

Page Migration in Dynamic Networks^{*}

Marcin Bienkowski¹ and Friedhelm Meyer auf der Heide²

¹ International Graduate School of Dynamic Intelligent Systems,
University of Paderborn, Germany
young@upb.de

² Heinz Nixdorf Institute and Computer Science Department,
University of Paderborn, Germany
fmadh@upb.de

1 Introduction

In the last couple of decades, network connected systems have gradually replaced centralized parallel computing machines. To provide smooth operation of network applications, the underlying system has to provide so-called *basic services*. One of the most crucial services is to provide a transparent access to data like variables, databases, memory pages, or files, which are shared by the instances of programs running at nodes of the network.

The traditional approach of storing the shared data in one or a few central repositories does not scale up well with the increase of the network size and is therefore inherently inefficient.

In this paper, we survey 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. This problem can be modeled as an online problem; several such models are discussed and will be presented in this survey. We will mainly deal with the classical, most basic of these data management problems, called *Page Migration*.

Our main focus will be on very recent results on page migration in a *dynamic* scenario: Here we assume that the network is no longer static, but it behaves like a mobile ad-hoc network, i.e., the nodes are allowed to move. Thus, we have to deal with two sources of online events, namely the requests from nodes to data items and the movements of the nodes. The new challenges both for modelling and for algorithm design and analysis arising from these *two adversaries* will be the main topic of this paper.

2 Static Networks

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

^{*} 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”.

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. To keep the bookkeeping overhead small, it is often required that only *one copy* of each object is stored in the system. Additionally, in a typical situation in the parallel environments, 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.

2.1 Page Migration

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 generated by sending a 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 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. Then the algorithm may move the page to an arbitrary node. Such a transaction incurs a cost which is greater by D , the page size factor, than the cost of sending one unit of data between these two nodes.

The goal is to compute a schedule of page movements to minimize the total cost. Furthermore, computing the 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 .

Competitive Analysis. To evaluate any online strategy, we use competitive analysis [35, 17], i.e. we compare the cost of an online solution to the cost of the optimal offline strategy. In the following we assume that an optimal solution is denoted by OPT, and for any algorithm ALG, $C_{\text{ALG}}(\mathcal{I})$ denotes the cost of this algorithm on input sequence $\mathcal{I} = (\sigma_t)_t$.

An online deterministic algorithm ALG is \mathcal{R} -competitive, if there exists a constant A , s.t. for any input sequence \mathcal{I} holds

$$C_{\text{ALG}}(\mathcal{I}) \leq \mathcal{R} \cdot C_{\text{OPT}}(\mathcal{I}) + A . \quad (1)$$

For a randomized algorithm ALG, we replace its cost in the definition above by its expectation $\mathbf{E}[C_{\text{ALG}}(\mathcal{I})]$. The expected value is taken over all possible random choices made by ALG. Additionally, we have to distinguish between the three adversary types: *oblivious*, *adaptive-online*, *adaptive-offline* (see e.g. [10]), depending on their knowledge of the random bits used by ALG.

Results. The PM problem was thoroughly investigated for different types of adversaries. While we shortly state the results below, for a gentle introduction to the algorithms mentioned here, we refer the reader to the survey by Bartal [7].

First randomized solutions presented by Westbrook [36] 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 et al. [9, 7]. 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 et al. [18]. It is matched only for certain topologies, like trees or uniform networks (see [18] and [22], respectively).

The first deterministic, phase-based, 7-competitive algorithm Move-To-Min was given by Awerbuch et al. [3]. The result was subsequently improved by the Move-To-Local-Min algorithm [8] attaining competitive ratio of 4.086. On the other hand, Chrobak et al. [18] showed a network with a lower bound of approximately 3.148.

2.2 Data Management

In this subsection we give a brief overview of extensions of Page Migration that allow more flexible data management in networks. Let n denote the number of nodes of the network. 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.

- How many copies of shared objects should be created?
- 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 et al. [9]. They present a randomized strategy that achieves an optimal competitive ratio of $\mathcal{O}(\log n)$ against an adaptive-online adversary, by a reduction to the online Steiner tree problem. Additionally, they show how

to get rid of the central control (which is useful for example for locating the nearest copy of the object) and create $\mathcal{O}(\log^4 n)$ -competitive algorithm which works in a distributed fashion. Awerbuch et al. [3] show that the randomization is not crucial by constructing deterministic algorithms (centralized and distributed ones) for file allocation problem attaining asymptotically the same ratios.

For uniform topologies Bartal et al. [9] showed an optimal 3-competitive deterministic algorithm. Lund et al. [22] gave a 3-competitive algorithm for trees based on *work functions* technique.

Memory Constraints. If multiple objects are present in the network and the local memory capacity at nodes is limited, then running file allocation scheme for each single object in the network might encounter some problems. Above all, it is not possible to copy an object into node's memory, if it is already full. Possibly, some other objects' 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, where file allocation solutions have to be combined with schemes known from *uni-processor paging* (see for example [1, 19, 24, 35]).

For uniform networks, Bartal et al. [9] presented the deterministic $\mathcal{O}(m)$ -competitive Distributed-Flush-When-Full algorithm, where m denotes the total number of copies that can be stored within the network. They also proved that this bound is tight by showing $\Omega(m)$ lower bound for competitiveness against an adaptive-online adversary. Awerbuch et al. [4] used randomized uni-processor paging algorithms [1, 19, 24] to get an up to a constant factor optimal algorithm HEAT & DUMP, which is $\mathcal{O}(\max\{\log(m - f), \log k\})$ -competitive against an oblivious adversary. In this context, f is the number of different objects in the network, and k is the maximum number of files that can be stored at any node. If we again restrict the number of copies of object to one, it results in a problem called *page migration with memory constraints*. Albers and Koga [2] presented deterministic and randomized algorithms for this problem, which are much simpler than their distributed paging counterparts, and attain competitive ratios $\mathcal{O}(n)$ and $\mathcal{O}(\log n)$, respectively.

For general networks Awerbuch et al. [5] adopted the model suggested primarily for uniprocessor paging [35], which goes beyond pure competitive analysis. In order to compensate the optimal offline algorithm advantage of knowing the future, Sleator and Tarjan [35] proposed limiting the memory capacity that the optimal algorithm has at its disposal. This extension, which is sometimes referred to as *resource augmentation*, allowed authors of [5] to present a deterministic $\mathcal{O}(\text{polylog } n)$ -competitive algorithm, under the assumption that the online algorithm has $\mathcal{O}(\text{polylog } n)$ times more memory than the optimal algorithm.

Optimizing Congestion. In case of wired networks the communication cost between a pair of nodes might be measured in terms of the load generated by sending the data through a communication link. All the algorithms presented above were designed to minimize the total communication load. A more chal-

lenging task it to derive fine-grained algorithms, whose objective is to minimize congestion, i.e. the maximum load on each single link.

Maggs et al. [23] developed a distributed data management strategy for tree networks, which was 3-competitive for the uniform model (the size of object equal to 1). The aforementioned 3-competitive algorithm for trees by Lund et al. [22] was proven to be also competitive with respect to congestion minimization, and worked for the non-uniform model. However, as it was based on computing work-functions, it was inherently centralized. Meyer auf der Heide et al. [25] fixed this deficiency, presenting the deterministic 3-competitive distributed strategy for trees.

However, the main result of [23] was *bisimulation technique*. It was shown that for some regular networks like meshes of clustered networks, the original problem instance can be, without enlarging congestion, mapped into a virtual network, a so called *access tree*. As mentioned above, solving the problem on a tree is relatively easy. Finally, the virtual tree was randomly mapped into the original network, so that, with high probability (w.h.p.), the congestion increases at most by a factor of $\mathcal{O}(\log n)$. This yields a randomized algorithm, which is $\mathcal{O}(\log n)$ -competitive against an oblivious adversary. Similar results for fat trees and hypercubic networks, as well as $\mathcal{O}(1)$ -competitive algorithms for uniform networks, were presented in [26, 37] and experimentally evaluated in [21]. Finally, Räcke [27, 28] showed that it is possible to construct access trees for any network topology, showing an $\mathcal{O}(\log^3 n)$ -competitive algorithm. This was subsequently improved to $\mathcal{O}(\log^2 n \cdot \log \log n)$ by Harrelson et al. [20].

Furthermore, Meyer auf der Heide et al. [26] and later Westermann [37] showed how to extend these strategies to respect the capacity constraints on the local memory modules. Their algorithms also exploit the paradigm of resource augmentation, giving the online algorithm $\mathcal{O}(\log n)$ times more memory than to the offline strategy. The competitive ratios are asymptotically the same as in the case without memory capacity restrictions.

3 Dynamic Networks

Basic services for mobile wireless networks and dynamically changing wired networks are a relatively new area. Some results have been achieved for topology control and routing in dynamic networks (see surveys by Rajaraman [29], and Scheideler [33], as well as the paper of Awerbuch et al. [6]). In comparison, data management solutions in dynamically changing networks are still in their infancy. Till recently, neither theoretical analysis was present in this area, nor a reasonable model of network changes was proposed. In particular, any model similar to described in [6], where we assume adversarial link failures, would give no chance to any data management scheme.

Hence, for theoretical modelling dynamics of networks, 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. Such a model can be motivated by a reality-close *pedestrian model* by Schindelbauer et al. [34]. The model of slow changes in the communication costs, formally defined in the next section, tries also to capture slow changes in available bandwidth in wired networks, which are inherently induced by other programs running in the network

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 et al. [5] in context of file allocation.

3.1 Models and Results

To model the Page Migration problem in dynamic networks we make the following assumptions. The network is modelled as a set of n mobile nodes (processors) labelled v_1, v_2, \dots, v_n . 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 .

Time is discrete and slotted into time steps $t = 0, 1, 2, \dots$. 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

$$d_t(v_a, v_b) := d(p_t(v_a), p_t(v_b)) . \quad (2)$$

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 is 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 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.

$$d(p_t(v_i), p_{t+1}(v_i)) \leq \delta , \quad (3)$$

for some fixed constant δ . Furthermore, if \mathcal{X} is a bounded metric space, then let λ denote its *diameter*, i.e. the maximum possible distance between two points from \mathcal{X} . For an unbounded space, $\lambda = \infty$.

Any two nodes are able to communicate directly with each other. 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

$$c_t(v_a, v_b) = d_t(v_a, v_b) + 1 , \quad (4)$$

if v_a and v_b are different nodes. Obviously, the communication within one node is free, i.e. if $v_a \equiv v_b$, then $c_t(v_a, v_b) = 0$. 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.

Naturally, the changes in the network themselves (described by $(\mathcal{C}_t)_{t=0}^T$ sequence) do not constitute a problem of its own. According to the described model of Page Migration, a copy of memory page of size D is stored at one of the network's nodes, initially at v_1 . In each time step t , exactly one node, denoted by 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. Further, we refer to them as *accesses* or *requests*. The requests σ_t create the sequence $(\sigma_t)_{t=1}^T$, complementary to the configuration sequence $(\mathcal{C}_t)_{t=0}^T$.³

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. A movement to $P'_{\text{ALG}}(t)$ incurs a cost $D \cdot c_t(P_{\text{ALG}}(t), P'_{\text{ALG}}(t))$.

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 *Dynamic Page Migration* (DPM) is to construct a schedule of page movements to minimize the total cost of communication for any pair of sequences $(\mathcal{C}_t)_t, (\sigma_t)_t$.

3.2 Competitive Analysis in Different Scenarios

Like in the Page Migration case, the problem of minimizing the total cost incurred is relatively 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. In fact, an easy algorithm using dynamic programming approach is able to find 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* scenario, where an algorithm must make its decisions (where to move the page)

³ Note that nodes issue requests from the first step. The initial configuration in time step 0 is introduced for simplifying notation only.

in time step t solely on the basis of the initial part of the input up to step t , i.e. on the sequence $\mathcal{C}_0, \mathcal{C}_1, \sigma_1, \mathcal{C}_2, \sigma_2, \dots, \mathcal{C}_t, \sigma_t$. To evaluate any online strategy for the DPM problem, we use competitive analysis. Since the input sequence consists of two practically independent streams, one describing the request patterns and one reflecting the changes in network topology, it is reasonable to assume that they are created by two separate adversarial entities, the request adversary and the configuration/network adversary. This separation yields different scenarios depending on ways in which these adversaries interact.

Adversarial (cooperative) scenario. The most straightforward modelling, which also creates the most difficult task to solve, arises when both adversaries may cooperate to create the combined input sequence. In fact, this is equivalent to having one adversary capable of constructing the whole input sequence and brings the problem back to the classical formulation of online analysis.

For this scenario, Bienkowski et al. [13] constructed a deterministic strategy, which is $\mathcal{O}(\min\{n \cdot \sqrt{D}, D, \lambda\})$ -competitive. Recall that λ denotes the maximum distance that can be achieved between two nodes. Their algorithm is up to a constant factor optimal, due to the matching lower bound for adaptive-online adversaries, given in [15]. Further, they show how to randomize this strategy to get a competitive ratio of $\mathcal{O}(\sqrt{D} \cdot \log n, D, \lambda)$ against an oblivious adversary. This result is up to a $\mathcal{O}(\sqrt{\log n})$ factor optimal in the common case $D \geq \log^3 n$, due to the lower bound of $\Omega(\min\{\sqrt{D} \cdot \log n, D^{2/3}, \lambda\})$ from [13]. All the presented competitive ratios are strict, which means that the constant A occurring in (1) is equal to zero.

The competitive ratios of the best possible algorithms for DPM problem are large, even against the weakest, oblivious adversaries. It can be inferred that the poor performance of algorithms for this scenario is caused by the fact that the network and request adversaries might combine their efforts in order to destroy our algorithm. If cooperation between them was forbidden, then one might hope for a provably better performance. However, it is semantically not clear what non-cooperativeness means. Therefore, it was proposed in [11, 14, 15] that the DPM problem could be analyzed in another extreme case, where one of the adversaries is replaced by a stochastic process. This leads to another two scenarios.

Brownian motion scenario. In this scenario the mobile nodes perform a random walk on a bounded area of diameter B , and the request adversary dictates which nodes issue requests during runtime. However, the adversary is “oblivious”, i.e. it has to create the whole request sequence $(\sigma_t)_t$ in advance, without knowledge of the actual configuration sequence $(\mathcal{C}_t)_t$ induced by a random walk. The definition of competitiveness has to be adapted appropriately to reflect the fact that the input sequence is created both by an adversary and a stochastic process. A deterministic algorithm ALG is \mathcal{R} -competitive with probability p , if there exists a constant A , s.t. for all request sequences $(\sigma_t)_t$ holds

$$\Pr_{(\mathcal{C}_t)_t} \left[C_{\text{ALG}}((\mathcal{C}_t)_t, (\sigma_t)_t) \leq \mathcal{R} \cdot C_{\text{OPT}}((\mathcal{C}_t)_t, (\sigma_t)_t) + A \right] \geq p, \quad (5)$$

where the probability is taken over all possible configuration sequences generated by the random movement.

The main result of [14], based on the preliminary result of [15] is an algorithm MAJ, which is $\mathcal{O}(\min\{\sqrt[4]{D}, n\} \cdot \text{polylog}(B, D, n))$ -competitive. This result holds for 1-dimensional areas if $B \leq \tilde{\mathcal{O}}(\sqrt{D})$, or for any constant-dimensional areas if $B \geq \tilde{\mathcal{O}}(\sqrt{D})$. The ratio is achieved w.h.p., i.e. the probability p occurring in (5) can be amplified to $1 - D^{-\alpha}$ by setting $A = \alpha \cdot A_0$ for a fixed constant A_0 .

Stochastic requests scenario. This is the scenario symmetric to the Brownian motion one. It is assumed that requests appear with some given frequencies, i.e. in step t , σ_t is a node chosen randomly according to a fixed probability distribution π . Analogously, a deterministic algorithm ALG is \mathcal{R} -competitive with probability p , if there exists a constant A , s.t. for all possible network topology changes (configuration sequences) $(\mathcal{C}_t)_t$ and all possible probability distributions π holds

$$\Pr_{(\sigma_t)_t} \left[C_{\text{ALG}}((\mathcal{C}_t)_t, (\sigma_t)_t) \leq \mathcal{R} \cdot C_{\text{OPT}}((\mathcal{C}_t)_t, (\sigma_t)_t) + A \right] \geq p , \quad (6)$$

where the probability is taken over all possible request sequences $(\sigma_t)_t$ generated according to π .

The Move-To-First-Request algorithm presented in [11] achieves strict $\mathcal{O}(1)$ -competitive ratio, w.h.p. In this context, high probability means that one can achieve probability $1 - D^{-\alpha}$, if the input sequence is sufficiently long. Moreover, the algorithm can be slightly modified to handle also the following cost function

$$c_t(v_a, v_b) = (d_t(v_a, v_b))^\beta + 1 , \quad (7)$$

for any constant β , still remaining $\mathcal{O}(1)$ -competitive. For the case of wireless radio networks, one can choose the parameter β to respect a *propagation exponent* of the medium (see for example [31]). For example by setting $\beta = 2$, the cost definition reflects the energy consumption used to send the message in the ideally free space along a given distance. Thus, this result minimizes, up to a constant factor, the total energy used in the system.

4 Algorithms and Lower Bounds

In this section we give some technically interesting results for the DPM model. First, we present MARK, the main building block of the $\mathcal{O}(\min\{n \cdot \sqrt{D}, D, \lambda\})$ upper bound for competitiveness in the adversarial scenario. Later, we show that this ratio is inherently high by showing a lower bound of $\Omega(\min\{\sqrt{D}, \lambda\})$ (which works even in two-node networks) for a randomized algorithm against an oblivious adversary. Finally, we present a simple majority algorithm, which is $\mathcal{O}(\log n)$ -competitive, w.h.p., in a very restricted version of the Brownian motion scenario.

4.1 Algorithm MARK.

The $\mathcal{O}(n \cdot \sqrt{D})$ -competitive deterministic MARK algorithm [13] for the adversarial scenario of the DPM problem was inspired by the Move-To-Min algorithm [3] for the regular Page Migration problem. Move-To-Min divides the whole input sequence works into chunks of length D . In any chunk, it serves all the requests, and move the page at the end of the chunk to a so called *gravity center*. A gravity center is a node, which would be the best place for a page in this chunk.

MARK works in chunks of length \sqrt{D} . This length constitutes a tradeoff – it has to be long enough to amortize the movement of the page against the cost of serving the requests in the chunk, and short enough to make the adversarial network changes negligible. However, it can be shown that any algorithm, which considers *only* gravity centers as candidates for the nodes holding the page, has no chance to be better than $\Omega(D)$ -competitive.

On the other hand, keeping the page close to the gravity center is, generally, a desirable thing. Hence, we consider the following marking scheme, which depends only on the input sequence. Chunks are grouped in epochs, each epoch begins with all nodes unmarked. First epoch starts with the beginning of the input. In each epoch we track A_i counters for the part of the epoch seen so far. A_i counter is the cost of an algorithm, which remains at v_i , and does not move. If such a counter exceeds D , then the corresponding node becomes marked. At the end of a chunk, in which all nodes are already marked, the current epoch ends, the scheme unmarks all nodes, and a new epoch begins.

MARK uses this scheme in the following way. It remains in a node till the end of chunk, in which this node gets marked, and then moves to any not yet marked node. Additionally, at the end of the last chunk in epoch, it moves to the gravity center associated with this chunk.

It can be proven, that even considering the adversarial movement of the nodes, if a node remains far away from the gravity center, A_i counter increases rapidly, which leads to marking the node.

Lemma 1 ([13]). *If at the end of a chunk I a node is not marked, then its distance to the gravity center is at most $\mathcal{O}(\sqrt{D})$.*

Thus, if MARK moves, it moves to the neighborhood of the gravity center. Denoting the sequence of chunks between two movements of MARK by *phase*, and using similar kind of amortized analysis (with adequately chosen potential function) as for Move-To-Min algorithm, the following can be shown.

Lemma 2 ([13]). *In each phase the amortized cost of MARK is not greater than the cost of OPT times $\mathcal{O}(\sqrt{D})$, plus an additive term of $\mathcal{O}(D \cdot \sqrt{D})$*

However, we may eradicate this additive term by resorting to the properties of the marking scheme. First, since in each phase at least one new node gets marked, the number of phases in one epoch is at most n . Second, the OPT's cost in one epoch is at least D . It follows from the case analysis: if OPT moves then it is charged at least D , otherwise it remains in one node v_i , and thus

its cost is equal to $A_i \geq D$. Hence, the additive terms in one epoch amount to $\mathcal{O}(n \cdot D \cdot \sqrt{D})$, which is at most $\mathcal{O}(n \cdot \sqrt{D})$ times the optimal cost. This concludes the proof of MARK's competitiveness.

Straightforward generalization of MARK, i.e. choosing not any, but a random not yet marked node, reduces the number of phases to $\log n$ and the competitive ratio to $\mathcal{O}(\sqrt{D} \cdot \log n)$. Choosing different chunks' length and a refined randomization presented in [12] yields a competitive ratio of $\mathcal{O}(\sqrt{D \cdot \log n})$ against oblivious adversary.

4.2 A Lower Bound against an Oblivious Adversary

Let $B_{\text{exp}} = \min\{\sqrt{D}, \lambda\}$. We construct a probability distribution π over inputs of arbitrary length and prove that for any deterministic algorithm DET, which knows this distribution, holds $\mathbf{E}_{\pi}[C_{\text{DET}}(\mathcal{I})] \geq \Omega(B_{\text{exp}}) \cdot \mathbf{E}_{\pi}[C_{\text{OPT}}(\mathcal{I})]$. Then, the lower bound of $\Omega(B_{\text{exp}})$ for any randomized algorithm against oblivious adversary follows directly from the Yao min-max principle [38, 17].

We divide input into phases, each of length $D + 2 \cdot B_{\text{exp}}$ steps. Each phase consists of *expanding part*, (B_{exp} steps), *main part* (D steps), and *contracting part* (also B_{exp} steps). Each phase begins with v_1 and v_2 occupying the same point in the space. Then within the expanding part, nodes are moved apart, so that in the t -th step of the expanding part the distance between them is $t - 1$. Throughout the whole main part the distance amounts to B_{exp} . Finally, in the contracting part nodes are moved closer to each other, so that at the end of the phase they meet again. Note that the movement of the nodes is fixed deterministically.

In the expanding part all the requests are issued at v_1 , and all the requests of the contracting one occur at v_2 . Further, in the main part, with probability $1/2$, all the requests are issued at v_1 , and, with probability $1/2$, all the requests are issued at v_2 .

We concentrate on one single phase P . It is relatively easy to show that OPT pays at most $\mathcal{O}(D)$ in each phase. On the other hand, a deterministic online algorithm DET can base its decisions only on the past requests. In particular, in the last step of the expanding part it has to decide whether to end this step at v_1 or v_2 . Independently of DET's choice, with probability $1/2$, all the next D requests in the main part are given at the opposite node. In this case DET has two options. If it moves the page within the main part, then it pays $D \cdot B_{\text{exp}}$. Otherwise, it pays $D \cdot B_{\text{exp}}$ for serving the requests during this part. Hence, the expected cost of DET in one phase is at least $\frac{1}{2} \cdot D \cdot B_{\text{exp}}$.

Thus, $\mathbf{E}_{\pi}[C_{\text{DET}}(P)] = \Omega(B_{\text{exp}}) \cdot C_{\text{OPT}}(P)$. Since we may construct arbitrarily long input sequences, the lower bound follows by linearity of expected value.

4.3 The Majority Algorithm

We analyze the Brownian motion scenario in a simplified setting, where only two nodes perform a random walk on a discrete ring of size $B = \sqrt{D}$. In each time

step the coordinate of a node, with probability $1/3$, increases by 1, decreases by 1, or remains the same.

Algorithm MAJ simply divides the input into phases P_1, P_2, P_3, \dots , each of length $B^2 = D$. At the end of each phase, it moves to the node which issued majority of requests in this phase.

We sketch a proof that MAJ is $\mathcal{O}(\log n)$ -competitive, w.h.p. We neglect the cost of MAJ in the first two phases, putting it into additive constant A , occurring in (5). The remaining phases are divided into three disjoint, alternating sets $\mathcal{M}_i = \{P_j : j \equiv i \pmod{3}\}$. Naturally, there exists a set \mathcal{M}_χ , which incurs at least $1/3$ of the total cost, hence we need to bound $C_{\text{MAJ}}(\mathcal{M}_\chi)$ only.

The crucial part is relating $C_{\text{MAJ}}(P_j)$ to $C_{\text{OPT}}(P_{j-1} \uplus P_j)$, for any phase P_j . We charge MAJ $\mathcal{O}(B)$ for any request issued not at the node holding its page, and $\mathcal{O}(D \cdot B)$ for moving its page.

Lemma 3 ([15, 14]). *For each phase P_j there exists a critical subphase $P'_j \subseteq P_{j-1} \uplus P_j$, of length $\Theta(\frac{1}{\log D}) \cdot D$, s.t. P'_j is similar to P_j .*

Similarity means that, under the assumption that within P'_j the distance between nodes is $\Omega(B)$, the cost of any algorithm ALG is $C_{\text{ALG}}(P'_j) = \Omega(1/\log D) \cdot C_{\text{MAJ}}(P_j)$. The key observation helping to prove the lemma above is that even if MAJ is at node v_1 within P_j , and all the requests are given at v_2 (and thus $C_{\text{MAJ}}(P_j) = \Omega(B \cdot D)$), then in the previous phase P_{j-1} the majority of requests must have been issued at v_1 . Thus, we are able to find a subphase with roughly the same number of requests of v_1 and v_2 .

Naturally, in the subphase nodes might be at a distance of $o(B)$. The following lemma assures that this is frequently not the case.

Lemma 4 ([14]). *Let P'_{j-3} and P'_j be two consecutive critical subphases. For any configuration at the end of P'_{j-3} , with a constant probability, within P'_j nodes are at the distance $\Omega(B)$.*

The proof utilizes two facts. First, at least B^2 steps separate P'_{j-3} and P'_j . The Markov chain induced by nodes' random walks converges relatively quickly (see [32]), i.e. after B^2 steps the position of nodes are almost uniform. Thus, with a constant probability nodes are at distance $\Omega(B)$ at the beginning of P'_j . Moreover, if their initial distance is $\Omega(B)$, then during $\mathcal{O}(B^2/\log B)$ steps, they approximately maintain this distance.

By Lemma 4 we get $C_{\text{MAJ}}(P_j) \leq \mathcal{O}(\log D) \cdot \mathbf{E}[C_{\text{OPT}}(P'_j)]$. Moreover, the expected value of this bound on OPT is taken only on the random walk in phases P_{j-2}, P_{j-1} , and P_j , and thus for different $P_j \in \mathcal{M}_\chi$, $C_{\text{OPT}}(P'_j)$ are independent random variables. Hence, we may use Hoeffding inequality [30] to show that $\sum_{P_j \in \mathcal{M}_\chi} C_{\text{OPT}}(P'_j)$ is sharply concentrated around its mean value.

References

1. D. Achlioptas, M. Chrobak, and J. Noga. Competitive analysis of randomized paging algorithms. *Theoretical Computer Science*, 234(1-2):203–218, 2000.

2. S. Albers and H. Koga. Page migration with limited local memory capacity. In *Proc. of the 4th Int. Workshop on Algorithms and Data Structures (WADS)*, pages 147–158, 1995.
3. 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.
4. B. Awerbuch, Y. Bartal, and A. Fiat. Heat & dump: Competitive distributed paging. In *Proc. of the 34th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 22–31, 1993.
5. B. Awerbuch, Y. Bartal, and A. Fiat. Distributed paging for general networks. *Journal of Algorithms*, 28(1):67–104, 1998. Also appeared in *Proc. of the 7th SODA*, pages 574–583, 1996.
6. B. Awerbuch, A. Brinkmann, and C. Scheideler. Anycasting in adversarial systems: routing and admission control. In *Proc. of the 30th Int. Colloq. on Automata, Languages and Programming (ICALP)*, pages 1153–1168, 2003.
7. Y. Bartal. Distributed paging. In *Dagstuhl Workshop on On-line Algorithms*, pages 97–117, 1996.
8. Y. Bartal, M. Charikar, and P. Indyk. On page migration and other relaxed task systems. *Theoretical Computer Science*, 268(1):43–66, 2001. Also appeared in *Proc. of the 8th SODA*, pages 43–52, 1997.
9. Y. Bartal, A. Fiat, and Y. Rabani. Competitive algorithms for distributed data management. *Journal of Computer and System Sciences*, 51(3):341–358, 1995. Also appeared in *Proc. of the 24th STOC*, pages 39–50, 1992.
10. 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.
11. M. Bienkowski. Dynamic page migration with stochastic requests. In *Proc. of the 17th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, 2005. To appear.
12. M. Bienkowski and J. Byrka. Bucket game with applications to set multicover and dynamic page migration. Unpublished manuscript, 2005.
13. M. Bienkowski, M. Dynia, and M. Korzeniowski. Improved algorithms for dynamic page migration. In *Proc. of the 22nd Symp. on Theoretical Aspects of Computer Science (STACS)*, pages 365–376, 2005.
14. M. Bienkowski and M. Korzeniowski. Dynamic page migration under brownian motion. In *Proc. of the European Conf. in Parallel Processing (Euro-Par)*, 2005. To appear.
15. M. Bienkowski, M. Korzeniowski, and 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)*, pages 64–73, 2004.
16. 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.
17. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
18. M. Chrobak, L. L. Larmore, N. Reingold, and J. Westbrook. Page migration algorithms using work functions. In *Proc. of the 4th Int. Symp. on Algorithms and Computation (ISAAC)*, pages 406–415, 1993.
19. A. Fiat, R. M. Karp, M. Luby, L. A. McGeoch, D. D. Sleator, and N. E. Young. Competitive paging algorithms. *Journal of Algorithms*, 12(4):685–699, 1991.

20. C. Harrelson, K. Hildrum, and S. Rao. A polynomial-time tree decomposition to minimize congestion. In *Proc. of the 15th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 34–43, 2003.
21. C. Krick, F. Meyer auf der Heide, H. Räcke, B. Vöcking, and M. Westermann. Data management in networks: Experimental evaluation of a provably good strategy. *Theory of Computing Systems*, 2:217–245, 2002. Also appeared in *Proc. of the 11th SPAA*, pages 165–174, 1999.
22. C. Lund, N. Reingold, J. Westbrook, and D. C. K. Yan. Competitive on-line algorithms for distributed data management. *SIAM Journal on Computing*, 28(3):1086–1111, 1999. Also appeared as On-Line Distributed Data Management in *Proc. of the 2nd ESA*, pages 202–214, 1994.
23. B. M. Maggs, F. Meyer auf der Heide, B. Vöcking, and M. Westermann. Exploiting locality for data management in systems of limited bandwidth. In *Proc. of the 38th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 284–293, 1997.
24. L. A. McGeoch and D. D. Sleator. A strongly competitive randomized paging algorithm. *Algorithmica*, 6(6):816–825, 1991.
25. F. Meyer auf der Heide, B. Vöcking, and M. Westermann. Provably good and practical strategies for non-uniform data management in networks. In *Proc. of the 7th European Symp. on Algorithms (ESA)*, pages 89–100, 1999.
26. F. Meyer auf der Heide, B. Vöcking, and M. Westermann. Caching in networks. In *Proc. of the 11th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 430–439, 2000.
27. H. Räcke. Minimizing congestion in general networks. In *Proc. of the 43rd IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 43–52, 2002.
28. H. Räcke. *Data management and routing in general networks*. PhD thesis, Universität Paderborn, 2003.
29. R. Rajaraman. Topology control and routing in ad hoc networks: a survey. *SIGACT News*, 33(2):60–73, 2002.
30. S. Rajesekaran, P. M. Pardalos, J. H. Reif, and J. Rolim. *Handbook of Randomized Computing*, volume II. Kluwer Academic Publishers, 2001.
31. T. S. Rappaport. *Wireless Communications: Principles and Practices*. Prentice Hall, 1996.
32. J. S. Rosenthal. Convergence rates for Markov chains. *SIAM Review*, 37(3):387–405, 1995.
33. C. Scheideler. Models and techniques for communication in dynamic networks. In *Proc. of the 19th Symp. on Theoretical Aspects of Computer Science (STACS)*, pages 27–49, 2002.
34. C. Schindelhauer, T. Lukovszki, S. Rührup, and K. Volbert. Worst case mobility in ad hoc networks. In *Proc. of the 15th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 230–239, 2003.
35. D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.
36. J. Westbrook. Randomized algorithms for multiprocessor page migration. *DI-MACS Series in Discrete Mathematics and Theoretical Computer Science*, 7:135–150, 1992.
37. M. Westermann. *Caching in Networks: Non-Uniform Algorithms and Memory Capacity Constraints*. PhD thesis, Universität Paderborn, 2000.
38. 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)*, pages 222–227, 1977.