Regret Minimization for Online Buffering Problems Using the Weighted Majority Algorithm*

Sascha Geulen
Dept. of Computer Science,
RWTH Aachen University
sgeulen@cs.rwth-aachen.de

Berthold Vöcking
Dept. of Computer Science,
RWTH Aachen University
voecking@cs.rwth-aachen.de

Melanie Winkler
Dept. of Computer Science,
RWTH Aachen University
winkler@cs.rwth-aachen.de

Abstract

Suppose a decision maker has to purchase a commodity over time with varying prices and demands. In particular, the price per unit might depend on the amount purchased and this price function might vary from step to step. The decision maker has a buffer of bounded size for storing units of the commodity that can be used to satisfy demands at later points in time. We seek for an algorithm deciding at which time to buy which amount of the commodity so as to minimize the cost. This kind of problem arises in many technological and economical settings like, e.g., battery management in hybrid cars and economical caching policies for mobile devices. A simplified but illustrative example is a frugal car driver thinking about at which occasion to buy which amount of gasoline.

Within a regret analysis, we assume that the decision maker can observe the performance of a set of expert strategies over time and synthesizes the observed strategies into a new online algorithm. In particular, we investigate the external regret obtained by the well-known Randomized Weighted Majority algorithm applied to our problem. We show that this algorithm does not achieve a reasonable regret bound if its random choices are independent from step to step, that is, the regret for T steps is $\Omega(T)$. However, one can achieve regret $O(\sqrt{T})$ when introducing dependencies in order to reduce the number of changes between the chosen experts. If the price functions satisfy a convexity condition then one can even derive a deterministic variant of this algorithm achieving regret $O(\sqrt{T})$.

Our more detailed bounds on the regret depend on the buffer size and the number of available experts. The upper bounds are complemented by a matching lower bound on the best possible external regret.

*Supported by the DFG GK/1298 "AlgoSyn" and UMIC Research Center

1 Introduction

We study online buffering problems dealing with the management of a storage or buffer for a commodity with varying prices and demands. Our setting is similar to the standard model for regret minimization. In particular, we assume that there is a number of experts corresponding to different strategies for managing the buffer. An online learning algorithm observes the performance of the experts and combines their policies with the objective to achieve a performance close to the performance of the best expert. The buffer makes the problem different from the standard setup in online learning since the algorithm now has a state. Switching between experts means switching between states and this might be costly. Indeed, this switching cost is the major challenge in applying known learning algorithms like the algorithms *Randomized Weighted Majority (RWM)* by Littlestone and Warmuth [16] and *Follow the Perturbed Leader (FPL)* by Kalai and Vempala [14] as these algorithms switch between experts frequently.

Informally, the online problems under consideration can be described as follows. A decision maker has to purchase a commodity over time. Time proceeds in discrete steps. In every step, the decision maker needs to satisfy a demand depending on environmental conditions that are not under control of the decision maker. A price function that might vary from step to step describes at which cost the decision maker can buy which amount of the commodity. The price per unit might be constant or depend on the amount purchased. The decision maker has a buffer of bounded size for storing units of the commodity that can be used to satisfy demands at later points in time. We seek for an algorithm deciding at which time to buy which amount of the commodity so as to minimize the cost.

An illustrative example of a buffering problem is a frugal car driver thinking about at which occasion to buy which amount of gasoline. In this example the price per unit varies over time, but can be assumed to be constant in every step, as typically the price for gasoline at a gas station does not depend on the amount that is bought by a single driver. Other examples are battery management in hybrid cars or economical caching policies for mobile devices. In these examples, the prices typically depend on the amount that is generated or purchased. For some applications like the battery management, the price functions may satisfy certain convexity assumptions. Some more information about these applications is given in Section 1.5.

Englert *et al.* [7] study online buffering problems within the framework of competitive analysis. They introduce the economical caching problem and give an online algorithm achieving the best possible competitive ratio against input sequences generated by an adversary. Although this work settles the problem in the competitive framework, the "optimal online algorithm" does not seem to be practical since it acts extremely precautiously in order to ensure a worst-case guarantee against the adversary. In fact, the design principle underlying this algorithm is called "thread-based" meaning that the algorithm, in each step, buys the minimal amount that is needed to ensure the competitiveness for all possible extensions of the price sequence. We believe that this kind of risk-averse behavior does not reflect the speculative nature of economical decision makers who may take the risk of buying more units than necessary when speculating on rising prices.

In this paper, we take a less pessimistic approach. As in the competitive framework, the online learning algorithm itself does not have any information about future prices. However, we assume that the algorithm can observe the performance of some experts (online strategies) in preceding steps. Each of these experts may have certain assumptions or knowledge about future prices. For example, one of these experts may assume that prices are set by an adversary and apply the "optimal online algorithm" from [7]. Other experts, may use stochastic prediction rules for estimating future prices [13, 8] or place their decisions based on practical experience and heuristics [5, 15]. Regardless of how these experts are chosen, the objective of the online learning algorithm is to come close to the performance of the best expert.

Before presenting our results, let us formally introduce the traditional model for expert based online learning and the adaption of this model for online buffering.

1.1 Online Learning and Weighted Majority Algorithm

In standard online learning the decision maker is equipped with N experts, numbered from 1 to N. The setup with respect to the cost is different from ours. (In particular, there is no buffer.) One might assume that an adversary chooses the cost for each expert in each step arbitrarily from $[0, 1]$. For expert i, we denote by c_i^t its cost in time step t and by $C_i^t = \sum_{k=1}^{t} c_i^k$ its accumulated cost until step t. In every step t, the decision maker selects an expert. The cost of the decision maker corresponds to the cost of the expert chosen in that step. Afterwards, it observes the cost vector $c^t \in [0, 1]^N$ of the experts.

Algorithm 1 (`Randomized Weighted Majority (RWM)`)

$w_i^1 = 1, q_i^1 = \frac{1}{N}$, for all i
for $t = 1, \dots, T$ **do**
  choose expert e^t at random according to $Q^t = (q_1^t, \dots, q_N^t)$
  $w_i^{t+1} = w_i^t (1-\eta)^{c_i^t}$, for all i
  $q_i^{t+1} = \frac{w_i^{t+1}}{\sum_{j=1}^{N} w_j^{t+1}}$, for all i
end for

The decision maker aims at choosing the experts in such a way that its cost is close to the cost of the best expert, that is, it aims at minimizing its regret. The decision maker corresponds to a (possibly randomized) algorithm $\mathcal{A}$. Consider a sequence of length T. Let $C_{\mathcal{A}}^T$ denote the expected cost accumulated by $\mathcal{A}$ until step T. Formally, the *(external) regret* of $\mathcal{A}$ on this sequence is

$$C_{\mathcal{A}}^T - C_{\text{best}}^T \ .$$

Observe that there are no assumptions on the quality of the experts or the relation between the experts. In general, if the decisions of each expert are arbitrarily bad, online learning algorithms cannot achieve good solutions. In particular, regret minimization does not mean to be competitive to an optimal offline algorithm, as in competitive analysis [3, 17]. However, one of the experts might use a strategy guaranteeing a competitive ratio against adversarial input sequences in which case the online learning algorithm can give (almost) the same performance guarantee as it achieves (almost) the performance of the best expert.

The *Randomized Weighted Majority (RWM)* algorithm of Littlestone and Warmuth [16] guarantees a regret bound of $O(\sqrt{T \log N})$, see also [2]. It is known that this is the best possible bound in the standard setting. The idea of this algorithm is to give each expert a probability of being chosen which depends on the cost that the expert has experienced in the past. The probability q_i that the strategy of expert i is chosen in the next time step is controlled by the current weight w_i of the expert which itself depends only on the weights in the steps up to $t-1$, the cost c_i^t and a parameter $\eta \in [0, \frac{1}{2}]$. The calculation of the weights and of the probabilities used by the algorithm is described in Algorithm 1. In the rest of the paper the probability for choosing expert i in round t is denoted by q_i^t, the corresponding weight is denoted by w_i^t and η is a parameter from $[0, \frac{1}{2}]$.

1.2 Extending the Online Learning Model towards the Buffering Problem

We study the following online buffering problem. A decision maker has to purchase a commodity over time. Time proceeds in discrete steps. In step t, the decision maker needs to satisfy a demand of $d^t \in [0,1]$ units of the commodity. The decision maker has a buffer of bounded size $B > 0$ for storing units of the commodity that can be used to satisfy demands at later points in time. In step t, it can purchase at most b^t units, where $b^t \in [d^t, B + d^t]$. The price per unit of the commodity varies over time and depends on the amount bought by the decision maker. It is described by a function $p^t : [0, b^t] \to [0,1]$. So the price for buying x units in step t is given by $xp^t(x)$. We seek for an algorithm deciding at which time to buy which amount of the commodity so as to minimize the cost.

In the context of the buffering problem, we assume that there are N experts corresponding to online algorithms and each expert is equipped with a buffer of size B. The expert decides how many units to buy in which step. Recall that the price per unit in step t depends on the purchased amount and is defined by the price function p^t. Observe that the experts may buy up to $B+1$ units per step so that the total price for the purchased units can be as high as $B+1$.

In our analysis, we account for the purchased units not at the time when the expert (or the online learning algorithm) buys the units but when it uses them to satisfy the demand. To formalize this, assume that every amount purchased by the expert (or the online algorithm) is put into the buffer and all demands are satisfied from the buffer in first-in first-out (FIFO) manner. The cost accounted for satisfying a demand with units bought in previous steps is equal to the price at which the units were bought. This accounting trick ensures that the cost of the experts (and the online algorithm) is at most one per step and it only decreases the accumulated cost of the experts up to an additive value of at most B. The cost by expert i in time step t is termed c_i^t, its cost accumulated until step t is denoted by $C_i^t = \sum_{k=1}^{t} c_i^k$ like in the standard model.

In every step, an online learning algorithm $\mathcal{A}$ chooses (possibly at random) one of the experts (or a linear combination of experts). In step t this choice depends only on the demands and prices until step $t-1$, i.e., on $d^1, \ldots, d^{t-1}, p^1, \ldots, p^{t-1}$. The term *choosing an expert* needs further clarification. If $\mathcal{A}$ has chosen expert i in step t, then it purchases the same amount of the commodity as expert i in step t with the following two exceptions: If the amount purchased by the expert together with the units in $\mathcal{A}$'s buffer does not suffice to cover the demand in this step, then $\mathcal{A}$ purchases a minimal amount necessary to satisfy the demand. (After such a step the buffer is empty.) If the amount purchased by the expert exceeds the demand in this step and the excess does not fit completely into $\mathcal{A}$'s buffer, then it purchases only the amount needed to fill the buffer up to the capacity. (After such a step the buffer contains B units.)

The (expected) cost of algorithm $\mathcal{A}$ in step t is termed $c^t_{\mathcal{A}}$, and the cost that $\mathcal{A}$ accumulates until step t is denoted by $C^t_{\mathcal{A}} = \sum_{k=1}^{t} c^k_{\mathcal{A}}$.

1.3 Our Contribution

We investigate the regret achieved by the RWM algorithm and variations of this algorithm on the buffering problem.

When describing the RWM algorithm, we did not specify that the random experiments in different steps are independent. In fact, in the standard setting such dependencies do not effect the expected cost of the RWM algorithm.[1] This is different when applying the algorithm to the buffering problem. In particular, one does not obtain a reasonable bound on the regret if experts are chosen using an independent random experiment for every step.

To see this, consider the following input sequence with fixed per unit prices (i.e., the price functions p^t are constant):

$$\begin{bmatrix} p^t \\ d^t \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \left(\begin{bmatrix} 0 \\ 1/4 \end{bmatrix} \begin{bmatrix} 1 \\ 1/4 \end{bmatrix} \begin{bmatrix} 0 \\ 1/4 \end{bmatrix} \begin{bmatrix} 1 \\ 1/4 \end{bmatrix} \right)^{T'} .$$

The sequence consists of an *initial step* with cost and demand equal to 0 followed by T' *rounds* of four steps with cost 0 or 1 (as specified above) and demand $1/4$ each. The length of the sequence is $T = 4T' + 1$. Suppose the buffer size is 1. The following two experts are given:

- The first expert purchases $1/2$ unit in the initial step and afterwards one unit in the third step of every round.

- The second expert purchases one unit in the first step of every round.

Obviously, both experts have cost 0 for the whole request sequence. We claim, however, that RWM has cost $\Theta(T)$ and, hence, the regret is $\Theta(T)$, too. RWM assigns probability $1/2$ to each of the experts in each step. If random experiments in different steps are independent, then in two consecutive rounds, with probability $1/16$ the algorithm selects in each step an expert that does not purchase a unit in this step. In this case, the buffer of RWM is empty in each step of the second round, which means that RWM has cost $1/2$ for serving the demands in the second round. Thus, for each round the expected cost of RWM is at least $1/32$.

A reason for the poor performance of RWM is that it changes the experts frequently and the chosen experts might have completely different filling levels in their buffers. We present a variant of RWM that uses dependencies in order to decrease the number of expert changes. The algorithm is called *Shrinking Dartboard (SD)* algorithm as the random experiments used by this algorithm are described in terms of a shrinking dartboard. We prove that the regret achieved by this algorithm is at most $O(\sqrt{BT \log N})$.

If the price functions in each step satisfy a convexity condition (e.g., if prices are constant) then we can derive even a deterministic variant of the Weighted Majority Algorithm with a good regret bound. The *Weighted Fractional (WF)* algorithm chooses linear combinations of experts rather than selecting experts at random. WF achieves a regret of at most $O(\sqrt{BT \log N})$, too. As this algorithm is deterministic, its regret guarantee holds even against an adaptive adversary.

Finally, we present a lower bound of $\Omega(\sqrt{BT \log N})$ for the regret showing that SD and WF achieve the optimal regret for the buffering problem up to constant factors. This lower bound holds even if prices are assumed to be constant.

[1] In the standard setting without buffer, the expected cost of the RWM algorithm is not effected by dependencies between different steps, unless the adversary who specifies the cost for the experts is *adaptive*, i.e., the cost vector presented for a step might depend on the random coin flips of the algorithm in previous steps.

1.4 Related Work

A special case of online buffering problems is the *economical caching* [7, 8] and the *one-way trading* problem [3, 6]. In one-way trading a price sequence is given, each price representing an exchange rate from dollar to yen. The task is to trade d dollars to a maximum number of yen. In economical caching, there is furthermore a previously unknown demand of yen at each time step which must be consumed from a buffer or bought for the current price. Economical caching was introduced in [7]. Englert et al. analyzed it in a worst case competitive analysis yielding general tight bounds for the competitive factor depending on the ratio between the lowest and highest exchange rate. [8] showed that if the price sequence is modeled by a random walk, a competitive factor which does not depend on this ratio can be achieved.

We study online buffering problems with online learning algorithms as described in Section 1.2. Online learning algorithms are used for regret minimization. A general introduction is given in [1, 2]. In this paper we use the external regret model [2, 4] with full information.

There are several algorithms for which the regret per time step converges against zero for a long time horizon. The following algorithms achieve a regret of $O(\sqrt{T \log N})$: The *Randomized Weighted Majority* (RWM) algorithm [16] weights each expert depending on its cost so far. In the randomized version the weights are used as probabilities for an expert to be chosen. It is known that RWM achieves the best possible bound for the standard setting. But this is no longer valid when applying the algorithm to the online buffering problem. A reason for this is that RWM changes the experts frequently.

[12] and [14] present the *Follow The Perturbed Leader* (FPL) algorithm that follows the expert which has achieved the lowest cost so far plus some perturbation. FPL reduces the number of expert changes to achieve good regret bounds. But in contrast to the Weighted Fractional (WF) algorithm presented in this paper FPL cannot achieve this bound against an adaptive adversary when applied to online buffering problems.

The WF algorithm can only achieve good regret bounds if the price functions satisfy a convexity property. This assumption is also made in [9, 18]. It is shown that if the cost of each expert is given by a convex function which may change over time, a gradient descent algorithm can be used to achieve good regret bounds. This algorithm cannot directly be used to achieve good regret bounds for online buffering problems, since the experts choose a fixed point of the convex function. In our model this would lead to very limited experts. An expert would then only be able to determine once the number units it wants to buy in every time step. This choice would be the same for all time steps. It could no longer depend on the current price or filling status of the buffer. This expert model is to limited and can therefore not be used to solve online buffering problems.

Online learning algorithms have been studied in many other research areas yielding a huge variety of results. Some of the possible applications for online learning are given in [10].

1.5 Applications

One possible application for our online learning algorithms is *battery management* for hybrid cars. In a hybrid car there are two engines, a combustion engine and an electrical engine. The energy for the electrical engine is taken from a battery of bounded capacity. The battery can be recharged using the combustion engine or regenerative energy, e.g., from the braking system. The torque is provided by both of these engines and the demand of torque depends on the acceleration requested by the driver and the topology of the route.

The online algorithm has to decide how much power is produced by each of the two engines. If more power is required than the combustion engine is producing, the remaining power must be used from the battery or it must be produced by the electrical engine by using regenerative energies. On the other hand, if more power is produced than currently needed, the additional power is not given to the drive shaft, but saved into the battery for later usage.

In our model, the demand specified in the input sequence corresponds to the energy needed for providing the requested torque as well as for electrical devices, e.g., for air conditioning. The amount that is purchased by the online algorithm corresponds to the energy produced by the combustion engine and the exploited regenerative energy. The price of the energy generated by the combustion engine is determined by the used amount of fuel, which itself depends on the torque and the gear. These price functions potentially satisfy the convexity assumption used in the analysis of the WF algorithm. The price of the exploited regenerative energy is 0.

In engineering, decisions about which engine should produce which amount of power are typically made based on engine operating maps [5] and certain knowledge of the route [13, 15]. Engine operating maps show the fuel consumption of the car for different operation states. The topology of the route can, e.g., be estimated by using the on-board navigation system. Heuristics based

Algorithm 2 (`Shrinking Dartboard (SD)`)

1: $w_i^1 = 1, q_i^1 = \frac{1}{N}$, for all i
2: choose expert e^1 at random according to $Q^1 = (q_1^1, \ldots, q_N^1)$
3: **for** $t = 2, \ldots, T$ **do**
4: $\quad w_i^t = w_i^{t-1}(1-\eta)^{c_i^{t-1}}$, for all i
5: $\quad q_i^t = \frac{w_i^t}{\sum_{j=1}^N w_j^t}$, for all i
6: $\quad$ with probability $\frac{w_{e^t}^t}{w_{e^t}^{t-1}}$ do not change expert, i.e., set $e^t = e^{t-1}$
7: $\qquad\qquad$ else choose e^t at random according to $Q^t = (q_1^t, \ldots, q_N^t)$
8: **end for**

on engine operating maps parametrized with typical driving conditions combined with different prediction models for the topology are suitable candidates for the experts used by our online learning algorithms.

Another application in a completely different context is *smart caching* of data streams on mobile devices, see also [7, 11]. Suppose a data stream can be fetched by a mobile device over different communication standards like, e.g., GSM, UMTS, WLAN, each for different cost, but not all services are always and anywhere available. The stream can be buffered by the mobile device in a storage of bounded size. We assume that the most expensive of these services is always available at a cost of one per data unit and any demand per step can always be satisfied using this expensive service. Other services can be used to download and buffer the data stream at lower cost only if they are available.

The best way for combining the different services depends on the users mobility and the availability of the services over time. Therefore, it is not possible to implement a fixed optimal strategy into the mobile device, but a good strategy shall be learned online. The experts for online learning in smart caching recommend which standard to use at which time step. They might recommend also to combine the different communication standards by using each standard to download a certain amount of the data in a time step.

In this context it might not be appropriate to assume that the price function is convex. Besides it might not be possible that every arbitrary amount of data can be downloaded per time step, but some services are restricted to a certain variety of different amounts. Under these assumptions, our algorithm WF cannot be applied as it combines experts in a fractional manner and its analysis assumes convexity of the price function. Let us remark, however, that SD can be applied without any assumptions on the price functions and even if different services are restricted to discrete amounts of data as long as the simulated experts satisfy the restrictions regarding the amounts that can be downloaded.

2 The Shrinking Dartboard Algorithm

We devise a variant of the Randomized Weighted Majority algorithm using dependencies between the random decisions in different steps in order to reduce the number of expert changes. Algorithm 2 specifies how the experts are chosen and introduces the notation for this section. Furthermore, let $W^t = \sum_{i=1}^N w_i^t$ denote the sum of weights in step t. The algorithm is called *Shrinking Dartboard (SD)* algorithm as it can be illustrated in terms of a dartboard shrinking over time.

Initially, the dartboard is a disc divided into N equally sized sectors, one for each expert. The total area covered by the disc has size N so that each sector has size 1. In step 1, SD chooses an expert by *throwing a dart* to the board, that is, it picks a point from the disc uniformly at random and chooses that expert into which sector this point falls. Over time, the expert's sectors shrink as illustrated in Figure 1. In particular, the size of the area covered by expert i's sector in step t, denoted by *allowed area* in step t, corresponds to the weight w_i^t as specified in Algorithm 2. In step $t > 1$, SD chooses an expert as follows: If the previously picked point is still in the allowed area, then SD does not change the expert. This happens with probability $w_{e^t}^t / w_{e^t}^{t-1}$ and corresponds to line 6 of Algorithm 2. Otherwise, SD *throws a new dart*, that is, it picks a point uniformly at random from the area covered by the sectors of all experts and chooses the expert into which sector this point falls. This happens with probability $1 - w_{e^t}^t / w_{e^t}^{t-1}$ and corresponds to line 7 of Algorithm 2.

Observe that the weights used by SD correspond to the weights of the original Randomized Weighted Majority algorithm (Algorithm 1). The following lemma shows that SD does not only use the same weights as RWM but both algorithms have the same probability distribution for choosing

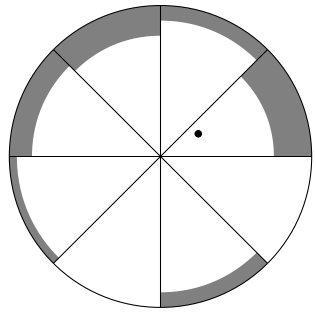

Figure 1: Probability distribution as a dartboard

an expert in a step. The difference, however, is that the random choices made by SD in different steps are not independent, but the selection of the expert in step t depends on the selection in step $t-1$.

Lemma 1 $\mathbf{Pr}\left[e^t = i\right] = q_i^t$, *for* $i \in \{1, \ldots, N\}$, $t \in \{1, \ldots, T\}$.

Proof: We use an induction on $1 \leq t \leq T$. For $t = 1$, the statement in the lemma follows immediately from the description of the algorithm. Now let $t \geq 2$. Expert i is selected in step t either because it was selected already in step $t-1$ and the corresponding dart is still in the allowed area (i.e., the expert is chosen by line 6 of Algorithm 2) or because a new dart is thrown and this dart hits i's sector (i.e., the expert is chosen by line 7). Hence,

$$\begin{aligned}
\mathbf{Pr}\left[e^t = i\right] &= \mathbf{Pr}\left[e^{t-1} = i\right] \cdot \frac{w_i^t}{w_i^{t-1}} + q_i^t \cdot \sum_{j=1}^{N} \mathbf{Pr}\left[e^{t-1} = j\right] \cdot \left(1 - \frac{w_j^t}{w_j^{t-1}}\right) \\
&= q_i^{t-1} \cdot \frac{w_i^t}{w_i^{t-1}} + q_i^t \cdot \sum_{j=1}^{N} q_j^{t-1} \cdot \left(1 - \frac{w_j^t}{w_j^{t-1}}\right) \\
&= \frac{w_i^{t-1}}{W^{t-1}} \cdot \frac{w_i^t}{w_i^{t-1}} + \frac{w_i^t}{W^t} \cdot \sum_{j=1}^{N} \frac{w_j^{t-1}}{W^{t-1}} \cdot \frac{w_j^{t-1} - w_j^t}{w_j^{t-1}} \\
&= \frac{w_i^t}{W^{t-1}} + \frac{w_i^t}{W^t} \cdot \frac{W^{t-1} - W^t}{W^{t-1}} \quad = \quad \frac{w_i^t}{W^t} \quad = \quad q_i^t \ .
\end{aligned}$$

■

Let D denote the number of expert changes during the execution of SD. The following lemma bounds the expected value of D in terms of the cost of the best expert.

Lemma 2 $\mathbf{E}\left[D\right] \leq 2\eta C_{\text{best}}^T + \ln N$.

Proof: D is bounded from above by the number of times line 7 of the algorithm is applied which corresponds to the number of darts that are thrown because the sector of the chosen expert shrinks. The probability for throwing a new dart in step $t \geq 2$ is

$$\alpha^t \;=\; \sum_{j=1}^{N} \mathbf{Pr}\left[e^{t-1} = j\right] \cdot \left(1 - \frac{w_j^t}{w_j^{t-1}}\right) \;=\; \frac{W^{t-1} - W^t}{W^{t-1}} \ ,$$

where the latter equation follows from the calculation in the proof of Lemma 1. Thus, $W^t = (1 - \alpha^t)W^{t-1}$.

The total weight W^{T+1} after step T can hence be expressed in terms of these probabilities, that is,

$$W^{T+1} \;=\; W^1 \prod_{t=1}^{T} (1 - \alpha^{t+1}) \;=\; N \prod_{t=1}^{T} (1 - \alpha^{t+1}) \ .$$

On the other hand,

$$W^{T+1} \;\geq\; (1 - \eta)^{C_{\text{best}}^T}$$

as the latter quantity corresponds to the weight of the best expert after step T. Combining these equations and applying the logarithm gives

$$C_{\text{best}}^T \ln(1-\eta) \leq \ln N + \sum_{t=1}^{T} \ln(1-\alpha^{t+1}) \ .$$

Using $\ln(1-\alpha^{t+1}) \leq -\alpha^{t+1}$ and $\ln(1-\eta) \geq -2\eta$, the expected number of thrown darts is thus

$$\sum_{t=1}^{T-1} \alpha^{t+1} \leq \ln N + 2\eta C_{\text{best}}^T \ .$$

■

We apply the lemmas above in the following regret analysis in which we compare the cost of SD with the cost of the best expert. As defined in the Section 1.2, C_i^t is the cost of the units expert i uses (rather than purchases) until step t, where we assume that units are consumed from the expert's buffer in FIFO manner and valuated with the price at which they were bought. Recall that c_i^t denotes the cost accounted for expert i in step t.

Theorem 3 *For $\eta \leq 1/2$, the expected cost of SD satisfies*

$$C_{\text{SD}}^T \leq (1+\eta+2\eta B) C_{\text{best}}^T + \frac{\ln N}{\eta} + B \ln N \ .$$

Setting $\eta = \min\{\sqrt{\ln N/(BT)}, 1/2\}$ yields $C_{\text{SD}}^T \leq C_{\text{best}}^T + O(\sqrt{BT \log N})$.

Proof: We claim that the cost of SD is bounded from above by

$$\sum_{t=1}^{T} c_{e^t}^t + DB \ .$$

In words, the cost of SD is bounded from above by the cost of the chosen experts plus the number of expert changes times the buffer size. To see this, consider the cost in a time period beginning with a step in which a new expert is chosen and ending with the last step before the next expert is chosen or the sequence ends. The cost of SD in this period can be split into three contributions.

(1) Cost due to units used and bought in this period by both SD and the expert.

(2) Cost for using those units that are stored in SD's buffer at the beginning of the period.

(3) Cost for using units bought during the period by SD to ensure feasibility.

The cost in (1) is upper bounded by $\sum_{t=1}^{T} c_{e^t}^t$. Observe that the sum of units covered by (2) and (3) is at most B. Hence, the difference between the cost of SD and the cost of the expert within such a period is at most B. Furthermore, in the very first period, the costs in (2) and (3) are 0 because both SD and the expert start with an empty buffer and purchase and use the same units. This gives the stated upper bound on C_{SD}^T as the number of periods without counting the first period is D.

Next we claim that

$$E\left[\sum_{t=1}^{T} c_{e^t}^t\right] \ \leq \ (1+\eta) C_{\text{best}}^T + \frac{\ln N}{\eta} \ .$$

This follows from Lemma 1 showing that the probability that SD chooses expert i in step t is equal to the probability that RWM chooses expert i in step t. The left hand term describes the cost of RWM assuming that the cost accounted for the learning algorithm in step t are $c_{e^t}^t$ (as in the standard setting of online learning). Thus, we can apply the well known upper bound on the cost of RWM holding for $\eta \leq 1/2$ (cf., e.g., [2]).

Together with the bound on the expected value of D in Lemma 2 this yields the first bound in the theorem.

Finally, assume $\eta \leq \sqrt{\ln N/(BT)}$. Combining the first bound of the theorem with the trivial bound $C_{\text{SD}}^T - C_{\text{best}}^T \leq T$, we obtain

$$\begin{aligned} C_{\text{SD}}^T - C_{\text{best}}^T &\leq (\eta + 2\eta B)T + \frac{\ln N}{\eta} + \min\{T, B \ln N\} \\ &= O\left(\sqrt{BT \log N} + \min\{T, B \log N\}\right) \\ &= O\left(\sqrt{BT \log N}\right), \end{aligned}$$

which gives the second bound stated in the theorem. ■

3 The Weighted Fractional Algorithm

The *Weighted Fractional (WF)* algorithm uses the same weights as the algorithms RWM and SD. However, instead of choosing an expert at random, it simulates the experts fractionally. That is, it purchases $x^t = \sum_{i=1}^{N} q_i^t x_i^t$ units in step t, where x_i^t is the amount purchased by expert i in the same step. Observe that this rule might lead to infeasibilities as the amount of purchased units together with the units in the buffer might not be enough to satisfy the demand in a step or the buffer might overflow. In these cases, WF enforces feasibility by purchasing a minimal amount of additional units or reducing the amount of bought units, respectively.

The following theorem bounds the cost of WF assuming that the price functions satisfy a convexity property. In particular, the function $f^t(x) = xp^t(x)$ that describes the cost incurred for buying an amount of x needs to be convex.

Theorem 4 *Suppose the functions $f^t(x)$, $x \in [0, b^t]$ are convex, for $1 \le t \le T$. Then the cost of WF satisfies*

$$C_{\mathrm{WF}}^T \le (1 + \eta + 2\eta B) C_{\mathrm{best}}^T + \frac{\ln N}{\eta} + B \ln N \ .$$

Setting $\eta = \min\{\sqrt{\ln N/(BT)}, 1/2\}$ yields $C_{\mathrm{WF}}^T \le C_{\mathrm{best}}^T + O(\sqrt{BT \log N})$.

Proof: We analyze WF by relating it to another algorithm called *k-SD*. This algorithm splits the buffer into $k \ge 1$ sub-buffers of size B/k each. For each of these sub-buffers, we simulate algorithm SD scaling down all demands as well as the amounts purchased by the experts by multiplying with $1/k$. Besides, we adapt the price function, that is, when buying x_j^t units for sub-buffer j in step t, algorithm k-SD assumes that this incurs *virtual cost* of $f^t(kx_j^t)/k$.

For $1 \le j \le k$, let L_j^T denote the expected virtual cost for sub-buffer j accumulated until step T. As k-SD simulates SD for every sub-buffer using an appropriate scaling, it holds $L_j^T = C_{\mathrm{SD}}^T/k$ so that

$$\sum_{j=1}^{k} L_j^T = C_{\mathrm{SD}}^T \ .$$

Now let us compare the sum of the virtual cost of k-SD with the true cost of this algorithms. For every time step t, we have

$$f^t\left(\sum_{j=1}^{k} x_j^t\right) \ \le \ \frac{1}{k}\sum_{j=1}^{k} f^t(kx_j^t)$$

by Jensen's inequality. Thus, we observe that the true cost incurred for any step t is upper bounded by the sum of the virtual cost for this steps. Combining this observation with the equation above gives

$$C_{k\text{-SD}}^T \le C_{\mathrm{SD}}^T \ ,$$

for every $k \ge 1$.

Let us introduce a slight modification to k-SD. The resulting algorithm is called k-SD'. As k-SD simulates SD on each sub-buffer, it needs to buy additional units in some time steps in order to ensure feasibility. These are at most B/k units for every thrown dart (new expert) for each sub-storage. k-SD' does not need to buy these additional units when the respective sub-buffer is empty but can defer this until all sub-buffers are empty. This does not increase the number of units that need to be bought, but prices for these units might change. However, in the analysis of SD, we estimated the prices for these units with the worst possible price. Hence, we can apply Theorem 3 not only to k-SD, but also to k-SD' and obtain

$$C_{k\text{-SD'}}^T \le (1 + \eta + 2\eta B) C_{\mathrm{best}}^T + \frac{\ln N}{\eta} + B \ln N \ ,$$

for every $k \ge 1$.

Now we let k go to infinity. Consider a fixed step t. By the law of large numbers, the sum of the amounts purchased by the experts chosen for the sub-buffers converges to its expectation. That is, the sum of purchased amounts over all sub-buffers converges to $\sum_{i=1}^{N} q_i^t x_i^t$. As a consequence, k-SD' (for $k \to \infty$) purchases the same amount per step as WF. Hence,

$$C_{\mathrm{WF}}^T = \lim_{k \to \infty} C_{k\text{-SD'}}^T \ .$$

Combining the last two equations yields the theorem. ■

4 Lower Bound

The following theorem shows that our upper bounds on the external regret achieved by SD and WF are tight up to constant factors, that is, one cannot achieve significant further improvements. Let us remark that the lower bound given in the theorem holds even if we restrict the input sequences to fixed prices per unit and if the experts purchase at most one unit per step.

Theorem 5 *For every integer T, there exists a stochastically generated sequence of length T together with N experts such that every learning algorithm $\mathcal{A}$ with a buffer of size B suffers a regret of $\Omega(\sqrt{BT \log N})$.*

Proof: The sequence consists of T' consecutive *rounds*. Each round consists of 3 *phases* each of which has B steps so that the sequence has a total length of $T = 3BT'$ steps. Prices are defined by constant functions, i.e., there is a fixed price p^t per unit in every step t. For simplicity in notation, we assume that prices are chosen from the interval [0,4] instead of [0,1]. In particular, the sequence of prices and demands has the following structure

$$\begin{bmatrix} p^t \\ d^t \end{bmatrix} = \left(\begin{bmatrix} 2 \\ 0 \end{bmatrix}^B \begin{bmatrix} \{0,4\} \\ 0 \end{bmatrix}^B \begin{bmatrix} 4 \\ 1 \end{bmatrix}^B \right)^{T'} .$$

Prices in the second phase of each round are chosen at random from $\{0, 4\}$. However, the choices within the same round are dependent, that is, for all steps within the second phase of the same round, we set the same price and this price is chosen uniformly at random from the set $\{0, 4\}$. Prices for different rounds are selected independently.

The N experts for the problem are defined as follows: In each round, every expert chooses independently one of the following two strategies each with probability 1/2.

a) The expert purchases B units in the first phase.

b) The expert purchases B units in the second phase.

In both cases, no further units are bought. In particular, the buffer is empty at the end and the beginning of each round.

Let F_i^t denote the random variable defining the cost of expert i in round t. Depending on the outcome of the price for round t and the strategy chosen by the expert, this variable takes the following values.

F_i^t	0	4
a	$2B$	$2B$
b	0	$4B$

When analyzing this random variable, we can assume that, a *column player* (the designer of the sequence) selects a column of this matrix and a *row player* (the considered expert) selects a row. Both of these choices are made independently, uniformly at random. As every entry of the matrix is chosen with probability 1/4, the expected value of F_i^t is $2B$. Thus, by linearity of expectation, the expected cost of an expert over the whole sequence is $2BT'$.

Next we analyze the expected cost of the *best* expert. Towards this end, let F^t be a random variable describing the average cost in the column chosen by the column player in round t, that is, if the column player chooses column 0 then $F^t = B$ and if the column player chooses column 1 then $F^t = 3B$. The expected value of this variable is $2B$ so that

$$E\left[\sum_{t=1}^{T'} F^t\right] = 2BT' = \frac{2}{3}T .$$

Now define $\Delta_i^t = F_i^t - F^t$. The values taken by the random variable Δ_i^t depend on the choices of the row and column players and are specified in the following matrix.

Δ_i^t	0	4
a	B	$-B$
b	$-B$	B

Observe that the random variables Δ_i^t, $1 \le i \le N$, $1 \le t \le T'$ are stochastically independent since each of these variables takes one of the value B or $-B$ with probability $1/2$ each, regardless of the outcome of the other variables. For $1 \le i \le N$, let $S_i = \sum_{t=1}^{T'} \Delta_i^t / B$. The random variables $S_1, \ldots, S_N$ are stochastically independent and each of them corresponds to the value of a fair random walk after T' steps. The expected minimum over N such variables is known to be $-\Theta(\sqrt{T' \log N})$, which gives

$$E\left[\min_i \left(\sum_{t=1}^{T'} \Delta_i^t\right)\right] = -\Theta(B\sqrt{T'\log N}) = -\Theta\left(\sqrt{BT\log N}\right) .$$

Hence, the expected cost of the best expert can be estimated by

$$\begin{aligned} E\left[\min_i \left(\sum_{t=1}^{T'} F_i^t\right)\right] &= E\left[\min_i \left(\sum_{t=1}^{T'} \left(F^t + \Delta_i^t\right)\right)\right] \\ &= E\left[\sum_{t=1}^{T'} F^t\right] + E\left[\min_i \left(\sum_{t=1}^{T'} \Delta_i^t\right)\right] \\ &= \frac{2}{3}T - \Theta\left(\sqrt{BT\log N}\right) . \end{aligned}$$

Finally, we show that any online learning algorithm $\mathcal{A}$ equipped with these experts cannot achieve an expected cost better than $2/3T$. W.l.o.g, $\mathcal{A}$ does not purchase any unit in the third phase of a round, but exactly B units during the first two phases of each round. In the first phase of a round, $\mathcal{A}$ does not have any information about the price in the second phase since the experts decisions (over the whole sequence) and costs (before entering the second phase) do not depend on this price and, hence, do not give any evidence about the price. Thus, $\mathcal{A}$'s decision about how many of the B units to purchase in the first and how many units to purchase in the second phase of a round is independent of the price selected for the second phase. As a consequence, the expected cost for each purchased unit is 2 and, hence, the expected cost per round is $2B$. Therefore, the expected cost of $\mathcal{A}$ for the whole sequence is $2BT' = 2/3\,T$ and, consequently, the regret is $\Theta\left(\sqrt{BT\log N}\right)$. ■

5 Discussion

We observed that RWM with independent coin flips in different steps fails to give reasonable regret bounds for buffering problems as experts have a state and switching between different states is expensive. We addressed this problem by changing the randomized selection in such a way that changes between experts are reduced. Let us remark that the same kind of problem occurs for algorithm Follow the Perturbed Leader (FPL) by Kalai and Vempala [14], too. The number of expert changes of FPL can be reduced in a similar fashion: One uses only one initial perturbation rather than independent perturbations in every step. In fact, one can deduce from the analysis in [14] that this results in the same regret bound as achieved by our algorithm SD. However, SD has one additional advantage: It can easily be transformed into a simple deterministic, fractional variant of SD guaranteeing optimal regret against an adaptive adversary, provided the price functions satisfy a convexity condition. The same kind of transformation cannot be directly applied to FPL as the probabilities for choosing an expert in a step are not available in closed form for FPL but would need to be extracted from the perturbation experiment.

In the model for online learning with buffer, we have assumed that experts have identical price functions. We want to point out, however, that this assumption is not necessary for our randomized algorithm. It is easy to check that the analysis for SD goes through even if different experts have different price functions within the same step. In contrast, for the deterministic, fractional algorithm WF, the assumption of identical price functions is crucial. It is an interesting question whether one can achieve a similar regret bound for a randomized or deterministic learning algorithm against an adaptive adversary even if experts have different and/or non-convex price functions.

References

[1] A. Blum. On-line algorithms in machine learning. In *Online Algorithms: The State of the Art*, volume 1442 of *Lecture Notes in Computer Science*, chapter 14, pages 306–325. Springer, 1998.

[2] A. Blum and Y. Mansour. Learning, regret minimization, and equilibria. In N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors, *Algorithmic Game Theory*, chapter 4, pages 79–101. Cambridge University Press, New York, NY, USA, 2007.

[3] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, New York, NY, USA, 1998.

[4] N. Cesa-Bianchi, Y. Freund, D. P. Helmbold, D. Haussler, R. E. Schapire, and M. K. Warmuth. How to use expert advice. In *Proceedings of the 25th ACM Symposium on Theory of Computing (STOC)*, pages 382–391, 1993.

[5] M. Ehsani, Y. Gao, and K. L. Butler. Application of Electrically Peaking Hybrid (ELPH) Propulsion System to a Full-Size Passenger Car with Simulated Design Verification. *IEEE Transactions on Vehicular Technology*, 48(6):1779–1787, 1999.

[6] R. El-Yaniv, A. Fiat, R. M. Karp, and G. Turpin. Optimal search and one-way trading online algorithms. *Algorithmica*, 30(1):101–139, 2001.

[7] M. Englert, H. Röglin, J. Spönemann, and B. Vöcking. Economical caching. In *Proceedings of the 26th Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 385–396, 2009.

[8] M. Englert, B. Vöcking, and M. Winkler. Economical caching with stochastic prices. In *Proceedings of the 5th Symposium on Stochastic Algorithms, Foundations and Applications (SAGA)*, pages 179–190, 2009.

[9] A. Flaxman, A. T. Kalai, and H. B. McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 385–394, 2005.

[10] D. P. Foster and R. Vohra. Regret in the on-line decision problem. *Games and Economic Behavior*, 29(1-2):7–35, 1999.

[11] S. Göbbels. *Smart Caching for Continuous Broadband Services in Intermittent Wireless Networks*. PhD thesis, RWTH Aachen University, Chair of Communication Networks (ComNets), 2009.

[12] J. Hannan. Approximation to bayes risk in repeated plays. *Contributions to the Theory of Games*, 3:97–139, 1957.

[13] L. Johannesson, M. Asbogard, and B. Egardt. Assessing the potential of predictive control for hybrid vehicle powertrains using stochastic dynamic programming. *IEEE Transactions on Intelligent Transportation Systems*, 8(1):71–83, 2007.

[14] A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

[15] C.-Y. Li and G.-P. Liu. Optimal fuzzy power control and management of fuel cell/battery hybrid vehicles. *Journal of Power Sources*, 192(2):525–533, 2009.

[16] N. Littlestone and M. Warmuth. The weighted majority algorithm. In *Proceedings of the 30th Symposium on Foundations of Computer Science (SFCS)*, pages 256–261, 1989.

[17] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[18] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 928–936, 2003.