, i.e. after a phase after which the adversary would normally move its page to the next position from the sequence r , but it has already used all elements of r . Furthermore, since the adversary jumps on the very beginning of the whole process and does not jump at the end of it, in the analysis we can assign the cost of moving its page to its cost in the metaphase which begins right after this movement.

Note, that for a given r the behaviour of the adversary ADV_r is completely deterministic and depends only on the behaviour of the algorithm ALG . We will now prove that if we choose r uniformly at random², then for any deterministic algorithm ALG , the expected cost of the adversary is smaller by a factor of $\Omega(\min\{n \cdot \sqrt{D}, D, \lambda\})$ than the cost of this algorithm.

LEMMA 7. *Let λ be the maximum allowed distance between two nodes and S be any positive integer. Then for the adversarial strategy defined above and for any deterministic algorithm ALG , holds:*

$$\mathbf{E}_r[C_{\text{ALG}} - \frac{1}{16} \min\{\lambda, D, n\sqrt{D}\} \cdot C_{\text{ADV}_r}] \geq 0,$$

²We can for example choose each element of r sequence uniformly at random from the set $\{0, \dots, n-1\}$

where the expected value is taken over all possible choices of the sequence r of length S .

PROOF. Fix any deterministic algorithm ALG . We consider i -th metaphase. Note, that the algorithm's behaviour can depend only on the past events (such as detecting the adversary's position in previous metaphases). In particular, till the last phase, it does not depend on the adversary's position r_i in this metaphase.

First, assume that the algorithm never detects the position of the adversary. Then the metaphase ends after the $\lfloor n/2 \rfloor$ -th long phase. We number all phases (within current metaphase) starting from 1. Let P_j be a node in which the algorithm has its page during the expanding part of j -th phase, and let X_j be the duration of the expanding part of phase j . Then the sequence of (P_j, X_j) fully characterizes the algorithm's behaviour in this metaphase and will be called *canonic sequence*. By \mathcal{S}_i we denote the set of all indexes of short phases and by \mathcal{L}_i the set of all indexes of long phases in the canonic sequence of metaphase i . Obviously, $|\mathcal{L}_i| = \lfloor \frac{n}{2} \rfloor$.

Now, when the canonic sequence is fixed, the performance of the algorithm depends only on r_i , the position of the adversary in metaphase i . In fact, the behaviour of the algorithm will follow the canonic sequence either till the phase in which the algorithm detects the position of the adversary (i.e. till the end of the first long phase j such that $P_j = r_i$) or till the end of the whole canonic sequence.

We prove now that in any metaphase the expected cost of the algorithm is greater than the expected cost of the adversary times $\frac{1}{16} \min\{\lambda, D, n\sqrt{D}\}$, where the expectation is taken over possible values of r_i , i.e. over the set $[n] := \{0, \dots, n-1\}$. This summed over all metaphases will yield Lemma 7. By $C_{\text{OPT}}(r_i)$ and $C_{\text{ALG}}(r_i)$ we denote the costs of the adversary and algorithm, respectively, in the current metaphase i , under the assumption that the adversary is in the node r_i during the metaphase. Then we have to prove that

$$\mathbf{E}_{r_i}[C_{\text{ALG}}(r_i)] \geq \frac{1}{16} \min\{\lambda, D, n\sqrt{D}\} \cdot \mathbf{E}_{r_i}[C_{\text{OPT}}(r_i)],$$

or since the r_i are distributed uniformly over the set $[n] = \{0, \dots, n-1\}$ it is equivalent to

$$\sum_{r_i \in [n]} C_{\text{ALG}}(r_i) \geq \frac{1}{16} \min\{\lambda, D, n\sqrt{D}\} \cdot \sum_{r_i \in [n]} C_{\text{OPT}}(r_i). \quad (3)$$

First we show an upper bound for the adversary's cost. In any short phase s , X_s is the length of the expanding part, the contracting part is at most as long as the expanding one. Since $\lambda \geq \sqrt{D}$, P_{ALG} is moved away during the whole expanding part and $K_s = \min\{X_s - 1, \lambda\} = X_s - 1$. Therefore, the adversary pays $2 \cdot \sum_{k=0}^{X_s-1} (k+1) = X_s^2 + X_s$ for serving requests if it is at the node P_s and $2 \cdot X_s$ if it is anywhere else. Thus, the total cost in short phases, summed over n possible choices of r_i , is $\sum_{s \in \mathcal{S}_i} (X_s^2 + X_s + (n-1) \cdot 2X_s) \leq \sum_{s \in \mathcal{S}_i} (X_s^2 + 2n \cdot X_s)$. In any long phase ℓ the adversary pays $2 \cdot \sum_{k=0}^{\sqrt{D}} (k+1) \leq D + 3\sqrt{D} + 2 < 2D$ for serving requests if it is at the same node as the algorithm and $2 \cdot X_\ell$ otherwise. There are at most $\lfloor n/2 \rfloor$ long phases, therefore the total cost in long phases is at most $\lfloor n/2 \rfloor \cdot 2D + n \cdot \sum_{\ell \in \mathcal{L}_i} 2X_\ell \leq n \cdot D + 2n \cdot \sum_{\ell \in \mathcal{L}_i} X_\ell$. Additionally, each of n adversaries

moves its page exactly once in a metaphase, along a distance 0, incurring a cost of D . Summing up, the total cost over all n choices of the r_i is

$$\sum_{r_i \in [n]} C_{\text{OPT}}(r_i) \leq \sum_{s \in \mathcal{S}_i} (X_s^2 + 2nX_j) + \sum_{\ell \in \mathcal{L}_i} (2nX_\ell) + nD + nD.$$

The lower bound on the total cost of n executions of the algorithm against n different choices of r_i is constructed as follows. At least $\lceil n/2 \rceil \geq n/2$ different values of r_i guarantee that the algorithm will not detect the adversary's position and will not end before finishing its canonic sequence. For all those values of r_i the algorithm has to pay for all short and long phases, from \mathcal{S}_i and \mathcal{L}_i , respectively. Since we assumed that the maximum allowed distance is $\lambda \geq \sqrt{D}$, the cost of moving the algorithm's page after a short phase s is equal to $DK_s = DX_s$. Now consider any long phase ℓ . If the length of its expanding part X_ℓ is not greater than λ , then the cost of moving the page at the end of the expanding part is $(X_\ell + 1) \cdot D \geq X_\ell \cdot D$. Otherwise, the cost of serving requests in the expanding part is $\sum_{k=0}^{\lambda-1} (k+1) + (X_\ell - \lambda) \cdot (\lambda + 1) \geq X_\ell \cdot \lambda - \frac{\lambda^2}{2} \geq \frac{1}{2}\lambda X_\ell$, and the cost of moving the page afterwards is $(\lambda + 1) \cdot D \geq \lambda D$. Summing up, the total cost over n possible choices of r_i is:

$$\sum_{r_i \in [n]} C_{\text{ALG}}(r_i) \geq \frac{1}{2}n \cdot \left(\sum_{s \in \mathcal{S}_i} DX_s + \sum_{\substack{\ell \in \mathcal{L}_i \\ X_\ell \leq \lambda}} DX_\ell + \sum_{\substack{\ell \in \mathcal{L}_i \\ X_\ell > \lambda}} \left(\frac{1}{2}\lambda X_\ell + \lambda D \right) \right)$$

The technical proof that the bounds on $\sum_{r_i \in [n]} C_{\text{OPT}}(r_i)$ and on $\sum_{r_i \in [n]} C_{\text{ALG}}(r_i)$ fulfill Inequality 3 can be found in the appendix. This finishes the proof of Lemma 7. \square

Now let ALG_{RAND} be any but fixed online randomized algorithm. ALG_{RAND} is equivalent to some probability distribution $\{p_i\}$ over the set of all possible deterministic algorithms ALG_i . We construct a matrix M whose rows are indexed with all possible deterministic algorithms ALG_i and whose columns are indexed with all possible S -element sequences of integers from the set $\{0, \dots, n-1\}$, i.e. columns are elements from $[n]^S$. We set the values in this matrix to $M_{i,r} := C_{\text{ALG}} - \frac{1}{16} \min\{\lambda, D, n\sqrt{D}\} \cdot C_{\text{ADV}_r}$. It follows from Lemma 7 that the average over each row is not smaller than 0. Then there exists (see [10, chapter 2] for a similar proof) a column r in which the average (weighted with p_i) is greater than 0. In other words, there exists a deterministic adversary ADV_r such that $\mathbf{E}[C_{\text{ALG}_{\text{RAND}}} - \frac{1}{16} \min\{\lambda, D, n\sqrt{D}\} \cdot C_{\text{ADV}_r}] \geq 0$, where the expected value is taken over all possible choices of the algorithm. Since we can prove it for any sequence length S , the competitive ratio of any randomized algorithm against cooperating adaptive-online adversaries is at least $\Omega(\min\{\lambda, D, n\sqrt{D}\})$. This finishes the proof of Theorem 6.

3. OBLIVIOUS COOPERATIVE

One could hope that the huge competitive ratio does not hold against non-adaptive adversaries. However, it appears that for the oblivious cooperative scenario there exists a lower bound which is almost as large as in the adaptive case.

THEOREM 8. *For any c -competitive algorithm for the dynamic page migration problem playing against cooperating oblivious configuration and sequence adversaries $c = \Omega(\min\{\sqrt{D}, \lambda\})$.*

PROOF. We will prove that even on a graph with two nodes connected by an edge, the competitive ratio of the algorithm is at least $\Omega(\min\{\sqrt{D}, \lambda\})$. Note that if the network has more than two nodes, the adversary can give requests only in v_0 and v_1 and put other nodes exactly in the same point of space \mathcal{X} as v_1 . Then for any algorithm ALG that uses the nodes v_2, v_3, \dots, v_{n-1} , there exist an algorithm ALG' which uses v_1 instead and achieves cost not larger than ALG .

We construct a probability distribution over inputs and prove that no deterministic algorithm which knows this distribution can achieve a competitive ratio better than $\Omega(\min\{\sqrt{D}, \lambda\})$. Using the Yao min-max theorem [8], Theorem 8 will be proven for any randomized algorithm against an oblivious adversary.

We divide the time into phases, each of length $D + 2 \cdot \min\{\lambda, \sqrt{D}\}$ steps. Each phase consists of *expanding part* which lasts for $\min\{\lambda, \sqrt{D}\}$ steps, *main part* lasting for D steps and *contracting part* lasting for $\min\{\lambda, \sqrt{D}\}$ steps. In i -th time step of expanding part the adversary moves v_1 so that $d(v_0, v_1) = i - 1$ and a request is issued at the node v_0 . Throughout the whole main part $d(v_0, v_1) = \min\{\lambda, \sqrt{D}\}$, and all the requests are issued at the same node — with probability 1/2 it is v_0 and with probability 1/2 it is v_1 . In i -th step of the contracting part $d(v_0, v_1) = \min\{\lambda, \sqrt{D}\} - i$ and the requests are issued at node v_1 .

For any randomly generated sequence of T phases, the optimal offline cost for serving all the requests is at most $C_{\text{OPT}} \leq T \cdot (D + 1 + \sum_{i=0}^{\min\{\lambda, \sqrt{D}\}-1} (i+1)) \leq T \cdot 3D$, because the optimal algorithm can move the page in the first step of the phase (paying 1 for serving request and D for jumping) to the node where all the D requests in the main part are. It pays additionally $(\sum_{i=0}^{\min\{\lambda, \sqrt{D}\}-1} (i+1))$ either in expanding part or in contracting part. On the other hand any deterministic algorithm has its page with probability 1/2 at the wrong node at the first time step of the main part. If it is the case and if it jumps within this part then it will pay $D \cdot \min\{\lambda, \sqrt{D}\}$ for moving the page, otherwise it will pay $D \cdot \min\{\lambda, \sqrt{D}\}$ for serving the requests during the main part. Summing it over all phases, we get that $\mathbf{E}[C_{\text{ALG}}] = T \cdot \frac{1}{2}D \cdot \min\{\lambda, \sqrt{D}\}$. Therefore the competitive ratio is at least

$$\mathcal{R}_{\text{OBL}} \geq \frac{\mathbf{E}[C_{\text{ALG}}]}{C_{\text{OPT}}} = \Omega(\min\{\lambda, \sqrt{D}\}),$$

which finishes the proof of Theorem 8 \square

We note that since the upper bound against two adaptive adversaries holds also against two oblivious ones, we obtain a competitive ratio of $\mathcal{O}(\min\{n \cdot \sqrt{D}, D, \lambda\})$ for the oblivious cooperative scenario. We note that this bound is up to a constant factor optimal in networks with a constant number of nodes.

4. HYBRID NON-COOPERATIVE

In this case we define our metric space \mathcal{X} to be a d -dimensional torus of diameter \sqrt{D} , where d is a fixed integer. d -dimensional torus is the same as a d -dimensional

mesh $[0, 1, 2, \dots, \sqrt{D} - 1]^d$ with added wrap-around edges, all of length 1. For any two nodes $v_x = (x_1, x_2, \dots, x_d)$ and $v_y = (y_1, y_2, \dots, y_d)$ we define the distance between them along the i -th dimension as $d_i(v_x, v_y) = \min\{|x_i - y_i|, \sqrt{D} - |x_i - y_i|\}$. Additionally, we will refer to d_1 as to *first dimension distance*. As the distance function on \mathcal{X} we take Manhattan distance, i.e. $d(v_x, v_y) = \sum_{1 \leq i \leq d} d_i(v_x, v_y)$.

Note that on such a torus (even for two nodes on a 1-dimensional torus and even for the oblivious cooperative scenario) the competitive ratio of any algorithm for the dynamic page migration is at least $\Omega(\sqrt{D})$. We show that if the configuration adversary is a stochastic process defined below, then the competitive ratio is significantly lower. Formally, we let the request adversary pick the request sequence (σ_t) and the initial configuration C_0 of n points in the space \mathcal{X} . The rest of the configuration sequence (C_t) is generated randomly according to the following rule:

For each $i \geq 1$, C_i is generated from C_{i-1} in the following way. For each node v from C_{i-1} a new position is chosen. Each coordinate of the current position of v with probability $1/3$ remains the same, increases by 1 or decreases by 1.

We can show that if the number of nodes is constant, then the expected competitive ratio is polylogarithmic in D . In this paper we prove it for the case of $n = 2$.

We present an algorithm ALG_{MAJ} , which works as follows. We divide the whole input into phases, each consisting of an interval of $K = \frac{1}{6} \cdot D \cdot \ln(2 \cdot \sqrt{D})$ time steps. The algorithm moves (optionally) only at the end of a phase to the node which issued the majority of requests in this phase. Formally, for any time interval I and a node number $j \in \{0, 1\}$, we define $I\#j$ be the number of requests issued in node v_j within time interval I . Let $\text{P}_{\text{ALG}}(i)$ denote the algorithm position throughout the whole i -th phase. If after the i -th phase I_i there exists a node $v_x \neq \text{P}_{\text{ALG}}(i)$, such that $I_i\#x > \frac{1}{2}K$, then the algorithm moves to the node v_x .

However, we are not able to prove a desired bound on the expected competitive ratio for short request sequences or for request sequences in which, after some early time step t , only one node accesses the page. We call a phase *empty* if it consists only of requests at one node. We say that an input sequence is *sufficiently long* if it consists of at least $\alpha \cdot D^{3/2} \cdot \log^2 D + 2$ non-empty phases, where α is some constant that we will specify later.

THEOREM 9. *In the hybrid non-cooperative scenario of the dynamic page migration problem, for any sufficiently long input sequence, the deterministic online algorithm ALG_{MAJ} achieves the expected competitive ratio of $\mathcal{O}(d \cdot \log^2 D)$ on two nodes in d -dimensional torus of diameter \sqrt{D} .*

PROOF. First, we will show that ALG_{MAJ} achieves the expected competitive ratio of $\mathcal{O}(\log^2 D)$ on a 1-dimensional torus i.e. a ring of \sqrt{D} points. In the following, we will always assume that if the algorithm has its page at a node other than the node issuing the request, then the algorithm will pay for serving the request the maximum possible cost of communication in the ring, which amount we will denote by $\mathcal{A} = 1 + \frac{\sqrt{D}}{2} = \mathcal{O}(\sqrt{D})$. If the algorithm moves the page,

we assume that the cost of such an operation is as large as possible, i.e. $D \cdot \mathcal{A}$.

We introduce a notion of a *configuration path*. A configuration path is just a sequence of configurations, such that for each two consecutive configurations C_i and C_{i+1} , the distance between the positions of any node v in both configurations is at most 1. The number of possible configurations of two nodes is $(\sqrt{D})^2 = D$. We can prove that if we begin from any configuration C , and the configuration is changed according to our stochastic process, then after $k \geq K$ steps we arrive at an *almost uniform* distribution of configurations. By an almost uniform distribution we mean that the probability of any configuration is at least $\frac{1}{4} \cdot \frac{1}{D}$. This observation is proven in the appendix.

OBSERVATION 10. *For any configuration C , after $k \geq K$ time steps the probability distribution over all possible configurations is almost uniform.*

However, if we take an interval which is relatively shorter than K , i.e. of length $L = \frac{1}{(24)^2} \cdot \frac{D}{4 \log D}$, then with a constant probability the starting configuration does not change much. In particular, the following technical lemma, proven in the appendix, holds.

LEMMA 11. *Let C_t be the configuration at time step t , and let the distance between two nodes in C_t be at least $\sqrt{D}/3$. Then at least $\frac{1}{4}$ of all configuration paths of length L starting at C_t have the property, that the distance of these two nodes is at least $\sqrt{D}/6$ on all configurations on the path.*

It is straightforward that in the first two phases the cost of the algorithm is at most $2 \cdot (K \cdot \mathcal{A} + D \cdot \mathcal{A}) = \mathcal{O}(D^{3/2} \log D)$. This creates an additive constant in the algorithm's cost.

We fix any phase $i \geq 3$, and denote the corresponding time interval by I . The main idea is to show that for each such interval I , we can find a consistent subinterval J which is *hard* even for the optimal algorithm. By the notion *hard* we mean that if such subinterval begins with an almost uniform distribution of configurations, then on the constant fraction of configuration paths in this interval, each algorithm (in particular the optimal algorithm) has a cost not much smaller than ALG_{MAJ} on I . Without loss of generality we can assume that at the beginning of I the algorithm has its page at node v_0 . We consider two cases.

- (a) If $I\#0 \geq \frac{1}{2}K$, then from the average argument, follows that there exists a subinterval $J \subseteq I$, such that $|J| = L$, $J\#0 = \frac{L}{K} \cdot I\#0 \pm 1$ and $J\#1 = \frac{L}{K} \cdot I\#1 \pm 1$. Then ALG_{MAJ} cost in this interval is at most

$$\begin{aligned} C_{\text{ALG}}(I) &\leq \mathcal{A} \cdot I\#1 \\ &\leq \mathcal{A} \cdot \min\{I\#0, I\#1\} \\ &= \mathcal{O}(\sqrt{D} \cdot \frac{K}{L}) \cdot \min\{J\#0, J\#1\} \\ &= \mathcal{O}(\sqrt{D} \cdot \log^2 D) \cdot \min\{J\#0, J\#1\} . \end{aligned}$$

- (b) If $I\#0 < \frac{1}{2}K$, then $I\#1 > \frac{1}{2}K$. Additionally, in the previous phase interval I_0 there were at most $\frac{1}{2}K$ requests at node v_1 , otherwise ALG would not have its page at v_0 at the beginning of the interval I . Therefore, there exists a subinterval $J \subseteq I_0 \cup I$ of length L , such that $J\#1 = \frac{1}{2} \cdot L \pm 1$. Then ALG_{MAJ} cost is at most

$$\begin{aligned}
C_{\text{ALG}}(I) &\leq K \cdot \mathcal{A} + D \cdot \mathcal{A} \\
&= \mathcal{O}(K \cdot \mathcal{A}) \\
&= \mathcal{O}(L \cdot \log^2 D \cdot \sqrt{D}) \\
&= \mathcal{O}(\sqrt{D} \cdot \log^2 D) \cdot \min\{J \neq 0, J \neq 1\}.
\end{aligned}$$

First, we show that interval J is hard. To prove it, we assume that J begins with an almost uniform distribution of configurations. If the position of the nodes v_0 and v_1 were chosen uniformly at random, then with probability at least $\frac{1}{3}$, they would be at the distance of at least $\sqrt{D}/3$. Since the distribution is almost uniform, the probability of this event is at least $\frac{1}{12}$. Then it follows from Lemma 11 that on the fraction of at least $\frac{1}{4} \cdot \frac{1}{12}$ of all configuration paths in the interval J , the distance between v_0 and v_1 is at least $\sqrt{D}/6$. On these configuration paths the performance of any algorithm is not much better than the performance of the algorithm ALG_{MAJ} in I . Actually, if any algorithm moves the page within interval J , then it pays at least $D \cdot \sqrt{D}/6$. Otherwise, it has its page always in the same node, paying for serving all requests coming from the other node, i.e. paying at least $\frac{\sqrt{D}}{6} \cdot \min\{J \neq 0, J \neq 1\}$. In either case its cost is at least $\Omega(\frac{1}{\log^2 D}) \cdot C_{\text{ALG}}(I)$.

We can find a relation between the cost of the optimal algorithm in interval J and the cost of ALG_{MAJ} in interval I on any configuration path, even in the situation where v_0 and v_1 are at the same point of the ring. The cost of any algorithm in interval J is then at least $\min\{D, J \neq 0, J \neq 1\} = \min\{J \neq 0, J \neq 1\} \geq \Omega(\frac{1}{\sqrt{D} \cdot \log^2 D}) \cdot C_{\text{ALG}}(I)$

Now, we have to justify the assumption that each interval J begins with an almost uniform distribution of configurations. If we take all the intervals I_i corresponding to the phases i of our algorithm, the corresponding J_i subintervals are not only dependent, but they can even overlap each other. This is because in the case (b) we look for a subinterval J_i in the intervals I_{i-1} and I_i . However, if we choose every third interval I_i , then the distances between two consecutive subintervals will be at least K . Thus, it follows from Observation 10 that if we fix a configuration path in one of J_i then at the beginning of subinterval J_{i+3} we have an almost uniform distribution over configurations. Formally, we divide the set of all phases into 3 subsets S_0 , S_1 and S_2 , where S_j is the set containing all phases $i \geq 3$ such that $i \bmod 3 = j$. Let S_{max} be this set among S_j , whose phases incur the largest cost on ALG_{MAJ} . It holds $\sum_{i \in S_{\text{max}}} C_{\text{ALG}}(I_i) \geq \frac{1}{3} \cdot \sum_{i \geq 3} C_{\text{ALG}}(I_i)$,

This leads to the conclusion that this can be considered as a following randomized process. In each phase i from S_{max} the algorithm has cost $C_{\text{ALG}}(I_i)$ and the optimal algorithm has cost $C_{\text{OPT}}(J_i)$ on the subinterval J_i which is at least $\frac{c}{\log^2 D} \cdot C_{\text{ALG}}(I_i)$ with some constant probability p and a constant c , and at least $\Omega(\frac{1}{\sqrt{D} \cdot \log^2 D}) \cdot C_{\text{ALG}}(I_i)$ with probability $1 - p$. We want to have a lower bound for the sum of optimal algorithm's cost, therefore we can assume that $C_{\text{OPT}}(J_i)$ is a random variable which is equal to $\frac{c}{\log^2 D} \cdot C_{\text{ALG}}(I_i)$ with probability p , and 0 otherwise.

We prove that after a sufficiently long request sequence (consisting of $\frac{12}{p^2} \cdot D^{3/2} \cdot \log^2 D + 2$ non-empty phases) with high probability the cost of the optimal algorithm is within a factor of $\mathcal{O}(\log^2 D)$ from the cost of ALG_{MAJ} incurred in

the phases $i \geq 3$. It follows from Hoeffding inequality [11] that:

$$\Pr \left[\sum_{i \in S_{\text{max}}} C_{\text{OPT}}(J_i) < \frac{c \cdot p}{2} \sum_{i \in S_{\text{max}}} \frac{C_{\text{ALG}}(I_i)}{\log^2 D} \right] \leq \frac{1}{D} \quad (4)$$

The proof of the inequality above can be found in the appendix. Therefore, with the probability at least $1 - \frac{1}{D}$ we have $C_{\text{OPT}}(\sigma_t, C_t) \geq \sum_{i \in S_{\text{max}}} C_{\text{OPT}}(J_i) = \Omega(\frac{1}{\log^2 D}) \cdot \sum_{i \in S_{\text{max}}} C_{\text{ALG}}(I_i) = \Omega(\frac{1}{\log^2 D}) \cdot \sum_{i \geq 3} C_{\text{ALG}}(I_i)$ and with the probability at most $\frac{1}{D}$, $C_{\text{OPT}}(\sigma_t, C_t) \geq \Omega(\frac{1}{\sqrt{D} \cdot \log^2 D}) \cdot \sum_{i \geq 3} C_{\text{ALG}}(I_i)$. Thus, the expected competitive ratio equals

$$\begin{aligned}
\mathcal{R}_{\text{ALG}} &= \mathbf{E}_{(C_t)} \left[\frac{C_{\text{ALG}}(\sigma_t, C_t) - \mathcal{O}(D^{3/2} \cdot \log D)}{C_{\text{OPT}}(\sigma_t, C_t)} \right] \\
&\leq \mathbf{E}_{(C_t)} \left[\frac{\sum_{i \geq 3} C_{\text{ALG}}(I_i)}{C_{\text{OPT}}(\sigma_t, C_t)} \right] \\
&= \mathcal{O}(\log^2 D) \cdot (1 - \frac{1}{D}) + \mathcal{O}(\sqrt{D} \cdot \log^2 D) \cdot \frac{1}{D} \\
&= \mathcal{O}(\log^2 D).
\end{aligned}$$

We can extend the proof for the ring to the case of a d -dimensional torus. Note that in the reasoning above, for each time step in which the algorithm had some cost we charged the algorithm \mathcal{A} , i.e. the maximum possible communication cost on the ring. Therefore, on the d -dimensional torus, we could still charge the optimal algorithm only for the communication along the first dimension and the cost of the algorithm will increase at most by a factor of d . If the metric were the Euclidean distance, then the cost of the algorithm would be increased by a factor of \sqrt{d} and the expected competitive ratio would be $\mathcal{O}(\sqrt{d} \cdot \log^2 D)$. This ends the proof of Theorem 9. \square

We note that if we replace this torus by a mesh of diameter \sqrt{D} , then we are able to prove asymptotically the same bound on the expected competitive ratio. We are still able to find $K = \mathcal{O}(D \cdot \log D)$, such that after K time steps the random walk of the nodes on the mesh yields almost uniform distribution. We could take the same value of $L = \Omega(\frac{D}{\log D})$ as in the proof of Theorem 9 and this assures that if two nodes are far apart from each other, then after L steps their distance remains large. Since the expected competitive ratio is up to a constant factor equal to the quotient $\frac{K}{L}$, Theorem 9 holds also for meshes. Furthermore, we can play with the constants inside the proof and allow the diameter to be $\Theta(\sqrt{D})$ instead of \sqrt{D} .

It is also possible to extend this proof to any constant number of nodes. We can also generalize it to n nodes, but it would yield a competitive ratio of $\Omega(n^2)$. However, we are convinced that the expected ratio of the dynamic page migration problem is at most polylogarithmic in n and D and can be achieved by more sophisticated algorithms. Analysis for other diameters of the torus (mesh) and other models of changes of the network is a subject of future work.

5. ACKNOWLEDGEMENTS

We would like to thank Gereon Frahling for an easier proof of Inequality 4 and anonymous referees for their helpful comments.

6. REFERENCES

- [1] B. Awerbuch, Y. Bartal, and A. Fiat. Competitive distributed file allocation. In *Proc. of the 25th ACM Symp. on Theory of Computing (STOC)*, pages 164–173, 1993.
- [2] Y. Bartal. Distributed paging. In *Online Algorithms*, pages 97–117, 1996.
- [3] Y. Bartal, M. Charikar, and P. Indyk. On page migration and other relaxed task systems. In *Proc. of the 8th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 43–52, 1997.
- [4] Y. Bartal, A. Fiat, and Y. Rabani. Competitive algorithms for distributed data management. In *Proc. of the 24th ACM Symp. on Theory of Computing (STOC)*, pages 39–50, 1992.
- [5] L. Becchetti, S. Leonardi, A. Marchetti-Spaccamela, G. Schäfer, and T. Vredeveld. Average case and smoothed competitive analysis of the multi-level feedback algorithm. In *Proc. of the 44th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 462–471, 2003.
- [6] S. Ben-David, A. Borodin, R. M. Karp, G. Tardos, and A. Wigderson. On the power of randomization in online algorithms. In *Proc. of the 22nd ACM Symp. on Theory of Computing (STOC)*, pages 379–386, 1990.
- [7] D. L. Black and D. D. Sleator. Competitive algorithms for replication and migration problems. Technical Report CMU-CS-89-201, Department of Computer Science, Carnegie-Mellon University, 1989.
- [8] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
- [9] M. Chrobak, L. L. Larmore, N. Reingold, and J. Westbrook. Page migration algorithms using work functions. In *Proc. of the ISAAC: 4th International Symposium on Algorithms and Computation*, 1993.
- [10] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
- [11] S. Rajesekaran, P. M. Pardalos, J. H. Reif, and J. Rolim. *Handbook of Randomized Computing*, volume II. Kluwer Academic Publishers, 2001.
- [12] J. S. Rosenthal. Convergence rates for Markov chains. *SIAM Review*, 37(3):387–405, 1995.
- [13] M. Scharbrodt, T. Schickinger, and A. Steger. A new average case analysis for completion time scheduling. In *Proc. of the 34th ACM Symp. on Theory of Computing (STOC)*, pages 170–178, 2002.
- [14] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. In *Communications of the ACM*, 28, pages 202–208, 1985.
- [15] J. Westbrook. Randomized algorithms for multiprocessor page migration. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 7, pages 135–150, 1992.

APPENDIX

A. THE PROOFS FOR SECTION 2

PROOF OF LEMMA 2. $P_{\text{ALG}} \equiv P_{\text{OPT}}$ implies $K = 0_E$, $F(K) = 0$ and $\mathcal{F}(K_0) = 0$. Since the algorithm is not in the node which issued the request, neither is the adversary.

Thus $c(Y) = c(X) = 1 + X$. Additionally for all $P_{R_i} \in \mathcal{R}$, $Y_i = X_i \leq \frac{5}{4}X < 2X$. Then we have

$$\begin{aligned}
\mathbf{E}[C_{\text{ALG}} + \Delta\Phi] &= c(X) + \sum_{P_{R_i} \in \mathcal{R}} \frac{\mathcal{B}(X)}{m} \cdot [D \cdot c(X_i) + \mathcal{F}(Y_i)] \\
&= 1 + X + \sum_{P_{R_i} \in \mathcal{R}} \frac{\mathcal{B}(X)}{m} \cdot [D \cdot (1 + X_i) + \mathcal{F}(X_i)] \\
&\leq 1 + X + \mathcal{B}(X) [D + D \cdot 2X + \mathcal{F}(2X)] \\
&\leq 1 + X + b \cdot \frac{1}{\sqrt{D}} (D + 2DX) + \mathcal{B}(X) \cdot \mathcal{F}(2X) \\
&\leq (1 + 2b \cdot \sqrt{D}) \cdot (1 + X) + \mathcal{B}(X) \cdot \mathcal{F}(2X) \\
&\leq \mathcal{O}(\sqrt{D}) \cdot c(Y) + \mathcal{B}(X) \cdot \mathcal{F}(2X) .
\end{aligned}$$

The first inequality follows from the monotonicity of $\mathcal{F}(\cdot)$ function. Now it is sufficient to prove that

$$\mathcal{B}(x) \cdot \mathcal{F}(2x) = \mathcal{O}(n\sqrt{D}) \cdot (1 + X) = \mathcal{O}(n\sqrt{D}) \cdot c(Y) .$$

First, we notice that $\mathcal{F}(2x)$ and $\mathcal{F}(x)$ are only constant factor away from each other.

$$\mathcal{F}(2x) = \mathcal{F}_B + (2x) \cdot f \cdot \sqrt{D} \cdot \min\{2x, \sqrt{D}\} \leq \mathcal{F}_B + 4 \cdot \mathcal{F}'(x) .$$

Now we have $\mathcal{B}(x) \cdot \mathcal{F}_B = \mathcal{B}(x) \cdot fnD\sqrt{D} =: (1)$. We consider three cases:

$$\begin{aligned}
\text{if } x \leq 1, & \quad (1) = bfn\sqrt{D} \\
\text{if } 1 < x \leq \sqrt{D}, & \quad (1) = bfn\sqrt{D} \cdot x \\
\text{if } x > \sqrt{D}, & \quad (1) \leq b \frac{1}{\sqrt{D}} \cdot fnD\sqrt{D} < bfn\sqrt{D} \cdot x
\end{aligned}$$

Analogously, $\mathcal{B}(x) \cdot \mathcal{F}'(x) = \mathcal{B}(x) \cdot fx\sqrt{D} \cdot \min\{x, \sqrt{D}\} =: (2)$. We consider four cases:

$$\begin{aligned}
\text{if } x \leq 1, & \quad (2) = b \cdot \frac{1}{D} \cdot f\sqrt{D} \cdot x^2 < bf \\
\text{if } 1 < x \leq \sqrt{D}, & \quad (2) = b \cdot \frac{x}{D} \cdot f\sqrt{D} \cdot x^2 \leq bf\sqrt{D} \cdot x \\
\text{if } \sqrt{D} < x \leq D, & \quad (2) = b \cdot \frac{1}{x} \cdot fD \cdot x < bf\sqrt{D} \cdot x \\
\text{if } x > D, & \quad (2) = b \cdot \frac{1}{D} \cdot fD \cdot x \leq bf \cdot x
\end{aligned}$$

In either case

$$\begin{aligned}
\mathcal{B}(x) \cdot \mathcal{F}(2x) &\leq \mathcal{B}(x) \cdot (\mathcal{F}_B + 4 \cdot \mathcal{F}'(x)) \\
&\leq bfn\sqrt{D} \cdot (1 + x) + 4 \cdot bf\sqrt{D} \cdot (1 + x) \\
&\leq \mathcal{O}(n\sqrt{D}) \cdot (1 + x) ,
\end{aligned}$$

which finishes the proof of Lemma 2. \square

PROOF OF LEMMA 3. In this proof we concentrate on a case $P_{\text{ALG}} \neq P_{\text{OPT}}$. First, we show how to bound the expression $\mathbf{E}[\Delta\Phi]$. It follows from Equation 2 that

$$\mathbf{E}[\Delta\Phi] = \mathcal{F}(K) - \mathcal{F}(K_0) + \sum_{P_{R_i} \in \mathcal{R}} \frac{\mathcal{B}(X)}{m} \cdot [\mathcal{F}(Y_i) - \mathcal{F}(K)] .$$

We denote $\mathcal{F}(K) - \mathcal{F}(K_0)$ by $\Delta\mathcal{F}(K)$, i.e. $\Delta\mathcal{F}(K)$ is the change induced by changing the distance between P_{ALG} and P_{OPT} . We note since the algorithm and the adversary have their pages at the different nodes ($K \neq 0_E$) it was also the case at the beginning of the time step, i.e. $K_0 \neq 0_E$. Since $\mathcal{F}(\cdot)$ is monotonically increasing, $\Delta\mathcal{F}(K)$ is maximized when K_0 is as small as possible. The distance between any two nodes cannot change by more than 2 per round, therefore

$\Delta\mathcal{F}(K) \leq \mathcal{F}(K) - \mathcal{F}(\max\{K-2, 0\})$. Thus for $K \leq \sqrt{D}$ we have

$$\Delta\mathcal{F}(K) \leq f\sqrt{D} \cdot (K^2 - (K-2)^2) \leq 4f\sqrt{D} \cdot K,$$

and for $K > \sqrt{D}$ we have

$$\begin{aligned} \Delta\mathcal{F}(K) &\leq \mathcal{F}(K) - \mathcal{F}(\max\{K-2, \sqrt{D}\}) + \Delta\mathcal{F}(\sqrt{D}) \\ &\leq fD \cdot (K - (K-2)) + 4f\sqrt{D} \cdot \sqrt{D} \\ &\leq 6fD \\ &\leq 6f\sqrt{D} \cdot K. \end{aligned}$$

Second, we find an upper bound for $\mathbf{E}[C_{\text{ALG}}]$. From the definition of $\mathcal{B}(\cdot)$ function we have $b \leq \mathcal{B}(X) \cdot D$. Then the algorithm's cost of servicing the request is equal to

$$c(X) \leq 1 + \frac{1}{b} \cdot \mathcal{B}(X) \cdot DX = 1 + \frac{\mathcal{B}(X)}{m} \sum_{P_{R_i} \in \mathcal{R}} \frac{1}{b} DX.$$

and the algorithm's expected cost of moving the page is $\frac{\mathcal{B}(X)}{m} \sum_{P_{R_i} \in \mathcal{R}} D \cdot (1 + X_i) \leq \frac{\mathcal{B}(X)}{m} \sum_{P_{R_i} \in \mathcal{R}} D \cdot (1 + \frac{5}{4} \cdot X)$ (see Equation 2 and Lemma 2). Thus, the total expected cost of the algorithm is

$$\begin{aligned} \mathbf{E}[C_{\text{ALG}}] &\leq 1 + \mathcal{B}(X) \cdot D + \frac{\mathcal{B}(X)}{m} \sum_{P_{R_i} \in \mathcal{R}} \left(\frac{1}{b} + \frac{5}{4}\right) DX \\ &\leq 1 + b \cdot \sqrt{D} + \frac{\mathcal{B}(X)}{m} \sum_{P_{R_i} \in \mathcal{R}} 2DX. \end{aligned}$$

Therefore, the total amortized cost per round is bounded by

$$\begin{aligned} \mathbf{E}[C_{\text{ALG}} + \Delta\Phi] &\leq \underbrace{1 + b \cdot \sqrt{D}}_{\text{constant part}} + \Delta\mathcal{F}(K) \quad (5) \\ &\quad + \underbrace{\frac{\mathcal{B}(X)}{m} \sum_{P_{R_i} \in \mathcal{R}} [2DX + \mathcal{F}(Y_i) - \mathcal{F}(K)]}_{\text{variable part}} \end{aligned}$$

To show that the lemma holds we consider the following two cases, depending on whether $P_{\text{OPT}} \in \mathcal{R}$.

1. $P_{\text{OPT}} \notin \mathcal{R}$. Since P_{OPT} lies outside the \mathcal{R} set, Y is large compared to X , i.e. $Y > X/4$. Additionally, $P_{\text{OPT}} \neq P_{\text{R}}$ implies $c(Y) = 1 + Y$. Since the *constant part* in Inequality 5 is smaller than $\mathcal{O}(\sqrt{D}) \cdot 1$, it is sufficient to prove that the $\Delta\mathcal{F}(K)$ term and the *variable part* is not greater than $\mathcal{O}(\sqrt{D}) \cdot Y$. First we prove the bound on $\Delta\mathcal{F}(K)$. The triangle inequality implies $K \leq Y + X \leq Y + 4Y = 5Y$. Then

$$\Delta\mathcal{F}(K) \leq 6f\sqrt{D} \cdot K \leq \mathcal{O}(\sqrt{D}) \cdot Y.$$

To bound the *variable part*, which we will further denote as VP , first we notice that the triangle inequality implies $Y_i \leq Y + X/4 < 2Y$ for all i . Therefore

$$\begin{aligned} VP &\leq \mathcal{B}(X)(2DX + \mathcal{F}(2Y) - \mathcal{F}(K)) \\ &\leq \mathcal{B}(X)(2D \cdot (4Y) + \mathcal{F}'(2Y)) \\ &\leq b \cdot \frac{1}{\sqrt{D}} \cdot (8DY + fD \cdot (2Y)) \\ &= \mathcal{O}(\sqrt{D}) \cdot Y. \end{aligned}$$

2. $P_{\text{OPT}} \in \mathcal{R}$. Since Y (and therefore also cost of the adversary) can be almost arbitrarily small, we cannot use the argument from the previous case. However, we are able to prove that the expected change in the potential is negative and the algorithm's cost can be paid from the potential function.

$P_{\text{OPT}} \in \mathcal{R}$ implies that $Y \leq X/4$, $K \geq X - Y \geq \frac{3}{4}X$ and that for all i , $Y_i \leq Y + X/4 \leq X/2$. We split the *variable part* depending on whether a node $P_{R_i} \in \mathcal{R}$ is equal to P_{OPT} .

$$\begin{aligned} VP &\leq \frac{\mathcal{B}(X)}{m} \sum_{P_{R_i} \in \mathcal{R}} 2DX \\ &\quad + \frac{\mathcal{B}(X)}{m} \sum_{\substack{P_{R_i} \in \mathcal{R} \\ P_{R_i} \neq P_{\text{OPT}}}} (\mathcal{F}(0_E) - \mathcal{F}(K)) \\ &\quad + \frac{\mathcal{B}(X)}{m} \sum_{\substack{P_{R_i} \in \mathcal{R} \\ P_{R_i} \neq P_{\text{OPT}}}} (\mathcal{F}(Y_i) - \mathcal{F}(K)) \\ &\leq \mathcal{B}(X) \cdot \left(2DX - \frac{\mathcal{F}(K)}{m} + \mathcal{F}(X/2) - \mathcal{F}(K) \right) \end{aligned}$$

Using the triangle inequality we obtain $\frac{3}{4}X \leq X - Y \leq K \leq X + Y \leq \frac{5}{4}X$. Applying $\mathcal{F}(K)/m \geq \mathcal{F}_B/m = fD\sqrt{D} \cdot \frac{m}{m} > fD\sqrt{D}$, we get

$$VP \leq \mathcal{B}(X) \left(2DX - fD^{3/2} + \mathcal{F}'(\frac{1}{2}X) - \mathcal{F}'(\frac{3}{4}X) \right).$$

We can prove that VP is negative and in absolute value it is greater than the sum of *constant part* and $\Delta\mathcal{F}(K)$. First, we note that $\frac{1}{16}\mathcal{B}(X) \cdot fD^{3/2} \geq b\frac{1}{D} \cdot fD^{3/2}$ is greater than the *constant part*, and therefore it is sufficient to prove that $VP' := VP - \frac{1}{16}\mathcal{B}(X) \cdot fD^{3/2} \geq \Delta\mathcal{F}(K)$.

We consider two cases. If $X < 2 \cdot \sqrt{D}$, then $\Delta\mathcal{F}(K) \leq 6f\sqrt{D}K \leq \frac{15}{2}f\sqrt{D}X$ and it can be easily checked that $\mathcal{B}(X) \geq \frac{1}{4} \cdot b \cdot \frac{X}{D}$. Substituting f by 22 and b by 40, we obtain

$$\begin{aligned} VP' &\leq \mathcal{B}(X) \cdot \left(2DX - \frac{15}{16}f \cdot D \cdot \sqrt{D} \right) \\ &\leq \frac{1}{4} \cdot b \cdot \frac{X}{D} \cdot \left(4 \cdot D\sqrt{D} - \frac{15}{16}f \cdot D\sqrt{D} \right) \\ &\leq -\Delta\mathcal{F}(K) \end{aligned}$$

Otherwise $X \geq 2 \cdot \sqrt{D}$ which implies $K > \sqrt{D}$ and we can bound $\Delta\mathcal{F}(K)$ by $\leq 6fD$. Additionally, in this case $\mathcal{B}(X) \geq b \cdot \frac{1}{X}$.

$$\begin{aligned} VP' &\leq \mathcal{B}(X) \cdot (2DX + \mathcal{F}'(1/2 \cdot X) - \mathcal{F}'(3/4 \cdot X)) \\ &\leq b \cdot \frac{1}{X} \cdot (2DX + fD(1/2 \cdot X) - fD(3/4 \cdot X)) \\ &\leq -\Delta\mathcal{F}(K) \end{aligned}$$

In either case the sum of VP , *constant part* and $\Delta\mathcal{F}(K)$ is smaller than 0.

This finishes the proof of Lemma 3 \square

PROOF OF THEOREM 4. In this proof we will reuse notation from the ALG_{DIST} algorithm and Theorem 1. This

time, let $\mathcal{F}(x) = 3 \cdot D \cdot c(x)$. Analogously to the proof of the $\mathcal{O}(n \cdot \sqrt{D})$ competitiveness we divide each time step into two parts, i.e.

1. OPT does not move the page. If all distances are 0_E then the proof follows trivially. Otherwise if $Y = 0_E$, then $X = K \neq 0_E$, $c(X) = 1 + X$ and we have

$$\begin{aligned} C_{\text{ALG}} + \Delta\Phi &= c(X) + D \cdot c(X) + \Delta\mathcal{F}(K) \\ &\quad + \mathcal{F}(Y) - \mathcal{F}(K) \\ &\leq (1 + D) \cdot c(X) + D - 3D \cdot c(X) \\ &\leq 0 . \end{aligned}$$

In other case $Y \neq 0_E$ and then

$$\begin{aligned} C_{\text{ALG}} + \Delta\Phi &= (1 + D) \cdot c(X) + D \\ &\quad + 3D \cdot c(Y) - 3D \cdot c(K) \\ &\leq 3D \cdot c(X) - 3D \cdot c(K) + 3D \cdot c(Y) \\ &\leq 3D \cdot (X + 1) - 3D \cdot K + 3D \cdot c(Y) \\ &\leq 3D \cdot Y + 3D + 3D \cdot c(Y) \\ &\leq 6D \cdot c(Y) \\ &\leq \mathcal{O}(D) \cdot C_{\text{OPT}} . \end{aligned}$$

2. OPT moves the page across non- 0_E distance, paying $D \cdot c(Y)$. We obtain

$$\begin{aligned} C_{\text{ALG}} + \Delta\Phi &= 0 + \mathcal{F}(X) - \mathcal{F}(K) \\ &\leq 3D \cdot (X + 1) - 3D \cdot K \\ &\leq 3D \cdot (Y + 1) \\ &= 3D \cdot c(Y) \\ &= \mathcal{O}(C_{\text{OPT}}) . \end{aligned}$$

In both cases we made extensive use of triangle inequality. Therefore, Theorem 4 holds. \square

PROOF OF INEQUALITY 3. We have the following bound on $\sum_{r_i \in [n]} C_{\text{OPT}}(r_i)$:

$$\sum_{r_i \in [n]} C_{\text{OPT}}(r_i) \leq \underbrace{\sum_{s \in \mathcal{S}_i} (X_s^2 + 2nX_s)}_{(*)} + \underbrace{\sum_{\ell \in \mathcal{L}_i} 2nX_\ell}_{(**)} + \underbrace{2nD}_{(***)}$$

Multiplying each summand by $\min\{\lambda, D, n\sqrt{D}\}$ we get

$$\begin{aligned} \min\{\lambda, D, n\sqrt{D}\} \cdot (*) &\leq n\sqrt{D} \sum_{s \in \mathcal{S}_i} X_s^2 + D \sum_{s \in \mathcal{S}_i} 2nX_s \\ &\leq \sum_{s \in \mathcal{S}_i} \left(n\sqrt{D} \cdot X_s \sqrt{D} + 2nDX_s \right) \\ &\leq \sum_{s \in \mathcal{S}_i} 3nDX_s , \end{aligned}$$

$$\min\{\lambda, D, n\sqrt{D}\} \cdot (***) \leq \sum_{\substack{\ell \in \mathcal{L}_i \\ X_\ell \leq \lambda}} 2nDX_\ell + \sum_{\substack{\ell \in \mathcal{L}_i \\ X_\ell > \lambda}} 2n\lambda X_\ell ,$$

$$\min\{\lambda, D, n\sqrt{D}\} \cdot (***) \leq n\sqrt{D} \cdot 2nD$$

$$\begin{aligned} &\leq 3 \sum_{\ell \in \mathcal{L}_i} 2nD\sqrt{D} \\ &\leq \sum_{\substack{\ell \in \mathcal{L}_i \\ X_\ell \leq \lambda}} 6nD\sqrt{D} + \sum_{\substack{\ell \in \mathcal{L}_i \\ X_\ell > \lambda}} 6nD\sqrt{D} \\ &\leq \sum_{\substack{\ell \in \mathcal{L}_i \\ X_\ell \leq \lambda}} 6nDX_\ell + \sum_{\substack{\ell \in \mathcal{L}_i \\ X_\ell > \lambda}} 6n\lambda D . \end{aligned}$$

In bounding the last expression we used $3|\mathcal{L}_i| = 3\lfloor n/2 \rfloor \geq n$ and $\lambda \geq \sqrt{D}$. Summing up, we get that

$$\min\{\lambda, D, n\sqrt{D}\} \cdot \sum_{r_i \in [n]} C_{\text{OPT}}(r_i) \leq 16 \cdot \sum_{r_i \in [n]} C_{\text{ALG}}(r_i) ,$$

which finishes the proof of Inequality 3. \square

B. THE PROOFS FOR SECTION 4

PROOF OF OBSERVATION 10. First, we note that if we start from any configuration C , then we can consider each node movement separately. Each node movement is a random walk on a ring of size \sqrt{D} , which is equivalent to a finite ergodic Markov chain, whose stationary distribution is uniform. Rosenthal in [12] gives explicit bounds on the convergence ratio of such a random walk, i.e. after $k \geq K = \frac{1}{6}D \cdot \ln(2\sqrt{D})$ steps the probability p_X of the random walk ending in some particular point X of the ring fulfills $p_X \in [\frac{1}{\sqrt{D}} - \epsilon, \frac{1}{\sqrt{D}} + \epsilon]$, where ϵ denotes the bound on variation distance

$$\epsilon \leq \sqrt{\frac{e^{-\frac{4\pi^2}{3(\sqrt{D})^2} \cdot k}}{1 - e^{-\frac{4\pi^2}{3(\sqrt{D})^2} \cdot k}}} .$$

Since $k \geq \frac{1}{6} \cdot D$, we have

$$\epsilon \leq \sqrt{\frac{e^{-\frac{4 \cdot 3^2}{3D} \cdot k}}{1/2}} \leq \sqrt{2} \cdot e^{-12 \cdot \frac{1}{6} \cdot \ln(2\sqrt{D})} ,$$

which reduces to

$$\epsilon \leq e^{\ln \sqrt{2} - 2 \ln(2\sqrt{D})} \leq e^{-\ln(2\sqrt{D})} = \frac{1}{2\sqrt{D}} .$$

Provided that C is the starting configuration, let $p_{C'}$ denote the probability of arriving after $k \geq K$ steps at configuration C' . Then $p_{C'}$ is the product of the probabilities of both nodes arriving at the given coordinates. Therefore $p_{C'} \geq (\frac{1}{\sqrt{D}} - \epsilon)^2 \geq (\frac{1}{2\sqrt{D}})^2 \geq \frac{1}{4} \cdot \frac{1}{D}$. \square

PROOF OF LEMMA 11. First we prove that for any node v_i , its position after $\ell \leq L$ steps differs from its starting position by more than $\frac{\sqrt{D}}{12} = 2\sqrt{L \cdot 4 \log D}$ with the probability at most $\frac{1}{12}$. We replace the random walk, induced by the node movement, by a sequence of ℓ Bernoulli trials in the following way. The node A starts at point 0 of an infinite line. With probability $1/2$, A decreases its coordinate by 1, and with probability $1/2$, increases it by 1. For any fixed random bits the distance between A and point 0 generated by this process is at least as large as the distance between the current and the starting position of v_i created by the random walk.

Let H_ℓ be a random variable denoting the number of positive outcomes in this Bernoulli process after ℓ steps. Obviously $\mu := \mathbf{E}[H_\ell] = \ell/2$. Then it is sufficient to prove that, with probability at most $\frac{1}{2D}$, the number of positive outcomes is greater than $E[H_\ell] + \sqrt{L \cdot 4 \log D}$ or smaller than $E[H_\ell] - \sqrt{L \cdot 4 \log D}$. Since H_ℓ is the sum of independent 0/1 variables, then by applying the Chernoff bound [11] we get

$$\begin{aligned} \Pr[H_\ell \leq \mu - \sqrt{L \cdot (\log 4D)}] &\leq \Pr[H_\ell \leq \mu - \sqrt{l \cdot (\log 4D)}] \\ &= \Pr[H_\ell \leq \mu(1 - 2\sqrt{\frac{\log 4D}{\ell}})] \\ &\leq e^{-\frac{1}{2} \cdot (2\sqrt{\frac{\log 4D}{\ell}})^2 \cdot \mu} \\ &\leq \frac{1}{4D} . \end{aligned}$$

Since the process is symmetric (the same probability for increase and decrease), the same bound applies to $\Pr[H \geq \mu + \sqrt{\ell \cdot (\log 4D)}]$. We can bound the probability that at any of the L steps the distance between the starting and the current positions of v_i are greater than $\frac{\sqrt{D}}{12}$ by $L \cdot \frac{1}{2D} \leq \frac{1}{2}$. Since the movement of two different nodes is independent, with probability at least $\frac{1}{4}$ the distance between two nodes does not change by more than $\sqrt{D}/12 + \sqrt{D}/12 = \sqrt{D}/6$ in L time steps. \square

PROOF OF INEQUALITY 4. $C_{\text{OPT}}(J_i)$ are the random variables fulfilling $0 \leq C_{\text{OPT}}(J_i) \leq \frac{c}{\log^2 D} \cdot C_{\text{ALG}}(I_i) =: B_i$. Moreover, $\mathbf{E}[C_{\text{OPT}}(J_i)] = p \cdot \frac{c}{\log^2 D} \cdot C_{\text{ALG}}(I_i)$ and

$$B_i \leq \frac{c}{\log^2 D} \cdot (D + K) \cdot \mathcal{A} \leq 2c \cdot \frac{D^{3/2}}{\log D} .$$

Further, we denote $\mathbf{E}[\sum_{i \in S_{\max}} C_{\text{OPT}}(J_i)]$ by μ . By applying

Hoeffding inequality [11] we obtain

$$\begin{aligned} \Pr \left[\sum_{i \in S_{\max}} C_{\text{OPT}}(J_i) < \frac{1}{2} \cdot p \sum_{i \in S_{\max}} B_i \right] \\ &= \Pr \left[\sum_{i \in S_{\max}} C_{\text{OPT}}(J_i) < \mu - \sum_{i \in S_{\max}} \frac{1}{2} \cdot p \cdot B_i \right] \\ &\leq \exp \left(- \frac{2 \cdot (\sum_{i \in S_{\max}} \frac{1}{2} \cdot p \cdot B_i)^2}{\sum_{i \in S_{\max}} B_i^2} \right) \\ &\leq \exp \left(- \frac{p^2 \cdot (\sum_{i \in S_{\max}} B_i)^2}{2 \cdot (\sum_{i \in S_{\max}} B_i) \cdot \max_{i \in S_{\max}} B_i} \right) \\ &= \exp \left(- \frac{p^2 \cdot \frac{c}{\log^2 D} \cdot \sum_{i \in S_{\max}} C_{\text{ALG}}(I_i)}{2 \cdot 2c \cdot \frac{D^{3/2}}{\log D}} \right) \\ &\leq \exp \left(- \frac{p^2 \cdot \frac{1}{3} \cdot \sum_{i \geq 3} C_{\text{ALG}}(I_i)}{4 \cdot D^{3/2} \cdot \log D} \right) \\ &\leq \frac{1}{D} , \end{aligned}$$

where in the last inequality we used the fact that the request sequence is sufficiently long, i.e. it consists of at least $\frac{12}{p^2} \cdot D^{3/2} \cdot \log^2 D$ non-empty phases, each incurring a cost at least 1 on the algorithm. \square