

# A 2-Competitive Algorithm For Online Convex Optimization With Switching Costs

Nikhil Bansal<sup>1</sup>, Anupam Gupta<sup>2</sup>, Ravishankar Krishnaswamy<sup>3</sup>,  
Kirk Pruhs<sup>4</sup>, Kevin Schewior<sup>5</sup>, and Cliff Stein<sup>6</sup>

- 1 Department of Mathematics and Computer Science, T. U. Eindhoven. Supported by NWO grant 639.022.211 and an ERC consolidator grant 617951.
- 2 Computer Science Department, Carnegie Mellon University.
- 3 Microsoft Research India. Part of this work was done when the author was at Columbia University and supported by NSF grant CCF-1349602.
- 4 Computer Science Department. University of Pittsburgh. Supported in part by NSF grants CCF-1115575, CNS-1253218, CCF-1421508, and an IBM Faculty Award.
- 5 Department of Mathematics, TU Berlin. Supported by the Deutsche Forschungsgemeinschaft within the research training group ‘Methods for Discrete Structures’ (GRK 1408).
- 6 Dept. of IEOR, Columbia University. Supported in part by NSF grants CCF-1349602 and CCF-1421161.

---

## Abstract

We consider a natural online optimization problem set on the real line. The state of the online algorithm at each integer time  $t$  is a location  $x_t$  on the real line. At each integer time  $t$ , a convex function  $f_t(x)$  arrives online. In response, the online algorithm picks a new location  $x_t$ . The cost paid by the online algorithm for this response is the distance moved, namely  $|x_t - x_{t-1}|$ , plus the value of the function at the final destination, namely  $f_t(x_t)$ . The objective is then to minimize the aggregate cost over all time, namely  $\sum_t (|x_t - x_{t-1}| + f_t(x_t))$ . The motivating application is rightsizing power-proportional data centers. We give a 2-competitive algorithm for this problem. We also give a 3-competitive memoryless algorithm, and show that this is the best competitive ratio achievable by a deterministic memoryless algorithm. Finally we show that this online problem is strictly harder than the standard ski rental problem.

**1998 ACM Subject Classification** F.2.2 Nonnumerical Algorithms and Problems: Sequencing and Scheduling

**Keywords and phrases** Stochastic, Scheduling

**Digital Object Identifier** 10.4230/LIPIcs.xxx.yyy.p

## 1 Introduction

We consider a natural online optimization problem on the real line. The state of the online algorithm after each integer time  $t \in \mathbb{Z}_{\geq 0}$  is a location on the line. At each integer time  $t$ , a convex function  $f_t(x)$  arrives online. In response, the online algorithm moves from its previous location  $x_{t-1} \in \mathbb{R}$  to a new location  $x_t \in \mathbb{R}$ . The cost paid by the online algorithm for this response is the distance moved, namely  $|x_t - x_{t-1}|$ , plus the value of the function at the final destination, namely  $f_t(x_t)$ . The objective is to minimize the aggregate cost  $\sum_t (|x_t - x_{t-1}| + f_t(x_t))$  over all time. We refer to this problem as *Online Convex Optimization with Switching Costs* (OCO). This problem is also referred to as *Smoothed Online Convex Optimization* in the literature.


© Nikhil Bansal and Anupam Gupta and Ravishankar Krishnaswamy and Kirk Pruhs and Kevin and Cliff Stein;

licensed under Creative Commons License CC-BY

Conference title on which this volume is based on.

Editors: Billy Editor and Bill Editors; pp. 1–14

Leibniz International Proceedings in Informatics

 **LIPICS** Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1.1 Motivation and Related Results

The OCO problem has been extensively studied recently, partly due to its application within the context of rightsizing power-proportional data centers, see for example [1, 15, 12, 14, 10, 11, 13]. In these applications, the data center consists of a homogeneous collection of servers/processors that are speed scalable and that may be powered down. The load on the data center varies with time, and at each time the data center operator has to determine the number of servers that will be operational. The standard assumption is that there is some fixed cost for powering a server on, or powering the server off. Most naturally this cost incorporates the energy used for powering up or down, but this cost may incorporate ancillary terms such as the cost of the additional wear and tear on the servers. As for the processor speeds, it is natural to assume that the speed of a processor is scaled linearly with its load (as would be required to maintain a constant quality of service), and that there is a convex function  $P(s)$  that specifies the power consumed as a function of speed. The most commonly used model for  $P(s)$  is  $s^\alpha + \beta$  for constants  $\alpha > 1$  and  $\beta$ . Here the first term  $s^\alpha$  is the dynamic power and the second term  $\beta$  is the static or leakage power. At each time, the state of the online algorithm represents the number of servers that are powered on. In a data center, there are typically sufficiently many servers so that this discrete variable can be reasonably be modeled a continuous one. Then, in response to a load  $L_t$  at time  $t$ , the data center operator decides on a number of servers  $x_t$  to use to handle this load. The algorithm pays a cost of  $|x_{t-1} - x_t|$  for either powering-up or powering-down servers, and a cost of  $x_t((L_t/x_t)^\alpha + \beta)$  for handling the load, which is the most energy efficient way to service the load  $L_t$  using  $x_t$  processors. Note that the function  $x_t((L_t/x_t)^\alpha + \beta)$  is convex in  $x_t$ , and hence this application can be directly cast in our general online model where  $f_t(x) = x((L_t/x)^\alpha + \beta)$ .

Lin et al. [12] observed that the offline problem can be modeled as a convex program, and thus is solvable in polynomial time, and that if the line/states are discretized, then the offline problem can be solved by a straight-forward dynamic program. They also give a 3-competitive deterministic algorithm. The algorithm computes (say via solving a convex program) the optimal solution to date if moving to the left on the line was free, and the optimal solution to date if moving to the right on the line was free, and then moves the least distance possible so that it ends up between the final states of these two solutions. Note that this algorithm solves a (progressively larger) convex program at each time. Andrew et al. [1] show that there is an algorithm with sublinear regret, but that  $O(1)$ -competitiveness and sublinear regret cannot be simultaneously achieved. They also claim that a particular randomized online algorithm, RBG, is 2-competitive, but this claim has been withdrawn [16].

The OCO problem is also related to several classic online optimization problems. It is a special case of the *metrical task system* problem in which the metric is restricted to be a line and the costs are restricted to be convex functions on the real line. The optimal deterministic competitive ratio for a general metrical task system is  $2n - 1$ , where  $n$  is the number of points in the metric [5], and the optimal randomized competitive ratio is  $\Omega(\log n / \log \log n)$  [4, 3], and  $O(\log^2 n \log \log n)$  [8]. The OCO problem is closely related to the *allocation problem* defined in [2], that arises when developing a randomized algorithm for the classic *k-server* problem using tree embeddings of the underlying metric space [7, 2]. In fact, the algorithm RBG in [1] is derived from a similar algorithm in [7] for this *k-server* “subproblem”. The classic *ski rental* problem, where randomized algorithms are allowed, is a special case of the OCO problem. The optimal competitive ratio for randomized algorithms for the ski rental problem is  $e/(e - 1)$  [9] and this translates to a matching lower bound for any online algorithm for the OCO problem. The ski rental problem where only deterministic

algorithms are allowed is a special case of the deterministic version of the OCO problem, and the optimal deterministic competitive ratio for the ski rental problem is exactly 2.

## 1.2 Our Results

**2-Competitive Algorithm.** Our main result, presented in Section 3, is a 2-competitive algorithm (thus we improve the upper bound on the optimal competitive ratio from 3 to 2). It will be convenient to first present a “fractional algorithm”  $\mathcal{A}$  that maintains a probability distribution  $p$  over locations. In Section 2 we show how to convert a fractional algorithm into a randomized algorithm, and how to convert any  $c$ -competitive randomized algorithm into a  $c$ -competitive deterministic algorithm. Although the observation that randomization is not helpful is straight-forward, as best as we can tell, it has not previously appeared in the literature on this problem. The deterministic algorithm that results from these two conversations maintains the invariant that the current location is the expected location given the probability distribution over the states that  $\mathcal{A}$  maintains.

We now describe the fractional algorithm  $\mathcal{A}$ . In response to the arrival of a new function  $f_t(x)$ , the algorithm  $\mathcal{A}$  computes a point  $x_r$  to the right of the minimizer  $x_m$  of  $f_t(x)$  such that the derivative of  $f_t(x_r)$  is equal to twice the total probability mass to the right of  $x_r$ . Similarly the algorithm  $\mathcal{A}$  computes a point  $x_l$  to the left of the minimizer  $x_m$  such that the (negative) derivative of  $f_t(x_l)$  is equal to twice the total probability mass to the left of  $x_l$ . Then, the probability mass at each state  $x \in [x_l, x_r]$  is increased by half the second derivative of  $f_t(x)$  at that point, while the probability mass for each state  $x \notin [x_l, x_r]$  is set to 0. A simple calculation shows that this operation, along with our choices of  $x_l$  and  $x_r$ , preserves the property that  $p$  is a valid probability distribution. One can convert such a probability distribution into a deterministic algorithm by initially picking a random number  $\gamma \in [0, 1]$ , and at any time  $t$ , moving to the state  $x_t$  such that the probability mass to the left of  $x_t$  in the current distribution is exactly  $\gamma$ .

The analysis of  $\mathcal{A}$  uses an amortized local competitiveness argument, using the potential function

$$\Phi(p, x^*) = 2 \int_{y=-\infty}^{\infty} |x^* - y| p(y) dy - \int_{x=-\infty}^{\infty} \int_{y=-\infty}^x p(x) p(y) (x - y) dx dy.$$

where  $x^*$  is the position of the adversary. The first term depends on the expected distance between  $\mathcal{A}$ ’s state and the adversary’s state, and the second term is proportional to the expected distance of two randomly drawn states from  $\mathcal{A}$ ’s probability distribution on states. This potential function can be viewed as a fractional generalization of the potential function used to show that the Double Cover algorithm is  $k$ -competitive for the  $k$ -server problem on a line metric [6].

**3-Competitive Memoryless Algorithm.** Our algorithm  $\mathcal{A}$  requires time and memory roughly proportional to the number of states and/or the number of time steps to date. Similarly, the 3-competitive algorithm from [12], requires solving a convex program (with the entire history) at each time step. However, as pointed out in [12], this may well be undesirable in settings where the data center operator wants to adapt quickly to changes in load. Previously it was not known if  $O(1)$ -competitiveness can be achieved by a “memoryless” algorithm. Intuitively in a memoryless algorithm the next state  $x_t$  only depends upon the previous state  $x_{t-1}$  and the current function  $f_t(x)$ . In Section 4 we show that  $O(1)$ -competitiveness is achievable by a memoryless algorithm — we give a simple memoryless algorithm  $\mathcal{M}$ , and show that it is 3-competitive. Given function  $f_t(x)$  at time  $t$ , this algorithm  $\mathcal{M}$  moves in the direction of the minimizer of  $f_t(x)$  until either it reaches the

minimizer, or it reaches a state where its movement cost equals twice the function cost of this state. The analysis is via an amortized local-competitiveness argument using the distance between the online algorithm's state and the adversary's state (times three) as the potential function.

**Lower Bounds.** In Section 5 we show a matching lower bound of 3 on the competitiveness of any deterministic memoryless online algorithm. We also give a general lower bound of 1.86 on the competitiveness of any algorithm, which shows that in some sense this problem is strictly harder than ski rental, which has an  $e/(e-1)$ -competitive randomized algorithm.

## 2 Reduction From Randomized to Deterministic

In this section, we explain how to convert a probability distribution over locations into randomized algorithm, and present a simple derandomization of any randomized algorithm.

**Converting a Fractional Algorithm into a Randomized Algorithm:** The randomized algorithm initially picks a number  $\gamma \in [0, 1]$  uniformly at random. Then the randomized algorithm maintains the invariant that at each time  $t$  the location  $x_t$  has the property that the probability mass to the left of  $x_t$  in the distribution for the fractional algorithm is exactly  $\gamma$ .

► **Theorem 2.1.** *For the OCO problem, if there is a  $c$ -competitive randomized algorithm  $\mathcal{R}$  then there is a  $c$ -competitive deterministic algorithm  $\mathcal{D}$ .*

**Proof.** Let  $\mathcal{R}$  denote the randomized algorithm, and let  $x_t$  denote the random variable for its position at time  $t$ . Then, our deterministic algorithm  $\mathcal{D}$  sets its location to be the expected location of  $\mathcal{R}$ , i.e., its location at time  $t$  is  $\mu_t := \mathbb{E}[x_t]$ . It is then a simple application of Jensen's inequality to observe that  $\mathcal{D}$ 's cost is at most  $\mathcal{R}$ 's expected cost for each time  $t$ . Indeed, first observe that  $\mathcal{D}$ 's cost at time  $t$  is  $|\mu_t - \mu_{t-1}| + f_t(\mu_t)$ , and  $\mathcal{R}$ 's expected cost is  $\mathbb{E}[|x_t - x_{t-1}|] + \mathbb{E}[f_t(x_t)]$ . Now, notice that both the absolute value function and the function  $f_t(\cdot)$  are convex functions, and therefore  $\mathcal{R}$ 's cost is at least  $|\mathbb{E}[x_t - x_{t-1}]| + f_t(\mathbb{E}[x_t])$ , which is precisely the cost incurred by the algorithm  $\mathcal{D}$ . Summing over all  $t$  completes the proof. ◀

## 3 The Algorithm A and its Analysis

In this section, we describe the online algorithm  $\mathcal{A}$  and prove that it is 2-competitive. For simplicity, we will assume that the functions  $f_t(x)$  are all continuous and smooth. That is, we assume that the first derivative  $f'_t(x)$  and second derivative  $f''_t(x)$  of  $f_t(x)$  are well defined functions. We also assume that  $f_t(x)$  has a unique bounded minimizer  $x_m$ , and  $f'_t(x_m) = 0$ . The assumptions are merely to simplify our presentation; we discharge these assumptions in Section 3.1.

The algorithm  $\mathcal{A}$  was informally described in the introduction, and is more formally described in Figure 1. At any time  $t$ , the state of algorithm  $\mathcal{A}$  is described by a probability distribution  $p_t(x)$  over the possible states  $x$ . So  $\int_a^b p_t(x)dx$  is the probability that  $x_t \in [a, b]$ .

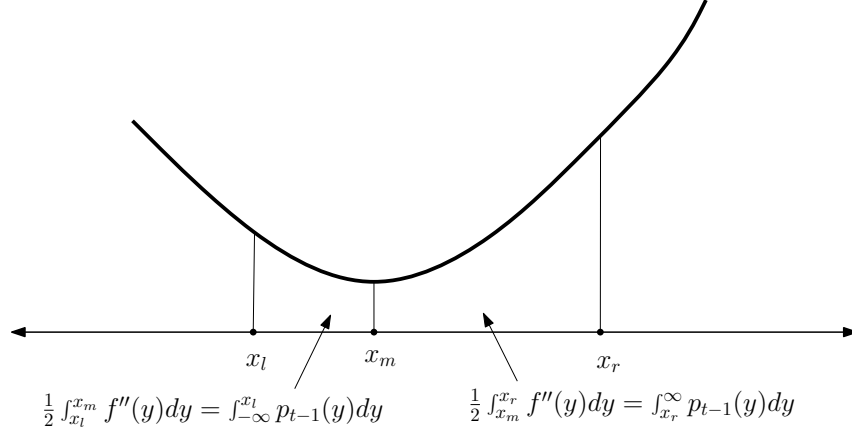
Before beginning our analysis of  $\mathcal{A}$ , let us introduce some notation. Let  $H_t = \mathbb{E}[f_t(x_t)] = \int_{y=-\infty}^{\infty} f_t(y)p_t(y)dy$  denote the *expected hit cost* for algorithm  $\mathcal{A}$  at time  $t$ . Let  $M_t = \mathbb{E}[|x_t - x_{t-1}|]$ , which is equal to the earthmover distance between the two probability distributions<sup>1</sup>, denote the *expected move cost* for algorithm  $\mathcal{A}$  at time  $t$ . Similarly, let  $x_t^*$  be

<sup>1</sup> Given two distributions, where each distribution is viewed as a unit amount of "dirt" piled on the line,

When a new function  $f_t(\cdot)$  arrives:

- (i) Let  $x_m = \operatorname{argmin} f_t(x)$  denote the minimizer of  $f_t$ ,  $x_r \geq x_m$  denote the point to the right of  $x_m$  where  $\frac{1}{2} \int_{x_m}^{x_r} f''(y) dy = \int_{x_r}^{\infty} p_{t-1}(y) dy$ .
- (ii) Let  $x_l \leq x_m$  denote the point to the left of  $x_m$  where  $\frac{1}{2} \int_{x_l}^{x_m} f''(y) dy = \int_{-\infty}^{x_l} p_{t-1}(y) dy$ .
- (iii) We update the probability density function of our online algorithm as  $p_t(x) = p_{t-1}(x) + \frac{1}{2} f''(x)$  for all  $x \in [x_l, x_r]$  and  $p_t(x) = 0$  for all other  $x$ .

■ **Figure 1** The 2-competitive Online Algorithm  $\mathcal{A}$



■ **Figure 2** Illustration of  $x_m, x_l$  and  $x_r$

the adversary's state after time  $t$ . Let  $H_t^* = f_t(x_t^*)$  be the hit cost for the adversary at time  $t$ , and  $M_t^* = |x_t^* - x_{t-1}^*|$  be the movement cost for the adversary at time  $t$ . The analysis will use the potential function:

$$\Phi(p, x_t^*) = \Phi_1(p, x_t^*) + \Phi_2(p)$$

where

$$\Phi_1(p, x_t^*) = 2 \int_{y=-\infty}^{\infty} |x_t^* - y| p(y) dy$$

and

$$\Phi_2(p) = - \int_{x=-\infty}^{\infty} \int_{y=-\infty}^x p(x)p(y)(x-y) dx dy.$$

Note that  $\Phi$  is initially zero. To see that  $\Phi$  is nonnegative, we show that  $\Phi_1(p, x_t^*) \geq -\Phi_2(p)$  as follows:

$$\begin{aligned} -\Phi_2(p) &= \int_{x=-\infty}^{\infty} \int_{y=-\infty}^x p(x)p(y)(x-y) dx dy \\ &= \frac{1}{2} \int_{x=-\infty}^{\infty} \int_{y=-\infty}^{\infty} p(x)p(y)|x-y| dx dy \\ &\leq \frac{1}{2} \int_{x=-\infty}^{\infty} \int_{y=-\infty}^{\infty} p(x)p(y)(|x-x_t^*| + |y-x_t^*|) dx dy \\ &= \frac{1}{2} \left( \int_{x=-\infty}^{\infty} p(x)|x-x_t^*| \int_{y=-\infty}^{\infty} p(y) dx dy + \int_{y=-\infty}^{\infty} p(y)|y-x_t^*| \int_{x=-\infty}^{\infty} p(x) dx dy \right) \end{aligned}$$

---

the earthmover distance (aka Wasserstein metric) is the minimum “cost” of turning one pile into the other, which is the amount of dirt that needs to be moved times the distance it has to be moved.

$$\begin{aligned}
&= \frac{1}{2} \left( \int_{x=-\infty}^{\infty} p(x)|x - x_t^*| dx + \int_{y=-\infty}^{\infty} p(y)|y - x_t^*| dy \right) \\
&= \int_{x=-\infty}^{\infty} p(x)|x - x_t^*| dx \\
&= \frac{1}{2} \Phi_1(p, x_t^*) \leq \Phi_1(p, x_t^*).
\end{aligned}$$

Thus, to prove that  $\mathcal{A}$  is 2-competitive it is sufficient to show that at all times  $t$ :

$$H_t + M_t + (\Phi(p_t, x_t^*) - \Phi(p_{t-1}, x_{t-1}^*)) \leq 2(H_t^* + M_t^*). \quad (1)$$

We first consider the effect on inequality (1) as the adversary moves from  $x_{t-1}^*$  to  $x_t^*$ . The only term which increases in the LHS of inequality (1) is the first term in  $\Phi$ , and this increase is at most  $2|x_{t-1}^* - x_t^*| = 2M_t^*$ , so inequality (1) holds. For the rest of the analysis we consider the effect on inequality (1) when the algorithm  $\mathcal{A}$  moves from  $p_{t-1}$  to  $p_t$ .

To make this easier we make several simplifying assumptions, and simplify our notation slightly. Without loss of generality, we assume that  $f_t(x_m) = 0$  (i.e., the minimum value is 0). Indeed, for general  $f_t$ , we can assume  $g_t(x) = f_t(x) - f_t(x_m)$  and prove the entire analysis for  $g_t$ , and finally add the valid inequality  $f_t(x_m) \leq 2f_t(x_m)$  to inequality (1) for  $g_t$  to get the corresponding inequality for  $f_t$ . (Here we use that the functions  $f_t$  are non-negative.) Also without loss of generality we will translate the points so that  $x_m = 0$ . To further simplify exposition, let us decompose  $f_t$  into two separate functions,  $f_t^>(x)$  and  $f_t^<(x)$ , where the former function is 0 for all  $x \leq x_m$  and  $f_t(x)$  otherwise, and likewise, the latter function is 0 for all  $x \geq x_m$  and  $f_t(x)$  otherwise. It is easy to see that  $f_t(x) = f_t^<(x) + f_t^>(x)$  for all  $x$ . Hence, we can imagine that we first feed  $f_t^>(\cdot)$  to the online algorithm, and then feed  $f_t^<(\cdot)$  to the online algorithm, and separately show inequality (1) for each of these functions. Henceforth, we shall assume that we are dealing with the function  $f_t^>(x)$ . Finally we will assume without loss of generality that  $x_m$  is the leftmost point with non-zero probability mass.

For notational simplicity, we avoid overuse of subscripts and superscripts by letting  $d$  denote  $x_r$ ,  $z$  denote  $x_t^*$ ,  $p$  denote the original distribution  $p_{t-1}$ , and  $q$  denote the resultant distribution  $p_t(\cdot)$ . So by the definition of the algorithm  $\mathcal{A}$ , we have in our new notation,  $\int_{x=0}^d \frac{1}{2} f''(x) dx = \int_{x=d}^{\infty} p(x) dx$ . Here are some simple facts used repeatedly in our analysis.

► **Fact 3.1.** For any smooth convex function  $f$ , and any values  $a, b$  and  $c$ ,

$$\int_{x=a}^b (c-x)f''(x) dx = (c-b)f'(b) - (c-a)f'(a) + f(b) - f(a).$$

**Proof.** This is an application of integration by parts. ◀

► **Fact 3.2.**  $\int_d^{\infty} p(x) dx = f'(d)/2$ . And hence,  $\int_{x=0}^d p(x) dx = 1 - f'(d)/2$ .

**Proof.** By the definition of  $\mathcal{A}$ , it is the case that  $\frac{1}{2} \int_0^d f''(x) dx = \int_d^{\infty} p(x) dx$ . Then note that  $\frac{1}{2} \int_0^d f''(x) dx = \frac{1}{2}(f'(d) - f'(0)) = \frac{1}{2}f'(d)$ , where the second equality follows because 0 is the minimizer of  $f$ . ◀

We now proceed bounding the various terms in the inequality (1).

► **Lemma 3.3.** The hit cost  $H_t$  is exactly  $\int_{x=0}^d f(x)p(x) dx + \frac{1}{2} \int_{x=0}^d f(x)f''(x) dx$ .

**Proof.** This follows from the definition of the hit cost, and the following facts: (i)  $f(x) = 0$  if  $x < 0$ , and (ii) the distribution  $q(x)$  is simply  $p(x) + \frac{1}{2}f''(x)$  for  $x \in [0, d]$  and 0 if  $x > d$ . ◀

► **Lemma 3.4.**  $M_t = \int_{x=d}^{\infty} x f(x) dx + \frac{f(d)}{2} - \frac{df'(d)}{2}$ .

**Proof.** We can view the updating of the probability distribution as a two step procedure. First, all the probability mass to the right of  $d$  moves to  $d$ , and then exactly a probability mass of  $\frac{1}{2}f''(x)$  moves from  $d$  to each point  $x \in [0, d]$ . Thus

$$\begin{aligned} M_t &= \int_0^d \frac{1}{2} f''(x)(d-x) dx + \int_d^{\infty} p(x)(x-d) dx \\ &= \frac{f(d)}{2} + \int_d^{\infty} xp(x) dx - \frac{df'(d)}{2} \end{aligned}$$

Here we used Fact 3.1 to simplify the first term, and Fact 3.2 to simplify the second term. ◀

► **Lemma 3.5.**  $\Phi_1(q, z) - \Phi_1(p, z) \leq 2f(z) - 2M_t$ .

**Proof.** First consider the case that  $z < d$ .

$$\begin{aligned} \Phi_1(q, z) - \Phi_1(p, z) &= 2 \int_0^{\infty} |x-z|(q(x) - p(x)) dx \\ &= \int_0^z (z-x)f''(x) dx + \int_z^d (x-z)f''(x) dx - 2 \int_d^{\infty} (x-z)p(x) dx \\ &= 2f(z) - f(d) - (z-d)f'(d) - 2 \int_d^{\infty} xp(x) dx + 2z \int_d^{\infty} p(x) dx \\ &= 2f(z) - f(d) - (z-d)f'(d) - 2 \int_d^{\infty} xp(x) dx + zf'(d) \\ &= 2f(z) - f(d) + df'(d) - 2 \int_d^{\infty} xp(x) dx \\ &= 2f(z) - 2M_t \end{aligned}$$

The first equality is by the definition of  $\Phi_1$ . The second equality is by the definition of the algorithm  $\mathcal{A}$ . The third equality is by application of Fact 3.1 and separating the last integral. The fourth equality is by Fact 3.2. The final equality is uses Theorem 3.4.

Now consider that case that  $z \geq d$ .

$$\begin{aligned} \Phi_1(q, z) - \Phi_1(p, z) &= 2 \int_0^{\infty} |x-z|(q(x) - p(x)) dx \\ &= \int_0^d (z-x)f''(x) dx - 2 \int_d^z (z-x)p(x) dx - 2 \int_z^{\infty} (x-z)p(x) dx \\ &= (z-d)f'(d) + f(d) - 2 \int_d^z (z-x)p(x) dx - 2 \int_z^{\infty} (x-z)p(x) dx \\ &= (z-d)f'(d) + f(d) - 2 \int_d^{\infty} (x-z)p(x) dx - 4 \int_d^z (z-x)p(x) dx \\ &\leq (z-d)f'(d) + f(d) - 2 \int_d^{\infty} (x-z)p(x) dx \\ &= (z-d)f'(d) + f(d) - 2 \int_d^{\infty} xp(x) dx + 2 \int_d^{\infty} zp(x) dx \\ &= -df'(d) + f(d) - 2 \int_d^{\infty} xp(x) dx + 2zf'(d) \\ &\leq 2f(z) - f(d) + df'(d) - 2 \int_d^{\infty} xp(x) dx \\ &= 2f(z) - 2M_t \end{aligned}$$

The first equality is by the definition of  $\Phi_1$ . The second equality is by the definition of the algorithm  $\mathcal{A}$ . The third equality is an application of integration by parts. The fourth equality follows from replacing the term  $2 \int_z^\infty (x-z)p(x) dx$  by  $2 \int_d^\infty (x-z)p(x) dx - 2 \int_d^z (x-z)p(x) dx$ . The first inequality from the fact that  $\int_d^z (z-x)p(x) dx \geq 0$  since  $z \geq d$ . The sixth equality uses Fact 3.2. The second inequality holds because, as  $f$  is convex,  $f(z) \geq f(d) + (z-d)f'(d)$ , and hence  $zf'(d) \leq f(z) - f(d) + df'(d)$ . The final equality is uses Theorem 3.4.  $\blacktriangleleft$

We now turn to analyzing  $\Phi_2(q) - \Phi_2(p)$ . We can express this as:

$$\begin{aligned}
& - \int_{x=0}^d \int_{y=0}^x (x-y) \left( p(x) + \frac{1}{2} f''(x) \right) \left( p(y) + \frac{1}{2} f''(y) \right) dy dx \\
& + \int_{x=0}^\infty \int_{y=0}^x (x-y) p(x) p(y) dy dx \\
& = - \underbrace{\frac{1}{4} \int_{x=0}^d \int_{y=0}^x (x-y) f''(x) f''(y) dy dx}_{T_1} - \underbrace{\frac{1}{2} \int_{x=0}^d \int_{y=0}^x (x-y) p(x) f''(y) dy dx}_{T_2} \\
& - \underbrace{\frac{1}{2} \int_{x=0}^d \int_{y=0}^x (x-y) f''(x) p(y) dy dx}_{T_3} + \underbrace{\int_{x=d}^\infty \int_{y=0}^x (x-y) p(x) p(y) dy dx}_{T_4} \tag{2}
\end{aligned}$$

We now bound the terms  $T_1$ ,  $T_2$ ,  $T_3$  and  $T_4$ .

► **Lemma 3.6.**  $T_1 = \frac{1}{4} \int_0^d f(x) f''(x) dx$

**Proof.** This follows by applying Fact 3.1 to the inner integral of  $T_1$ .  $\blacktriangleleft$

► **Lemma 3.7.**  $T_2 = \frac{1}{2} \int_0^d f(x) p(x) dx$ .

**Proof.** This follows by applying Fact 3.1 to the inner integral of  $T_2$ .  $\blacktriangleleft$

► **Lemma 3.8.**  $T_3 = -\frac{f'(d)}{2} \int_{x=0}^d x p(x) dx + \left( \frac{df'(d)}{2} - \frac{f(d)}{2} \right) \left( 1 - \frac{f'(d)}{2} \right) + \frac{1}{2} \int_{x=0}^d f(x) p(x) dx$ .

**Proof.**

$$\begin{aligned}
T_3 &= \frac{1}{2} \int_{x=0}^d \int_{y=0}^x (x-y) p(y) f''(x) dy dx \\
&= \frac{1}{2} \int_{y=0}^d \int_{x=y}^d (x-y) p(y) f''(x) dx dy \\
&= -\frac{1}{2} \int_{y=0}^d p(y) \int_{x=y}^d (y-x) f''(x) dx dy \\
&= -\frac{1}{2} \int_{y=0}^d p(y) [(y-d)f'(d) + f(d) - f(y)] dy \\
&= -\frac{f'(d)}{2} \int_{y=0}^d y p(y) dy + \left( \frac{df'(d)}{2} - \frac{f(d)}{2} \right) \int_{y=0}^d dy + \frac{1}{2} \int_{y=0}^d f(y) p(y) dy \\
&= -\frac{f'(d)}{2} \int_{x=0}^d x p(x) dx + \left( \frac{df'(d)}{2} - \frac{f(d)}{2} \right) \left( 1 - \frac{f'(d)}{2} \right) + \frac{1}{2} \int_{x=0}^d f(x) p(x) dx
\end{aligned}$$

The second equality follows as the order of integration is just reversed. The fourth equality is an application of Fact 3.1. The last equality uses Fact 3.2.  $\blacktriangleleft$



► **Lemma 3.9.**  $T_4 \leq \int_d^\infty xp(x) dx - \frac{f'(d)}{2} \int_0^d xp(x) dx - \frac{df'(d)^2}{4}$ .

**Proof.**

$$\begin{aligned} T_4 &= \int_{x=d}^\infty \int_{y=0}^x (x-y)p(x)p(y) dy dx \\ &= \int_{x=d}^\infty \int_{y=0}^d (x-y)p(x)p(y) dx dy + \int_{y=d}^\infty \int_{x=y}^\infty (x-y)p(x)p(y) dx dy \end{aligned} \quad (3)$$

The first expression in (3) can be rewritten as

$$\begin{aligned} \int_{x=d}^\infty \int_{y=0}^d (x-y)p(x)p(y) dx dy &= \int_{x=d}^\infty xp(x) dx \int_{y=0}^d p(y) dy - \int_{x=d}^\infty p(x) dx \int_{y=0}^d yp(y) dy \\ &= \left(1 - \frac{f'(d)}{2}\right) \int_{x=d}^\infty xp(x) dx - \frac{f'(d)}{2} \int_{y=0}^d yp(y) dy \end{aligned}$$

The second equality follows by Fact 3.2. Similarly, for the second expression in (3), we get

$$\begin{aligned} \int_{y=d}^\infty \int_{x=y}^\infty (x-y)p(x)p(y) dx dy &\leq \left(\int_{y=d}^\infty p(y) dy\right) \int_{x=d}^\infty (x-d)p(x) dx \\ &= \left(\int_{y=d}^\infty p(y) dy\right) \int_{x=d}^\infty xp(x) dx - d \int_{x=d}^\infty \int_{y=d}^\infty p(y)p(x) dy dx \\ &= \frac{f'(d)}{2} \int_{x=d}^\infty xp(x) dx - \frac{df'(d)^2}{4} \end{aligned}$$

Here, the inequality uses  $(x-y) \leq (x-d)$ , since  $y \geq d$ , and the last equality uses Fact 3.2 again. Summing the expressions (and replacing the variable  $y$  by  $x$ ) completes the proof. ◀

We now use Theorems 3.3 to 3.9 to show that inequality (1) holds as follows:

$$\begin{aligned} &H_t + M_t + \Phi(p_t) - \Phi(p_{t-1}) \\ &\leq \int_{x=0}^d f(x)p(x) dx + \frac{1}{2} \int_{x=0}^d f(x)f''(x) dx + 2f(z) - \int_{x=d}^\infty xf(x) dx - \frac{f(d)}{2} + \frac{df'(d)}{2} \\ &\quad - \frac{1}{4} \int_0^d f(x)f''(x) dx - \frac{1}{2} \int_0^d f(x)p(x) dx \\ &\quad + \frac{f'(d)}{2} \int_{x=0}^d xp(x) dx - \left(\frac{df'(d)}{2} - \frac{f(d)}{2}\right) \left(1 - \frac{f'(d)}{2}\right) - \frac{1}{2} \int_{x=0}^d f(x)p(x) dx \\ &\quad + \int_d^\infty xp(x) dx - \frac{f'(d)}{2} \int_0^d xp(x) dx - \frac{df'(d)^2}{4} \\ &= 2f(z) + \frac{1}{4} \int_0^d f(x)f''(x) dx - \frac{f(d)f'(d)}{4} \\ &= 2f(z) + \frac{1}{4} \left( f(d)f'(d) - \int_{y=0}^d (f'(y))^2 dy \right) - \frac{f(d)f'(d)}{4} \\ &\leq 2f(z) \end{aligned}$$

The first equality follows by canceling identical terms. The second equality is an application of integration by parts. This proves inequality (1) and hence the 2-competitiveness of our algorithm.

### 3.1 Discharging the Assumptions

We now explain how to modify the algorithm and analysis if some of our simplifying assumptions do not hold. If the functions are piecewise linear, then in the algorithm we can suitably discretize the integral into a summation, and replace the second derivative at a point by the difference in slopes between consecutive points and increase the probability at each point by this difference amount. The analysis then goes through mostly unchanged. If the minimizer is at infinity, then the analysis goes through pretty much unchanged except that we can not translate so that the minimizer is at 0, and we have to explicitly keep  $x_m$  instead of 0 in the limits of the integration.

## 4 Memoryless Algorithm

In this section we present a simple 3-competitive memoryless algorithm  $\mathcal{M}$ . The action of  $\mathcal{M}$  at time  $t$  depends only upon the past state  $x_{t-1}$  and the current function  $f_t(x)$ . The algorithm  $\mathcal{M}$  is described informally in the introduction, and more formally in Figure 3. We adopt the same notation from the previous section using  $x_t$  and  $x_t^*$  to denote the locations of the algorithm and of the adversary, using  $H_t^*$  and  $M_t^*$  to denote the move and hit cost for the adversary, and we remove the expectations from the algorithm's costs, so now  $H_t = f_t(x_t)$  and  $M_t = |x_t - x_{t-1}|$ ,

When a new function  $f_t(\cdot)$  arrives:

- (i) Let  $x_m = \operatorname{argmin} f_t(x)$  denote the minimizer of  $f_t$ .
- (ii) Move in the direction of  $x_m$  until we reach either (a) a point  $x$  s.t.  $|x - x_{t-1}| = f_t(x)/2$  or (b) the minimizer  $x_m$ . Whichever happens first, set  $x_t$  to be that point.

■ **Figure 3** The 3-competitive Memoryless Algorithm  $\mathcal{M}$

► **Theorem 4.1.** *The online algorithm  $\mathcal{M}$  is 3-competitive for the ACO problem.*

**Proof.** We use the potential function  $\Phi(x, x^*) = 3|x - x^*|$ . Clearly  $\Phi$  is initially zero, and always nonnegative. Thus it will be sufficient to show that for each time step:

$$H_t + M_t + (\Phi(x_t, x_t^*) - \Phi(x_{t-1}, x_{t-1}^*)) \leq 3(H_t^* + M_t^*). \quad (4)$$

Two simple observations are that if  $x_{t-1} = x_m$  then the algorithm does not move and, secondly that for all  $t$ ,  $M_t \leq H_t/2$ . We now argue that equation (4) always holds. Indeed, we can upperbound the change in potential by first making the adversary move and then moving the algorithm's point. Using the triangle inequality and the definition  $M_t^* = |x_t^* - x_{t-1}^*|$ ,

$$\Phi(x_{t-1}, x_t^*) - \Phi(x_{t-1}, x_{t-1}^*) \leq 3M_t^* \quad . \quad (5)$$

Therefore, we will assume that the optimal solution has already moved to  $x_t^*$ , and show that

$$H_t + M_t + \Phi(x_t, x_t^*) - \Phi(x_{t-1}, x_t^*) \leq 3H_t^* \quad . \quad (6)$$

Adding equation (5) and equation (6) gives us equation (4), completing the proof. To establish eq. (6) we now consider two cases, based on the relative values of  $H_t$  and  $H_t^*$ .

Case 1: Suppose that  $H_t \leq H_t^*$ . We upper bound the change in potential from the algorithm moving by  $3M_t$  (again using the triangle inequality) and using the fact that  $M_t \leq H_t/2$ , and the inequality defining the case to obtain:

$$H_t + M_t + (\Phi(x_t, x_t^*) - \Phi(x_{t-1}, x_t^*)) \leq H_t + H_t/2 + 3M_t \leq 3H_t \leq 3H_t^*.$$

Case 2: Suppose that  $H_t > H_t^*$ . In this case, all of the algorithm's movement must have been towards  $x_t^*$ , since it was moving in the direction of decreasing value but did not reach  $x_t^*$ . Thus, the algorithm's movement must *decrease* the potential function by  $3M_t$ . Furthermore, since the algorithm is not at  $x_m$ , it must be the case that  $M_t = H_t/2$ . We therefore have

$$H_t + M_t + (\Phi(x_t, x_t^*) - \Phi(x_{t-1}, x_t^*)) \leq H_t + H_t/2 - 3M_t \leq 0 \leq 3H_t^*.$$

This completes the proof. ◀

## 5 Lower Bounds

We first show that no memoryless deterministic algorithm can be better than 3-competitive. We then show that the competitive ratio of every algorithm is at least 1.86.

### 5.1 Lower Bound for Memoryless Algorithms

We show that no memoryless deterministic algorithm  $B$  can be better than 3-competitive. The first issue is that the standard definition of memorylessness, that the next state only depends on the current state and the current input, is problematic for the OCO problem. Because the state is a real number, any algorithm can be converted into an algorithm in which all the memory is encoded in the very low order bits of the current state, and is thus memoryless under this standard definition. Intuitively we believe that the notion of memoryless for the setting of OCO should mean that the algorithm's responses don't depend on the scale of the line (e.g. whether distance is measured in meters or kilometers), and the algorithm's responses are bilaterally symmetric (so the algorithm's response would be the mirror of its current response if the function and the location were mirrored around the function minimizer). We formalize this in the setting that all functions are "vee-shaped", that is they have the form  $f_t(x) = a|x - b|$  for some constants  $a \geq 0$  and  $b$ . Our lower bound only uses such functions. In this setting, say that an algorithm is memoryless if the ratio of the distance that algorithm moves to the distance from the previous location to the minimizer, namely  $(|x_t - x_{t-1}|/|x_{t-1} - b|)$  depends only on  $a$ , the slope of the vee-shaped function. We can assume without any real loss of generality that a memoryless algorithm always moves towards the minimizer, as any algorithm without this property cannot be  $O(1)$ -competitive.

Assume that the initial position is the origin of the line. The first function that arrives is  $\varepsilon|x - 1|$  for some small slope  $\varepsilon$ . We consider two cases. In the first case, assume that the distance  $\delta$  that  $B$  moves is less than  $\varepsilon/2$ . Thus  $B$ 's hit cost is at least  $\varepsilon(1 - \delta) \geq \varepsilon(1 - \varepsilon/2) = \varepsilon - \varepsilon^2/2$ . In that case we continue bringing in copies of the function  $\varepsilon|x - 1|$ . By the definition of memorylessness,  $B$  will maintain the invariant that the ratio of its hit cost to its movement cost is  $(\varepsilon(1 - \delta))/\delta \geq 2 - \varepsilon$ . This continues until  $B$  gets very close to 1. Thus  $B$ 's move cost is asymptotically 1, and its hit cost is at least  $2 - \varepsilon$ . Thus  $B$ 's cost is asymptotically 3. A cost of 1 is achievable by moving to the state 1 when the first function arrives.

Now consider the case that the distance  $\delta$  that  $B$  moves in response to the first function is more than  $\varepsilon/2$ . In this case we bring many copies of the function  $\varepsilon|x|$ , until  $B$  has returned

to very near the origin. Thus  $B$ 's movement cost is approximately  $2\delta$ . By our assumption of memorylessness,  $x_t = \delta(1 - \delta)^{t-1}$ . Thus  $B$ 's hit cost is asymptotically

$$\varepsilon(1 - \delta) + \sum_{t=2}^{\infty} \varepsilon\delta(1 - \delta)^{t-1} = 2\varepsilon(1 - \delta).$$

Thus  $B$ 's total cost is at least  $2\varepsilon + 2\delta(1 - \varepsilon)$ . Using the fact that  $\delta \geq \varepsilon/2$  in this case,  $B$ 's cost is at least  $3\varepsilon - 2\varepsilon^2$ . A cost of  $\varepsilon$  is achievable by never leaving state 0.

## 5.2 General Lower Bound

We now prove a lower bound on the competitive ratio of any online algorithm.

► **Theorem 5.1.** *There is no  $c$ -competitive algorithm for the OCO problem when  $c < 1.86$ .*

**Proof.** By Lemma 2.1, we can restrict to deterministic algorithms without loss of generality. Let  $\mathcal{O}$  be an arbitrary  $c$ -competitive deterministic algorithm.

We now define our adversarial strategy. The initial position is 0. Then some number of functions of the form  $\varepsilon|1 - x|$  arrive. We will be interested in the limit as  $\varepsilon$  approaches 0. Then some number, possibly none, of functions of the form  $\varepsilon|x|$  arrive.

For the deterministic algorithm, let  $b(s)$  denote the position of  $\mathcal{O}$  after  $s/\varepsilon$  functions of the type  $\varepsilon|1 - x|$  have arrived. Intuitively, if  $b(s)$  is too small for large enough  $s$ , then it has a high hit cost on the first  $s/\varepsilon$  functions whereas the optimal solution would have moved immediately to the point 1 only incurring the moving cost. Alternately, if the position  $b(s)$  is sufficiently far to the right (i.e., close to 1), then the adversary can introduce a very long sequence of requests of type  $\varepsilon|x|$ , forcing the algorithm to eventually move back to 0 incurring the movement cost of  $b(s)$  again. In this case, the optimal solution would have stayed at 0.

Formally, the total function cost  $\mathcal{O}$  at time  $s/\varepsilon$  is at least  $b(s) + \int_0^s (1 - b(y)) dy$ . Now, if the adversary introduces an infinite sequence of functions of the form  $\varepsilon|x|$ , then the best that the online algorithm can do is to move immediately to the origin incurring an additional movement cost of  $b(s)$ . Meanwhile, the optimal solution would have stayed at 0 throughout incurring a total cost of  $s$ . Hence, if the online algorithm is  $c$ -competitive, we must have, for all  $s$ ,

$$2b(s) + \int_0^s (1 - b(y)) dy \leq cs. \quad (7)$$

Alternately, if the functions  $\varepsilon|1 - x|$  keep appearing forever, the online algorithm eventually moves to 1 and its total cost is therefore at least  $1 + \int_0^\infty (1 - b(y)) dy$  and the optimal solution would have moved to 1 at time 0 and only incurred the movement cost of 1. Hence, we also have

$$1 + \int_0^\infty (1 - b(y)) dy \leq c. \quad (8)$$

This establishes the dichotomy using which we complete our lower bound proof. Indeed, define  $G(s) = \int_0^s (1 - b(y)) dy$ . Then,  $G'(s) = 1 - b(s)$  and we can write (7) as, for all  $s$  we have

$$G'(s) \geq \frac{1}{2} (2 - cs + G(s)) \quad (9)$$

and (8) is simply

$$G(\infty) \leq c - 1.$$

Now, notice that in order to minimize  $G(\infty)$ , we may assume that (9) is satisfied at equality for all  $s$  (this can only reduce  $G(s)$ , which in turn reduces  $G'(s)$  further), which in turn gives us a unique solution to  $G$ .

Now, writing (9) as equality and differentiating w.r.t  $s$ , we get the first-order differential equation  $b(s) = 2b'(s) - c + 1$ . It is a simple calculation to verify that its unique solution satisfying  $b(0) = 0$  is  $b(s) = (c - 1) \cdot (e^{s/2} - 1)$ . But now, we can plug this into  $G(\infty)$  to get that

$$\int_0^{2 \ln \frac{c}{c-1}} (1 - b(s)) ds + 1 = \int_0^{2 \ln \frac{c}{c-1}} \left(1 - (c - 1) (e^{s/2} - 1)\right) ds + 1 \leq c.$$

Evaluation of the integral and simplification yields

$$2 \ln \frac{c}{c-1} - (c - 1) \left( \frac{2c}{c-1} - 2 \ln \frac{c}{c-1} - 2 \right) + 1 = 2c \ln \frac{c}{c-1} - 1 \leq c,$$

which is false for  $c < 1.86$ . ◀

We conjecture that the optimal competitive ratio for the general problem is strictly less than 2, and is achieved for the special case where all functions are of the form  $\varepsilon|x|$  or  $\varepsilon|x - 1|$ . It is implausible that our lower bound for this special case is tight. Intuitively, the optimal competitive ratio would be 2 if and only if the optimally competitive algorithm doesn't accelerate the rate of probability mass transfer, whereas it seems beneficial to accelerate the rate of probability mass transfer.

## Acknowledgements

We thank Adam Wierman for his assistance and for many stimulating conversations. We thank Neal Barcelo and Michael Nugent for their assistance with the general lower bound.

---

## References

- 1 Lachlan L. H. Andrew, Siddharth Barman, Katrina Ligett, Minghong Lin, Adam Meyerson, Alan Roytman, and Adam Wierman. A tale of two metrics: Simultaneous bounds on competitiveness and regret. In *COLT 2013 - The 26th Annual Conference on Learning Theory, June 12-14, 2013, Princeton University, NJ, USA*, pages 741–763, 2013.
- 2 Nikhil Bansal, Niv Buchbinder, and Joseph Naor. Towards the randomized k-server conjecture: A primal-dual approach. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 40–55, 2010.
- 3 Yair Bartal, Béla Bollobás, and Manor Mendel. Ramsey-type theorems for metric spaces with applications to online problems. *J. Comput. Syst. Sci.*, 72(5):890–921, 2006.
- 4 Yair Bartal, Nathan Linial, Manor Mendel, and Assaf Naor. On metric ramsey-type phenomena. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing, STOC '03*, pages 463–472, New York, NY, USA, 2003. ACM.
- 5 Allan Borodin, Nathan Linial, and Michael E. Saks. An optimal on-line algorithm for metrical task system. *J. ACM*, 39(4):745–763, 1992.

- 6 M. Chrobak, H. Karloff, T. Payne, and S. Vishwanathan. New results on server problems. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '90*, pages 291–300, Philadelphia, PA, USA, 1990. Society for Industrial and Applied Mathematics.
- 7 Aaron Côté, Adam Meyerson, and Laura Poplawski. Randomized k-server on hierarchical binary trees. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, STOC '08*, pages 227–234, New York, NY, USA, 2008. ACM.
- 8 Amos Fiat and Manor Mendel. Better algorithms for unfair metrical task systems and applications. *SIAM J. Comput.*, 32(6):1403–1422, 2003.
- 9 Anna R. Karlin, Mark S. Manasse, Lyle A. McGeoch, and Susan S. Owicki. Competitive randomized algorithms for nonuniform problems. *Algorithmica*, 11(6):542–571, 1994.
- 10 Minghong Lin, Zhenhua Liu, Adam Wierman, and Lachlan L. H. Andrew. Online algorithms for geographical load balancing. In *2012 International Green Computing Conference, IGCC 2012, San Jose, CA, USA, June 4-8, 2012*, pages 1–10, 2012.
- 11 Minghong Lin, Adam Wierman, Lachlan L. H. Andrew, and Eno Thereska. Online dynamic capacity provisioning in data centers. In *2011 49th Annual Allerton Conference on Communication, Control, and Computing, Allerton Park & Retreat Center, Monticello, IL, USA, 28-30 September, 2011*, pages 1159–1163, 2011.
- 12 Minghong Lin, Adam Wierman, Lachlan L. H. Andrew, and Eno Thereska. Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Trans. Netw.*, 21(5):1378–1391, 2013.
- 13 Minghong Lin, Adam Wierman, Alan Roytman, Adam Meyerson, and Lachlan L. H. Andrew. Online optimization with switching cost. *SIGMETRICS Performance Evaluation Review*, 40(3):98–100, 2012.
- 14 Zhenhua Liu, Minghong Lin, Adam Wierman, Steven H. Low, and Lachlan L. H. Andrew. Greening geographical load balancing. In *SIGMETRICS 2011, Proceedings of the 2011 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, San Jose, CA, USA, 07-11 June 2011 (Co-located with FCRC 2011)*, pages 233–244, 2011.
- 15 Kai Wang, Minghong Lin, Florin Ciucu, Adam Wierman, and Chuang Lin. Characterizing the impact of the workload on the value of dynamic resizing in data centers. In *Proceedings of the IEEE INFOCOM 2013, Turin, Italy, April 14-19, 2013*, pages 515–519, 2013.
- 16 Adam Wierman. Personal communication, 2015.