Online Page Migration with ML Advice

Piotr Indyk MIT
Frederik Mallmann-Trenn King's College London
Slobodan Mitrović UC Davis
Ronitt Rubinfeld MIT

Abstract

We consider online algorithms for the *page migration problem* that use predictions, potentially imperfect, to improve their performance. The best known online algorithms for this problem, due to Westbrook'94 and Bienkowski et al'17, have competitive ratios strictly bounded away from 1. In contrast, we show that if the algorithm is given a prediction of the input sequence, then it can achieve a competitive ratio that tends to 1 as the prediction error rate tends to 0. Specifically, the competitive ratio is equal to $1+O(q)$, where q is the prediction error rate. We also design a "fallback option" that ensures that the competitive ratio of the algorithm for *any* input sequence is at most $O(1/q)$. Our result adds to the recent body of work that uses machine learning to improve the performance of "classic" algorithms.

1 Introduction

Recently, there has been a lot of interest in using machine learning to design improved algorithms for various computational problems. This includes work on data structures Kraska et al. (2018); Mitzenmacher (2018), online algorithms Lykouris and Vassilvitskii (2018); Purohit et al. (2018); Gollapudi and Panigrahi (2019a); Rohatgi (2020), combinatorial optimization Khalil et al. (2017); Balcan et al. (2018), similarity search Wang et al. (2016), compressive sensing Mousavi et al. (2015); Bora et al. (2017) and streaming algorithms Hsu et al. (2019). This body of work is motivated by the fact that modern machine learning methods are capable of discovering subtle structure in collections of input data, which can be utilized to improve the performance of algorithms that operate on similar data.

In this paper we focus on learning-augmented *online algorithms*. An on-line algorithm makes non-revocable decisions based only on the part of the input seen so far, without *any* knowledge of the future. It is thus natural to consider a relaxation of the model where the algorithm has access to (imperfect) predictors of the future input that could be used to improve the algorithm performance. Over the last couple of years this line of research has attracted growing attention in the machine learning and algorithms literature, for classical on-line problems such as caching Lykouris and Vassilvitskii (2018); Rohatgi (2020), ski-rental and scheduling Purohit et al. (2018); Gollapudi and Panigrahi (2019b); Lattanzi et al. (2020) and graph matching Kumar et al. (2019). Interestingly, most of the aforementioned works conclude that the "optimistic" strategy of simply following the predictions, i.e., executing the optimal solution computed off-line for the predicted input, can lead to a highly sub-optimal performance even if the prediction error is small [1]. For instance, for the caching problem, even a single misprediction can lead to an unbounded competitive ratio Lykouris and Vassilvitskii (2018).

In this paper we show that, perhaps surprisingly, the aforementioned "optimistic" strategy leads to near-optimal performance for some well-studied on-line problems. We focus on the problem of *page migration* Black and Sleator (1989) (a.k.a. *file migration* Bienkowski (2012) or *1-server with excursions* Manasse et al. (1990)). Here, the algorithm is given a sequence s of points (called *requests*) $s_1, s_2, \ldots$ from a metric space (X, d), in an online fashion. The state of the algorithm is also a point from (X, d). Given the next request s_i, the algorithm moves to its next state a_i (at the cost of $D \cdot d(a_{i-1}, a_i)$, where D is a parameter), and then "satisfies" the request s_i (at the cost of $d(a_i, s_i)$). The objective is to satisfy all requests while

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

[1] To the best of our knowledge the only problem for which this strategy is known to result in an optimal algorithm is the online bipartite matching, see Section 1.2 for more details

minimizing the total cost. The problem has been a focus on a large body of research, see e.g., Awerbuch et al. (1993); Westbrook (1994); Chrobak et al. (1997); Bartal et al. (1997); Khorramian and Matsubayashi (2016); Bienkowski et al. (2017). The best known algorithms for this problem have competitive ratios of 4 (a deterministic algorithm due to Bienkowski et al. (2017)), 3 (a randomized algorithm against adaptive adversaries due to Westbrook (1994)) and $2.618\ldots$ (a randomized algorithm against oblivious adversaries due to Westbrook (1994)). The original paper Black and Sleator (1989) also showed that the competitive ratio of any deterministic algorithm must be at least 3, which was recently improved to $3+\epsilon$ for some $\epsilon > 0$ by Matsubayashi (2015).

1.1 Our results

Suppose that we are given a *predicted* request sequence $\hat{s}$ that, in each interval of length ϵD, differs from the actual sequence s on at most a fraction q of positions, where $\epsilon, q \in (0,1)$ are the parameters (note that the lower the values of ϵ and q are, the stronger our assumption is). Under this assumption we show that the optimal off-line solution for $\hat{s}$ is a $(1+\epsilon)(1+O(q))$-competitive solution for s as long as the parameter $q > 0$ is a small enough constant. Thus, the competitive ratio of this prediction-based algorithm improves over the state of the art even if the number of errors is linear in the sequence length, and tends to 1 when the error rate tends to 0.[2] Furthermore, to make the algorithm robust, we also design a "fallback option", which is triggered if the input sequence violates the aforementioned assumption (i.e., if the fraction of errors in the suffix of the current input sequence exceeds q). The fallback option ensures that the competitive ratio of the algorithm for *any* input sequence is at most $O(1/q)$. Thus, our final algorithm produces a near-optimal solution if the prediction error is small, while guaranteeing a constant competitive ratio otherwise.

For the case when the underlying metric is *uniform*, i.e., all distances between distinct points are equal to 1, we further improve the competitive ratio to $1+O(q)$ under the assumption that each interval of length D differs from the actual sequence in at most qD positions. That is, the parameter ϵ is not needed in this case. Moreover, any algorithm has a competitive ratio of at least $1+\Omega(q)$.

It is natural to wonder whether the same guarantees hold even when the predicted sequence differs from the actual sequence on at most a fraction of q positions distributed *arbitrarily* over $\hat{s}$, as opposed to over chunks of length εD. We construct a simple example that shows that such a relaxed assumption results in the same lower bound as for the classical problem.

[2] Note that if each interval of length D has at most a fraction of q of errors, then it is also the case that each interval of length $\sqrt{q}D$ has at most a fraction of $\sqrt{q}$ of errors. Thus, if q tends to 0, the competitive ratio tends to 1 even if the interval length remains fixed.

1.2 Related Work

Multiple variations of the page migration problem have been studied over the years. For example, if the page can be *copied* as well as moved, the problem has been studied under the name of *file allocation*, see e.g., Bartal et al. (1995); Awerbuch et al. (2003); Lund et al. (1998). Other formulations add constraints on nodes capacities, allow dynamically changing networks etc. See the survey Bienkowski (2012) for an overview.

There is a large body of work concerning on-line algorithms working under stochastic or probabilistic assumptions about the input Unc (2016). In contrast, in this paper we do not make such assumptions, and allow *worst case* prediction errors (similarly to Lykouris and Vassilvitskii (2018); Kumar et al. (2019); Purohit et al. (2018)). Among these works, our prediction error model (bounding the fraction of mispredicted requests) is most similar to the "agnostic" model defined in Kumar et al. (2019). The latter paper considers online matching in bipartite graphs, where a prediction of the graph is given in advance, but the final input graph can deviate from the prediction on d vertices. Since each vertex impacts at most one matching edge, it directly follows that d errors reduce the matching size by at most d. In contrast, in our case a single error can affect the cost of the optimum solution by an arbitrary amount. Thus, our analysis requires a more detailed understanding of the properties of the optimal solution.

Multiple papers studied on-line algorithms that are given a small number of bits of *advice* Boyar et al. (2017) and show that, in many scenarios, this can improve their competitive ratios. Those algorithms, however, typically assume that the advice is error-free.

2 Preliminaries

2.1 Page Migration

In the classical version, the algorithm is given a sequence s of points (called *requests*) $s = (s_i)_{i\in[n]}$ from a metric space (X,d), in an online fashion. The state of the algorithm (i.e., the *page*), is also a point from (X,d). Given the next request s_i, the algorithm moves to its next state a_i (at the cost of $D\cdot d(a_{i-1},a_i)$, where $D>1$ is a parameter), and then "satisfies" the request s_i (at the cost of $d(a_i,s_i)$). The objective is to satisfy

all requests while minimizing the total cost. We can consider a version of this problem where the algorithm is given, prior to the arrival of the requests, a *predicted sequence* $\hat{s} = (\hat{s}_i)_{i\in[n]}$. The (final) sequence s is generated adversarially from $\hat{s}$ and an arbitrary *adversarial sequence* $s^\star = (s^*_i)_{i\in[n]}$. That is either $s_i = \hat{s}_i$ or $s_i = s^*_i$. The initial input to the (online) algorithm is $\hat{s}$. In addition, at every step i, the algorithm receives the request s_i for which it outputs in the same step the location from where it serves the request. If we do not make any assumptions on how well s is predicted by $\hat{s}$, then the problem is no easier than the classical online version. On the other hand, if $s = \hat{s}$, then one obtains an optimal online algorithm, by simply computing the optimal offline algorithm. The interesting regime lies in between these two cases. We will make the following assumption throughout the paper, which roughly speaking demands that a $1-q$ fraction of the input is correctly predicted and that the q fraction of errors is somewhat spread out.

Definition 1 (Number of mismatches $m(\cdot)$)**.** *Let $\mathcal{I}$ be an interval of indices. We define $m(\mathcal{I}) \stackrel{\text{def}}{=} \sum_{t\in\mathcal{I}} 1_{s_t\neq\hat{s}_t}$ to be the number of mismatches between s and $\hat{s}$ within the interval $\mathcal{I}$.*

Assumption 1. *Consider an interval $\mathcal{I}$ of s of length εD. It holds that for any $\mathcal{I}$ we have $m(\mathcal{I}) \leq q\varepsilon D$.*

Remark 1. *Relaxing Assumption 1 by allowing the adversary to change an arbitrary q fraction of the input results in the same lower bound as for the classical problem. To see this, consider an arbitrary instance on qn elements that gives a lower bound of c in the classical problem. Call this sequence of elements* adversarial*. Let $\hat{s}$ consists of n elements being equal to the starting point. That is, $\hat{s}$ is simply the starting position replicated n times. Let s be equal to the sequence $\hat{s}$ whose suffix of length qn is replaced by the adversarial sequence. Now, on s defined in this way no algorithm can be better than c-competitive. Hence, in general this relaxation of Assumption 1 gives no advantage.*

Our main results hold for *general metric space*, where for all $p, p', p'' \in X$ all of the following hold: $d(p,p) = 0$, $d(p,p') > 0$ for $p \neq p'$, $d(p,p') = d(p',p)$, and $d(p,p'') \leq d(p,p') + d(p',p'')$. We obtain better results for *uniform metric space*, where, $d(p,p') = 1$ for $p \neq p'$.

2.2 Notation

Given a sequence s, we use s_i to denote the i-th element of s. For integers i and j, such that $1 \leq i \leq j$, we use $s_{[i,j]}$ to denote the subsequence of s consisting of the elements $s_i, \ldots, s_j$.

Let p_0 denote the start position for all algorithms, i.e., the position of the page at time 0.

Given an algorithm B that pays cost C for serving n requests, we denote by C_{t_1,t_2} the cost paid by B during the interval $[t_1, t_2]$. We sometimes abuse notation and write C_t as a shorthand for $C_{0,t}$. In particular, C denotes $C_{0,n}$ as well as C_n. This notation is the most often used in the context of our algorithm ALG and the optimal solution OPT, whose total serving costs are A and O, respectively.

3 Proof Overview

We now present our main algorithm and provide intuition about its correctness.

Our two main contributions are: algorithm ALG that is $(1+O(q))$-competitive provided Assumption 1; and, a black-box reduction from ALG to a $O(1/q)$-competitive algorithm $\text{ALG}^{\text{ROBUST}}$ when Assumption 1 does not hold. In Section 3.1 we present an overview of ALG, while an overview of $\text{ALG}^{\text{ROBUST}}$ is given in Section 3.2.

3.1 ALG under assumption Assumption 1

Algorithm ALG (given as Algorithm 1) simply computes the optimal *offline* solution based on $\hat{s}$ and moves pages accordingly.

Algorithm 1 ALG$(s_i, \hat{s})$

Input:

- Request s_i
- A prediction $\hat{s}$ (s and $\hat{s}$ are defined in Section 2)

1: Let a_i be the position of the page in the optimal offline algorithm at the i-th request with respect to $\hat{s}$.
2: Move the page to a_i and serve the request s_i.

The main challenge in proving that ALG still performs well in the online setting lies in leveraging the optimality of ALG with respect to the offline sequence. The reason for this is that, due to s and $\hat{s}$ not being identical, OPT and ALG may be on different page locations throughout all the requests. In addition to that, we have no control over which q fraction of any interval of length D is changed nor to what it is changed. In particular, if $s_i \neq \hat{s}_i$, then s_i and $\hat{s}_i$ could be very far from each other. To circumvent this, we use the following way to argue about the offline optimality, that is, about the optimality computed with respect to $\hat{s}$.

We think of ALG (OPT, respectively) as a sequence of page locations that are defined with respect to $\hat{s}$ (s,

respectively). These page locations do not change even if, for instance, the i-th online request to ALG deviates from $\hat{s}_i$. Let A_t (O_t, respectively) be the cost of ALG (OPT, respectively) serving t requests given by $s_{[1,t]}$. Similarly, let $\hat{A}_t$ ($\hat{O}_t$, respectively) be the cost of ALG (OPT, respectively) for serving the oracle subsequence $\hat{s}_{[1,t]}$. In particular, A_n is the cost of ALG (optimal on $\hat{s}$) on the final sequence s, whereas $\hat{O}_n$ is the cost of the optimal algorithm for s on the predicted sequence $\hat{s}$. It is convenient to think of $\hat{O}_n$ as the 'evil twin' of A_n.

We have, due to optimality of ALG on the offline sequence,

$$\begin{aligned} A_n - O_n &= A_n - \hat{A}_n + \hat{A}_n - O_n \\ &\leq A_n - \hat{A}_n + \hat{O}_n - O_n. \end{aligned} \tag{1}$$

The intuition behind the right-hand side of inequality above is best explained pictorially, which we do in Fig. 1. Here ALG is at a and OPT is at o. In the depicted example the actual request is s while the predicted one is $\hat{s}$. This causes $A_n - \hat{A}_n$ to increase, however, at the same time, $\hat{O}_n - O_n$ decreases by almost the same amount.[3] In fact, one can show that for such a request the right hand side of Eq. (1) will increase by no more than $2d(a, o)$. For requests that are predicted correctly, i.e., $s = \hat{s}$, the costs of ALG and OPT do not change. It remains to bound $d(a_t, o_t)$, which we do next. By triangle inequality, it holds that

$$d(a_t, o_t) \leq d(a_t, s_t) + d(o_t, s_t) \tag{2}$$

$$\leq A_t - A_{t-1} + O_t - O_{t-1}, \tag{3}$$

Consider an interval $(t_{i-1}, t_i]$. Let $c_{move}^{(t_{i-1},t_i]}$ be the total sum of moving costs for both OPT and ALG for the requests in the interval $(t_{i-1}, t_i]$. As a reminder (see Definition 1), for a given interval $\mathcal{I}$, $m(\mathcal{I})$ is the number of mismatches between s and $\hat{s}$ within $\mathcal{I}$. From Eq. (3), we derive

$$A_n - O_n \leq 2 \sum_i m((t_{i-1}, t_i]) \cdot \tag{4}$$

$$\frac{A_{t_i} - A_{t_{i-1}} + O_{t_i} - O_{t_{i-1}} - c_{move}^{(t_{i-1},t_i]}}{t_i - t_{i-1}}. \tag{5}$$

We would like the right hand side of Eq. (5) to be small, implying that $A_n - O_n$ is small as well. To understand the nature of the right hand side of Eq. (5) and what is required for it to be small, assume for a moment that $m((t_{i-1}, t_i]) = \alpha(t_i - t_{i-1})$. Then, the rest of the summation telescopes to $A_n - O_n$, and

[3] We oversimplified here, since the right hand side of (1) only holds for the sum of all points, but a similar argument can be made for a single requests.

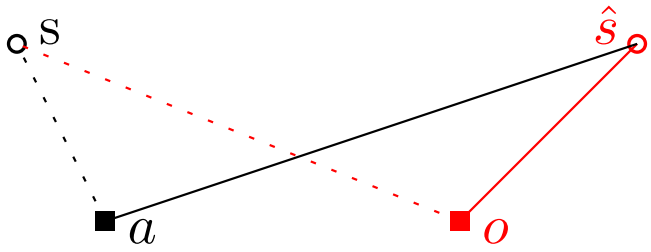


Figure 1: A pictorial representation of Eq. (1).

Eq. (5) reduces to $A_n - O_n \leq 2\alpha(A_n + O_n)$. Now, if α is sufficiently small, e.g., $\alpha \leq 2q$, then we are able to upper-bound Eq. (5) by $4q(A_n + O_n)$ and derive

$$\frac{A_n}{O_n} \leq \frac{1+4q}{1-4q},$$

which gives the desired competitive factor.

So, to utilize Eq. (5), in our proof we will focus on showing that $m((t_{i-1}, t_i])$ is sufficiently smaller than $t_i - t_{i-1}$. However, this can be challenging as OPT is allowed to move often, potentially on every request which results in $t_i - t_{i-1}$ being very small. But, if $t_i - t_{i-1}$ is too small, then Assumption 1 gives no information about $m((t_{i-1}, t_i])$. However, if intervals $t_i - t_{i-1}$ are large enough, e.g., at least βD for some positive constant β, then from Assumption 1 we would be able to conclude that $\alpha = O(q)$. Since in principle OPT can move in every step, we design 'lazy' versions of OPT and ALG that only move $O(1)$ times in any interval of length D. This will enable us to argue that $t_i - t_{i-1}$ is not too small. It turns out that the respective competitive factors of the lazy versions with respect to the original versions is very close, allowing us prove

$$\frac{A_n}{O_n} \approx \frac{A_n^{lazy}}{O_n^{lazy}} \leq (1+\varepsilon)\frac{1+O(q)}{1-O(q)}.$$

3.2 ALG$^{\text{robust}}$, a robust version of ALG

We now describe ALG$^{\text{ROBUST}}$. This algorithm follows a "lazy" variant of ALG as long as Assumption 1 holds, and otherwise switches to ALG$^{\text{ONLINE}}$. Instead of using ALG directly, we use a 'lazy' version of ALG that works as follows: Follow the optimal offline solution given by ALG with a delay of $6qD$ steps. Let ALG$^{\text{LAZY}}$ be the corresponding algorithm. We point out that performing some delay with respect to ALG is crucial here. To see that, consider the following example in the case of uniform metric spaces: $s = \{0\}^n$ and $\hat{s} = \{1\}^n$, and let the starting location be 0. According to ALG, the page should be moved from 0 to 1 in the very beginning, incurring the cost of D. On the

other hand, OPT never moves from 0. If $\text{ALG}^{\text{ROBUST}}$ would follow ALG until it realizes that the fraction of errors is too high, it would already pay the cost of at least D, leading to an unbounded competitive ratio. However, if $\text{ALG}^{\text{ROBUST}}$ delays following ALG, then it gets some "slack" in verifying whether the predicted sequence properly predicts requests or not. As a result, when Assumption 1 holds, this delay increases the overall serving cost by a factor $O(1+O(q))$, but in turn achieves a bounded competitive ratio when this assumption does not hold.

While serving requests, $\text{ALG}^{\text{ROBUST}}$ also maintains the execution of $\text{ALG}^{\text{ONLINE}}$, i.e., $\text{ALG}^{\text{ROBUST}}$ maintains where $\text{ALG}^{\text{ONLINE}}$ would be at a given point in time, in case a fallback is needed. Now $\text{ALG}^{\text{ROBUST}}$ simply executes ALG^{LAZY} unless we find a violation of Assumption 1 is detected. Once such a violation is detected, the algorithm switches to $\text{ALG}^{\text{ONLINE}}$ by moving its location to $\text{ALG}^{\text{ONLINE}}$'s current location. From there on $\text{ALG}^{\text{ONLINE}}$ is executed.

We now present the intuition behind the proof for the competitive factor of the algorithm.

3.2.1 Case when Assumption 1 holds.

In this case $\text{ALG}^{\text{ROBUST}}$ is ALG^{LAZY}, and the analysis boils down to proving competitive ratio of ALG^{LAZY}. We show that ALG^{LAZY} is $(1+O(q))$-competitive to ALG, which is, as we argued in the previous section, $1+O(q)$ competitive to OPT. To see this, we employ the following charging argument: whenever ALG moves from a to a' it pays $D \cdot d(a,a')$. The lazy algorithm eventually pays the same moving cost of less.

However, in addition, the serving cost of ALG^{LAZY} for each of the $6qD$ requests is potentially increased, as ALG^{LAZY} is not at the same location as ALG. Nevertheless, by triangle inequality, the cost due to the movement from a to a' of ALG reflect to an increase in the serving cost of ALG^{LAZY} by at most $d(a,a')$. In total over all the $6qD$ requests and per each move of ALG from a to a', ALG^{LAZY} pays at most $6qDd(a,a')$ extra cost compared to ALG. Considering all migrations, this gives a $1+O(q)$ competitive factor.

3.2.2 Case when Assumption 1 is violated.

The case where Assumption 1 is violated (say at time t') is considerably more involved. We then have

$$\begin{aligned}\text{ALG}^{\text{ROBUST}} \leq &\text{ALG}^{\text{LAZY}}(0,t') + \text{ALG}^{\text{ONLINE}}(t'+1,n)\\ &+ D \cdot d(a,a'),\end{aligned}$$

and we seek to upper-bound each of these terms by $O(\text{OPT}/q)$. While the upper-bound holds directly for $\text{ALG}^{\text{ONLINE}}(t'+1,n)$, showing the upper-bound for other terms is more challenging.

The key insight is that, due to the optimality of ALG,

$$d(a,p_0) \leq OPT(t')/(qD), \tag{6}$$

which can be proven as follows. If ALG migrates its page to a location that is far from the starting location p_0, then there have to be, even when taking into account noise, at least $4qD$ page requests that are far from p_0. OPT also has to serve these requests (either remotely or by moving), and hence has to pay a cost of at least $qD \cdot d(a,p_0)$. Equipped with this idea, we can now bound $D \cdot d(a,a')$ in terms of $\text{OPT}(t')/q$. To bound $\text{ALG}^{\text{LAZY}}(0,t')$ we need one more idea. Namely, we compare $\text{ALG}^{\text{LAZY}}(0,t')$ to the optimal solution that has a constraint to be at the same position as ALG^{LAZY} at time t'. A formal analysis is given in Appendix B.

4 The Analysis of ALG

Now we analyze ALG (Algorithm 1). As discussed in Section 3.1, our main objective is to establish Eq. (5), which we do in Section 4.1. That upper-bound will be directly used to obtain our result for uniform metric spaces, as we present in Appendix A.1. To construct our algorithm for general-metric spaces, in Appendix A.2 we build on ALG by first designing its "lazy" variant. As the final result, we show the following. Recall that q is the fraction of symbols that the adversary is allowed to change in any sequence of length εD of the predicted sequence.

Theorem 2. *If Assumption 1 holds with respect to parameter ε, then we obtain the following results:*

(A) There exists a $(1+\varepsilon)\cdot(1+O(q))$-competitive algorithm for the online page migration problem.

(B) There exists a $(1+O(q))$-competitive algorithm for the online page migration problem in uniform metric spaces.

We defer the proof of Theorem 2 to Appendix A. Note that in the prediction-free case Matsubayashi (2015) show that the competitive ratio of any deterministic algorithm must be at least $3+\epsilon$ for some $\epsilon > 0$. In the case of having access to predictions, Theorem 2 is asymptotically optimal with respect to q. Namely, any algorithm is at least $1+\Omega(q)$ competitive; even in the uniform metric case. To see this consider the following binary example where the algorithm starts at position 0. The advice is $\hat{s} = \underbrace{111\cdots1111}_{(1-q)D}\underbrace{000\cdots000}_{2qD}$. The final sequence is $s = \begin{cases} \hat{s} & \text{w.p. } 1/2 \\ \underbrace{111\cdots1111}_{(1+q)D} & \text{otherwise.}\end{cases}$

In the first case OPT simply stays at 0 since moving costs D; in the second case, OPT goes immediately to 1. Note that ALG can only distinguish between the sequences after $(1-q)D$ steps at which point it is doomed to have an additional cost of qD with probability at least $1/2$ depending on the sequence s.

4.1 Establishing Eq. (5)

In our proofs we will use the following corollary of Assumption 1.

Corollary 3. *If Assumption 1 holds, then for any interval $\mathcal{I}$ of length $\ell > \varepsilon D$ it holds $m(\mathcal{I}) \leq 2q\ell$.*

Proof. This statement follows from the fact that each such $\mathcal{I}$ can be subdivided into $k \geq 1$ intervals of length exactly εD and at most one interval $\mathcal{I}'$ of length less than εD. On one hand, the total number of mismatches for these intervals of length exactly εD is upper-bounded by $qk\varepsilon D \leq q\ell$. On the other hand, since $\mathcal{I}'$ is a subinterval of an interval of length εD, it holds $m(\mathcal{I}') \leq q\varepsilon D < q\ell$. The claim now follows. □

Most of our analysis in this section proceeds by reasoning about intervals where neither ALG nor OPT moves. Let $t_1, t_2 \dots$ be the time steps at which either OPT or ALG move. The final product of this section will be an upper-bound on $A_n - O_n$ as given by Eq. (5)[4], i.e.,

$$\begin{aligned} &A_n - O_n \\ \leq &2 \sum_i m((t_{i-1}, t_i]) \\ &\cdot \frac{A_{t_i} - A_{t_{i-1}} + O_{t_i} - O_{t_{i-1}} - c_{move}^{(t_{i-1},t_i]}}{t_i - t_{i-1}}. \end{aligned}$$

We begin by rewriting and upper-bounding $A_t - O_t$ as follows

$$\begin{aligned} A_t - O_t &= A_t - \hat{A}_t + \hat{A}_t - O_t \\ &\leq A_t - \hat{A}_t + \hat{O}_t - O_t, \end{aligned} \tag{7}$$

where we used that $\hat{A}_t \leq \hat{O}_t$ as $\hat{A}_t$ is the optimum for $\hat{s}$. Consider a fixed interval $\mathcal{I} = (t_{i-1}, t_i]$. Then, by triangle inequality, it holds

$$\begin{aligned} d(a_t, o_t) &\leq d(a_t, s_t) + d(o_t, s_t) \\ &\leq A_t - A_{t-1} + O_t - O_{t-1}.^{5} \end{aligned} \tag{8}$$

Let $c_{move}^{(t_{i-1},t_i]}$ be the sum of moving costs for OPT and ALG in $(t_{i-1}, t_i]$. Note that

$$\begin{aligned} &A_{t_i} - A_{t_{i-1}} + O_{t_i} - O_{t_{i-1}} \\ =&\sum_{t \in (t_{i-1},t_i]} (A_t - A_{t-1} + O_t - O_{t-1}) \\ \geq &c_{move}^{(t_{i-1},t_i]} + d(a_{t_i}, o_{t_i})|t_i - t_{i-1}|, \end{aligned} \tag{9}$$

where the inequality comes from Eq. (8) applied to every time step in $(t_{i-1}, t_i]$ and the fact that ALG or OPT must have moved inducing a cost of at least $c_{move}^{(t_{i-1},t_i]}$. The following notation is used to represent the difference between serving s_i and $\hat{s}_i$ by ALG

$$\begin{aligned} A[t-1,t] &:= A_t - \hat{A}_t - (A_{t-1} - \hat{A}_{t-1}) \\ &= d(a_t, s_t) - d(a_t, \hat{s}_t). \end{aligned}$$

Note that this holds even when ALG moves since the moving costs for the oracle sequence and on the final sequence are the same and therefore cancel each other out. Similarly to $A[t-1,t]$, let

$$\begin{aligned} \hat{O}[t-1,t] &:= \hat{O}_t - O_t - (\hat{O}_{t-1} - O_{t-1}) \\ &= d(o_t, \hat{s}_t) - d(o_t, s_t). \end{aligned}$$

Consider now any $t \in [1, n]$. By triangle inequality we have

$$\begin{aligned} &A[t-1,t] + \hat{O}[t-1,t] \\ =&d(a_t, s_t) - d(o_t, s_t) + d(o_t, \hat{s}_t) - d(a_t, \hat{s}_t) \\ \leq &(d(a_t, o_t) + d(o_t, s_t)) - d(o_t, s_t) \\ &+ (d(a_t, \hat{s}_t) + d(a_t, o_t)) - d(a_t, \hat{s}_t) \\ =&2d(a_t, o_t) \\ \overset{(9)}{\leq} &2 \frac{A_{t_i} - A_{t_{i-1}} + O_{t_i} - O_{t_{i-1}} - c_{move}^{(t_{i-1},t_i]}}{t_i - t_{i-1}}. \end{aligned} \tag{10}$$

Let $\Delta_i = A_{t_i} - \hat{A}_{t_i} + \hat{O}_{t_i} - O_{t_i}$, where $\Delta_0 = 0$ by definition. Note that

$$\begin{aligned} A_n - O_n \overset{(7)}{\leq} A_n - \hat{A}_n + \hat{O}_n - O_n &= \sum_i (\Delta_i - \Delta_{i-1}) \\ &= \sum_i \sum_{t \in (t_{i-1},t_i]} \Big(A[t-1,t] + \hat{O}[t-1,t] \Big). \end{aligned}$$

Recall that, for a given interval $\mathcal{I}$ the function $m(\mathcal{I})$ denotes the number of mismatches between s and $\hat{s}$ within $\mathcal{I}$ (see Definition 1). Now, as for t such that $s_t = \hat{s}_t$ we have $A[t-1,t] = \hat{O}[t-1,t] = 0$, the last

[4] As a reminder, A_t (O_t, respectively) is the cost of ALG (OPT, respectively) at time for the sequence $s_{[1,t]}$.

chain of inequalities further implies

$$\begin{aligned} & A_n - O_n \\ \overset{Eq.\ (10)}{\leq} & \sum_i \sum_{t \in (t_{i-1}, t_i]} 1_{s_t \neq \hat{s}_t} \\ & \cdot 2 \frac{A_{t_i} - A_{t_{i-1}} + O_{t_i} - O_{t_{i-1}} - c_{move}^{(t_{i-1}, t_i]}}{t_i - t_{i-1}} \\ \leq & 2 \sum_i m((t_{i-1}, t_i]) \\ & \cdot \frac{A_{t_i} - A_{t_{i-1}} + O_{t_i} - O_{t_{i-1}} - c_{move}^{(t_{i-1}, t_i]}}{t_i - t_{i-1}}. \end{aligned} \quad (11)$$

This establishes the desired upper-bound on $A_n - O_n$. As discussed in Section 3.1, this upper-bound is used to derive our non-robust results for uniform (Appendix A.1) and general (Appendix A.2) metric spaces. The main task in those two sections will be to show that $m((t_{i-1}, t_i])$ is sufficiently smaller than $t_i - t_{i-1}$.

5 Experiments

We evaluate our approach on two synthetic data sets, and compare it to the state of the art algorithm for page migration due to Westbrook Westbrook (1994). The two data sets are obtained by generating "predicted" sequences of points in the plane, and then perturbing each point by independent Gaussian noise to obtain "actual" sequences. The predicted sequence is fed to our algorithm, while the actual sequence forms an input of the online algorithm. Recall that our algorithm sees the actual sequence only in the online fashion.

5.1 Data sets

The predicted sequences of the two sets of points are generated as follows:

1. **Line process:** the t-th point $(\hat{X}_1(t), \hat{X}_2(t))$ is equal to $(t, 0)$.

2. **Brownian motion process:** the t-th point $\hat{X}(t)$ is equal to $\hat{X}(t-1) + (\Delta_1(t), \Delta_2(t))$, where $\Delta_t(t)$ and $\Delta_2(t)$ are i.i.d. random variables chosen from $N(0, 1)$.

Note that the predicted line process is completely deterministic whereas the Brownian motion points has, by definition, Gaussian noise. In both cases, the actual sequence is generated by adding (additional) Gaussian noise to the predicted sequence: the t-th request $X(t)$ in the actual sequence is equal to $\hat{X}(t) + (N_1(t), N_2(t))$, where $N_1(t), N_2(t)$ are i.i.d. random variables chosen from $N(0, \sigma^2)$. The value of σ varies, depending on the specific experiment. An example Brownian motion sequence is depicted in Fig. 2.

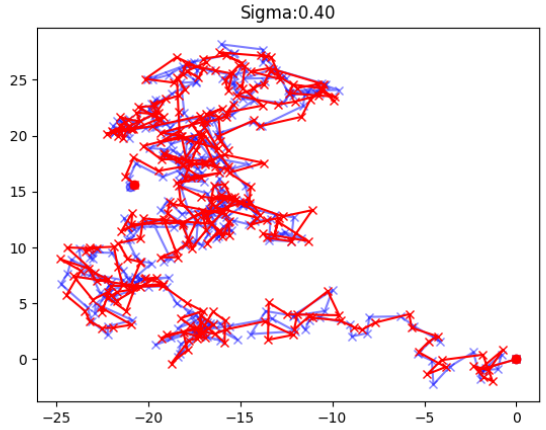


Figure 2: An example of Brownian motion sequence. The predicted sequence is in blue, the actual sequence is in red.

5.2 Set up

We use the two data sets to compare the following three algorithms:

- PREDICT refers to our algorithm, which computes the optimum solution for the predicted sequence (by using standard dynamic programming) and follows that optimum to serve actual requests.

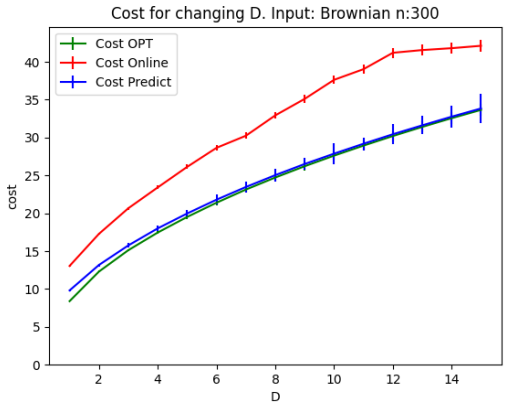
- OPT is the optimum *offline* algorithm executed on the actual sequence. This optimum is computed by using the same dynamic programming as in the implementation of PREDICT.

- ONLINE is state-of-the-art online randomized algorithm for page migration that achieves 2.62-approximation in expectation. This algorithm is described in Section 4.1 of Westbrook (1994). Since it is randomized, on each input we perform 100 runs of ONLINE and as the output report the average of all the runs. The standard deviation is smaller than 5%.

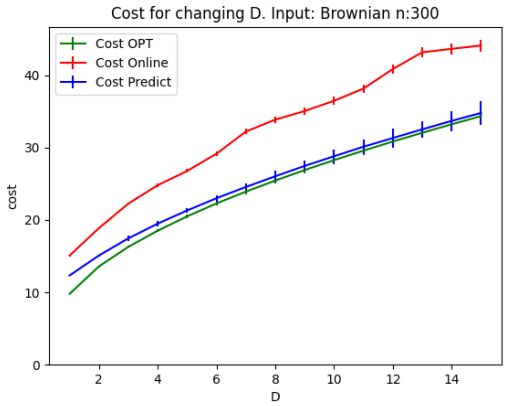
For both data sets, we depict the costs of the three algorithms as a function of either D or σ. See the text above each plots for the specification.

5.2.1 Results

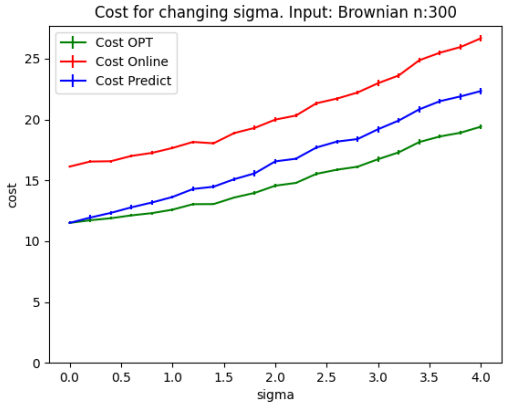
The results for the Brownian motion data set are depicted in Fig. 3. The top two figures show the cost incurred by each algorithm for fixed values of σ and



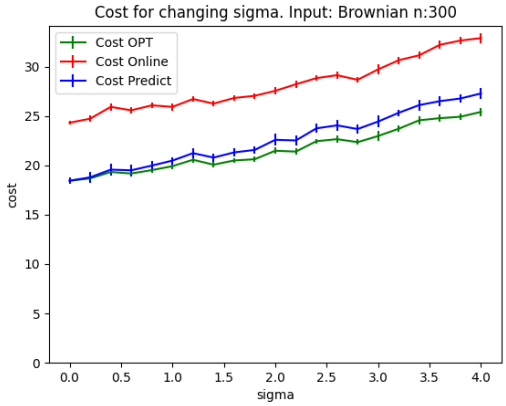
(a) Fixed sigma= 0.8, varying D.



(b) Fixed sigma= 1.6, varying D.



(c) Fixed $D = 2$, varying sigma.



(d) Fixed $D = 5$, varying sigma.

Figure 3: Comparison between PREDICT, OPT and ONLINE on Brownian motion data set.

different values of D, while the bottom two figures show the costs for fixed values of D while σ varies. Not surprisingly, for low values of σ, the costs PREDICT and OPT are almost equal, since the predicted and the actual sequences are very close to each other. As the value of σ increases, their costs starts to diverge. Nevertheless, the benefit of predictions is clear, as the cost of PREDICT is significantly lower than the cost of ONLINE. Interestingly, this holds even though the fraction of requests predicted *exactly* is very close to 0. Each point in the plots is obtained by averaging over 15 runs. The standard deviation are also depicted in the figures as vertical lines (in some cases the standard deviation is so small that it is not visible in the plots).

The results for the Line data set is depicted in Fig. 6 (see Appendix C). They are qualitatively similar to those for Brownian motion.

More complex experiments. In Fig. 4 we present the results of a more complex experiment where we average 10 adversarial sequences, e.g., data observed in the past, and use it as a prediction for PREDICT. These adversarial sequences are obtained by perturbing the same ground truth. Notice that this means that in order to obtain a prediction we do not use the ground truth directly. Each point in Fig. 4 is obtained by averaging over 5 runs. The standard deviation is so small that it is not visible in the plot.

6 Conclusion

In this paper we studied online Page Migration, which is one of the classical online problems, in the setting where there is a predicted sequence of yet-to-be-seen requests. When this sequence predicts the online sequence reasonably well, our algorithm provides better approximation than state-of-the-art approach for classical online setting. It is an interesting open question to improve the constant in our competitive factor to obtain tight bounds.

Arguably the most interesting continuation of this

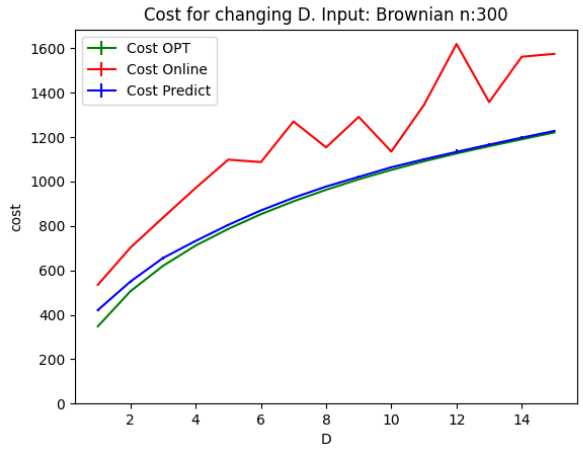


Figure 4: Comparison between PREDICT, OPT and ONLINE where the prediction for PREDICT is obtained from past data.

work is to use ML tools to obtain the predicted sequence. For example, a neural network could be trained on the sequence seen so far to predict where the future requests will be. Combining this with our robust algorithm will likely give very strong results when the sequence has a somewhat predictable structure. Characterizing such predictable structures theoretically will likely yield new interesting models which has potentially application beyond Page Migration.

Acknowledgements

S. Mitrović was supported by the Swiss NSF grant No. P400P2_191122/1, NSF award CCF-1733808, and FinTech@CSAIL. R. Rubinfeld was supported by the NSF TRIPODS program (awards CCF-1740751 and DMS-2022448), NSF award CCF-2006664, and FinTech@CSAIL. F. Mallmann-Trenn was supported by the EPSRC grant EP/W005573/1.

References

(2016). *Special Semester on Algorithms and Uncertainty*. https://simons.berkeley.edu/programs/uncertainty2016.

Awerbuch, B., Bartal, Y., and Fiat, A. (1993). Competitive distributed file allocation. In *STOC*, volume 93, pages 164–173.

Awerbuch, B., Bartal, Y., and Fiat, A. (2003). Competitive distributed file allocation. *Information and Computation*, 185(1):1–40.

Balcan, M.-F., Dick, T., Sandholm, T., and Vitercik, E. (2018). Learning to branch. In *International Conference on Machine Learning*, pages 353–362.

Bartal, Y., Charikar, M., and Indyk, P. (1997). On page migration and other relaxed task systems. *SODA*.

Bartal, Y., Fiat, A., and Rabani, Y. (1995). Competitive algorithms for distributed data management. *Journal of Computer and System Sciences*, 51(3):341–358.

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

Bienkowski, M., Byrka, J., and Mucha, M. (2017). Dynamic beats fixed: On phase-based algorithms for file migration. *ICALP*.

Black, D. L. and Sleator, D. D. (1989). *Competitive algorithms for replication and migration problems*. Carnegie-Mellon University. Department of Computer Science.

Bora, A., Jalal, A., Price, E., and Dimakis, A. G. (2017). Compressed sensing using generative models. In *International Conference on Machine Learning*, pages 537–546.

Boyar, J., Favrholdt, L. M., Kudahl, C., Larsen, K. S., and Mikkelsen, J. W. (2017). Online algorithms with advice: A survey. *ACM Computing Surveys (CSUR)*, 50(2):19.

Chrobak, M., Larmore, L. L., Reingold, N., and Westbrook, J. (1997). Page migration algorithms using work functions. *Journal of Algorithms*, 24(1):124–157.

Gollapudi, S. and Panigrahi, D. (2019a). Online algorithms for rent-or-buy with expert advice. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2319–2327.

Gollapudi, S. and Panigrahi, D. (2019b). Online algorithms for rent-or-buy with expert advice. In *International Conference on Machine Learning*, pages 2319–2327.

Hsu, C.-Y., Indyk, P., Katabi, D., and Vakilian, A. (2019). Learning-based frequency estimation algorithms. In *International Conference on Learning Representations*.

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., and Song, L. (2017). Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, pages 6348–6358.

Khorramian, A. and Matsubayashi, A. (2016). Uniform page migration problem in euclidean space. *Algorithms*, 9(3):57.

Kraska, T., Beutel, A., Chi, E. H., Dean, J., and Polyzotis, N. (2018). The case for learned index structures. In *Proceedings of the 2018 International Conference on Management of Data*, pages 489–504.

Kumar, R., Purohit, M., Schild, A., Svitkina, Z., and Vee, E. (2019). Semi-online bipartite matching. *ITCS*.

Lattanzi, S., Lavastida, T., Moseley, B., and Vassilvitskii, S. (2020). Online scheduling via learned weights. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1859–1877. SIAM.

Lund, C., Reingold, N., Westbrook, J., and Yan, D. (1998). Competitive on-line algorithms for distributed data management. *SIAM Journal on Computing*, 28(3):1086–1111.

Lykouris, T. and Vassilvitskii, S. (2018). Competitive caching with machine learned advice. In *International Conference on Machine Learning*, pages 3302–3311.

Manasse, M. S., McGeoch, L. A., and Sleator, D. D. (1990). Competitive algorithms for server problems. *Journal of Algorithms*, 11(2):208–230.

Matsubayashi, A. (2015). A 3+ omega (1) lower bound for page migration. In *2015 Third International Symposium on Computing and Networking (CANDAR)*, pages 314–320. IEEE.

Mitzenmacher, M. (2018). A model for learned bloom filters and optimizing by sandwiching. In *Advances in Neural Information Processing Systems*, pages 464–473.

Mousavi, A., Patel, A. B., and Baraniuk, R. G. (2015). A deep learning approach to structured signal recovery. In *Communication, Control, and Computing (Allerton), 2015 53rd Annual Allerton Conference on*, pages 1336–1343. IEEE.

Purohit, M., Svitkina, Z., and Kumar, R. (2018). Improving online algorithms via ml predictions. In *Advances in Neural Information Processing Systems*, pages 9661–9670.

Rohatgi, D. (2020). Near-optimal bounds for online caching with machine learned advice. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1834–1845. SIAM.

Wang, J., Liu, W., Kumar, S., and Chang, S.-F. (2016). Learning to hash for indexing big data - a survey. *Proceedings of the IEEE*, 104(1):34–57.

Westbrook, J. (1994). Randomized algorithms for multiprocessor page migration. *SIAM Journal on Computing*, 23(5):951–965.

Online Page Migration with ML Advice
Supplementary Material

A Proofs Missing from Section 4

A.1 Uniform Metric Spaces – Theorem 2 (B)

We now use the upper-bound on $A_n - O_n$ given by Eq. (11) to show that ALG is $(1+O(q))$-competitive under Assumption 1, i.e., we show Theorem 2 (A). We distinguish between two cases: $t_i - t_{i-1} \geq D$; and $t_i - t_{i-1} < D$.

A.1.1 Case $t_i - t_{i-1} \geq D$.

In this case, by Corollary 3 we have $m((t_{i-1}, t_i]) \leq 2q|t_i - t_{i-1}|$. Plugging this into Eq. (11) we derive

$$\begin{aligned} A_n - O_n &\leq 2\sum_i m((t_{i-1}, t_i]) \cdot \frac{A_{t_i} - A_{t_{i-1}} + O_{t_i} - O_{t_{i-1}}}{t_i - t_{i-1}} \\ &\leq 4q \sum_i (A_{t_i} - A_{t_{i-1}} + O_{t_i} - O_{t_{i-1}}) \\ &= 4q(A_n + O_n). \end{aligned}$$

A.1.2 Case $t_i - t_{i-1} < D$.

We proceed by upper-bounding all the terms in Eq. (11). As the interval $(t_{i-1}, t_i]$ is a subinterval of $(t_{i-1}, t_{i-1}+D]$, we have

$$m((t_{i-1}, t_{i-1} + D]) \leq m((t_{i-1}, t_i]) \leq qD.$$

Also, observe that trivially it holds

$$A_{t_i} - A_{t_{i-1}} + O_{t_i} - O_{t_{i-1}} \leq 2|t_i - t_{i-1}| + c_{move}^{(t_{i-1}, t_i]}. \tag{12}$$

Combining the derived upper-bounds, we establish

$$\begin{aligned} A_n - O_n &\overset{\text{Eq. (11)}}{\leq} 2\sum_i m((t_{i-1}, t_i]) \cdot \frac{A_{t_i} - A_{t_{i-1}} + O_{t_i} - O_{t_{i-1}} - c_{move}^{(t_{i-1}, t_i]}}{t_i - t_{i-1}} \\ &\overset{\text{Eq. (12)}}{\leq} 2\sum_i qD \frac{2(t_i - t_{i-1}) + c_{move}^{(t_{i-1}, t_i]} - c_{move}^{(t_{i-1}, t_i]}}{t_i - t_{i-1}} \qquad (13) \\ &= 4q \sum_i D. \qquad (14) \end{aligned}$$

To conclude this case, note that by definition either ALG or OPT moves within $(t_{i-1}, t_i]$, incurring the cost of at least D. Therefore, $A_{t_i} - A_{t_{i-1}} + O_{t_i} - O_{t_{i-1}} \geq D$. This together with Eq. (14) implies

$$A_n - O_n \leq 4q \sum_i (A_{t_i} - A_{t_{i-1}} + O_{t_i} - O_{t_{i-1}}) = 4q(A_n + O_n).$$

A.1.3 Combining the two cases.

We have concluded that in either case it holds $A_n - O_n \leq 4q(A_n + O_n)$ and hence we derive

$$\frac{A_n}{O_n} \leq \frac{1+4q}{1-4q}.$$

This concludes the analysis for uniform metric spaces.

A.2 General Metric Spaces – Theorem 2 (A)

As in the uniform case, our goal for general metric spaces is to use Eq. (5) for proving the advertised competitive ratio. However, as we discussed in Section 3.1, the main challenge in applying Eq. (5) lies in upper-bounding the ratio between $m((t_{i-1}, t_i])$ and $t_i - t_{i-1}$ by a small constant, ideally much smaller than 1. Unfortunately, this ratio can be as large as 1 as OPT (or ALG) could possibly move on every single request. To see that, consider the scenario in which all the requests are on the x-axis and are requested in their increasing order of their location. Then, for all but potentially the last D requests, OPT would move from request to request. To bypass this behavior of OPT and ALG, we define and analyze their "lazy" variants, i.e., variants in which OPT and ALG are allowed to move only at the i-th request when i is a multiple of εD. We now state the algorithm.

A.2.1 Our Algorithm ALG^{lazy}

We use the following algorithm ALG^{LAZY}: Compute the optimal offline solution (on $\hat{s}$) while only moving on multiples of εD. Let A^{LAZY} be the cost of the solution s and let $\widehat{A^{\text{LAZY}}}$ be the cost of the solution on $\hat{s}$. Note that there can be better offline algorithms for $\hat{s}$, however ALG^{LAZY} has the minimal cost among all online algorithms that are only allowed to move every multiple of εD.

A.2.2 Proof

We also need to consider a lazy version of OPT, which we do in the following lemma. There we show that making any algorithm lazy does not increase the cost by more than a factor of $(1+\varepsilon)$. In particular, we will show $O^{\text{LAZY}} \leq (1+\varepsilon)\text{OPT}$. Let A_t^{LAZY} and O_t^{LAZY} denote their costs at time t.

Lemma 4. *Let $\varepsilon \in (0, 1]$. Consider an arbitrary prefix w of length t of a sequence of requests. Let B_t be the cost of* any *algorithm ALG_B serving w. Let $B_t^{\text{LAZY}\prime}$ be the cost of the algorithm that* has *to move at every time step that is a multiple of εD (and is not allowed to move at any other time step), and to move to the position where ALG_B is at that time step. Then, we have*

$$B_t^{\text{LAZY}\prime} \leq (1+\varepsilon)B_t.$$

Proof. Let x_i be the distance of the i-th move and y_i be the cost for serving the i-th request remotely. Then,

$$B_t = D\sum_i x_i + \sum_i y_i.$$

Now we relate B_t and $B_t^{\text{LAZY}\prime}$. $B_t^{\text{LAZY}\prime}$ has two components: the moving cost and the cost for serving remotely. By triangle inequality, the moving cost is upper-bounded by $D\sum_i x_i$. Consider now interval $\mathcal{I}_j \in [j\varepsilon D+1, (j+1)\varepsilon D]$ for some integer j. To serve point $i \in \mathcal{I}_j$ remotely, the cost is, by triangle inequality, at most the cost of y_i plus the cost of traversing all the points with indices in $\mathcal{I}_j$ where ALG_B has moved to. Thus the cost per request $i \in \mathcal{I}_j$ is upper-bounded by $y_i + \sum_{k\in\mathcal{I}_j} x_k$. Note that the summation $\sum_{k\in\mathcal{I}_j} x_k$ is charged to εD requests. Hence, summing over all the intervals gives

$$B_t^{\text{LAZY}\prime} \leq D\sum_i x_i + \sum_i y_i + \varepsilon D\sum_i x_i \leq (1+\varepsilon)B_t.$$

□

Define O^{LAZY} as the cost of the optimal algorithm for s that is *allowed* to move only at time steps which are multiple of εD. Similarly as in Lemma 4, we have $O_n^{\text{LAZY}} \leq (1+\varepsilon)O_n$. Thus,

$$\frac{A_n^{\text{LAZY}}}{O_n} \leq (1+\varepsilon)\frac{A_n^{\text{LAZY}}}{O_n^{\text{LAZY}}}. \tag{15}$$

Now we need to upper-bound $\frac{A_n^{\text{LAZY}}}{O_n^{\text{LAZY}}}$. We will do that by showing that the same statements as we developed in Section 4.1 hold for A^{LAZY} and O^{LAZY}. To that end, observe that to derive Eq. (7) we used the fact that $\hat{A} \leq \hat{O}$. Notice that the analog inequality $\widehat{A^{\text{LAZY}}} \leq \widehat{O^{\text{LAZY}}}$ holds, since ALG^{LAZY} is the the optimal offline algorithm that only moves every multiple of εD.

Hence, we can obtain the derivation Eq. (11) for $A_n^{\text{LAZY}} - O_n^{\text{LAZY}}$

$$A_n^{\text{LAZY}} - O_n^{\text{LAZY}} \leq 2 \sum_i m((t_{i-1}, t_i]) \cdot \frac{A_{t_i}^{\text{LAZY}} - A_{t_{i-1}}^{\text{LAZY}} + O_{t_i}^{\text{LAZY}} - O_{t_{i-1}}^{\text{LAZY}}}{t_i - t_{i-1}}. \tag{16}$$

Since for the lazy versions we have $|t_i - t_{i-1}| = \varepsilon D$, Assumption 1 implies $m((t_{i-1}, t_i]) \leq \varepsilon q D$. Plugging this into Eq. (16) gives

$$A_n^{\text{LAZY}} - O_n^{\text{LAZY}} \leq 2q \sum_i \left(A_{t_i}^{\text{LAZY}} - A_{t_{i-1}}^{\text{LAZY}} + O_{t_i}^{\text{LAZY}} - O_{t_{i-1}}^{\text{LAZY}} \right) = 2q(A_n^{\text{LAZY}} + O_n^{\text{LAZY}}).$$

From Eq. (15) we establish

$$\frac{A_n^{\text{LAZY}}}{O_n} \leq (1+\varepsilon)\frac{A_n^{\text{LAZY}}}{O_n^{\text{LAZY}}} \leq (1+\varepsilon)\frac{1+2q}{1-2q}.$$

This concludes the proof of Theorem 2 (A).

B Robust Page Migration

So far we designed algorithms for the online page migration problem that have small competitive ratio when Assumption 1 holds. In this section we build on those algorithm and design a (robust) algorithm that performs well even when Assumption 1 does not hold, while still retaining competitiveness when Assumption 1 is true. We refer to this algorithm by $\text{ALG}^{\text{ROBUST}}$. For $\text{ALG}^{\text{ROBUST}}$ we prove the following.

Theorem 5. *Let γ be the competitive ratio of* ALG *for the online page migration problem, and let q be a positive number less than $1/24$. If Assumption 1 holds, then* $\text{ALG}^{\text{ROBUST}}$ *is $\gamma \cdot (1 + O(q))$-competitive, and otherwise* $\text{ALG}^{\text{ROBUST}}$ *is $O(1/q)$-competitive.*

Using our techniques it is straight-forward to obtain an arbitrary trade-off between the two competitive ratios. Fix an arbitrary $x \geq 1$, then Algorithm $\text{ALG}^{\text{ROBUST}}$ is $(1 + O(x \cdot q))$-competitive if Assumption 1 holds and $O(1/(x \cdot q))$-competitive otherwise.

B.1 Algorithm $\text{ALG}^{\text{robust}}$

Let $\text{ALG}^{\text{ONLINE}}$ refer to an arbitrary online algorithm for the problem, e.g., Westbrook (1994). We now define $\text{ALG}^{\text{ROBUST}}$. This algorithm switches from ALG to $\text{ALG}^{\text{ONLINE}}$ when it detects that Assumption 1 does not hold. Instead of using ALG directly, we use a "lazy" version of ALG that works as follows. Follow the optimal offline solution given by ALG with a delay of $6qD$ steps. Let ALG^{LAZY} be the corresponding algorithm. (A lazy version for different setup of parameters was presented in Appendix A.2.)

Throughout its execution, $\text{ALG}^{\text{ROBUST}}$ maintains/tracks in its memory the execution of $\text{ALG}^{\text{ONLINE}}$ on the prefix of s seen so far. That is, $\text{ALG}^{\text{ROBUST}}$ maintains where $\text{ALG}^{\text{ONLINE}}$ would be at a given point in time in case a fallback is needed. Now $\text{ALG}^{\text{ROBUST}}$ simply executes ALG^{LAZY} unless we find a violation of Assumption 1 is detected. Once such a violation is detected, the algorithm switches to $\text{ALG}^{\text{ONLINE}}$ by moving its location to $\text{ALG}^{\text{ONLINE}}$'s current location. From there on $\text{ALG}^{\text{ONLINE}}$ is executed.

We now analyze $\text{ALG}^{\text{ROBUST}}$ and show that in case Assumption 1 holds, then ALG and $\text{ALG}^{\text{ROBUST}}$ are close in terms of total cost, and otherwise the cost of $\text{ALG}^{\text{ROBUST}}$ is at most $O(1/q)$ larger than that of $\text{ALG}^{\text{ONLINE}}$.

B.1.1 Case 1: Assumption 1 holds for the entire sequence.

In this case $\text{ALG}^{\text{ROBUST}}$ executes ALG^{LAZY} throughout. Following the same argument for $\varepsilon = 6q$ as given for $A^{\text{LAZY}\prime}$ in the proof of Lemma 4, we have

$$A_t^{\text{LAZY}} \leq (1 + 6q)A_t. \tag{17}$$

Thus,

$$A^{\text{ROBUST}} = A_n^{\text{LAZY}} \leq (1+6q)A_n \leq \gamma(1+O(q))O,$$

where we used the assumption that ALG is γ-competitive. This completes this case.

B.1.2 Case 2: Assumption 1 is violated at the t-th request.

Let $t' = t - qD + 1$. Note that up to this point in time no violation occurred. We define the following: a is the position of ALG^{LAZY} at time t'; a' is the position of $\text{ALG}^{\text{ONLINE}}$ at time $t'+1$; o is the position of OPT at time t'; and, $O^p_{0,t''}$ is the cost of OPT up to time t'' where we demand that OPT is at position p at t''.

In the following, we assume the following holds. We defer the proof of its correctness for later.

$$d(a, p_0) \leq O_{t'}/(qD). \tag{18}$$

Intuitively, this means that we can bound the distance from the starting position by the cost of OPT.

Using Eq. (18), we get,

$$A^{\text{ROBUST}} \leq A^{\text{LAZY}}_{t'} + A^{\text{ONLINE}}_{t'+1,n} + D \cdot d(a, a'). \tag{19}$$

As $O^a_{0,t'}$ and $A^{\text{LAZY}}_{0,t'}$ are at the same position at time t', inequality $A^{\text{LAZY}}_{0,t'} \leq (1+c_1 q)O^a_{0,t'}$ follows from Eq. (17) for a suitable constant c_1. Note that $O_{t'} \geq D \cdot d(p_0, o)$, which holds since this cost is already incurred by moving to o, where we used triangle inequality.

Next, using triangle inequality again, we get

$$\begin{aligned}
&A^{\text{LAZY}}_{0,t'} \\
&\leq (1+c_1 q)O^a_{0,t'} \\
&\leq (1+c_1 q)\left(O^o_{0,t'} + D \cdot d(a, o)\right) \\
&\leq (1+c_1 q)\left(O^o_{0,t'} + D \cdot d(a, p_0) + D \cdot d(p_0, o)\right) \\
&\leq (1+c_1 q)\left(O^o_{0,t'} + O_{t'}/q + O_{t'}\right) \\
&= O(O_{t'}/q).
\end{aligned} \tag{20}$$

Furthermore, using Eq. (18), triangle inequality and a simple lower bound on $A^{\text{LAZY}}_{0,t'}$ as well as Eq. (20), we get,

$$\begin{aligned}
D \cdot d(a, a') &\leq D \cdot d(a, p_0) + D \cdot d(p_0, a') \\
&\leq O_{t'}/q + A^{\text{ONLINE}}_{0,t'} \\
&\leq 2O_{t'}/q.
\end{aligned} \tag{21}$$

Thus, plugging Eq. (21) and Eq. (20) into Eq. (19) and using $A^{\text{ONLINE}} \leq O(O_n)$, we get

$$\begin{aligned}
A^{\text{ROBUST}} &\leq A^{\text{LAZY}}_{t'} + A^{\text{ONLINE}}_{t'+1,n} + D \cdot d(a, a') \\
&= O(O_{t'}/q) + O(O_n) + 2O_{t'}/q \\
&= O(O_n/q).
\end{aligned}$$

Thus, it only remains to prove Eq. (18), as we do using the following lemma. That lemma shows that if ALG moves its page to a location that is far from p_0, then this means that there must be pages that are far from p_0. Later we will show that OPT pays considerable cost to serve them, even if done remotely. See Fig. 5 for an illustration of the lemma.

Lemma 6. *Let $\mathcal{P} = p_1, p_2, \ldots$ be the sequence of page locations that* ALG *produces. Let p_{max} be the furthest point with respect to p_0 a page is moved to by the* ALG*, i.e.,*

$$p_{max} \stackrel{\text{def}}{=} \arg\max_{p_i} d(p_i, p_0).$$

In case that there are several pages at p_{max}, we let p_{max} be the first among them. Let $d_{max} \stackrel{\text{def}}{=} d(p_{max}, p_0)$.

Let P be the maximal consecutive sequence of $\mathcal{P}$ including p_{max} consisting of pages that are each at distance at least $r \stackrel{\text{def}}{=} d_{max}/4$ from p_0. Then, for $q < 1/24$, it holds that the page locations in P serve together at least $6qD$ points at distance r from p_0 in the oracle sequence.

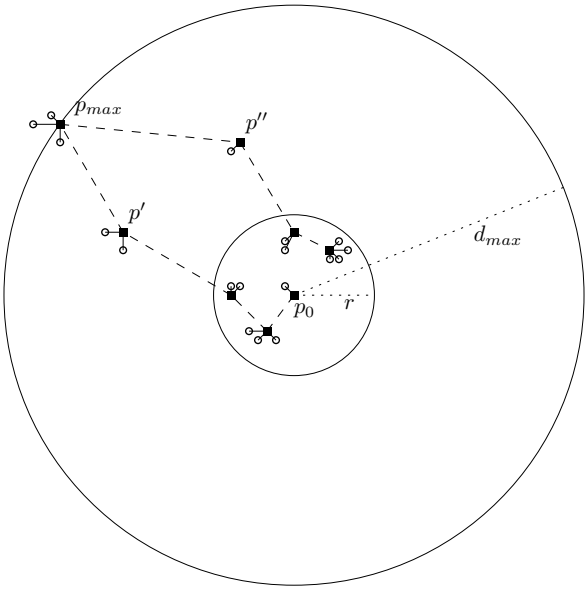


Figure 5: An illustration of Lemma 6, where we argue that the reason we moved a page to a location far away (at distance d_{max}) from p_0 means that there must be many points that are at least at distance $r = d_{max}/4$ from p_0. OPT will have to serve most of these points as well. The squares denote location of pages, the small circles denote page requests, the solid lines between squares and small circles depict a remotely served request. The dashed lines denote the movement of the page. The sequence P consists of p', p_{max} and p''.

Proof. The proof proceeds by contradiction. Suppose that P serves fewer than $6qD$ points in the oracle sequence. We will show that a better solution consists of replacing the sequence P by simply moving to p_0 and serving all points remotely from there. Since P is a maximal sequence of $\mathcal{P}$ including p_{max} such that each page location is at distance r from p_0, ALG moves by at least $d_{max} - r$ within P. Hence, the cost of ALG using the page locations P is at least

$$D(d_{max} - r) + \sum d_i, \tag{22}$$

where the $\sum d_i$ represents the distances to pages served remotely from the page locations in P (depicted as solid lines connected to p, p_{max} and p' in Fig. 5). Consider a request s that is served from location p in the original (using P) solution. In the new solution, where all points are served from p_0, serving any request has, by triangle inequality, a cost of at most $d(p_0, p) + d(p, s) \leq d_{max} + d(p, s)$. Moreover, observe that the sequence P consists of at most $6qD$ locations. This is because otherwise there would be a location that does not serve any points. Putting everything together, the cost of the new solution is at most

$$2Dr + 6qDd_{max} + \sum d_i, \tag{23}$$

where the $2Dr$ accounts for moving the page from the location preceding P to p_0 (the cost of at most Dr) and moving the page from p_0 to the location just after P (also the cost of at most Dr). Recall that $r = d_{max}/4$. Thus, Eq. (23) is cheaper than the solution Eq. (22) for q small enough (i.e., for $q < 1/24$), which contradicts the optimality of ALG of the oracle sequence. □

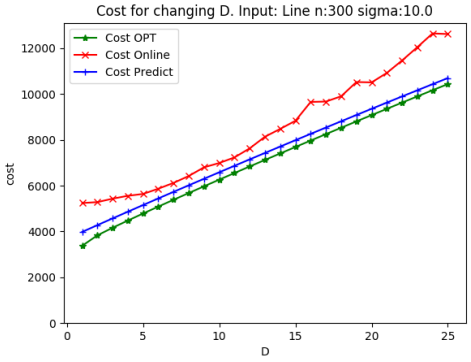
By Lemma 6, we conclude that there are at least $6qD$ points at distance r from p_0 in the oracle sequence. Note that the final sequence s will contain at least $6qD - 2qD$ of these points, due to our assumption on noise and

the fact that up to the first violation of Assumption 1 were detected as time t. OPT has to serve these points as well and thus

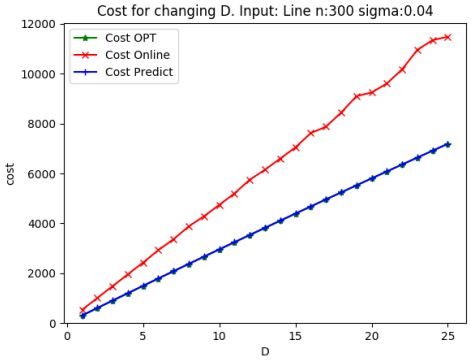
$$O_{t'} \geq (6qD - 2qD)r = 4qD \cdot d_{max}/4 \geq qD \cdot d(a, p_0),$$

which yields Eq. (18) and therefore completes the proof.

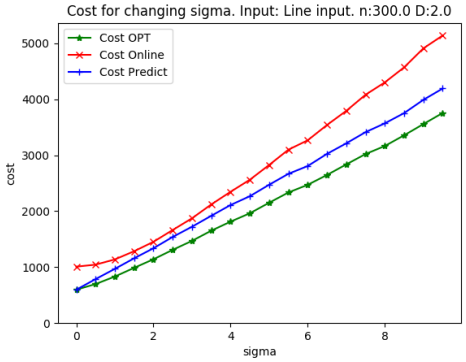
C Additional Experiments



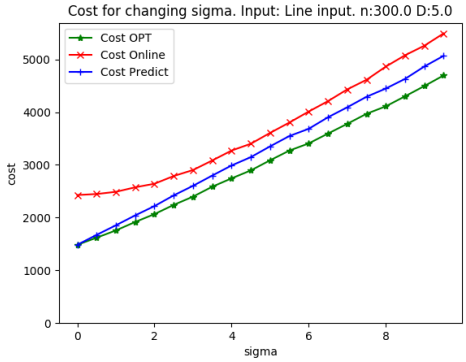
(a) Fixed sigma, varying D.



(b) Fixed sigma, varying D.



(c) Fixed $D = 2$, varying sigma.



(d) Fixed $D = 5$, varying sigma.

Figure 6: Comparison between PREDICT, OPT and ONLINE on Line data set.