Dynamic Beats Fixed: On Phase-Based Algorithms for File Migration*†

Marcin Bienkowski[1], Jarosław Byrka[2], and Marcin Mucha[3]

1 **Institute of Computer Science, University of Wrocław, Wrocław, Poland**
`mbi@cs.uni.wroc.pl`
2 **Institute of Computer Science, University of Wrocław, Wrocław, Poland**
`jby@cs.uni.wroc.pl`
3 **Institute of Informatics, University of Warsaw, Warsaw, Poland**
`mucha@mimuw.edu.pl`

Abstract

In this paper, we construct a deterministic 4-competitive algorithm for the online file migration problem, beating the currently best 20-year old, 4.086-competitive MTLM algorithm by Bartal et al. (SODA 1997). Like MTLM, our algorithm also operates in phases, but it adapts their lengths dynamically depending on the geometry of requests seen so far. The improvement was obtained by carefully analyzing a linear model (factor-revealing LP) of a single phase of the algorithm. We also show that if an online algorithm operates in phases of fixed length and the adversary is able to modify the graph between phases, no algorithm can beat the competitive ratio of 4.086.

1998 ACM Subject Classification F.1.2 [Modes of Computation] Online Computation, G.1.6 [Optimization] Linear programming, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases file migration, factor-revealing linear programs, online algorithms, competitive analysis

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.13

1 Introduction

Consider the problem of managing a shared data item among sets of processors. For example, in a distributed program running in a network, nodes want to have access to shared files, objects or databases. Such a file can be stored in the local memory of one of the processors and when another processor wants to access (read from or write to) this file, it has to contact the processor holding the file. Such a transaction incurs a certain cost. Moreover, access patterns to this file may change frequently and unpredictably, which renders any static placement of the file inefficient. Hence, the goal is to minimize the total cost of communication by moving the file in response to such accesses, so that the requesting processors find the file "nearby" in the network.

The *file migration* problem serves as the theoretical underpinning of the application scenario described above. The problem was coined by Black and Sleator [13] and was initially called *page migration*, as the original motivation concerned managing a set of memory pages

* Extended abstract; the full version is available at `https://arxiv.org/abs/1609.00831`.

† Partially supported by Polish National Science Centre grants 2016/22/E/ST6/00499 and 2015/18/E/ST6/00456. The work of M. Mucha is part of a project TOTAL that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 677651).

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 13; pp. 13:1–13:14

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

in a multiprocessor system. There the data item was a single memory page held at a local memory of a single processor.

Most subsequent work referred to this problem as *file migration* and we will stick to this convention in this paper. The file migration problem assumes the *non-uniform model*, where the shared file is much larger than a portion accessed in a single time step. This is typical when in one step a processor wants to read a single unit of data from a file or a record from a database. On the other hand, to reduce the maintenance overhead, it is assumed that the shared file is indivisible, and can be migrated between nodes only as a whole. This makes the file migration much more expensive than a single access to the file. As the knowledge of future accesses is either partial or completely non-existing, the accesses to the file can be naturally modeled as an online problem, where the input sequence consists of processor identifiers, which sequentially try to access pieces of the shared file.

1.1 The Model

The studied network is modeled as an edge-weighted graph or, more generally, as a metric space $(\mathcal{X}, d)$ whose point set $\mathcal{X}$ corresponds to processors and d defines the distances between them. There is a large indivisible *file* (historically called *page*) of size D stored at a point of $\mathcal{X}$. An input is a sequence of space points $r_1, r_2, r_3, \ldots$ denoting processors requesting access to the file. This sequence is presented in an online manner to an algorithm. More precisely, we assume that the time is slotted into steps numbered from 1. Let ALG_t denote the position of the file at the end of step t and ALG_0 be the initial position of the file. In step $t \geq 1$, the following happens:

1. A requesting point r_t is presented to the algorithm.
2. The algorithm pays $d(\text{ALG}_{t-1}, r_t)$ for serving the request.
3. The algorithm chooses a new position ALG_t for the file (possibly $\text{ALG}_t = \text{ALG}_{t-1}$) and moves the file to ALG_t paying $D \cdot d(\text{ALG}_{t-1}, \text{ALG}_t)$.

After the t-th request, the algorithm has to make its decision (where to migrate the file) exclusively on the basis of the sequence up to step t. To measure the performance of an online strategy, we use the standard competitive ratio metric [14]: an online deterministic algorithm Alg is *c-competitive* if there exists a constant γ, such that for any input sequence $\mathcal{I}$, it holds that $C_{\text{ALG}}(\mathcal{I}) \leq c \cdot C_{\text{OPT}}(\mathcal{I}) + \gamma$, where C_{ALG} and C_{OPT} denote the costs of Alg and Opt (optimal *offline* algorithm) on $\mathcal{I}$, respectively. The minimum c for which Alg is c-competitive is called the *competitive ratio* of Alg.

1.2 Previous Work

The problem was stated by Black and Sleator [13], who gave 3-competitive deterministic algorithms for uniform metrics and trees and conjectured that 3-competitive deterministic algorithms were possible for any metric space.

Westbrook [26] constructed randomized strategies: a 3-competitive algorithm against adaptive-online adversaries and a $(1 + \phi)$-competitive algorithm (for D tending to infinity) against oblivious adversaries, where $\phi \approx 1.618$ denotes the golden ratio. By the result of Ben-David et al. [10] this asserted *the existence* of a deterministic algorithm with the competitive ratio at most $3 \cdot (1 + \phi) \approx 7.854$.

The first explicit deterministic construction was the 7-competitive algorithm Move-To-Min (Mtm) by Awerbuch et al. [2]. Mtm operates in phases of length D, during which the algorithm *remains at a fixed position*. In the last step of a phase, Mtm migrates the file to

a point that minimizes the sum of distances to all requests $r_1, r_2, \ldots, r_D$ presented in the phase, i.e., to a minimizer of the function $f_{\text{MTM}}(x) = \sum_{i=1}^{D} d(x, r_i)$.

The ratio has been subsequently improved by the algorithm Move-To-Local-Min (Mtlm) by Bartal et al. [8]. Mtlm works similarly to Mtm, but it changes the phase duration to $c_0 \cdot D$ for a constant c_0, and when computing the new position for the file, it also takes the migration distance into consideration. Namely, it chooses to migrate the file to a point that minimizes the function

$$f_{\text{MTLM}}(x) = D \cdot d(v_{\text{MTLM}}, x) + \tfrac{c_0+1}{c_0} \sum_{i=1}^{c_0 \cdot D} d(x, r_i) \ ,$$

where v_{MTLM} denotes the point at which Mtlm keeps its file during the phase. The algorithm is optimized by setting $c_0 \approx 1.841$ being the only positive root of the equation $3c^3 - 8c - 4 = 0$. For such c, the competitive ratio of Mtlm is $R_0 \approx 4.086$, where R_0 is the largest (real) root of the equation $R^3 - 5R^2 + 3R + 3 = 0$. Their analysis is tight.

It is worth noting that most of the competitive ratios given above hold when D tends to infinity. In particular, for Mtlm we assume that $c_0 \cdot D$ is an integer and the ratio of $1 + \phi$ of Westbrook's algorithm [26] is achieved only in the limit.

Better deterministic algorithms are known only for some specific graph topologies. There are 3-competitive algorithms for uniform metrics and trees [13], and $(3 + 1/D)$-competitive strategies for three-point metrics [23]. Chrobak et al. [15] showed $2 + 1/(2D)$-competitive strategies for continuous trees and products of trees, e.g., for $\mathbb{R}^n$ with ℓ_1 norm. Furthermore, a $(1 + \phi)$-competitive algorithm for $\mathbb{R}^n$ under any norm was also given in [15].

A straightforward lower bound of 3 for deterministic algorithms was given by Black and Sleator [13] and later adapted to randomized algorithms against adaptive-online adversaries by Westbrook [26]. The currently best lower bound for deterministic algorithms is due to Matsubayashi [22], who showed a lower bound of $3 + \varepsilon$ that holds for any value of D, where ε is a constant that does not depend on D. This renders the file migration problem one of the few natural problems, where a known lower bound on the competitive ratio of any deterministic algorithm is strictly larger than the competitive ratio of a randomized algorithm against an adaptive-online adversary.

Finally, improved results were given for a simplified model where $D = 1$: the competitive ratio for deterministic algorithms is then known to be between 3.164 and 3.414 [21].

1.3 Our Contribution

We propose a new deterministic algorithm that dynamically decides on the length of the phase based on the geometry of requests received in the initial part of each phase. This improves the 20-years old result of Bartal et al. [8].

The improvement was obtained by carefully analyzing a linear model (factor revealing LP) of a single phase of the algorithm. It allowed us to identify some key tight examples for the previous analysis, suggested a nontrivial construction of the new algorithm, and facilitated a systematic search within the design parameter space.

More precisely, for a fixed algorithm Alg (from a relatively broad class), we create a maximization LP with the following property: if the competitive ratio of Alg is at least R, then so is the value of LP. A solution to the LP contains a succinct description of a metric space along with a short description of a single-phase input, both constituting a lower bound for Alg. Hence, the value of LP is an upper bound on Alg's competitiveness. We discuss the details of the LP approach in Section 4.

The way the algorithm was obtained is perhaps unintuitive. Nevertheless, the final algorithm is an elegant construction involving only essentially integral constants. By studying

the dual solution, we managed to extract a compact, human-readable, combinatorial upper bound based on path-packing arguments and to obtain the following result.

▶ **Theorem 1.** *There exists a deterministic 4-competitive algorithm for the file migration problem.*

We also show that improvement of MTLM would not be possible by just selecting different parameters for a phase-based algorithm operating with a fixed phase length. Our construction shows that an analysis that treats each phase separately (e.g., the one employed for MTLM [8]) cannot give better bounds on the competitive ratio than 4.086. (A weaker lower bound of 3.847 for algorithms that use fixed phase length was given by Bartal et al. [8].)

▶ **Theorem 2.** *Fix any algorithm* ALG *that operates in phases of fixed length. Assume that between the phases, the adversary can arbitrarily modify the graph while keeping the distance between the files of* OPT *and* ALG *unchanged. Then, the competitive ratio of* ALG *is at least R_0 (for D tending to infinity), where $R_0 \approx 4.086$ is the competitive ratio of algorithm* MTLM*.*

1.4 Other Related Work

The file migration problem has been generalized in a few directions. When we lift the restriction that the file can only be migrated and not copied, the resulting problem is called *file allocation* [9, 2, 18]. It makes sense especially when we differentiate read and write requests to the file; for the former, we need to contact only one replica of the file; for the latter all copies need to be updated. The attainable competitive ratios become then worse: the best deterministic algorithm is $O(\log n)$-competitive [2]; the lower bound of $\Omega(\log n)$ holds even for randomized algorithms and follows by a reduction from the online Steiner tree problem [9, 17].

The file migration problem has been also extended to accommodate memory capacity constraints at nodes (when more than one file is used) [1, 3, 4, 6], dynamically changing networks [4, 12], and different objective functions (e.g., minimizing congestion) [19, 25]. For a more systematic treatment of the file migration and related problems, see surveys [7, 11]. For more applied approaches, see the survey [16] and the references therein.

2 4-Competitive Algorithm Dynamic-Local-Min

We start with an insight concerning the hard inputs for the MTLM algorithm [8]. We identified two classes of tight instances for MTLM: *bipartite* and *linear* (cf. Figure 1). It can be shown that if the algorithm knew in advance on which instance it was run, it could improve its performance by changing the phase length. Namely, for bipartite instances a longer phase would help the algorithm, whereas a shorter phase would be beneficial for linear instances.

To decide the length of the phase, we need to measure the level of request concentration as compared to the distance from the current position of an algorithm to the center of requests. Intuitively, observing that (from some time) requests are concentrated around a certain point motivates the algorithm to shorten the phase and quickly move to the "center of the requests". If, on the other hand, requests are scattered and the current algorithm's position is essentially in the middle of the observed requests, it appears reasonable to wait longer before moving the file. This rule agrees with the desired behavior of the algorithm on linear and bipartite instances.

Turning the above intuition into an effective phase extension rule is not trivial. We present an algorithm based on a rule that we have extracted from an optimization process

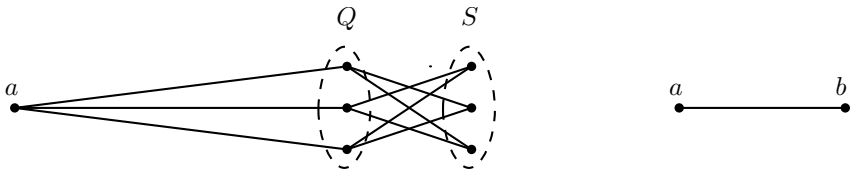


Figure 1 The geometries of selected tight instances for MTLM. In both cases, the algorithm starts at point a. In the *linear* instance (on the right), the requests are initially given at a and then later at b, and the algorithm is expected to migrate the file to point b. In the *bipartite* instance (on the left) the requests are given at nodes from set S and the algorithm is expected to migrate the file to one of the nodes from set Q.

using a natural linear model of the amortized phase-based analysis. This linear model is quite complex and we present it in Section 4. It can be seen as an alternative (computer-based) proof for the performance guarantee of our algorithm. Such proof technique might be interesting on its own and useful for analyzing other online games played on metric spaces.

2.1 Notation

For succinctness, we introduce the following notions. For any two points $v_1, v_2 \in \mathcal{X}$, let $[v_1, v_2] = D \cdot d(v_1, v_2)$. We extend this notation to sequences of points, i.e., $[v_1, v_2, \ldots, v_j] = [v_1, v_2] + [v_2, v_3] + \ldots + [v_{j-1}, v_j]$. Moreover, if $v \in \mathcal{X}$ is a point and $S \subseteq \mathcal{X}$ is a multiset of points, then

$$[v, S] = [S, v] = D \cdot \tfrac{1}{|S|} \textstyle\sum_{x \in S} d(v, x) \ ,$$

i.e., $[v, S]$ is the average distance from v to a point of S times D. We extend the sequence notation introduced above to sequences of points and multisets of points, e.g., $[v, S, u, T] = [v, S] + [S, u] + [u, T]$. The symbol $[S, T]$ is not defined for multisets S, T; we will only use this notation for sequences that do not contain two consecutive multisets.

Observe that the sequence notation allows for easy expressing of the triangle inequality: $[v_1, v_2] \leq [v_1, v_3, v_2]$; we will extensively use this property. Note that the following "multiset" version of the triangle inequality also holds: $[v_1, v_2] \leq [v_1, S, v_2]$.

2.2 Algorithm definition

We propose a new phase-based algorithm that dynamically decides on the length of the current phase, which we call Dynamic-Local-Min (DLM). DLM operates in phases, but it chooses their lengths depending on the geometry of requests seen in the initial part of the phase. Roughly speaking, when it recognizes that the currently seen requests more closely resemble a linear tight example for MTLM, it ends the phase after $1.75\,D$ steps. Otherwise, it assumes that the presented graph is more in the flavor of the bipartite construction, and ends the phase only after $2.25\,D$ steps.

For any step t, we denote the position of DLM's file at the end of step t by DLM_t and that of OPT by OP_t. We identify the requests with the points where they are issued.

Assume a phase starts in step $t+1$; that is, DLM_t is the position of DLM at the very beginning of a phase. Within the phase, DLM waits $1.75\,D$ steps and at step $t + 1.75\,D$, it finds a node v_g that minimizes the function

$$g(v) = [\text{DLM}_t, v, \mathcal{R}_1, v, \mathcal{R}_2] = [\text{DLM}_t, v] + 2 \cdot [v, \mathcal{R}_1] + [v, \mathcal{R}_2] \ ,$$

where $\mathcal{R}_1$ is the multiset of the requests from steps $t+1,\ldots,t+D$ and $\mathcal{R}_2$ is the multiset of the subsequent requests from steps $t+D+1,\ldots,t+1.75\,D$.

If $g(v_g) \le 1.5 \cdot [\textsc{dlm}_t, \mathcal{R}_2]$, the algorithm moves its file to v_g, and ends the current phase. Intuitively, this condition corresponds to detecting if there exists a point that is substantially closer to the first $1.75\,D$ requests of the phase than the current position. The condition is constructed in a way to be conclusive for each of the possible outcomes. If indeed such point exists, then by moving the file to this point, we get much closer to the file of Opt. If there is no such good point, then also the optimal solution is experiencing some request related costs. Then, we may afford to wait a little longer and meanwhile get a more accurate estimation of the possible location of the file of Opt.

That is, if $g(v_g) > 1.5 \cdot [\textsc{dlm}_t, \mathcal{R}_2]$, Dlm waits the next $0.5\,D$ steps and (in step $t+2.25\,D$) it moves its file to the point v_h, where v_h is the minimizer of the function

$$h(v) = [\textsc{dlm}_t, v] + [v, \mathcal{R}_1] + 1.25 \cdot [v, \mathcal{R}_2] + 0.75 \cdot [v, \mathcal{R}_3] \ .$$

$\mathcal{R}_3$ is the multiset of the last $0.5\,D$ requests from the prolonged phase (from steps $t+1.75\,D+1,\ldots,t+2.25\,D$). Also in this case, the next phase starts right after the file movement.

Note that the *short phase* consists of D requests denoted $\mathcal{R}_1$ followed by $0.75\,D$ requests denoted $\mathcal{R}_2$, while the *long phase* consists additionally of $0.5\,D$ requests denoted $\mathcal{R}_3$. We will say that the short phase consists of *two parts*, $\mathcal{R}_1$ and $\mathcal{R}_2$, and the long phase consists of *three parts*, $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$.

2.3 DLM Analysis

We start with a lower bound on Opt. The following bound is an extension of the bound given implicitly in [8]; its proof is given in the full version of the paper.

▶ **Lemma 3.** *Let $\mathcal{R}$ be a subsequence of at most $2D$ consecutive requests from the input issued at steps $t+1, t+2, \ldots, t+|\mathcal{R}|$. Then, $4 \cdot C_{\mathrm{OPT}}(\mathcal{R}) \ge (2|\mathcal{R}|/D) \cdot [\textsc{op}_t, \mathcal{R}, \textsc{op}_{t+|\mathcal{R}|}] + (4 - 2|\mathcal{R}|/D) \cdot [\textsc{op}_t, \textsc{op}_{t+|\mathcal{R}|}]$.*

We define a potential function at (the end of) step t as $\Phi_t = 3 \cdot [\textsc{dlm}_t, \textsc{op}_t]$. It suffices to show that in any (short or long) phase consisting of steps $t+1, t+2, \ldots, t+\ell$, during which requests $\mathcal{R}$ are given, it holds that

$$C_{\mathrm{ALG}}(\mathcal{R}) + \Phi_{t+\ell} \le 4 \cdot C_{\mathrm{OPT}}(\mathcal{R}) + \Phi_t \ . \tag{1}$$

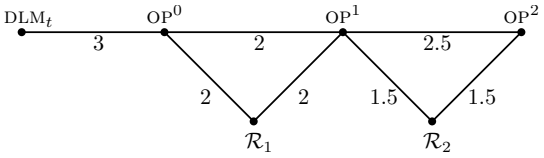
Theorem 1 follows immediately by summing the above bound over all phases of the input.

2.3.1 Proof for a short phase

We consider any short phase $\mathcal{R}$ consisting of part $\mathcal{R}_1$, spanning steps $t+1,\ldots,t+D$, and part $\mathcal{R}_2$, spanning steps $t+D+1,\ldots,t+1.75\,D$. For succinctness, we define $\textsc{op}^0 = \textsc{op}_t$, $\textsc{op}^1 = \textsc{op}_{t+D}$ and $\textsc{op}^2 = \textsc{op}_{t+1.75\,D}$. By Lemma 3 applied to $\mathcal{R}_1$ and $\mathcal{R}_2$,

$$\begin{aligned} 4 \cdot C_{\mathrm{OPT}}(\mathcal{R}) + \Phi_t \ &= \ 3 \cdot [\textsc{dlm}_t, \textsc{op}^0] + 4 \cdot C_{\mathrm{OPT}}(\mathcal{R}_1) + 4 \cdot C_{\mathrm{OPT}}(\mathcal{R}_2) \\ &\ge \ 3 \cdot [\textsc{dlm}_t, \textsc{op}^0] + 2 \cdot [\textsc{op}^0, \textsc{op}^1] + 2 \cdot [\textsc{op}^0, \mathcal{R}_1, \textsc{op}^1] \\ &\quad + 2.5 \cdot [\textsc{op}^1, \textsc{op}^2] + 1.5 \cdot [\textsc{op}^1, \mathcal{R}_2, \textsc{op}^2] \ . \end{aligned} \tag{2}$$

We treat the amount (2) as our budget. This is illustrated below; the coefficients are written as edge weights.



Budget = 4 OPT
+ initial potential
(short phase case)

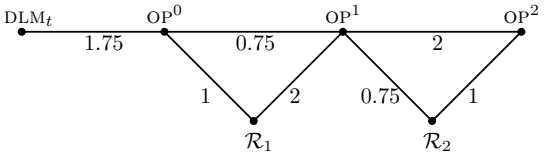
Now, we bound $C_{\text{ALG}}(\mathcal{R}) + \Phi_{t+1.75\,D}$ using the definition of ALG and the triangle inequality.

$$\begin{aligned}
&C_{\text{ALG}}(\mathcal{R}) + \Phi_{t+1.75\,D} \\
&\quad = C_{\text{ALG}}(\mathcal{R}_1) + C_{\text{ALG}}(\mathcal{R}_2) + 3[v_g, \text{OP}^2] \\
&\quad \le [\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + [\text{DLM}_t, v_g] + 3 \cdot [v_g, \text{OP}^2] \\
&\quad \le [\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + [\text{DLM}_t, v_g] + 2 \cdot [v_g, \mathcal{R}_1, \text{OP}^2] + [v_g, \mathcal{R}_2, \text{OP}^2] \\
&\quad = [\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + 2 \cdot [\text{OP}^2, \mathcal{R}_1] + [\text{OP}^2, \mathcal{R}_2] \\
&\qquad\quad + [\text{DLM}_t, v_g] + 2 \cdot [v_g, \mathcal{R}_1] + [v_g, \mathcal{R}_2] \\
&\quad = [\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + 2 \cdot [\text{OP}^2, \mathcal{R}_1] + [\text{OP}^2, \mathcal{R}_2] + g(v_g)\ . \qquad (3)
\end{aligned}$$

The first four summands of (3) can be bounded as

$$\begin{aligned}
&[\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + 2 \cdot [\text{OP}^2, \mathcal{R}_1] + [\text{OP}^2, \mathcal{R}_2] \qquad (4) \\
&\quad \le [\text{DLM}_t, \text{OP}^0, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \text{OP}^0, \text{OP}^1, \mathcal{R}_2] + 2 \cdot [\text{OP}^2, \text{OP}^1, \mathcal{R}_1] + [\text{OP}^2, \mathcal{R}_2]\ ,
\end{aligned}$$

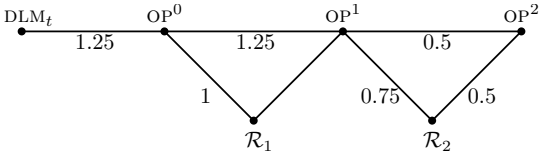
and their total weights in the final expression are depicted below.



DLM cost
+ final potential
(part 1, short phase)

For the last summand of (3), $g(v_g)$, we use the fact that v_g is a minimizer of the function g (and hence $g(v_g) \le g(\text{OP}^0)$), and the property of the short phase ($g(v_g) \le 1.5 \cdot [\text{DLM}_t, \mathcal{R}_2]$). Therefore,

$$\begin{aligned}
g(v_g) \ &\le\ 0.5 \cdot g(\text{OP}^0) + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] \\
&\le\ 0.5 \cdot [\text{DLM}_t, \text{OP}^0, \mathcal{R}_1, \text{OP}^0, \mathcal{R}_2] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] \qquad (5) \\
&\le\ 0.5 \cdot [\text{DLM}_t, \text{OP}^0, \mathcal{R}_1, \text{OP}^0, \text{OP}^1, \text{OP}^2, \mathcal{R}_2] + 0.75 \cdot [\text{DLM}_t, \text{OP}^0, \text{OP}^1, \mathcal{R}_2]\ .
\end{aligned}$$



DLM cost
+ final potential
(part 2, short phase)

By combining (3), (4) and (5) (or simply adding the edge coefficients on the last two figures) we observe that the budget ((2), i.e., the edge coefficients on the first figure) is not exceeded. This implies 4-competitiveness, i.e., that (1) holds for any short phase.

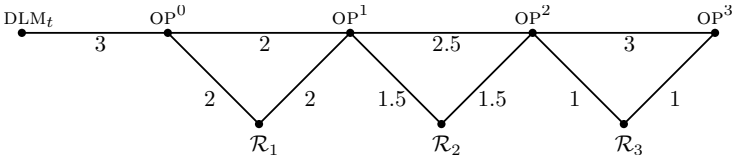
2.3.2 Proof for a long phase

We consider any long phase $\mathcal{R}$ consisting of part $\mathcal{R}_1$, spanning steps $t+1, \ldots, t+D$; part $\mathcal{R}_2$, spanning steps $t+D+1, \ldots, t+1.75 \cdot D$; and part $\mathcal{R}_3$, spanning steps $t+1.75 \cdot D+1, \ldots, t+2.25 \cdot D$. Similarly to the proof for a short phase, we define $\text{OP}^0 = \text{OP}_t$, $\text{OP}^1 = \text{OP}_{t+D}$,

$\text{OP}^2 = \text{OP}_{t+1.75\,D}$, and $\text{OP}^3 = \text{OP}_{t+2.25\,D}$. We emphasize that the positions of Opt in a long and a short phase can be completely different.

By Lemma 3, we obtain a bound very similar to that for a short phase; again, we treat it as a budget and depict its coefficients as edge weights.

$$\begin{aligned} 4 \cdot C_{\text{OPT}}(\mathcal{R}) + \Phi_t &= 3 \cdot [\text{DLM}_t, \text{OP}^0] + 4 \cdot C_{\text{OPT}}(\mathcal{R}_1) + 4 \cdot C_{\text{OPT}}(\mathcal{R}_2) + 4 \cdot C_{\text{OPT}}(\mathcal{R}_3) \\ &\geq 3 \cdot [\text{DLM}_t, \text{OP}^0] + 2 \cdot [\text{OP}^0, \text{OP}^1] + 2 \cdot [\text{OP}^0, \mathcal{R}_1, \text{OP}^1] \\ &\quad + 2.5 \cdot [\text{OP}^1, \text{OP}^2] + 1.5 \cdot [\text{OP}^1, \mathcal{R}_2, \text{OP}^2] \\ &\quad + 3 \cdot [\text{OP}^2, \text{OP}^3] + [\text{OP}^2, \mathcal{R}_3, \text{OP}^2] \ . \end{aligned} \tag{6}$$



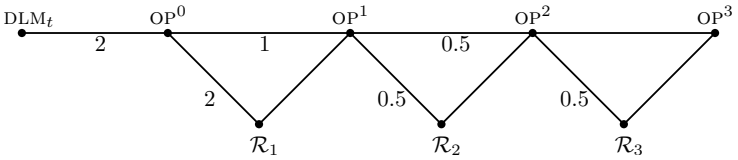
Budget = 4 OPT + initial potential (long phase case)

Now, we bound $C_{\text{ALG}}(R) + \Phi_{t+2.25\,D}$, using the definition of Alg and the triangle inequality.

$$\begin{aligned} &C_{\text{ALG}}(\mathcal{R}) + \Phi_{t+2.25\,D} \\ &\quad = C_{\text{ALG}}(\mathcal{R}_1) + C_{\text{ALG}}(\mathcal{R}_2) + C_{\text{ALG}}(\mathcal{R}_3) + 3 \cdot [v_h, \text{OP}^3] \\ &\quad = [\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + 0.5 \cdot [\text{DLM}_t, \mathcal{R}_3] + [\text{DLM}_t, v_h] + 3 \cdot [v_h, \text{OP}^3] \\ &\quad \leq [\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + 0.5 \cdot [\text{DLM}_t, \mathcal{R}_3] + [\text{DLM}_t, v_h] \\ &\qquad + [v_h, \mathcal{R}_1, \text{OP}^3] + 1.25 \cdot [v_h, \mathcal{R}_2, \text{OP}^3] + 0.75 \cdot [v_h, \mathcal{R}_3, \text{OP}^3] \\ &\quad = [\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + 0.5 \cdot [\text{DLM}_t, \mathcal{R}_3] \\ &\qquad + [\text{OP}^3, \mathcal{R}_1] + 1.25 \cdot [\text{OP}^3, \mathcal{R}_2] + 0.75 \cdot [\text{OP}^3, \mathcal{R}_3] + h(v_h) \ . \end{aligned} \tag{7}$$

Since Dlm has not migrated the file after the first two parts, $g(v) \geq 1.5 \cdot [\text{DLM}_t, \mathcal{R}_2]$ for any node v. Therefore $0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] \leq 0.5 \cdot g(\text{OP}^0) = 0.5 \cdot [\text{DLM}_t, \text{OP}^0, \mathcal{R}_1, \text{OP}^0, \mathcal{R}_2] \leq 0.5 \cdot [\text{DLM}_t, \text{OP}^0, \mathcal{R}_1, \text{OP}^0, \text{OP}^1, \mathcal{R}_2]$. Using this and the triangle inequality, the first three summands of (7) can be bounded and depicted as follows:

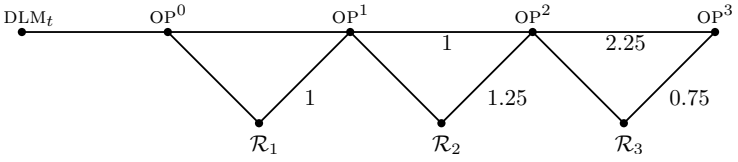
$$\begin{aligned} &[\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + 0.5 \cdot [\text{DLM}_t, \mathcal{R}_3] \\ &\quad \leq [\text{DLM}_t, \text{OP}^0, \mathcal{R}_1] + 0.5 \cdot [\text{DLM}_t, \text{OP}^0, \mathcal{R}_1, \text{OP}^0, \text{OP}^1, \mathcal{R}_2] \\ &\qquad + 0.5 \cdot [\text{DLM}_t, \text{OP}^0, \text{OP}^1, \text{OP}^2, \mathcal{R}_3] \ . \end{aligned} \tag{8}$$



DLM cost + final potential (part1, long phase)

The next three summands of (7) can be also bounded appropriately:

$$\begin{aligned} &[\text{OP}^3, \mathcal{R}_1] + 1.25 \cdot [\text{OP}^3, \mathcal{R}_2] + 0.75 \cdot [\text{OP}^3, \mathcal{R}_3] \\ &\quad \leq [\text{OP}^3, \text{OP}^2, \text{OP}^1, \mathcal{R}_1] + 1.25 \cdot [\text{OP}^3, \text{OP}^2, \mathcal{R}_2] + 0.75 \cdot [\text{OP}^3, \mathcal{R}_3] \ . \end{aligned} \tag{9}$$

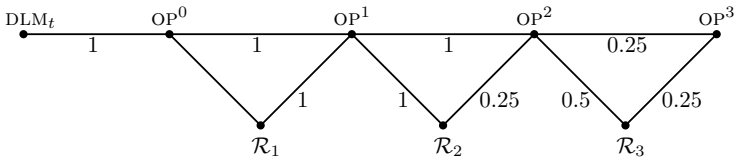


DLM cost + final potential (part 2, long phase)

Lastly, for bounding $h(v_h)$, we use the fact that v_h is a minimizer of h, and hence

$$\begin{aligned} h(v_h) &\leq h(\text{OP}^1) \\ &= [\text{OP}^1, \text{DLM}_t] + [\text{OP}^1, \mathcal{R}_1] + 1.25 \cdot [\text{OP}^1, \mathcal{R}_2] + 0.75 \cdot [\text{OP}^1, \mathcal{R}_3] \\ &\leq [\text{OP}^1, \text{OP}^0, \text{DLM}_t] + [\text{OP}^1, \mathcal{R}_1] + [\text{OP}^1, \mathcal{R}^2] + 0.25 \cdot [\text{OP}^1, \text{OP}^2, \mathcal{R}_2] \\ &\qquad + 0.5 \cdot [\text{OP}^1, \text{OP}^2, \mathcal{R}_3] + 0.25 \cdot [\text{OP}^1, \text{OP}^2, \text{OP}^3, \mathcal{R}_3] \ . \end{aligned} \tag{10}$$

Note that in (10) we split some of the paths and choose the longer ones, so that the budgets on edges are not violated. Bound (10) is depicted on the figure below.



DLM cost
+ final potential
(part 3, long phase)

By combining (7), (8), (9) and (10) (or simply adding edge coefficients on the last three figures), we observe that the budget ((6), i.e., the edge coefficients on the first figure) is not exceeded. This implies 4-competitiveness, i.e., that (1) holds for any long phase and concludes the proof of Theorem 1.

3 Lower Bound for Phase-Based Algorithms

A *fixed-phase algorithm* chooses phase length $c \cdot D$ and after every $c \cdot D$ requests it makes a migration decision solely on the basis of its current position and the last $c \cdot D$ requests. We now proceed to argue that no fixed-phase algorithm ALG can beat the competitive ratio $R_0 \approx 4.086$ achieved by MTLM [8] (cf. Theorem 2). As already stated in the introduction, to ensure that the algorithm cannot base its choices on the previous phases, we will give the adversary an additional power: it may modify the graph between the phases of ALG, as long as the distance between nodes keeping the files of ALG and OPT (v_{ALG} and v_{OPT}, respectively) is preserved. We emphasize that the analysis of MTLM [8] essentially uses this model: each phase is analyzed completely separately from others. The full proof is given in the full version of the paper; here we informally highlight its key ideas.

The adversarial construction consists of many epochs, each consisting of some number of phases. At the beginning and at the end of an epoch, ALG and OPT keep their files at the same node. We define three adversarial strategies, called *plays*: *linear*, *bipartite*, and *finishing*. Each play consists of one or more phases. A prerequisite for each given play is a particular distance between v_{ALG} and v_{OPT}. Each play will have some properties: it will incur some cost on ALG and OPT and will end with v_{ALG} and v_{OPT} in a specific distance.

In the first phase of an epoch, when initially $v_{\text{ALG}} = v_{\text{OPT}}$, the adversary uses the *linear* play (the generated graph is a single edge of length 1), so that at the end of the phase, $d(v_{\text{ALG}}, v_{\text{OPT}}) = 1$. For such phase P, we have $C_{\text{ALG}}(P) \geq R_0 \cdot C_{\text{OPT}}(P) - (1/(1-2\alpha)) \cdot D$, where $\alpha = 1/(R_0 - 1)$. Note that in this phase alone, the adversary does not enforce the desired competitive ratio of R_0, but it increases the distance between v_{OPT} and v_{ALG}.

In each of the next L phases, the adversary employs the *bipartite* play; the graph used corresponds to a tight bipartite example for MTLM, cf. Figure 1). Let f be the value of $d(v_{\text{ALG}}, v_{\text{OPT}})$ at the beginning of a phase. If the algorithm plays well, then at the end of the phase this distance decreases to $2\alpha \cdot f$. Furthermore, neglecting lower order terms, for such phase P, it holds that $C_{\text{ALG}}(P) \geq R_0 \cdot C_{\text{OPT}}(P) + f \cdot D$, i.e., the inequality $C_{\text{ALG}}(P) \geq R_0 \cdot C_{\text{OPT}}(P)$ holds with the slack $f \cdot D$. The sum of these slacks over L phases

is $\sum_{i=0}^{L-1}(2\alpha)^i \cdot D$, which tends to $(1/(1-2\alpha)) \cdot D$ when L grows. Hence, after one linear and L bipartite phases (for a large L and neglecting lower order terms again), the cost paid by Alg is at least R_0 times the cost paid by Opt and the distance between their files is negligible.

Finally, to decrease the distance between v_{ALG} and v_{OPT} to zero, the adversary uses a third type of play, the *finishing* one. This play incurs a negligible cost and it forces the positions of Alg and Opt files to coincide, which ends an epoch.

4 Linear Program for File Migration

In this section, we present a linear programming model for the analysis of both algorithm Mtlm by Bartal et al. [8] and our algorithm Dlm.

4.1 LP analysis of MTLM-like algorithms

In our approach, we analyze any Mtlm-like algorithm Alg. We use our notion of distances from Section 2.1. Alg will be a variant of Mtlm parameterized with two values β and δ. The length of its phase is $\delta \cdot D$ and the initial point of Alg is denoted by A_0. We denote the set of requests within a phase by $\mathcal{R}$. At the end of a phase, Alg migrates the file to a point A_1 that minimizes the function

$$f(x) = [A_0, x] + \beta \cdot [x, \mathcal{R}] \ .$$

As in the amortized analysis of the algorithm Mtlm [8], we will use a potential function equal to ϕ times the distance between the files of Alg and Opt, where ϕ is a parameter used in the analysis. We let O_0 and O_1 denote the initial and final position of Opt during the studied phase, respectively. Then, the amortized cost of Alg in a single phase is $C_{\text{ALG}} = \delta \cdot [A_0, \mathcal{R}] + [A_0, A_1] + \phi([A_1, O_1] - [A_0, O_0])$.

The following factor-revealing LP mimics the proof given in [8]. Namely, it encodes inequalities that are true for any phase and a graph on which Alg can be run. Its goal is to maximize the ratio between C_{ALG} and C_{OPT}: as an instance can be scaled, we set $C_{\text{OPT}} = 1$ and we maximize C_{ALG}. Let $V = \{A_0, A_1, O_0, O_1\}$ and $V' = V \cup \{\mathcal{R}\}$.

maximize C_{ALG}

subject to:

$$C_{\text{ALG}} = \delta \cdot [A_0, \mathcal{R}] + [A_0, A_1] + \phi \cdot ([A_1, O_1] - [A_0, O_0])$$
$$C_{\text{OPT}} = 1$$
$$C_{\text{OPT}} = C^{\text{req}}_{\text{OPT}} + C^{\text{move}}_{\text{OPT}}$$
$$C^{\text{move}}_{\text{OPT}} \geq [O_0, O_1]$$
$$2 \cdot C^{\text{req}}_{\text{OPT}} + \delta \cdot C^{\text{move}}_{\text{OPT}} \geq \delta \cdot [O_0, \mathcal{R}] + \delta \cdot [O_1, \mathcal{R}]$$
$$f(A_1) \leq f(v) \qquad \text{for all } v \in V$$
$$0 \leq [v_1, v_3] \leq [v_1, v_2] + [v_2, v_3] \qquad \text{for all } v_1, v_2, v_3 \in V'$$

As $\mathcal{R}$ is a set of requests, it does not necessarily correspond to a single point in the studied metric. Nevertheless, our notion of average distances (i.e., $[v_1, v_2]$) allows us to write the triangle inequalities for any pair of objects from set $V \cup \{\mathcal{R}\}$.

In the LP above, $C^{\text{req}}_{\text{OPT}}$, $C^{\text{move}}_{\text{OPT}}$ denote the cost of Opt for serving the request and the cost of Opt for migrating the file, respectively. The inequality $2 \cdot C^{\text{req}}_{\text{OPT}} + \delta \cdot C^{\text{move}}_{\text{OPT}} \geq \delta \cdot [O_0, \mathcal{R}] + \delta \cdot [O_1, \mathcal{R}]$ is a counterpart of the relation guaranteed by Lemma 3.

For any choice of parameters β, δ, and ϕ, the LP above finds an instance that maximizes the competitive ratio of Alg. Note that such instance is not necessarily a certificate that Alg indeed performs poorly: in particular, inequalities that lower-bound the cost of Opt might not be tight. However, the opposite is true: if the value of C_{ALG} returned by the LP is ξ, then for any possible instance the ratio is at most ξ.

Let $c_0 = 1.841$ be the phase length of Mtlm. Setting $\delta = c_0$ and $\beta = \phi = 1 + c_0$ yields that the optimal value of the LP is $R_0 \approx 4.086$, which can be interpreted as a numerical counterpart of the original analysis in [8]. To obtain a formal mathematical proof, one may take a dual solution to the LP. It gives the coefficients that multiplied by the corresponding LP inequalities and summed over all inequalities yield a proof that the amortized cost of Mtlm in any phase is at most R_0 times the cost of Opt. Summed over all phases, this implies that Mtlm is R_0-competitive.

Among other advantages, this approach allows us to numerically find the instances that are tight for the current analysis (cf. Section 2 and Figure 1): linear and bipartite instances can be obtained this way.

4.2 LP analysis of DLM-like algorithms

Now we show how to adapt the LP from the previous section to analyze Dlm-type algorithms. Recall that after $1.75\,D$ requests, Dlm evaluates the geometry of the so-far-received requests and decides whether to continue this phase or not. Although the final parameters of Dlm are elegant numbers (multiplicities of 1/4), they were obtained by a tedious optimization process using the LP we present below. Furthermore, the LP below does not give us an explicit rule for continuing the phase; it only tells that Dlm is successful either in a short or in a long phase.

Recall that in a phase, Dlm considers three groups of consecutive $\delta_i \cdot D$ requests: $\mathcal{R}_1$, $\mathcal{R}_2$, and $\mathcal{R}_3$, where δ_1, δ_2, and δ_3 are parameters of Dlm. First, assume that Dlm always processes three parts and afterwards it moves the file to a point A_3 that minimizes the function

$$h(x) = [A_0, x] + \beta_1 \cdot [x, \mathcal{R}_1] + \beta_2 \cdot [x, \mathcal{R}_2] + \beta_3 \cdot [x, \mathcal{R}_3] \ ,$$

where β_i are parameters that we choose later. We denote the strategy of an optimal algorithm by OptL (short for Opt-Long). Let O_0^L, O_1^L, O_2^L and O_3^L denote the trajectory of OptL (O_0^L is the initial position of the OptL's file at the beginning of the phase, and O_i^L is its position right after the i-th part of the phase). Analogously to the previous section, we obtain the following LP.

maximize C_{ALGL}

subject to:

$$C_{\text{ALGL}} = [A_0, A_3] + \textstyle\sum_{i=1,2,3} \delta_i \cdot [A_0, \mathcal{R}_i] + \phi \cdot ([A_3, O_3^L] - [A_0, O_0^L])$$

$$C_{\text{OPTL}} = 1$$

$$C_{\text{OPTL}} = \textstyle\sum_{i=1,2,3} (C^{\text{req}}_{\text{OPTL}}(i) + C^{\text{move}}_{\text{OPTL}}(i))$$

$$C^{\text{move}}_{\text{OPTL}}(i) \geq [O_{i-1}^L, O_i^L] \qquad \text{for } i = 1, 2, 3$$

$$2 \cdot C^{\text{req}}_{\text{OPTL}}(i) + \delta \cdot C^{\text{move}}_{\text{OPTL}}(i) \geq \delta_i \cdot [O_{i-1}^L, \mathcal{R}_i] + \delta_i \cdot [O_i^L, \mathcal{R}_i] \qquad \text{for } i = 1, 2, 3$$

$$h(A_3) \leq h(v) \qquad \text{for all } v \in V$$

$$0 \leq [v_1, v_3] \leq [v_1, v_2] + [v_2, v_3] \qquad \text{for all } v_1, v_2, v_3 \in V'$$

This time $V = \{A_0, A_3, O_0^L, O_1^L, O_2^L, O_3^L\}$ and $V' = V \cup \{\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3\}$.

We note that such parameterization alone does not improve the competitive ratio, i.e., for any choice of parameters δ_i and β_i, the objective value of the LP above is at least $R_0 \approx 4.086$.

However, as stated in Section 2.1, DLM verifies if after two parts it can migrate its file to a node A_2 being the minimizer of the function

$$g(x) = [A_0, x] + \beta'_1 \cdot [x, \mathcal{R}_1] + \beta'_2 \cdot [x, \mathcal{R}_2] \ ,$$

where β'_i are parameters that we choose later.

In our analysis, we gave an explicit rule whether the migration to A_2 should take place. However, for our LP-based approach, we follow a slightly different scheme. Namely, if the migration to A_2 guarantees that the amortized cost in the short phase (the first two parts) is at most 4 times the cost of *any* strategy for the short phase, then DLM may move to A_2 and we immediately achieve competitive ratio 4 on the short phase. Otherwise, we may add additional constraints to the LP, stating that the competitive ratio of an algorithm which moves to A_2 is at least 4 (against any chosen strategy OPTS). Analogously to OPTL, the trajectory of OPTS is described by three points: O_0^S, O_1^S, and O_2^S. This allows us to strengthen our LP by adding the following inequalities:

$$\begin{aligned}
&C_{\text{ALGS}} = [A_0, A_2] + \textstyle\sum_{i=1,2} \delta_i \cdot [A_0, \mathcal{R}_i] + \phi \cdot ([A_2, O_2^S] - [A_0, O_0^S]) \\
&C_{\text{OPTS}} = \textstyle\sum_{i=1,2} (C^{\text{req}}_{\text{OPTS}}(i) + C^{\text{move}}_{\text{OPTS}}(i)) \\
&C^{\text{move}}_{\text{OPTS}}(i) \geq [O_{i-1}^S, O_i^S] && \text{for } i = 1, 2 \\
&2 \cdot C^{\text{req}}_{\text{OPTS}}(i) + \delta \cdot C^{\text{move}}_{\text{OPTS}}(i) \geq \delta_i \cdot [O_{i-1}^S, \mathcal{R}_i] + \delta_i \cdot [O_i^S, \mathcal{R}_i] && \text{for } i = 1, 2 \\
&g(A_2) \leq g(v) && \text{for all } v \in V \\
&C_{\text{ALGS}} \geq 4 \cdot C_{\text{OPTS}}
\end{aligned}$$

We also change V to $\{A_0, A_3, O_0^L, O_1^L, O_2^L, O_3^L, O_0^S, O_1^S, O_2^S\}$, both in new and in old inequalities.

When we choose $\phi = 3$, fix phase length parameters to be $\delta_1 = 1$, $\delta_2 = 0.75$, $\delta_3 = 0.5$ and parameters for functions g and h to be $\beta'_1 = 2$, $\beta'_2 = 1$, $\beta_1 = 1, \beta_2 = 0.25$ and $\beta_3 = 0.75$, we obtain that the value of the above LP is 4. Again, this can be interpreted as a numerical argument that DLM is indeed a 4-competitive algorithm.

5 Conclusions

While in the last decade factor-revealing LPs became a standard tool for analysis of approximation algorithms, their application to online algorithms so far have been limited to online bipartite matching and its variants (see, e.g., [24, 20]) and for showing lower bounds [5]. In this paper, we successfully used the factor-revealing LP to bound the competitive ratio of an algorithm for an online problem defined on an arbitrary metric space. We believe that similar approaches could yield improvements also for other online graph problems.

References

1 Susanne Albers and Hisashi Koga. Page migration with limited local memory capacity. In *Proc. 4th Int. Workshop on Algorithms and Data Structures (WADS)*, pages 147–158, 1995.

2 Baruch Awerbuch, Yair Bartal, and Amos Fiat. Competitive distributed file allocation. In *Proc. 25th ACM Symp. on Theory of Computing (STOC)*, pages 164–173, 1993.

3 Baruch Awerbuch, Yair Bartal, and Amos Fiat. Heat & Dump: Competitive distributed paging. In *Proc. 34th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 22–31, 1993.

4 Baruch Awerbuch, Yair Bartal, and Amos Fiat. Distributed paging for general networks. *Journal of Algorithms*, 28(1):67–104, 1998.

5 Yossi Azar, Ilan Reuven Cohen, and Alan Roytman. Online lower bounds via duality. In *Proc. 28th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1038–1050, 2017.

6 Yair Bartal. *Competitive Analysis of Distributed On-line Problems – Distributed Paging*. PhD thesis, Tel-Aviv University, 1995.

7 Yair Bartal. Distributed paging. In *Dagstul Workshop on On-line Algorithms*, pages 97–117, 1996.

8 Yair Bartal, Moses Charikar, and Piotr 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 Yair Bartal, Amos Fiat, and Yuval Rabani. Competitive algorithms for distributed data management. *Journal of Computer and System Sciences*, 51(3):341–358, 1995.

10 Shai Ben-David, Allan Borodin, Richard M. Karp, Gabor Tardos, and Avi Wigderson. On the power of randomization in online algorithms. *Algorithmica*, 11(1):2–14, 1994.

11 Marcin Bienkowski. Migrating and replicating data in networks. *Computer Science – Research and Development*, 27(3):169–179, 2012.

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

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

14 Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

15 Marek Chrobak, Lawrence L. Larmore, Nick Reingold, and Jeffery Westbrook. Page migration algorithms using work functions. *Journal of Algorithms*, 24(1):124–157, 1997.

16 Bezalel Gavish and Olivia R. Liu Sheng. Dynamic file migration in distributed computer systems. *Communications of the ACM*, 33(2):177–189, 1990.

17 Makoto Imase and Bernard M. Waxman. Dynamic Steiner tree problem. *SIAM Journal on Discrete Mathematics*, 4(3):369–384, 1991.

18 Carsten Lund, Nick Reingold, Jeffery Westbrook, and Dicky C. K. Yan. Competitive on-line algorithms for distributed data management. *SIAM Journal on Computing*, 28(3):1086–1111, 1999.

19 Bruce M. Maggs, Friedhelm Meyer auf der Heide, Berthold Vöcking, and Matthias Westermann. Exploiting locality for data management in systems of limited bandwidth. In *Proc. 38th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 284–293, 1997.

20 Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing LPs. In *Proc. 43rd ACM Symp. on Theory of Computing (STOC)*, pages 597–606, 2011.

21 Akira Matsubayashi. Uniform page migration on general networks. *International Journal of Pure and Applied Mathematics*, 42(2):161–168, 2008.

22 Akira Matsubayashi. A 3+Omega(1) lower bound for page migration. In *Proc. 3rd Int. Symp. on Computing and Networking (CANDAR)*, pages 314–320, 2015.

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

24 Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. Adwords and generalized online matching. *Journal of the ACM*, 54(5), 2007.

25 Friedhelm Meyer auf der Heide, Berthold Vöcking, and Matthias Westermann. Provably good and practical strategies for non-uniform data management in networks. In *Proc. 7th European Symp. on Algorithms (ESA)*, pages 89–100, 1999.

26 Jeffery Westbrook. Randomized algorithms for the multiprocessor page migration. *SIAM Journal on Computing*, 23:951–965, 1994.