Proceedings of Machine Learning Research vol 75:1–19, 2018 31st Annual Conference on Learning Theory

An Optimal Learning Algorithm for Online Unconstrained Submodular Maximization

Tim Roughgarden TIM@CS.STANFORD.EDU
Stanford University

Joshua R. Wang JOSHUA.WANG@CS.STANFORD.EDU
Stanford University

Editors: Sebastien Bubeck, Vianney Perchet and Philippe Rigollet

Abstract

We consider a basic problem at the interface of two fundamental fields: *submodular optimization* and *online learning*. In the *online unconstrained submodular maximization (online USM) problem*, there is a universe $[n] = \{1, 2, \ldots, n\}$ and a sequence of T nonnegative (not necessarily monotone) submodular functions arrive over time. The goal is to design a computationally efficient online algorithm, which chooses a subset of $[n]$ at each time step as a function only of the past, such that the accumulated value of the chosen subsets is as close as possible to the maximum total value of a fixed subset in hindsight. Our main result is a polynomial-time no-$\frac{1}{2}$-regret algorithm for this problem, meaning that for every sequence of nonnegative submodular functions, the algorithm's expected total value is at least $\frac{1}{2}$ times that of the best subset in hindsight, up to an error term sublinear in T. The factor of $\frac{1}{2}$ cannot be improved upon by any polynomial-time online algorithm when the submodular functions are presented as value oracles. Previous work on the offline problem implies that picking a subset uniformly at random in each time step achieves zero $\frac{1}{4}$-regret.

A byproduct of our techniques is an explicit subroutine for the two-experts problem that has an unusually strong regret guarantee: the total value of its choices is comparable to twice the total value of either expert on rounds it did not pick that expert. This subroutine may be of independent interest.

Keywords: Online learning; submodular optimization

1. Introduction

The problem we study, *online unconstrained submodular maximization (online USM)*, lies in the intersection of two fundamental fields: *submodular optimization* and *online learning*.

Submodular optimization. A nonnegative real-valued set function $f : 2^{[n]} \to \mathbb{R}_+$ defined on the ground set $[n] = \{1, 2, \ldots, n\}$ is *submodular* if it exhibits diminishing returns, in the sense that

$$f(S \cup \{i\}) - f(S) \leq f(T \cup \{i\}) - f(T)$$

whenever $T \subseteq S$ and $i \notin S$.[1] Submodular functions can be used to model a wide array of important problems, and for this reason have been extensively studied for decades in theoretical computer science (e.g. Dughmi (2011)), combinatorial optimization (e.g. Vondrak (2007)), economics (e.g. Milgrom (2004)), and machine learning (e.g. Bach (2013)). Perhaps the most basic problems in

1. Note that f is not assumed to be monotone.

submodular optimization are to minimize or maximize a submodular function (without constraints). While the former problem admits (highly non-trivial) polynomial-time algorithms (Grötschel et al., 1988; Iwata et al., 2001; Schrijver, 2000), unconstrained maximization is hard to approximate better than a factor of $\frac{1}{2}$ in polynomial time (Feige et al., 2011; Dobzinski and Vondrák, 2012). Indeed, many fundamental NP-hard problems are special cases of unconstrained submodular maximization (USM), including undirected and directed versions of graph and hypergraph cut problems (e.g. Goemans and Williamson (1995); Halperin and Zwick (2001)), maximum facility location problems (e.g. Ageev and Sviridenko (1999)), and certain restricted satisfiability problems (e.g. Guruswami and Khot (2005)). Also, approximation algorithms for the USM problem have been used as subroutines in many other algorithms, including those for social network marketing (Hartline et al., 2008), market expansion (Dughmi et al., 2012), and the computation of the least core value in a cooperative game (Schulz and Uhan, 2013).

Online learning. The goal in online learning is to make good decisions over time with knowledge only of the past. In the standard "experts" setup, there is a known set A of actions and a time horizon T. At each time step $t = 1, 2, \ldots, T$, the online algorithm has to first choose a action $a^t \in A$, and an adversary subsequently chooses a reward vector $r^t : A \to [0, 1]$. Given a history of actions $a^1, \ldots, a^T$ and reward vectors $r^1, \ldots, r^T$, the *regret* of the algorithm is the difference between the maximum total reward $\max_{a \in A} \sum_{t=1}^{T} r^t(a)$ of a fixed action in hindsight and the total reward $\sum_{t=1}^{T} r^t(a^t)$ earned by the algorithm. The goal in online learning is to design algorithms with expected regret $o(T)$ as $T \to \infty$.

Ignoring computational issues, this goal is well understood: there are randomized algorithms (like "Follow the Perturbed Leader" (Kalai and Vempala, 2005) and "Multiplicative Weights" (Cesa-Bianchi et al., 2007; Freund and Schapire, 1997)) with worst-case expected regret $O(\sqrt{T \log |A|})$, and no algorithm can do better (see e.g. Cesa-Bianchi and Lugosi (2006)). However, the generic algorithms that achieve this regret bound require computation at least linear in $|A|$ at each time step. Thus, when the action space A has size exponential in the parameters of interest, these algorithms are not computationally efficient.

Online USM. We consider the natural online learning version of the USM problem. There is a universe $[n] = \{1, 2, \ldots, n\}$, known in advance. Actions correspond to subsets of the universe, and submodular functions arrive online.

- At each time step $t = 1, 2, \ldots, T$:
  - The algorithm picks a probability distribution p^t over subsets of $[n]$.
  - An adversary picks a submodular function $f^t : 2^{[n]} \to [0, 1]$.
  - A subset S^t is chosen according to the distribution p^t, and the algorithm reaps a reward of $f^t(S^t)$.
  - The adversary reveals f^t to the algorithm.

The goal is to design a computationally efficient online algorithm with worst-case expected regret as small as possible. Applying the generic no-regret algorithms to this problem requires per-step computation exponential in n. Indeed, unless $RP = NP$, *there does not exist a polynomial-time no-regret algorithm for the online USM problem.*[2] This negative result motivates following in

2. Standard arguments show that any polynomial-time (randomized) no-α-regret algorithm for online USM yields a polynomial-time randomized $(\alpha+\epsilon)$-approximation algorithm for the offline USM problem for every constant $\epsilon > 0$.

the footsteps of Kakade et al. (2009) and defining, for $\alpha \in [0,1]$, the *α-regret* of an algorithm (w.r.t. actions $S^1, \ldots, S^T$ and functions $f^1, \ldots, f^T$) as the difference between α times the cumulative reward of the best fixed action in hindsight and that earned by the algorithm:

$$\alpha \cdot \max_{S \subseteq [n]} \sum_{t=1}^{T} f^t(S) - \sum_{t=1}^{T} f^t(S^t). \tag{1}$$

A *no-α-regret algorithm* is one whose worst-case expected α-regret is bounded by $O(T^c)$ for some constant $c < 1$ (with the big-O suppressing any dependence on n). The worst case is taken over the adversary's choice of functions, and the expectation is over the coin flips of the algorithm. A basic question is:

> *What is the largest constant α such that there exists a computationally efficient no-α-regret algorithm for online USM?*

By "efficient," we mean that the number of operations performed by the algorithm in each time step is bounded by some polynomial function of n, the size of the universe.[3]

Our main result is a tight answer to this question:

> *$\alpha = \frac{1}{2}$ is achievable, and no larger value of α can be achieved (unless $RP = NP$).*

Prior to our work, the best result known (which follows from Feige et al. (2011)) was that $\alpha = \frac{1}{4}$ can be achieved by picking a subset uniformly at random in every time step.

Offline-to-online reductions. Our results also contribute to the burgeoning line of work on "offline-to-online reductions." Here, the question is whether or not an efficient α-approximate oracle for the offline version of a problem (i.e., computing the best strategy in hindsight, given a sequence of implicitly defined reward vectors) can be translated in "black-box" fashion to an efficient no-α-regret online algorithm.

The existing offline-to-online reductions apply only to linear online optimization problems (Awerbuch and Kleinberg, 2008; Fujita et al., 2013; Kalai and Vempala, 2005; Kakade et al., 2009) or require an exact best-response oracle (Dudik et al., 2017; Zinkevich, 2003), and thus do not apply to the USM problem. Meanwhile, Hazan and Koren (2016) prove that there is no fully general black-box reduction: there exists a (somewhat artificial) problem such that, even with an exact oracle for the offline version of the problem, achieving sublinear regret requires a super-polynomial amount of computation. Thus for some problems, there is a fundamental difference between what is possible offline versus online. It remains an open question whether or not there is a "natural" optimization problem with a provable separation between its offline and online versions.

The basic idea is to feed the offline input f into the online algorithm over and over again, and return the best of the subsets output by the online algorithm. Since Dobzinski and Vondrák (2012) prove that offline USM is hard to approximate to within a factor better than $\frac{1}{2}$ (assuming $NP \neq RP$), the same lower bound carries over to the online version of the problem.

3. Unless otherwise noted, we assume that each submodular function f in the input: (i) has description length polynomial in n; and (ii) given a subset $S \subseteq [n]$, the value $f(S)$ of f can be evaluated in time polynomial in n. All of our results also hold in the "value oracle" model, with submodular functions given as "black boxes" that support value queries. Here, our online algorithm uses only polynomially many (in n) value queries and polynomial additional computation. The lower bound continues to apply and becomes unconditional in the value oracle model (following Feige et al. (2011)).

Online USM is arguably one of the most natural online problems where the state-of-the-art is silent on whether or not there are online guarantees matching what is possible offline, and this paper resolves this question (in the positive).

1.1. Related Work

Feige et al. (2011) were the first to rigorously study the general USM problem. They showed that a uniformly random subset S achieves a $\frac{1}{4}$-approximation (in expectation). They also provided an algorithm, based on noisy local search, with an approximation guarantee of $\frac{2}{5}$. Finally, they proved that in the value oracle model, achieving an approximation of $\frac{1}{2} + \epsilon$ requires an exponential number of queries in the worst case. The noisy local search technique was improved slightly by Oveis Gharan and Vondrák (2011) and further by Feldman et al. (2011). A breakthrough occurred when Buchbinder et al. (2015b) showed that a simple strategy could be used to achieve a (tight) $\frac{1}{2}$ approximation ratio. Their algorithm was randomized, but was later derandomized by Buchbinder and Feldman (2016). The initial lower bound in Feige et al. (2011) was generalized by Dobzinski and Vondrák (2012), who proved the same bound even for succinctly represented functions (polynomial description and evaluation time), conditioned on $RP \neq NP$.

Online submodular *minimization* is considered by Hazan and Kale (2012). Here, the offline problem can be solved exactly with a polynomial number of value queries (e.g. Grötschel et al. (1988)), and the main result in Hazan and Kale (2012) is an efficient no-regret algorithm for the online setting. Some extensions to online submodular minimization with constraints are given by Jegelka and Blimes (2011). Streeter and Golovin (2009) considered a fairly general online submodular maximization problem. In particular, their problem captures the online problem where the algorithm receives a series of monotone submodular functions and wants to maximize them subject to a knapsack constraint.

Finally, Buchbinder et al. (2015a) study a problem that they call "online submodular maximization," but where there is only a single function and the elements of the universe arrive over time. This version of the problem is in the tradition of competitive online algorithms rather than no-regret learning algorithms, and hence is quite different from the online USM problem that we study.

1.2. Our Techniques

We now provide an overview of the main ideas used in obtaining a no-$1/2$-regret algorithm for Online USM. The overall argument is divided into two phases. In the first phase, whose main result is captured in Theorem 1, we propose a general class of algorithms for the Online USM algorithm, based on the Buchbinder et al. analysis for the offline problem (Buchbinder et al., 2015b). This class is parameterized by our choice of subroutine, and the main result of this phase states that the performance of our algorithm with respect to Online USM is precisely characterized by the performance of its subroutine with respect to a specific task: the USM Balance Subproblem. Stopping here already yields a novel result: using a no-regret algorithm for the (two) experts problem, such as Multiplicative Weights (see, e.g., Cesa-Bianchi and Lugosi (2006)), as our subroutine would give us a no-$1/3$-regret algorithm for Online USM. The previously best known is returning a uniform random point in every round, which is a no-$1/4$-regret algorithm.

In the second phase of our argument, we focus our efforts on designing a good subroutine for the USM Balance Subproblem. The main result is Theorem 3 in which we prove that our proposed

subroutine satisfies the condition which results in a no-$1/2$-regret algorithm for Online USM.[4] Using any algorithm with a no-regret guarantee for the (two) experts problem is provably insufficient; for *every* such algorithm the result is an algorithm for Online USM (when using the aforementioned no-regret algorithm as a subroutine) with *linear* expected $1/2$-regret. In other words, the binary-action task we are attempting to solve really is distinct from the experts problem. Roughly speaking, the situtation in the USM Balance Subproblem is as follows. The algorithm wants to make progress, but at the same time the adversary is advancing its own goals on two different fronts. The problem is named after the need to balance the losses incurred on these two fronts; an algorithm that focuses on the experts problem only considers the total loss. To complicate matters further, one of the possible actions may have a negative value, and choosing such an action incurs two types of loss: it both sets back the algorithm while advancing the adversary's agenda. One key technical contribution is a potential-based analysis, which succinctly captures the relationship between the algorithm's state and the status of the no-regret guarantee we want to prove; much of the complexity is hidden in identifying the appropriate potential functions.

Finally, we conclude by generalizing the analysis to work against adaptive adversaries. The main contribution here is a covariance-based argument which guards against the adaptive adversary blowing up the variance of our algorithm by choosing its future inputs to depend on the results of past coin flips.

1.3. Organization

The first phase of our proof is conducted in Section 2; we provide a framework for Online USM, identify the subproblem of interest, and give our main reduction. Our proposed subroutine and main result are stated in Section 3, and the proof is carried out in Appendix A. Finally, in Appendix B, we discuss the generalization to adaptive adversaries.

2. An Online USM Framework

We begin by presenting our framework for Online USM, which is based on the BFNS offline algorithms (Buchbinder et al., 2015b).

In order to make the offline problem tractable, these algorithms transform the task of choosing a subset $S \subseteq [n]$, which has 2^n possible choices, into the n tasks of choosing whether element i should be in S or not, each of which have just two possible choices. To be more specific, we begin with two candidate solutions: X_0 as the empty set and Y_0 as the entire universe. We then proceed in n iterations. In iteration i, we want to make the two candidate solutions agree on element i. Hence we must either add i to X_{i-1} or remove i from Y_{i-1}. To decide which, we compute the marginal values of these two options according to our function f. In particular, let:

$$\alpha_i = f(X_{i-1} \cup \{i\}) - f(X_{i-1}),$$
$$\beta_i = f(Y_{i-1} \setminus \{i\}) - f(Y_{i-1}).$$

Roughly speaking, we want to favor the larger of these two values. Due to submodularity, $\alpha_i + \beta_i \geq 0$ always (since $X_{i-1} \subseteq Y_{i-1}$; the two sets agree on all elements up to $i-1$ after which

4. It is also possible to apply Blackwell's Approachability Theorem (Blackwell, 1956) to get a subroutine which satisfies this condition as well. (Thanks to anonymous reviewer for pointing this out.) We nevertheless provide our own subroutine along with its proof, as this makes the entire online USM algorithm and its analysis more explicit.

Y_{i-1} has everything and X_{i-1} has nothing). The analysis in Buchbinder et al. (2015b) shows that deterministically picking based on the larger value gives a $1/3$-approximation overall. However, randomly choosing to include i with probability $\frac{\alpha_i}{\alpha_i+\beta_i}$ and to remove i with probability $\frac{\beta_i}{\alpha_i+\beta_i}$ can[5] improve this to a $\frac{1}{2}$-approximation overall.

This suggests an online framework which uses specialized binary-action subroutines to make these smaller decisions. Algorithm 1 implements this idea, using the after-the-fact marginal values of the most recent submodular function to provide feedback to its subroutines.

Algorithm 1: Online USM Framework.

input : Subroutine A (binary-action), submodular functions $\{f^t : [n] \to [0,1]\}_t$
output: Subsets $\{S^t \subseteq [n]\}_t$
Run n copies of A: $A_1, \ldots, A_n$.
for round $t = 1$ to T **do**
  Initialize subset $X_0^t \leftarrow \{\}$ and subset $Y_0^t \leftarrow [n]$.
  for $i = 1$ to n **do**
    Ask A_i whether i should be in S^t.
    If A_i says yes, set $X_i^t \leftarrow X_{i-1}^t \cup \{i\}$ and $Y_i^t \leftarrow Y_{i-1}^t$.
    If A_i says no, set $X_i^t \leftarrow X_{i-1}^t$ and $Y_i^t \leftarrow Y_{i-1}^t \setminus \{i\}$.
  end
  Output X_n^t for round t, and receive as input the submodular function f^t.
  for $i = 1$ to n **do**
    Let $\alpha_i^t \leftarrow f^t(X_{i-1}^t \cup \{i\}) - f^t(X_{i-1}^t)$.
    Let $\beta_i^t \leftarrow f^t(Y_{i-1}^t \setminus \{i\}) - f^t(Y_{i-1}^t)$.
    Report (α_i^t, β_i^t) to A_i as the rewards for yes and no, respectively.
  end
end

What guarantees do we need on the subroutine in our framework to get a no-regret guarantee for Online USM? We present the necessary guarantees as another online problem, which we call the USM Balance Subproblem.

2.1. The USM Balance Subproblem

The USM Balance Subproblem is a binary-action online problem. In each round t, the algorithm chooses "yes" or "no" and then the adversary reveals a point (α^t, β^t). Based on the algorithm's decision and the adversary's point, three quantities are updated. The algorithm has a total accumulated reward, denoted R_{alg}. The adversary accumulates two *separate* piles of missed opportunities, which will be denoted C_{yes} and C_{no}.

The adversary's point (α^t, β^t) lies in $\mathbb{R}^2$ subject to three constraints:

- $-1 \leq \alpha^t \leq +1$,
- $-1 \leq \beta^t \leq +1$, and

5. Only necessary when both α_i and β_i are both positive. If only one value is positive, we need to always pick that choice.

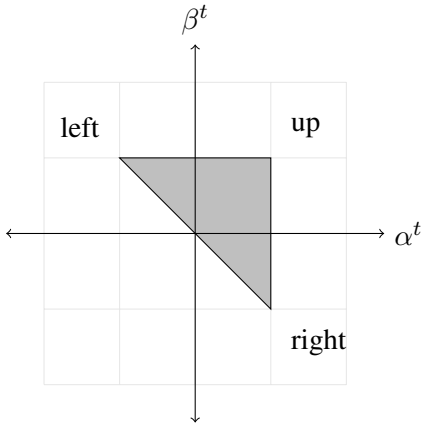

Figure 1: The USM Balance Subproblem adversary's possible moves are convex combinations of up $(+1, +1)$, right $(+1, -1)$, and left $(-1, +1)$.

- $\alpha^t + \beta^t \geq 0$.

The allowed space of points is illustrated in Figure 1. When the algorithm chooses yes, R_{alg} increases by $\frac{1}{2}\alpha^t$ and C_{no} increases by β^t. If it instead chooses no, then R_{alg} increases by $\frac{1}{2}\beta^t$ and C_{yes} increases by α^t.[6]

We say that the α-regret of an algorithm for the USM Balance Subproblem is

$$\alpha \cdot \max(C_{yes}, C_{no}) - R_{alg}.$$

As usual, we say that an algorithm has no-α-regret if its worst-case expected α-regret is bounded by $O(T^c)$ for some constant $c < 1$, with the big-O supressing any dependence on n. The worst case is still taken over the adversary's choice of points, and the expectation is over coin flips of the algorithm. If we have a no-α-regret algorithm for the USM Balance Subproblem, its R_{alg} is comparable to the better of C_{yes} and C_{no}, in expectation.

We have been building up to the following theorem reducing Online USM to the USM Balance Subproblem:

Theorem 1 *For any constant $\alpha > 0$, when given a subroutine A with $g(T)$ α-regret for the USM Balance Subproblem, Algorithm 1 has $O(n \cdot g(T))$ $\frac{\alpha}{1+\alpha}$-regret.*

Proof In this proof, we use Z_i^+ to denote the rounds where the subroutine A_i returned yes; Z_i^-, no.

For the purposes of comparison, we also track the evolution of a *third* set. For each round t, define OPT_0^t to be offline optimal set (the best fixed set over all rounds, independent of t). Let $OPT_i^t = OPT_{i-1}^t \cup \{i\}$ if $t \in Z_i^+$ and $OPT_i^t = OPT_{i-1}^t \setminus \{i\}$ if $t \in Z_i^-$. In English, OPT_i^t begins (when $i = 0$) at the optimal answer and has its entries changed to match the decisions made within round t until it finishes (when $i = n$) at the algorithm's choice: $OPT_n^t = X_n^t = Y_n^t$. Intuitively, our

6. Why is the algorithm reward seemingly half of what it should be? The heart of the matter is that because our Online USM analysis is keeping track of two candidate solutions, it winds up double-counting progress. We correct for this factor with our rewards.

Option	Increase of $f^t(X_i^t) + f^t(Y_i^t)$	Decrease of $f^t(OPT_i^t)$
Choosing i	α_i^t	0
Not Choosing i	β_i^t	$\leq \alpha_i^t$

Table 1: How values change when $i \in OPT$.

algorithm will perform well if it manages to grow $f^t(X_i^t)$ and/or $f^t(Y_i^t)$ while lowering the value of $f^t(OPT_i^t)$ relatively little in comparison.

Armed with these three evolving sets, we now want to know how the decisions of our subroutines impact their values. Suppose that in round t, the ith subroutine says yes. By construction, we know that:

- $f^t(X_i^t) = f^t(X_{i-1}^t) + \alpha_i^t$,
- $f^t(Y_i^t) = f^t(Y_{i-1}^t)$,
- if element i was in OPT, then $f^t(OPT_i^t) = f^t(OPT_{i-1}^t)$ because $OPT_i^t = OPT_{i-1}^t$,
- if element i was not in OPT, then $f^t(OPT_i^t) \geq f^t(OPT_{i-1}^t) - \beta_i^t$ by the submodularity of f^t, noting that Y_{i-1}^t is a superset of OPT_{i-1}^t.

By the same reasoning, when the ith subroutine says no, all of the following happen:

- $f^t(X_i^t) = f^t(X_{i-1}^t)$,
- $f^t(Y_i^t) = f^t(Y_{i-1}^t) + \beta_i^t$,
- if element i was not in OPT, then $f^t(OPT_i^t) = f^t(OPT_{i-1}^t)$, again because $OPT_i^t = OPT_{i-1}^t$, and
- if element i was in OPT, then $f^t(OPT_i^t) \geq f^t(OPT_{i-1}^t) - \alpha_i^t$, again by submodularity of f^t, noting that X_{i-1}^t is a subset of OPT_{i-1}^t.

Table 1 depicts these changes for the case where element i is in OPT.

Now, fix an element $i \in [n]$. Summing the first two bullets above (for both cases) over all the rounds, we have:

$$\sum_t \left[f^t(X_i^t) - f^t(X_{i-1}^t) + f^t(Y_i^t) - f^t(Y_{i-1}^t) \right] = \sum_{t \in Z_i^+} \alpha_i^t + \sum_{t \in Z_i^-} \beta_i^t. \tag{2}$$

We finish by summing the last two bullets above (for both cases) over all the rounds.

$$\begin{aligned} \sum_{t=1}^{T} \left(f^t(OPT_{i-1}^t) - f^t(OPT_i^t) \right) &\leq \begin{cases} \sum_{t \in Z_i^-} \alpha_i^t & \text{if } i \in OPT \\ \sum_{t \in Z_i^+} \beta_i^t & \text{if } i \notin OPT \end{cases} \\ &\leq \max \left(\sum_{t \in Z_i^-} \alpha_i^t, \sum_{t \in Z_i^+} \beta_i^t \right) \end{aligned} \tag{3}$$

We must now discuss an important issue before we can proceed with the proof. Suppose that our subroutine A_i is only effective against oblivious adversaries, not adaptive adversaries. We must ensure that its input (namely the sequence $(\alpha_i^t, \beta_i^t)_t$) does not depend on its output. Fortunately, this is the case. In addition to depending on the actual adversary, this sequence depends on the output of subroutines $A_1, \ldots, A_{i-1}$ over all rounds. If the actual adversary is oblivious, then it does not depend on the output of A_i. Since they come before A_i, none of $A_1, \ldots, A_{i-1}$ depend on A_i's output either (in particular, their inputs do not, so their outputs cannot either)! Put another way, the dependency graph between our subroutines is a directed acyclic graph. If this was not the case, we would have required that they be impervious to adaptive adversaries, in order to handle each other's output. Luckily, we may safely proceed with algorithms that just handle oblivious adversaries. We may now invoke the α-regret guarantee for our subroutine A_i. Written out completely, the definition of α-regret for the USM Balance Subproblem states that:

$$\alpha \cdot \mathbf{E}\left[\max\left(\underbrace{\sum_{t \in Z_i^-} \alpha_i^t}_{C_{yes}}, \underbrace{\sum_{t \in Z_i^+} \beta_i^t}_{C_{no}}\right)\right] - \mathbf{E}\left[\underbrace{\sum_{t \in Z_i^+} \frac{1}{2}\alpha_i^t + \sum_{t \in Z_i^-} \frac{1}{2}\beta_i^t}_{R_{alg}}\right] \leq g(T). \tag{4}$$

where the expectation is over the random coin flips of A_i and also $A_1, \ldots, A_{i-1}$. We now combine lines 2-4.

$$\alpha \cdot \mathbf{E}\left[\sum_{t=1}^{T} \left(f^t(OPT_{i-1}^t) - f^t(OPT_i^t)\right)\right]$$
$$-\frac{1}{2}\mathbf{E}\left[\sum_t \left[f^t(X_i^t) - f^t(X_{i-1}^t) + f^t(Y_i^t) - f^t(Y_{i-1}^t)\right]\right] \leq g(T)$$

This implements our stated plan; the growth of $f^t(X_i^t)$ and/or $f^t(Y_i^t)$ roughly dominates the amount that $f^t(OPT_i^t)$ drops. We now sum over the elements $i \in [n]$.

$$\alpha \cdot \mathbf{E}\left[\sum_{t=1}^{T} \left(\underbrace{f^t(OPT_0^t)}_{=f^t(OPT)} - \underbrace{f^t(OPT_n^t)}_{=f^t(ALG^t)}\right)\right]$$
$$-\frac{1}{2}\mathbf{E}\left[\sum_t \left[\underbrace{f^t(X_n^t)}_{=f^t(ALG^t)} - \underbrace{f^t(X_0^t)}_{=f^t(\emptyset)\geq 0} + \underbrace{f^t(Y_n^t)}_{=f^t(ALG^t)} - \underbrace{f^t(Y_0^t)}_{=f^t([n])\geq 0}\right]\right] \leq n \cdot g(T)$$

We finish with some slight rearranging.

$$\alpha \cdot \mathbf{E}\left[\sum_{t=1}^{T}\left(f^t(OPT) - f^t(ALG^t)\right)\right] - \mathbf{E}\left[\sum_t f^t(ALG^t)\right] \leq n \cdot g(T)$$

$$\alpha \cdot \sum_{t=1}^{T} f^t(OPT) - (1+\alpha) \cdot \mathbf{E}\left[\sum_t f^t(ALG^t)\right] \leq n \cdot g(T)$$

$$\frac{\alpha}{1+\alpha} \cdot \sum_{t=1}^{T} f^t(OPT) - \cdot \mathbf{E}\left[\sum_t f^t(ALG^t)\right] \leq \frac{1}{1+\alpha} \cdot n \cdot g(T)$$

Such a subroutine gives Algorithm 1 the stated $\frac{\alpha}{1+\alpha}$-regret. ■

With this reduction in hand, we can make a key observation. We claim that any no-regret algorithm for the (two) experts problem is also a no-$\frac{1}{2}$-regret algorithm for the USM Balance Subproblem. Suppose that an algorithm has $g(T)$ regret for the (two) experts problem and it says yes in rounds Z_i^+ and no in rounds Z_i^-.

$$\mathbf{E}\left[\sum_{t=1}^{T} \alpha_i^t\right] - \mathbf{E}\left[\sum_{t\in Z_i^+} \alpha_i^t + \sum_{t\in Z_i^-} \beta_i^t\right] \leq g(T)$$

$$\mathbf{E}\left[\sum_{t\in Z_i^-} \alpha_i^t\right] - \mathbf{E}\left[\sum_{t\in Z_i^-} \beta_i^t\right] \leq g(T)$$

$$\mathbf{E}\left[\sum_{t=1}^{T} \beta_i^t\right] - \mathbf{E}\left[\sum_{t\in Z_i^+} \alpha_i^t + \sum_{t\in Z_i^-} \beta_i^t\right] \leq g(T)$$

$$\mathbf{E}\left[\sum_{t\in Z_i^+} \beta_i^t\right] - \mathbf{E}\left[\sum_{t\in Z_i^+} \alpha_i^t\right] \leq g(T)$$

$$\mathbf{E}\left[\underbrace{\sum_{t\in Z_i^-} \alpha_i^t}_{=C_{yes}} + \underbrace{\sum_{t\in Z_i^+} \beta_i^t}_{=C_{no}}\right] - \mathbf{E}\left[\underbrace{\sum_{t\in Z_i^-} \beta_i^t + \sum_{t\in Z_i^+} \alpha_i^t}_{=2R_{alg}}\right] \leq 2 \cdot g(T)$$

$$\frac{1}{2} \cdot \mathbf{E}[\max(C_{yes}, C_{no})] - \mathbf{E}[R_{alg}] \leq g(T)$$

In other words, the algorithm also has $g(T)$ $\frac{1}{2}$-regret for the USM Balance Subproblem, as we claimed earlier. Combining this with Theorem 1, we have arrived at the following partial result:

Corollary 2 *When given a subroutine A with $g(T)$ regret for the (two) experts problem, Algorithm 1 has $O(n \cdot g(T))$ $1/3$-regret.*

Even without this proof, we might have expected that a claim like Corollary 2 should be true. After all, over time, a good algorithm for the two experts problem learns to pick the better (on average) expert. This corresponds to making an offline greedy decision, which according to the Buchbinder et al. (2015b) analysis is good enough to get a $1/3$-approximation. However, there are some subtleties that can occur. For example, the subroutine can sometimes make mistakes, possibly picking a negative value over a positive one sometimes. The original Buchbinder et al. (2015b) analysis did not need to account for the possibility of such events, but our proofs implicitly handle them.

3. An Optimal No-$\frac{1}{2}$-Regret Algorithm for Online USM

We have now identified a clear goal. In this section, we successfully give a no-regret algorithm for the USM Balance Subproblem. Note that this is the optimal value of α in terms of α-regret, since Theorem 1 also transforms inapproximability of Online USM (nothing better than $1/2$) into inapproximability of the USM Balance Subproblem (nothing better than 1). Due to our unusual definition of α-regret for the USM Balance Subproblem, this was nonobvious. It is perhaps suprising that such a simple algorithm manages to obtain the optimal approximation ratio; the brunt of the work is in the analysis.

Algorithm 2: USM BALANCER

Initialize $x \leftarrow \frac{1}{2}\sqrt{T}$.
for round $t = 1$ to T **do**
  Compute probability $p^t \leftarrow \frac{x}{\sqrt{T}}$.
  Choose the item with probability p^t for round t, and receive the point (α^t, β^t),
  Write (α^t, β^t) as the convex combination $c_u(+1,+1) + c_r(+1,-1) + c_\ell(-1,+1)$.
  Perform update $x \leftarrow x + (1-2p^t)c_u + c_r - c_\ell$.
  Cap x back into the interval $[0, \sqrt{T}]$.
end

Our proposed subroutine is Algorithm 2.[7] We defer the proof of its regret to Appendix A.

Theorem 3 USM BALANCER *solves the USM Balance Subproblem with* $O(\sqrt{T})$ *regret.*

Corollary 4 *Algorithm 1 has* $O(n\sqrt{T})$ $1/2$*-regret when using* USM BALANCER *as a subroutine.*

Proof We combine the guarantee about USM BALANCER given by Theorem 3 with the reduction in Theorem 1. ■

References

A. A. Ageev and M. I. Sviridenko. An 0.828 approximation algorithm for the uncapacitated facility location problem. *Discrete Applied Mathematics*, 93:149156, 1999.

7. As presented, our algorithm needs to know the time horizon T. This dependence can be removed with a standard trick: simply guess $T = 1$, and double T while restarting the algorithm everytime the current guess is violated.

B. Awerbuch and R. D. Kleinberg. Online linear optimization and adaptive routing. *Journal of Computer and System Sciences*, 74(1):97–114, 2008.

F. Bach. Learning with submodular functions: A convex optimization perspective. *Foundations and Trends in Machine Learning*, 6(2-3):145–373, 2013.

David Blackwell. An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6(1):1–8, 1956.

N. Buchbinder, M. Feldman, and R. Schwartz. Online submodular maximization with preemption. In *Proceedings of SODA*, pages 1202–1216, 2015a.

Niv Buchbinder and Moran Feldman. Deterministic algorithms for submodular maximization problems. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 392–403. Society for Industrial and Applied Mathematics, 2016.

Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. *SIAM Journal on Computing*, 44(5): 1384–1402, 2015b.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. 2006.

N. Cesa-Bianchi, Y. Mansour, and G. Stolz. Improved second-order bounds for prediction with expert advice. *Machine Learning*, 66(2–3):321–352, 2007.

Shahar Dobzinski and Jan Vondrák. From query complexity to computational complexity. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1107–1116. ACM, 2012.

M. Dudik, N. Haghtalab, H. Luo, R. E. Scahpire, V. Sygkanis, and J. Wortman Vaughan. Oracle-efficient online learning and auction design. In *Proceedings of* FOCS, 2017.

S. Dughmi. Submodular functions: Extensions, distributions, and algorithms. a survey. arXiv:0912.0322v4, 2011.

S. Dughmi, T. Roughgarden, and Mukund Sundararajan. Revenue submodularity. *Theory of Computing*, 8:95–119, 2012. Article 5.

Uriel Feige, Vahab S Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

Moran Feldman, Joseph Naor, and Roy Schwartz. Nonmonotone submodular maximization via a structural continuous greedy algorithm. *Automata, Languages and Programming*, pages 342–353, 2011.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

T. Fujita, K. Hatano, and E. Takimoto. Combinatorial online prediction via metarounding. In *Proceedings of* ALT, pages 68–82, 2013.

M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*. Springer-Verlag, Berlin, 1988.

V. Guruswami and S. Khot. Hardness of Max 3-SAT with no mixed clauses. In *Proceedings of CCC*, 2005.

E. Halperin and U. Zwick. Combinatorial approximation algorithms for the maximum directed cut problem. In *Proceedings of SODA*, pages 1–7, 2001.

J. Hartline, V. Mirrokni, and M. Sundararajan. Optimal marketing strategies over social networks. In *Proceedings of the 17th international conference on World Wide Web (WWW)*, pages 189–198, 2008.

E. Hazan and S. Kale. Online submodular minimization. *Journal of Machine Learning Research*, 13:2903–2922, 2012.

E. Hazan and T. Koren. The computational power of optimization in online learning. In *Proceedings of* STOC, 2016.

S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM*, 48(4):761–777, 2001.

S. Jegelka and J. Blimes. Online submodular minimization for combinatorial structures. In *Proceedings of* ICML, 2011.

Sham M Kakade, Adam Tauman Kalai, and Katrina Ligett. Playing games with approximation algorithms. *SIAM Journal on Computing*, 39(3):1088–1106, 2009.

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

P. Milgrom. *Putting Auction Theory to Work*. 2004.

Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1098–1116. Society for Industrial and Applied Mathematics, 2011.

A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000.

A. S. Schulz and N. A. Uhan. Approximating the least core value and least core of cooperative games with supermodular costs. *Discrete Optimization*, 10:163–180, 2013.

Matthew Streeter and Daniel Golovin. An online algorithm for maximizing submodular functions. In *Advances in Neural Information Processing Systems*, pages 1577–1584, 2009.

J. Vondrak. *Submodularity in Combinatorial Optimization*. PhD thesis, Charles University, 2007.

M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of* ICML, 2003.

Appendix A. Proof of Theorem 3

In this Appendix, we prove our guarantee for USM BALANCER.

Reminder of Theorem 3 USM BALANCER *solves the USM Balance Subproblem with* $O(\sqrt{T})$ *regret.*

Proof We need to begin by discussing expectations. The precise inequality we need to prove is actually

$$\mathbb{E}\left[\max\left(C_{yes}, C_{no}\right) - R_{alg}\right] \leq O(\sqrt{T}). \tag{5}$$

However, we would rather prove this inequality:

$$\max\left(\mathbb{E}C_{yes}, \mathbb{E}C_{no}\right) - \mathbb{E}R_{alg} \leq O(\sqrt{T}). \tag{6}$$

We wish we could use linearity of expectation to make Inequalities 5 and 6 equivalent. However, the guarantee we want to prove has a max inside the expectation, so we cannot freely swap the two. Our first task is to show that Inequality 6 is sufficient.

We now argue that the two random variables may as well have the same expectation. Assume without loss of generality that $\mathbb{E}C_{yes} \geq \mathbb{E}C_{no}$. Let C'_{no} be a random variable equal to $C_{no} + \mathbb{E}C_{yes} - EC_{no}$, so it has mean $\mathbb{E}C_{yes}$ as well. This inequality is hence stronger than Inequality 5:

$$\mathbb{E}\left[\max\left(C_{yes}, C'_{no}\right) - R_{alg}\right] \leq O(\sqrt{T}).$$

However, Inequality 6 is equivalent to:

$$\max\left(\mathbb{E}C_{yes}, \mathbb{E}C'_{no}\right) - \mathbb{E}R_{alg} \leq O(\sqrt{T}).$$

We want to prove the following, so that we can add it to the latter to get the former:

$$\mathbb{E}\max\left(C_{yes}, C'_{no}\right) - \mathbb{E}C_{yes} \leq O(\sqrt{T}).$$

Let $(x)^+$ denote the positive part of x, i.e., $(x)^+ = \max(0, x)$. We prove the following stronger statement:

$$\begin{aligned} \mathbb{E}\left(\max\left(C_{yes}, C'_{no}\right) - \mathbb{E}C_{yes}\right)^+ &\leq O(\sqrt{T}) \\ \mathbb{E}\left(\max\left(C_{yes} - \mathbb{E}C_{yes}, C'_{no} - \mathbb{E}C_{yes}\right)\right)^+ &\leq O(\sqrt{T}). \end{aligned} \tag{7}$$

Fortunately, against an oblivious adversary, C_{yes} is a weighted (all weights are at most a constant) sum of independent Bernoulli random variables (the coin tosses we perform each round based on p^t). They are independent since the adversary must fix a sequence up front, to which our online algorithm always chooses the same probabilities p^t for. Since variances add over independent variables, this means the variance of C_{yes} is $O(T)$. Similarly, the variance of C_{no} is $O(T)$ as well

(although the two are not independent, since they use the same coins). We then apply Jensen's inequality to transform our variance bounds into bounds on the expected amount variables may exceed their means.

$$\mathbb{E}\left[(C_{yes} - \mathbb{E}C_{yes})^2\right] \leq O(T) \qquad \mathbb{E}\left[(C_{no} - \mathbb{E}C_{no})^2\right] \leq O(T)$$
$$\mathbb{E}\left[|C_{yes} - \mathbb{E}C_{yes}|\right] \leq O(\sqrt{T}) \qquad \mathbb{E}\left[|C_{no} - \mathbb{E}C_{no}|\right] \leq O(\sqrt{T})$$
$$\mathbb{E}\left[(C_{yes} - \mathbb{E}C_{yes})^+\right] \leq O(\sqrt{T}) \qquad \mathbb{E}\left[(C_{no} - \mathbb{E}C_{no})^+\right] \leq O(\sqrt{T})$$

Since C'_{no} is just a translated version of C_{no}, this guarantee holds for C'_{no} as well. This is now good enough to prove Inequality 7, because for any two positive numbers x, y, we know that $\max(x, y) \leq x + y$.

We have now finished justifying why Inequality 6 is sufficient, and can proceed to the main proof. Our strategy is as follows. We do not try to analyze R_{alg}, C_{yes}, and C_{no} by themselves. Instead, we add the potential functions Φ_{alg}, Φ_{yes}, and Φ_{no} to them, respectively. We will show that the algorithm's sum is at least as much as the better of the adversary's two sums. Here are our three potential functions and their derivatives:

- $\Phi_{alg}(x) = \frac{\sqrt{T}}{8} - \frac{1}{8}\frac{(2x-\sqrt{T})^2}{\sqrt{T}}$ with derivative $\Phi'_{alg}(x) = -\frac{1}{2}\frac{(2x-\sqrt{T})}{\sqrt{T}} = \frac{1}{2}(1-2p^t)$.
- $\Phi_{yes}(x) = \frac{1}{2}\frac{(\sqrt{T}-x)^2}{\sqrt{T}}$ with derivative $\Phi'_{yes}(x) = \frac{(x-\sqrt{T})}{\sqrt{T}} = (p^t - 1)$.
- $\Phi_{no}(x) = \frac{1}{2}\frac{x^2}{\sqrt{T}}$ with derivative $\Phi'_{no}(x) = \frac{x}{\sqrt{T}} = p^t$.

The potential functions depend on the algorithm's current value for x. Since the algorithm maintains x to be in the interval $[0, \sqrt{T}]$, these potential functions always fall in the range $[0, \frac{\sqrt{T}}{2}]$. Since all potentials are bounded in magnitude by $O(\sqrt{T})$, it suffices to prove the following, which we attempt to maintain as an invariant over steps:

$$\mathbb{E}(R_{alg} + \Phi_{alg}) + O(\sqrt{T}) \geq \max\left(\mathbb{E}(C_{yes} + \Phi_{yes}), \mathbb{E}(C_{no} + \Phi_{no})\right). \tag{8}$$

There are only two ways that the algorithm affects rewards, costs, or potentials. The first way is that the algorithm may cap x back into the interval $[0, \sqrt{T}]$. Since this does not involve interaction with the adversary, only the potential functions change in value. However, Φ_{alg} is a quadratic which is maximized at $x = \sqrt{T}/2$, Φ_{yes} is a quadratic which is minimized at $x = \sqrt{T}$, and Φ_{no} is a quadratic which is minimized at $x = 0$. Hence, capping x only moves it closer to the maximum (resp. minimum) of one of these functions, and so only increases (resp. decreases) the function value, in our favor. Hence, this process maintains Invariant 8.

The second way is that the algorithm chooses the item with probability p^t and interacts with the adversary. This results in changes to the rewards and costs as well as an update to x, which changes the potentials. Recall that the adversary's possible points are depicted in Figure 1, and that the adversary's point is always a convex combination of three extremal choices: right $(+1, -1)$, left $(-1, +1)$ and up $(+1, +1)$.

We need to understand how the potential functions change as x is updated. Notice that the update to x never changes it by more than 1. We observe that when x changes by at most 1, all the

potential derivatives change by at most $\frac{1}{\sqrt{T}}$. Formally, let there be a constant δ such that $|\delta| \leq 1$.

$$\begin{aligned}
\Phi'_{alg}(x+\delta) - \Phi'_{alg}(x) &= \left[-\frac{1}{2}\frac{(2(x+\delta)-\sqrt{T})}{\sqrt{T}}\right] - \left[-\frac{1}{2}\frac{(2x-\sqrt{T})}{\sqrt{T}}\right] \\
&= -\frac{\delta}{\sqrt{T}} \\
\Phi'_{yes}(x+\delta) - \Phi'_{yes}(x) &= \left[\frac{(x+\delta-\sqrt{T})}{\sqrt{T}}\right] - \left[\frac{(x-\sqrt{T})}{\sqrt{T}}\right] \\
&= \frac{\delta}{\sqrt{T}} \\
\Phi'_{no}(x+\delta) - \Phi'_{no}(x) &= \left[\frac{x+\delta}{\sqrt{T}}\right] - \left[\frac{x}{\sqrt{T}}\right] \\
&= \frac{\delta}{\sqrt{T}}
\end{aligned}$$

We can now approximate the amount that the potentials themselves change. Let Φ be one of the potential functions, and remember that $|\delta| \leq 1$.

$$\begin{aligned}
\Phi(x+\delta) - \Phi(x) &= \int_x^{x+\delta} \Phi'(y)dy \\
&= \int_x^{x+\delta} \left(\Phi'(x) \pm \frac{1}{\sqrt{T}}\right) dy \\
&= (x+\delta-x)\left(\Phi'(x) \pm \frac{1}{\sqrt{T}}\right) \\
&= \delta \cdot \Phi'(x) \pm \frac{1}{\sqrt{T}}
\end{aligned}$$

Suppose the adversary chooses the extreme point "right" $(+1, -1)$. Then R_{alg} increases by $\frac{1}{2}(2p^t - 1)$, C_{yes} increases by $(1 - p^t)$, and C_{no} increases by $-p^t$. Our algorithm responds by increasing x by 1, which affects the potentials according to our previous analysis.

$$\begin{aligned}
\Phi_{alg}(x+1) - \Phi_{alg}(x) &= 1 \cdot \Phi'_{alg}(x) \pm \frac{1}{\sqrt{T}} \\
&= \frac{1}{2}(1-2p^t) \pm \frac{1}{\sqrt{T}} \\
\Phi_{yes}(x+1) - \Phi_{yes}(x) &= 1 \cdot \Phi'_{yes}(x) \pm \frac{1}{\sqrt{T}} \\
&= (p^t - 1) \pm \frac{1}{\sqrt{T}} \\
\Phi_{no}(x+1) - \Phi_{no}(x) &= 1 \cdot \Phi'_{no}(x) \pm \frac{1}{\sqrt{T}} \\
&= p^t \pm \frac{1}{\sqrt{T}}
\end{aligned}$$

In other words, temporarily ignoring our $\pm\frac{1}{\sqrt{T}}$ error bounds, the changes to the potential functions cancel with the changes to the rewards and costs with respect to the sums in Invariant 8. By symmetry, the same happens when the adversary chooses the extreme point "left" $(-1, +1)$; everything cancels except for the error terms.

The only remaining extreme point is "up" $(+1, +1)$. For this case, R_{alg} increases by $\frac{1}{2}$, C_{yes} increases by $(1-p^t)$, and C_{no} increases by p^t. Our algorithm responds by changing x by $(1-2p^t)$. This again affects the potentials.

$$\Phi_{alg}(x+1-2p^t) - \Phi_{alg}(x) = \frac{1}{2}(1-2p^t)^2 \pm \frac{1}{\sqrt{T}}$$
$$\Phi_{yes}(x+1-2p^t) - \Phi_{yes}(x) = (1-2p^t)(p^t-1) \pm \frac{1}{\sqrt{T}}$$
$$\Phi_{no}(x+1-2p^t) - \Phi_{no}(x) = (1-2p^t)p^t \pm \frac{1}{\sqrt{T}}$$

The net effect, hiding error terms, is that $R_{alg} + \Phi_{alg}$ increases by $\frac{1}{2}(1 + (1-2p^t)^2) \geq \frac{1}{2}$, while $C_{yes} + \Phi_{yes}$ increases by $2p^t(1-p^t) \leq \frac{1}{2}$ and $C_{no} + \Phi_{no}$ also increases by $2p^t(1-p^t) \leq \frac{1}{2}$. Hence for this move we maintain Invariant 8, not accounting for error terms.

We have maintained the invariant for the three extremal moves. However, all other adversary moves are just convex combinations of these three moves, and the algorithm reacts with a convex combination of the appropriate replies. Hence the invariant is maintained for all of the adversary's choice of move.

It remains to briefly discuss the $\pm\frac{1}{\sqrt{T}}$ error we pick up. We pick up this error each round, and there are T total rounds, so the total error regarding our rewards, costs, and potentials is $\pm\sqrt{T}$. We still get Invariant 8, but have to increase the constant in the $O(\sqrt{T})$ term by one. Since the invariant was enough to finish the proof, we are now done. ■

Appendix B. Adaptive Adversaries

When analyzing an online algorithm, we may consider oblivious adversaries or adaptive adversaries. Oblivious adversaries fix the entire input sequence up front, unable to react to the decisions of the online algorithm. On the other hand, adaptive adversaries can choose the next piece of the input to depend on what the algorithm has done so far. For example, for the standard experts problem, the multiplicative weights algorithm works even against adaptive adversaries (Kalai and Vempala, 2005).

Our techniques work against adaptive adversaries as well. Our framework for Online USM simply deterministically decomposes the problem. Deterministic algorithms are not affected by the oblivious/adaptive swap, because even an oblivious adversary knows what the deterministic online algorithm will do and can hence simulate an adaptive adversary. If we use multiplicative weights as the subroutine for our framework, then we inherit its immunity to adaptive adversaries when producing a no $\frac{1}{3}$-regret algorithm.

Is our USM BALANCER capable of handling adaptive adversaries as well? It turns out that the answer is yes. We do not need to make any changes to the algorithm, but fixing the proof is a little tricky. We will need to move away from expectations, because the coin flips of our algorithm and adversarial input sequence are now intertwined. We begin by observing that we really managed

to prove the following invariant (with some slight rearranging), which is true for any adversarial sequence:

$$\max\left(\sum_{t=1}^{T} p^t\beta^t + \Phi_{no}, \sum_{t=1}^{T}(1-p^t)\alpha^t + \Phi_{yes}\right) - \left(\sum_{t=1}^{T}\frac{1}{2}p^t\alpha^t + \sum_{t=1}^{T}\frac{1}{2}(1-p^t)\beta^t + \Phi_{alg}\right) \leq O(\sqrt{T})$$

and we want to wind up with the following regret guarantee in expectation over the coin flips of the algorithm:

$$\max\left(\sum_{t\in Z_i^+}\beta^t, \sum_{t\in Z_i^-}\alpha^t\right) - \left(\sum_{t\in Z_i^+}\frac{1}{2}\alpha^t + \sum_{t\in Z_i^-}\frac{1}{2}\beta^t\right) \leq O(\sqrt{T}).$$

As we have suggestively hinted at by lining up matching terms, the difference between these guarantees boils down to the following question: how much do we expect a sum of Bernoulli variables to differ from their means? The issue we are faced with is that an adaptive adversary may choose the mean of a future variable to depend on the result of a past variable. Nevertheless, we show that even under this condition Bernoulli variables minus their means will have covariance zero (note that the later means are random variables as well):

Lemma 5 *Consider two random variables X_1, X_2 determined by the following process:*

1. *An adversary selects a probability $p_1 \in [0,1]$.*
2. *X_1 is drawn as a Bernoulli variable with mean p_1.*
3. *The adversary looks at X_1 and then selects a probability $p_2 \in [0,1]$.*
4. *X_2 is drawn as a Bernoulli varaible with mean p_2.*

Then the covariance of $X_1 - p_1$ and $X_2 - p_2$ is zero.

Proof We first use the definition of covariance and simplify, noting that X_1 and p_1 have the same expectation, as do X_2 and p_2.

$$\begin{aligned}\text{cov}(X_1 - p_1, X_2 - p_2) &= \mathbb{E}\left[\left((X_1 - p_1) - \mathbb{E}\left[X_1 - p_1\right]\right)\left((X_2 - p_2) - \mathbb{E}\left[X_2 - p_2\right]\right)\right] \\ &= \mathbb{E}\left[\left(X_1 - p_1\right)\left(X_2 - p_2\right)\right]\end{aligned}$$

Next, we note that the expected value of $X_2 - p_2$ is always zero even if we condition on the values of X_1 and p_1.

$$\begin{aligned}\mathbb{E}\left[\left(X_1 - p_1\right)\left(X_2 - p_2\right) \mid X_1, p_1\right] &= 0 \\ \mathbb{E}\left[\left(X_1 - p_1\right)\left(X_2 - p_2\right)\right] &= 0 \\ \text{cov}(X_1 - p_1, X_2 - p_2) &= 0\end{aligned}$$

This completes the proof. ■

As a result of Lemma 5 we can bound the difference between matching sums of our invariant and desired regret guarantee. For example, consider the two sums $\sum_{t=1}^{T} p^t \beta^t$ and $\sum_{t \in Z_i^+} \beta^t$. What is the expected difference between them? We know that the variance of a single term in the sum is $O(1)$, since each term is the difference between a Bernoulli random variable and its mean, times a value β^t which is at most one. By Lemma 5, any pair of differences has covariance zero. Hence the overall variance between these sums is the desired $O(T)$. We again apply Jensen's to turn a variance bound into a bound on the expected deviation.

$$\mathbb{E}\left[\left(\sum_{t \in Z_i^+} \beta^t - \sum_{t=1}^{T} p^t \beta^t\right)^2\right] \leq O(T)$$

$$\mathbb{E}\left[\left|\sum_{t \in Z_i^+} \beta^t - \sum_{t=1}^{T} p^t \beta^t\right|\right] \leq O(\sqrt{T})$$

$$\mathbb{E}\left[\left(\sum_{t \in Z_i^+} \beta^t - \sum_{t=1}^{T} p^t \beta^t\right)^+\right] \leq O(\sqrt{T})$$

However, there was nothing special about this pair of sums, so we get similar inequalities for the other three pairs of sums.

$$\mathbb{E}\left[\left(\sum_{t \in Z_i^-} \alpha^t - \sum_{t=1}^{T} (1 - p^t) \alpha^t\right)^+\right] \leq O(\sqrt{T})$$

$$\mathbb{E}\left[\left(-\sum_{t \in Z_i^+} \frac{1}{2} \alpha^t + \sum_{t=1}^{T} \frac{1}{2} p^t \alpha^t\right)^+\right] \leq O(\sqrt{T})$$

$$\mathbb{E}\left[\left(-\sum_{t \in Z_i^-} \frac{1}{2} \beta^t + \sum_{t=1}^{T} \frac{1}{2} (1 - p^t) \beta^t\right)^+\right] \leq O(\sqrt{T})$$

We conclude by again noting that for any positive numbers x, y, $\max(x, y) \leq x + y$, which lets us swap expectation and max as before. The final step is noting that by construction, potentials are always $O(\sqrt{T})$ in magnitude.