Online Function Tracking with Generalized Penalties*

Marcin Bienkowski[1] and Stefan Schmid[2]

[1] Institute of Computer Science, University of Wrocław, Poland
[2] Deutsche Telekom Laboratories / TU Berlin, Germany

Abstract. We attend to the classic setting where an observer needs to inform a tracker about an arbitrary time varying function $f : \mathbb{N}_0 \to \mathbb{Z}$. This is an optimization problem, where both wrong values at the tracker and sending updates entail a certain cost. We consider an online variant of this problem, i.e., at time t, the observer only knows $f(t')$ for all $t' \leq t$. In this paper, we generalize existing cost models (with an emphasis on concave and convex penalties) and present two online algorithms. Our analysis shows that these algorithms perform well in a large class of models, and are even optimal in some settings.

1 Introduction

Online function tracking has a wide range of applications. For instance, consider a sensor network where a node measures physical properties (e.g., oxygen levels) at a certain location, and needs to report this data to a sink node collecting the measurements of multiple nodes in order to, e.g., raise an alarm if necessary. There is a natural tradeoff between communication and energy costs (how often is the sink informed?) and accuracy (how accurate is the information at the sink?). Function tracking also finds applications in publish/subscribe systems or organization theory where similar tradeoffs exist.

This paper attends to a two-party version of the problem where a node observing a certain function f informs a tracking node. Our main objective is to devise online algorithms for the observing node, which guarantee that the overall cost (sum of update costs and penalties for inaccuracies) is *competitive* — for any possible sequence of function changes — to the cost of an optimal offline algorithm knowing all values of f in advance. This simple two-party instantiation already requires non-trivial solutions [12]. In this paper, we consider an arbitrary function f and different classes of penalty functions.

1.1 Model

We consider a situation where an observer node wants to keep a tracker node informed about a certain function $f : \text{Time } (\mathbb{N}_0) \to \mathbb{Z}$ evolving over time in synchronous time steps (*rounds*). Let $f(t)$ be the actual function value observed

* Supported by MNiSW grants number N N206 2573 35 and N N206 1723 33.

H. Kaplan (Ed.): SWAT 2010, LNCS 6139, pp. 359–370, 2010.

at round t. Let $\textsc{Alg}_t$ denote the value at the tracker at time t, specified by an algorithm Alg; we say that the algorithm is in state $\textsc{Alg}_t$. Initially, at time $t = 0$, $\textsc{Alg}_0 = f(0)$. When the time is clear from the context, we drop the time index.

We study the design of algorithms that allow the observer to inform the tracker about the current values of $f(t)$. In each round t, the following happens:

1. The function f can assume a new arbitrary value $f(t) \in \mathbb{Z}$.
2. The algorithm may change its state $\textsc{Alg}_t$ to any integer paying fixed *update cost* C; otherwise $\textsc{Alg}_t = \textsc{Alg}_{t-1}$.
3. The algorithm pays *penalty* $\Psi(|\textsc{Alg}_t - f(t)|)$, where $\Psi : \mathbb{N}_0 \to \mathbb{N}_0$ is a general function that specifies the cost of a given inaccuracy (e.g., $\Psi(x) = x$).

For succinctness, we abuse notation and will sometimes write $\Psi(x, y)$ meaning $\Psi(|x-y|)$. In this paper, we use the reasonable assumption that $\Psi(x)$ grows monotonically in x, i.e., the penalty cost never decreases for larger errors. Moreover, without loss of generality, we assume that $\Psi(0) = 0$; otherwise, the competitive ratio only improves.

Our main objective is to find an ideal trade-off between *update cost* (informing the tracker about new values) and *penalty cost* (difference between $f(t)$ and $\textsc{Alg}_t$):

$$\begin{aligned} \text{Cost} &= \text{Cost}_{\text{update}} + \text{Cost}_{\text{penalty}} \\ &= C \cdot \sum\nolimits_{t=0}^{T} (\textsc{Alg}_t \neq \textsc{Alg}_{t+1}) + \sum\nolimits_{t=0}^{T} \Psi(\textsc{Alg}_t, f(t)) \ , \end{aligned} \tag{1}$$

where T is the total number of rounds (chosen by the adversary). In other words, our cost function counts the number of updates made by an algorithm and accumulates penalties at the tracker over time. For any input sequence (i.e., the sequence of function changes over time) σ and algorithm Alg, by $\textsc{Alg}(\sigma)$ we denote the cost of Alg on σ.

We assume that at time t, the algorithm only knows the function values $f(t')$ for $t' \leq t$, but has no information about upcoming values. We are in the realm of online algorithms and competitive analysis [3], i.e., we want to compare the performance of an online algorithm Alg with an optimal offline algorithm Opt. An algorithm is ρ-competitive if there exists a constant γ, such that for any input σ, it holds that

$$\textsc{Alg}(\sigma) \leq \rho \cdot \textsc{Opt}(\sigma) + \gamma \ . \tag{2}$$

For a randomized algorithm, we replace the cost of Alg by its expected value and we consider *oblivious* adversaries [3], which do not have access to random bits of the algorithm. For succinctness, we will sometimes use the terminology from the request-answer games [3], saying that in round t a *request* occurred at $f(t)$.

1.2 Related Work

The tradeoff between accuracy and update or transmission cost has challenged researchers from various fields for many years. A classic example of this tradeoff

is known as the TCP acknowledgement problem [6]. In the design of Internet transfer protocols such as the TCP protocol, an important question concerns the times when acknowledgments (ACKs) are sent from the receivers to a sender (to inform about the successful reception of packets). In many protocols, a delay algorithm is employed to acknowledge multiple ACK packets with a single message. The main goal of these protocols is to save bandwidth (and other overhead) while still guaranteeing small delays. Aggregating ACKs has similarities with function tracking as in some sense, the number of to be acknowledged packets can be regarded as the to be tracked function. Karlin et al. [8] gave an optimal $e/(e-1)$-competitive randomized online algorithm for a single link. There are also many variations of the theme, e.g., where the goal is to minimize the maximum delays of the packets [1], to minimize the total time elapsed while packets are waiting at the leaf node [7], to meet fixed deadlines [2] or to find schedules on tree topologies [9,11].

While our model is reminiscent of the TCP acknowledgment problem, there are crucial differences. First of all, we track an arbitrary function f that can both increase and decrease over time, whereas the number of ACKs can only become larger if no message is sent, which means that the to be tracked function is essentially monotonic. A more general aggregation function has already been proposed in [11]; however as there the value at the tracker is updated with a delay, the offline algorithm is unrealistically strong as it can always anticipate function changes and update the values before observing them. We note however that their *offline* solution, running in time quadratic in number of function changes, works also in our model.

In the field of distributed tracking (e.g., [4,5]), a coordinator seeks to keep track of the online inputs distributed over several sites. This problem can be regarded as a generalization of the model studied here. However, these results are still not applicable in our setting. For instance, [4] only considers monotonic functions, and [5] only allows a site to send the current function values, which is trivial in our case.

The closest work to ours is the SODA 2009 paper by Yi and Zhang [12]. In our terminology, they consider a special case with update cost $C = 1$, and the penalty function $\Psi(x) = 0$ for $x \leq \Delta$ and ∞ otherwise (Δ is a fixed constant). They present a deterministic algorithm, which achieves an asymptotically optimal competitive ratio of $\Theta(\log \Delta)$. They also generalize their algorithms to the multidimensional case, i.e., they are able to track functions whose values are integer vectors.

1.3 Our Contributions and Paper Organization

We present a simple online algorithm MED that achieves good competitive ratios for a large class of penalty functions (see Section 2.1). For example, our analysis shows that MED performs particularly well for concave penalty functions, where it achieves a ratio of $\mathcal{O}(\log C / \log \log C)$. This bound is matched for linear penalty functions, for which we show a lower bound of $\Omega(\log C / \log \log C)$ (see

Section 3.1). The same lower bound also holds for randomized algorithms (even against oblivious adversaries).

In Section 2.2, we propose an alternative algorithm SET which is $\mathcal{O}(\log \Delta)$-competitive for convex penalty functions, where $\Delta = \min\{x : \Psi(x) \geq C\}$ (see Section 2.2). This is a generalization of the bound in [12]; in their paper, Ψ can only assume values from $\{0, \infty\}$. We prove that for certain classes of convex functions, this bound is optimal (again, even for randomized algorithms).

Further, we observe that MED behaves well for a class of functions with "bounded growth". In particular, for polynomial penalty functions $\Psi(x) = x^\alpha$, MED is $\mathcal{O}(4^\alpha \cdot \log C / \log\log C)$-competitive and SET is $\mathcal{O}(\max\{1, \frac{1}{\alpha} \cdot \log C\})$-competitive. Thus, by choosing the better of the two algorithms MED and SET, we get a competitive ratio of $\mathcal{O}(\log C / \log\log\log C)$ for all choices of α, i.e., for all polynomial penalty functions.

2 Algorithms

All our algorithms follow the *accumulate-and-update paradigm*: they wait until the total penalty (since the last update) exceeds the threshold $\Theta(C)$ and then they update the value. Henceforth, such a subsequence between two consecutive updates is called a *phase*. In the simplest case, when f is non-decreasing, the problem becomes a discrete variant of the TCP acknowledgement problem [6].

Observation 1. *If f changes monotonically, then the algorithm which updates the value at the end of the phase to the last observed value is 4-competitive.*

The proof is similar to the one presented in [6] and is omitted. However, in the general case, updating always to the last observed value is bad, as the adversary can exploit this strategy.

One may see the choice of the new value as a pursuit of the optimal algorithm: we imagine that both the online as well as the optimal offline algorithm OPT are processing the input in parallel; then the algorithm wants to have a state as close to OPT's state as possible.

Where can OPT be found? A straightforward answer is that its state should be close to the recent requests. Indeed, if the penalty function grows fast at the beginning (e.g., it is concave), OPT has to be relatively close to the requests (otherwise, it accrues a high cost). For such functions, we construct the algorithm MED, which, roughly speaking, changes it state to the median of the recent requests and in this way decreases the distance between its state and the state of OPT. However, if the penalty function is relatively flat at the beginning (e.g., it is convex), then there are many states which are similarly well-suited for the optimal algorithm. In this case, in the construction of our second algorithm, SET, we use an approach which bears some resemblance to the *work function technique* (see, e.g., [10]). Namely, we track a set of states with the property that an algorithm which remains at such states pays little, i.e., the states are potential candidates for OPT. By choosing our position in the middle of such a set, in each phase the cardinality of the set decreases by a constant factor.

The intuitions above are formalized in the upcoming sections.

2.1 Concave Penalties and the Median Strategy

In this section, we present an online algorithm MED pursuing a median strategy and derive an upper bound on its competitive ratio on concave functions and functions of bounded growth. Later, we prove that its competitive ratio is asymptotically optimal for linear penalty functions.

Definition 1 (Growth). *Let $f : \mathbb{N}_0 \to \mathbb{N}_0$ be a monotonic function with $f(0) = 0$. The* growth *of f is defined as* $\max_{x \geq 1}\{f(2x)/f(x)\}$

For example, the growth of any concave function is at most 2. To give another example, $f(x) = c \cdot x^\alpha$ has growth 2^α.

Observation 2 (Triangle Inequality). *Let Ψ be a penalty function of growth at most β. For any three integers a, b, and c, it holds that $\Psi(a,c) \leq \beta \cdot (\Psi(a,b) + \Psi(b,c))$, since $\max\{|a-b|, |b-c|\} \geq |a-c|/2$ and since Ψ is monotonic. Consequently, $\Psi(a,b) \geq \Psi(a,c)/\beta - \Psi(b,c)$.*

The online algorithm MED we introduce here is based on a median strategy. MED partitions the input sequence into *phases*, each phase consisting of several rounds; the first phase starts with the beginning of the input, i.e., $f(0) = \text{MED}_0$. A phase is defined as a period of time, in which MED does not update the tracker but monitors the total penalty paid so far in this phase. Let t_0 be the first round of the current phase. If in one round t, the function $f(t)$ changes abruptly and is far away from MED_{t-1}, i.e., $\Psi(\text{MED}_{t-1}, f(t)) > C$, then MED updates $\text{MED}_t := f(t)$. Otherwise, if the accumulated sum of differences up to the current round would exceed or be equal to C, i.e., $\sum_{i=t_0}^{t} \Psi(\text{MED}_i, f(i)) \geq C$, then $\text{MED}_t := \widetilde{x}$ where $\widetilde{x}$ is the the *median* of of the function values in this phase. (In case of two medians the tie is broken arbitrarily.) In either case, if MED changes its state, the current phase ends and the new begins in the next round.

First, we bound the cost of MED in any phase.

Lemma 1. *Assume that the growth of the penalty function is bounded by β. Consider a phase P and let σ_P be the input sequence of P. Then,* $\text{MED}(\sigma_P) \leq 2 \cdot (\beta + 1) \cdot C$.

Proof. By the definition of MED, its accumulated penalty in all rounds except for the last one is at most C, and MED pays update cost C for changing its state in the last round t. In the case that MED changes its state to the last value $f(t)$, the total cost is $2C$ as no additional penalty accrues. Otherwise, MED updates to the median value $\widetilde{x}$ and in the last round it pays $\Psi(\widetilde{x}, f(t)) \leq \beta \cdot (\Psi(\text{MED}_{t-1}, \widetilde{x}) + \Psi(\text{MED}_{t-1}, f(t)))$. As the median is chosen among all the requests in P and for any $i \in P$, $\Psi(\text{MED}_{t-1}, f(i)) \leq C$, the total cost in the last round is at most $2\beta \cdot C$. □

Next, we turn our attention to OPT. In the following, a phase in which OPT pays less than α is called *α-constrained*. The main idea for proving the competitiveness

of MED is as follows. Phases which are not $\mathcal{O}(C)$-constrained — e.g., phases in which OPT updates — are trivial, as OPT incurs a cost of $\Omega(C)$ in them. On the other hand, in α-constrained phases with small α, the possible distance between OPT and MED becomes smaller: the less OPT pays, the faster the MED's state converges to the state of OPT. We show next that if OPT tries to pay $o(C)$ in a single phase, then after a sequence of $\mathcal{O}(\log C / \log\log C)$ phases, MED's state becomes equal to OPT's state, which entails a OPT cost $\Omega(C)$ in the next phase. This idea is formalized in the two lemmas below.

Lemma 2. *Assume that the growth of the penalty function is at most β. Fix any α-constrained phase P, starting at round t_0 and ending at t_1, for a given $\alpha < C/(3\beta)$. Assume that* OPT *is in state ξ throughout P. Then, it holds that $\Psi(\text{MED}_{t_1}, \xi) \leq 2C$ and*

$$\frac{\Psi(\text{MED}_{t_1}, \xi)}{\Psi(\text{MED}_{t_0}, \xi)} \leq \frac{2 \cdot \alpha}{C/\beta - \alpha} \ .$$

Proof. First, we consider the case that MED updates its state to the last request in P, i.e., $\text{MED}_{t_1} = f(t_1)$. By the definition, $\Psi(\xi, f(t)) \leq \alpha$ for all $t \in \{t_0, t_0+1, ..., t_1\}$. Then, by Observation 2, $\Psi(\text{MED}_{t_0}, \xi) \geq \Psi(\text{MED}_{t_0}, f(t_1))/\beta - \Psi(\xi, f(t_1)) > C/\beta - \alpha$. Finally, $\Psi(\text{MED}_{t_1}, \xi) = \Psi(f(t_1), \xi) \leq \alpha$, and the lemma holds.

Second, we consider the case that P ends with MED updating its state to the median. Since all the requests are at distance at most C from MED_{t_0}, initially $\Psi(\text{MED}_{t_0}, \xi) \leq 2C$ (as otherwise OPT would pay C for each request). Thus, in the following, we show that $\Psi(\text{MED}_{t_1}, \xi)/\Psi(\text{MED}_{t_0}, \xi) \leq 2\alpha/(C/\beta - \alpha)$. As $2\alpha/(C/\beta - \alpha) \leq 1$, this implies both parts of the claim. Let n be the number of rounds in P and let $\widetilde{x}$ be the median of the corresponding n requests, denoted by $x_1, x_2, \ldots, x_n$. By Observation 2, we obtain a lower bound for $\Psi(\text{MED}_{t_0}, \xi)$:

$$\begin{aligned} n \cdot \Psi(\text{MED}_{t_0}, \xi) \ &\geq \ \sum_{i=1}^{n} \left(\frac{1}{\beta} \cdot \Psi(\text{MED}_{t_0}, x_i) - \Psi(\xi - x_i) \right) \\ &= \frac{1}{\beta} \cdot \sum_{i=1}^{n} \Psi(\text{MED}_{t_0}, x_i) - \sum_{i=1}^{n} \Psi(\xi, x_i) \\ &\geq C/\beta - \alpha \ . \end{aligned}$$

Moreover, by the median definition, it follows that at least half of the requests in P are further from ξ than the median is, and thus $(n/2) \cdot \Psi(\xi, \widetilde{x}) \leq \text{OPT}(\sigma_P) \leq \alpha$. This implies that $\Psi(\text{MED}_{t_1}, \xi) = \Psi(\widetilde{x}, \xi) \leq 2\alpha/n$. Comparing $\Psi(\text{MED}_{t_1}, \xi)$ to $\Psi(\text{MED}_{t_0}, \xi)$ immediately yields the lemma:

$$\frac{\Psi(\text{MED}_{t_1}, \xi)}{\Psi(\text{MED}_{t_0}, \xi)} \leq \frac{2\alpha/n}{(C/\beta - \alpha)/n} = \frac{2\alpha}{C/\beta - \alpha} \ . \qquad \square$$

Lemma 3. *Assume that the growth of the penalty function is at most β. There exists $\ell = \Theta(\log C/\log\log C)$, such that in any subsequence τ consisting of consecutive $2\ell+1$ phases,* $\textsc{Opt}(\tau) = \Omega(C/\beta)$.

Proof. Fix any input sequence σ and any contiguous subsequence τ consisting of $2\ell+1$ phases. (The exact value of ℓ is discussed later.)

If Opt changes its state within τ, then the lemma follows trivially. Thus, in the remainder of the proof, we assume that throughout τ, Opt is in state ξ. We look at the prefix τ' of 2ℓ phases of τ (i.e., ignoring the last phase). Let $B = \frac{1}{\beta} \cdot C/\sqrt{\log C}$. We assume that C is sufficiently large, i.e., $B \le C/(3\beta)$. We consider three cases:

1. τ' contains a phase for which Opt pays at least $\frac{1}{3} \cdot C/\beta$. In this case, the claim follows trivially.
2. τ' contains ℓ phases for which Opt pays at least B. Then, $\textsc{Opt}(\tau') \ge \ell \cdot B = \Omega(C/\beta)$.
3. All phases of τ' are $(\frac{1}{3} \cdot C/\beta)$-constrained and at least $\ell+1$ of them are additionally B-constrained. We show that this implies the existence of a phase in τ', at the end of which $\Psi(\textsc{Med}, \xi) = 0$.

   By Lemma 2, we can make three key observations for this case: (1) in all phases of τ', the distance between Med and ξ does not increase; (2) after the first phase of τ', the distance between Med and ξ becomes at most $2C$; (3) in each of the next ℓ B-constrained phases, $\Psi(\textsc{Med}, \xi)$ decreases by a factor of $q := 2B/(C/\beta - B) = \Theta(1/\sqrt{\log C})$. Let $\ell = \log_{1/q}(4C) = \Theta(\log C/\log\log C)$. Thus, at the end of these ℓ phases, $\Psi(\textsc{Med}, \xi)$ decreases to at most $1/2$, i.e., it becomes 0.

   We consider the next phase of τ, during which $\Psi(\textsc{Med}, \xi) = 0$ and we denote the requests in this phase by $x_1, x_2, \ldots, x_n$. By Observation 2, the cost of Opt in this phase is $\sum_{i=1}^{n} \Psi(\xi, x_i) \ge \sum_{i=1}^{n} (\Psi(\textsc{Med}, x_i)/\beta - \Psi(\textsc{Med}, \xi)) \ge C/\beta$ (as by the construction of Med, $\sum_{i=1}^{n} \Psi(\textsc{Med}, x_i) \ge C$). □

Theorem 3. Med *is $\mathcal{O}(\beta^2 \cdot \log C/\log\log C)$-competitive for penalty functions Ψ of growth at most β.*

Proof. Fix any input sequence σ and partition it into subsequences of length $2\ell+1$ phases, where ℓ is as in the proof of Lemma 3. Fix any such subsequence τ. By Lemma 3, $\textsc{Opt}(\tau) = \Omega(C/\beta)$ and by Lemma 1, $\textsc{Med}(\tau) \le (2\ell+1) \cdot 2(\beta+1) \cdot C$. Summing over all the subsequences of σ, we obtain that the competitive ratio is

$$\rho = \mathcal{O}\left(\frac{\beta C}{C/\beta} \cdot \log C/\log\log C\right) = \mathcal{O}(\beta^2 \log C/\log\log C) \ .$$

Finally, we observe that after partitioning σ, we might get a subsequence shorter than $2\ell+1$ at the end. However, by Lemma 1, this contributes only a constant term to the overall cost, and hence it does not influence the competitive ratio (cf. Equation 2). □

2.2 Convex Penalties and the Set Strategy

In the previous section, we have observed that MED performs particularly well for concave penalty functions. We now turn our attention to a different algorithm SET which is inspired by [12]: there, it is shown that such a strategy performs well under "0-or-∞" penalties, which is in some sense an extreme case of convexity. Thus, we seek to generalize the approach of [12] to an entire class of penalty functions, and provide a performance analysis.

The algorithm SET works as follows. First, on the basis of the penalty function Ψ, it computes a parameter $\Delta = \min\{x : \Psi(x) \geq C\}$. We call this value the *C-gap* of Ψ. This means that if an algorithm's state is at distance Δ from the request, then the algorithm pays at least C, and Δ is the smallest distance with this property.

SET keeps track of a set S, centered at its current state, consisting of consecutive integers. At the beginning, $S = [\text{SET}_0 - \Delta, \text{SET}_0 + \Delta] \cap \mathbb{Z}$, where $\text{SET}_0 = f(0)$. Again, in one *phase*, SET remains in the same state. Similarly to the MED algorithm, SET computes the penalties accumulated since the beginning of a phase. If this cost exceeds C, then SET changes its state as described below and a new phase starts.

For any point $x \in S$, SET computes the accumulated penalty of an algorithm A_x which remains at x during the whole phase. Among all points $x \in S$, we choose the leftmost (ℓ) and the rightmost (r) point for which $A_x \leq C/2$. Let S' be the set of all integers in $[\ell, r]$. Now SET distinguishes two cases. If S' is nonempty, then we set $S := S'$, otherwise we choose set S to contain all the integers from range $[z - \Delta, z + \Delta]$, where z is the latest request. In the second case, we say that an *epoch* has ended and a new epoch starts with the next phase. In either case, SET moves to the median of the new set S.

Below, we analyze the performance of the algorithm SET for convex penalty functions. We start with a simple property of set S' chosen at the end of each phase.

Observation 4. *Assume that the penalty function Ψ is convex. Let $S' = \{\ell, \ell + 1, \ldots, r\}$ be the set computed by* SET *at the end of phase P. Then $A_x(P) \leq C/2$ for any $x \in S'$ (and not only for $x \in \{\ell, r\}$).*

Proof. The function of the cumulative penalty over a fixed period is also convex (as the sum of convex functions is convex). This function is bitonic (i.e., first monotonically decreasing and then monotonically increasing), which implies the observation. □

In the lemmas below, we use the above observation, i.e., we assume that Ψ is a convex function.

Lemma 4. *In any phase, the cost of* SET *is at most $5C/2$.*

Proof. As in the proof of Lemma 1, the total penalty for all the requests but the last one is at most C. The cost of changing state is also C. Thus, we have to show that the penalty associated with the last request y is at most $C/2$. If

set S' is non-empty, then SET changes its state to a median of S'. Hence, by Observation 4, the penalty associated with y is at most $C/2$. If S' is empty, then SET changes its state to y, in which case the penalty is zero. □

Lemma 5. *In any two consecutive epochs E_{i-1} and E_i, the cost of* OPT *is at least $C/2$.*

Proof. The lemma follows trivially if OPT changes its state in these epochs, so we assume it does not. Let S_i be the set S of the algorithm SET at the beginning of epoch E_i. If OPT's state is in S_i, then its cost is at least $C/2$ in at least one phase of E_i. Otherwise, the OPT state is outside S_i. Then, we consider the last request of E_{i-1}, which, by the definition of SET is given at the center of S_i, i.e., at a distance of at least $\Delta+1$ from the state of OPT. Thus, the penalty for OPT associated with this request is at least C. □

Theorem 5. *For any convex penalty Ψ with C-gap equal to Δ,* SET *is $\mathcal{O}(\log \Delta)$-competitive.*

Proof. By Lemmas 4 and 5, it suffices to show that the number of phases in a single epoch is at most $O(\log \Delta)$. At the beginning of any epoch, $span(S)$ (defined as the distance between the rightmost and the leftmost point of S) is 2Δ. Fix any phase that is not the last phase in an epoch. Then a set S' chosen at the end is non-empty. Let x be the state of SET at the beginning of this phase. Since $A_x \geq C$, S' cannot contain x, i.e., the median of S. Thus, $span(S') \leq span(S)/2$, which means that $span(S)$ decreases at least by a factor of 2 in each phase. This may happen only $O(\log \Delta)$ times. □

It follows from [12] that this result is asymptotically tight in the following sense: for any Δ there exists a convex penalty function Ψ, so that the competitive ratio of any online algorithm is $\Omega(\log \Delta)$. In Section 3, we will show that this lower bound holds also for randomized algorithms. The function Ψ for which our lower bound holds is any (possibly convex) function satisfying $\Psi(x) = 0$ for $x < \Delta$ and $\Psi(x) \geq C$ otherwise.

Remark. Note that our results are also applicable to convex penalty functions Ψ with the additional hard constraint that the difference between reported and observed value must not exceed T. Convexity was used only to obtain the property guaranteed in Observation 4; however, this property also holds for a function $\Psi(x)$ that is derived from a convex function Ψ' and a threshold T, such that $\Psi(x) = \Psi'(x)$ for $x \leq T$ and $\Psi(x) = \infty$ for $x > T$.

3 Lower Bounds

Next, we show that our algorithms MED and SET are asymptotically optimal in the classes of convex and concave functions, respectively. Note that we are not claiming their optimality for every such function, but for a quite broad subset of them. We emphasize that our lower bound holds even for randomized algorithms against oblivious adversaries.

3.1 Linear Penalties

We prove that our deterministic algorithm MED is asymptotically optimal for linear penalty functions.

Theorem 6. *For a given penalty function $\Psi(x) = a \cdot x$ (for any $a > 0$), the competitive ratio of any randomized algorithm is at least $\Omega(\log C / \log\log C)$.*

To prove this theorem, we employ a standard min-max approach. We fix an arbitrary deterministic online algorithm DET and generate a probability distribution π over input sequences in such a way that if an input sequence σ is chosen according to π, the following conditions hold:

1. $\textsc{Opt}(\sigma) = \mathcal{O}(C)$
2. $\mathbf{E}_\pi[\textsc{Det}(\sigma)] = \Omega(C \cdot \log C / \log\log C)$, where the expectation is taken over the random choice of the input;

Our construction below can be repeated an arbitrary number of times. Along with the second condition above, this ensures that the cost of the algorithm cannot be hidden in the additive constant in the definition of the competitive ratio (see Eq. 2). Then the lower bound for any randomized online algorithm follows immediately by the Yao min-max principle [3].

We now describe how to randomly choose an input σ, that is, we will implicitly create a probability distribution π over input sequences. Let $[a,b]_\mathbb{N}$ be the set $[a,b] \cap \mathbb{N} = \{a, a+1, \ldots, b\}$. Let s be the largest integer i for which $(\log C)^i \le C$; clearly $s = \Theta(\log_{\log C} C) = \Theta(\log C / \log\log C)$. First, we create a sequence of $s+1$ random sets: $R_0 \supseteq R_1 \supseteq R_2 \supseteq \ldots \supseteq R_s$. $R_0 = [0, (\log C)^s - 1]_\mathbb{N}$. The remaining sets are chosen iteratively in the following manner. We partition R_i into $\log C$ disjoint contiguous subsets of the same size; R_{i+1} is chosen uniformly at random among them. For example, R_1 is chosen amongst the following sets: $[0, (\log C)^{s-1} - 1]_\mathbb{N}, [(\log C)^{s-1}, 2 \cdot (\log C)^{s-1} - 1]_\mathbb{N}, \ldots, [(\log C)^s - (\log C)^{s-1}, (\log C)^s - 1]_\mathbb{N}$. Note that the construction implies that R_s contains a single integer. The sequence σ associated with sets $R_0, R_1, \ldots, R_s$ consists of s phases, numbered from 1. In phase i there are $\lceil C / (a \cdot |R_{i-1}|) \rceil$ requests given at the leftmost integer from the set R_i.

Below, we present two lemmas, which directly imply the two conditions above, and thus also Theorem 6.

Lemma 6. *For any initial state of* OPT *and input σ generated in the way described above,* $\textsc{Opt}(\sigma) = O(C)$.

Proof. Let $R_0 \supseteq R_1 \supseteq \ldots \supseteq R_s$ be the sequence of sets associated with σ. Let x be the only element of the set R_s. The strategy for an offline (possibly not optimal) algorithm OFF is to change its state to x at the very beginning of σ and remain there for the whole σ.

Clearly, the update cost is C. In phase i, the distance between x and the requests is at most $|R_i|$, and thus the total penalty paid by OFF is at most $\mathcal{O}(a \cdot |R_i| \cdot \lceil C / (a \cdot |R_{i-1}|) \rceil) = \mathcal{O}(C / \log C)$. Thus, the total cost in the entire sequence is $\textsc{Off}(\sigma) \le C + s \cdot O(C / \log C) = \mathcal{O}(C)$. As $\textsc{Opt}(\sigma) \le \textsc{Off}(\sigma)$, the lemma follows. □

Lemma 7. *For any deterministic online algorithm* DET *and input* σ *generated randomly in the way described above,* $\mathbf{E}_\pi[\text{DET}(\sigma)] = \Omega(C \cdot \log C / \log\log C)$.

Proof. Any sequence σ consists of $s = \Theta(\log C / \log\log C)$ phases. It is therefore sufficient to show that the expected cost of DET in any phase is at least $\Omega(C)$.

We consider the moment at the very beginning of phase i, even before its first request is presented to DET. At that moment, DET knows the set R_{i-1} and is at some fixed state x (not necessarily but possibly from R_{i-1}). When the first request from phase i is revealed to DET, it immediately learns R_i, but it is already too late. For any fixed state x, the expected distance between x and the leftmost point of R_i is at least $\Omega(|R_{i-1}|)$. Thus, DET has two choices. It may change its state, paying C or it may remain at x paying in expectation the total penalty of at least $\Omega(a \cdot \lceil C/(a \cdot |R_{i-1}|)\rceil \cdot |R_{i-1}|) = \Omega(C)$. □

3.2 Convex Penalties

The following theorem shows the asymptotic optimality of the algorithm SET in the class of convex functions. We note that the deterministic variant of this theorem is already known (cf. Theorem 2.1 of [12]) and our proof can be viewed as its adaptation.

Theorem 7. *For any* Δ*, there exists a convex penalty function* Ψ*, whose* C*-gap is* Δ *and the competitive ratio of any randomized algorithm is at least* $\Omega(\log \Delta)$.

Proof. We consider the convex penalty function used in [12], i.e., $\Psi(x) = 0$ for $x \le \Delta$ and $\Psi(x) = \infty$ otherwise. In fact, for our proof to work, we require only that $\Psi(x) = 0$ for $x \le \Delta$ and $\Psi(x) = \Omega(C)$ otherwise.

Our approach is similar in flavor to the proof of Theorem 6, so we just concentrate on the differences. Once again, we randomly construct a family of sets $R_0, R_1, \ldots R_s$. This time $s = \lfloor \log \Delta \rfloor$ and $R_0 = [0, 2^s - 1]_{\mathbb{N}}$. To construct R_i out of R_{i-1}, we divide R_{i-1} into two equal contiguous halves and R_i is chosen randomly among them. The sequence consists now of s rounds, numbered from 1. In round i, the requests are given at x_i, such that the distance between x_i and any point of R_i is at most Δ and the distance between x_i and any point of $R_{i-1} \setminus R_i$ is greater than Δ.

OPT can serve the whole sequence without penalties changing its state to the only integer of R_s at the beginning (paying C for the state change). On the other hand, any deterministic algorithm DET at the beginning of round i knows set R_{i-1}, but does not know which half will be chosen as R_i, and with probability $1/2$ it is in "the wrong half". Thus, with probability $1/2$, when the request is presented to DET, it either has to pay the penalty $\Omega(C)$ or change its state paying C. Hence, the expected cost of DET on such a sequence is $\Omega(C \cdot s) = \Omega(C \log \Delta)$, i.e., the ratio of $\mathbf{E}[\text{DET}]$ divided by OPT is at least $\Omega(\log \Delta)$. Again, using the Yao min-max principle, the same bound even holds for any randomized algorithm (against an oblivious adversary). □

4 Conclusions

This paper studies generalized penalty functions for the problem of approximating a function f (that is revealed gradually) by a piecewise constant function g, where the cost depends on the number of value changes of g plus the error cost summed over the discrete sampling points.

We believe that our work opens several interesting directions for future research. First, our results raise the question whether MED and SET can be combined in order to have the advantages of both worlds in penalty functions beyond concave and convex models. Another research direction is the study of different penalty functions in multi-dimensional tracking $f : \mathbb{N}_0 \to \mathbb{Z}^d$ and the analysis of the gains that can be obtained with the line predictions of [12]. Finally, distributed settings remain to be explored where there are multiple observers at different sites.

References

1. Albers, S., Bals, H.: Dynamic TCP acknowledgement: Penalizing long delays. In: Proc. of the 14th ACM-SIAM Symp. on Discrete Algorithms (SODA), pp. 47–55 (2003)
2. Becchetti, L., Korteweg, P., Marchetti-Spaccamela, A., Skutella, M., Stougie, L., Vitaletti, A.: Latency constrained aggregation in sensor networks. In: Proc. of the 14th European Symp. on Algorithms (ESA), pp. 88–99 (2006)
3. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press, Cambridge (1998)
4. Cormode, G., Muthukrishnan, S., Yi, K.: Algorithms for distributed functional monitoring. In: Proc. of the 19th ACM-SIAM Symp. on Discrete Algorithms (SODA), pp. 1076–1085 (2008)
5. Davis, S., Edmonds, J., Impagliazzo, R.: Online algorithms to minimize resource reallocations and network communication. In: Proc. of the 9th Int. Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX), pp. 104–115 (2006)
6. Dooly, D.R., Goldman, S.A., Scott, S.D.: On-line analysis of the TCP acknowledgment delay problem. Journal of the ACM 48(2), 243–273 (2001)
7. Frederiksen, J.S., Larsen, K.S.: Packet bundling. In: Proc. of the 8th Scandinavian Workshop on Algorithm Theory (SWAT), pp. 328–337 (2002)
8. Karlin, A.R., Kenyon, C., Randall, D.: Dynamic TCP acknowledgement and other stories about e/(e - 1). Algorithmica 36(3), 209–224 (2003)
9. Khanna, S., Naor, J., Raz, D.: Control message aggregation in group communication protocols. In: Proc. of the 29th Int. Colloq. on Automata, Languages and Programming (ICALP), pp. 135–146 (2002)
10. Koutsoupias, E., Papadimitriou, C.H.: On the k-server conjecture. Journal of the ACM 42(5), 971–983 (1995)
11. Pignolet, Y.A., Schmid, S., Wattenhofer, R.: Tight bounds for delay-sensitive aggregation. Discrete Mathematics and Theoretical Computer Science (DMTCS) 12(1) (2010)
12. Yi, K., Zhang, Q.: Multi-dimensional online tracking. In: Proc. of the 19th ACM-SIAM Symp. on Discrete Algorithms (SODA), pp. 1098–1107 (2009)