MaxHedge: Maximising a Maximum Online

Stephen Pasteris
University College London
London, UK
s.pasteris@cs.ucl.ac.uk

Fabio Vitale
Sapienza University
Italy & INRIA Lille, France
fabio.vitale@inria.fr

Kevin Chan
Army Research Lab
Adelphi, MD, USA
kevin.s.chan.civ@mail.mil

Shiqiang Wang
IBM Research
Yorktown Heights, NY, USA
wangshiq@us.ibm.com

Mark Herbster
University College London
London, UK
m.herbster@cs.ucl.ac.uk

Abstract

We introduce a new online learning framework where, at each trial, the learner is required to select a subset of actions from a given known action set. Each action is associated with an energy value, a reward and a cost. The sum of the energies of the actions selected cannot exceed a given energy budget. The goal is to maximise the cumulative *profit*, where the profit obtained on a single trial is defined as the difference between the *maximum reward* among the selected actions and the *sum of their costs*. Action energy values and the budget are known and fixed. All rewards and costs associated with each action change over time and are revealed at each trial only after the learner's selection of actions. Our framework encompasses several online learning problems where the environment changes over time; and the solution trades-off between minimising the costs and maximising the *maximum* reward of the selected subset of actions, while being constrained to an action energy budget. The algorithm that we propose is efficient and general that may be specialised to multiple natural online combinatorial problems.

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89. Copyright 2019 by the author(s).

1 Introduction

In this paper we propose a novel online framework where learning proceeds in a sequence of trials and the goal is to select, at each trial, a *subset of actions* maximising a profit while taking into account a certain constraint. More precisely, we are given a finite set of actions enumerated from 1 to n. Each action is associated with three values: (i) a *cost*, (ii) an amount of *energy* (both of which are required to perform the given action), as well as (iii) a *reward*. Both the cost and the reward associated with each action may change over time, are unknown at the beginning of each trial, and their values are all revealed after the learner's selection. The energy associated with each action is instead known by the learner and does not change over time. At each trial, the learner is required to select a subset of actions such that the *sum* of their energies does not exceed a *fixed energy budget*. The goal of the learner is to maximise the cumulative *profit*, where the profit obtained on a single trial is defined as the difference between the *maximum reward* among the selected actions and the *sum of their costs*. We denote by T the total number of trials.

Our framework is general and flexible in the sense that it encompasses several online learning problems. In the general case, the main challenge lies in the fact that the rewards are not known at the beginning of each trial, and the learner's profit depends only on the *maximum* reward among the selected subset of actions, instead of the *sum* of all their rewards. In particular, it is worth mentioning three different problems which can be seen as special cases of our online learning framework; these are variants of the Facility Location problem, the 0-1 Knapsack problem, and the Knapsack Median problem.

When all action energy values are equal to 0, we obtain an online learning variant of the Facility Location problem (see, e.g., [4], [21], [18]). The goal of this specific problem may be viewed as selecting and opening a subset of facilities at each given trial, to service a sequence of users which arrive one at a time. At any given trial t, each action's cost may be interpreted as the cost for opening a facility for the t-th user. The reward associated with a given facility represents what the user potentially gains when it is opened. More specifically, the rewards can be seen as quantities dependent on the distance between the user and the facilities in some metric space. In this context, it is reasonable to assume that the profit obtained on a single trial depends *only on the maximum reward* among the opened facilities taking into account the facility costs. This represents a natural setting for the Facility Location problem, because in practical scenarios users may arrive sequentially and each arrival requires a connection to an open facility. Online versions of the Facility Location problem have been studied in several contexts (see, e.g., [5], [20]). However, as far as we are aware, our work is the first study of this specific online setting for the Facility Location problem. Moreover, this dynamic model is natural and interesting within this context[1], because in practice the location of the next user, which in turn determines the reward associated with each facility, is often unknown to the learner. At each trial, after the user's location is known, the rewards are then revealed, because disclosing the user's location enables to compute all distances between the facilities and the user, which are previously unknown. In fact, this corresponds to assuming all rewards are revealed at the end of the trial.

When, instead, all rewards are equal to 0 and all costs are negative, the problem can be seen as an online learning variant of the 0-1 Knapsack problem (see, e.g., [6], [19]). In this case, each action corresponds to an item whose weight is equal to the associated energy, the energy budget represents the knapsack capacity, and the absolute value of each action cost can be viewed as the corresponding item value. Our formulation makes the problem challenging especially because the item values are revealed only after the learner's selection.

Finally, when all costs are equal to 0, we obtain an online learning variant of the Knapsack Median problem ([1], [15]). In this problem (which is a generalization of the k-median problem – see, e.g., [2], [10]), we are given a set of clients and facilities, as well as a distance metric. The goal is to open a subset of facilities such that the total connection cost (distance to nearest open facility) of all clients is minimized, while the sum of the open facility weights is limited by a budget threshold.

[1]Similar arguments hold also for motivating the other two special cases mentioned in this section.

In our framework, at each trial the rewards can express the closeness of each facility, and the action energy budget represents therefore the threshold of sum of the opened facility weights.

This problem has practical applications in multiple domains. In computer networks, network administrators may want to understand where to place network monitors or intrusion detection systems. Network packets or malicious attacks are related to the events playing a crucial role in this scenario, and a limited amount of network resources are available to detect or observe network behavior. Another class of application examples include municipal emergency services. A service center needs to deploy responders (police, paramedics, fire rescue), and with limited resources, personnel must be deployed sparingly. A further application is related to deploying program instances in a distributed computing environment (e.g. distributed cloud). These systems must respond to user requests and are subject to constraints. Both costs and rewards may be unknown in practical real-world scenarios at the beginning of each trial.

For our general problem we propose and rigorously analyse a very scalable algorithm called MaxHedge based on the complex interplay between satisfying the energy budget constraint and bounding the profit by a concave function, which in turn is related to the online gradient descent algorithm. Moreover, the total time required per trial by our learning strategy is quasi-linear in n. We measure the performance of proposed solution with respect to the difference between its cumulative profit and a discounted cumulative profit of the best fixed subset of actions.

In summary, our framework captures several real-world problems, where the environment changes over time and the solutions trade-off between minimising the costs and maximising the *maximum* reward among the selected subset of actions, while being constrained to an action energy budget. The framework is very general and the proposed algorithm is very efficient and may be specialised to several natural online combinatorial problems. Finally we provide a guarantee on the rewards achieved and costs incurred as compared to the best fixed subset of actions.

1.1 Related Work

The closest work to our online learning framework is perhaps addressed in [13], where the authors describe an online learning algorithm for structured concepts that are formed by components. Each component and each concept can be respectively seen as an action and a feasible subset of actions. Despite several similarities, the algorithm they proposed cannot be used when the rewards are non-zero, because we focus on the *maximum* reward among the selected subsets of actions, whereas in [13] the profit corresponds to the *sum* of the rewards of all selected components/actions. When all rewards are instead equal to 0, then we have the online variant of the 0-1 Knapsack problem described above as one of the three special cases of our general problem. In Appendix C we prove that if the algorithm presented in [13] could handle the Knapsack problem, then the classical version of the Knapsack problem could be solved in polynomial time, which therefore implies that it cannot address this problem unless $P = NP$.

The Hedge Algorithm described in [7] obtains a regret linear in n. However, as we have exponentially many possible subsets of selected actions, a vanilla application of Hedge would require an exponential amount of time and space to solve our problem.

Another class of problems and algorithms that are not far from ours is represented by online decision problems where efficient strategies use, as a subroutine, an approximation algorithm for choosing a concept to maximise an inner product ([8], [12], [11]). Again, these learning strategies cannot handle the case of non-zero reward. They also have a significantly higher time complexity than MaxHedge in the case of zero reward.

In [3] , [9], and [22], the authors address the problem of online maximisation of non-negative monotone submodular functions. Although our profit is submodular, it is not necessarily either non-negative or monotone. However, as we shall show in Appendix D, we could, for the facility location special case, combine much of the mechanics of our paper with the second algorithm of [3], essentially doing gradient ascent with the exact expected profit instead of an approximate expected profit (as is done in MaxHedge). Even though this new algorithm could be as efficient as the one presented in our paper, its theoretical guarantees are worse. Note that in the Knapsack and Knapsack Median special cases, the profit is indeed monotone and non-negative. For these special cases, the approach presented in [22] comes close to solving the problem, but only guarantees that the expectation of the total energy does not exceed the budget rather than the actual total energy. In addition, for the Knapsack Median problem, [22] uses quadratic time and space. As far as we are aware, there does not exist any trivial reduction to use [3] or [9] for these special cases.

Finally, in [16] the authors address a problem of online minimisation of submodular function. Our paper maximises, instead of minimising, a submodular function (the profit). This is a very different problem.

Some of the techniques behind the development of MaxHedge were inspired by [25].

2 Preliminaries and Problem Setup

2.1 Preliminaries

Vectors in $\mathbb{R}^n$ will be indicated in bold. Given a vector $\boldsymbol{v} \in \mathbb{R}^n$ we define v_i to be its i-th component. Given vectors $\boldsymbol{v}, \boldsymbol{x} \in \mathbb{R}^n$ define $\langle \boldsymbol{v}, \boldsymbol{x} \rangle := \sum_{i=1}^n v_i x_i$. Define $\mathcal{P}$ to be the set of n-dimensional real vectors in which every component is non-negative. Given a closed convex set $C \subseteq \mathbb{R}^n$ and a vector $\boldsymbol{x} \in \mathbb{R}^n$, we define $\Pi_C(\boldsymbol{x})$ as the projection (under the Euclidean norm) of $\boldsymbol{x}$ onto C. Let $\mathbb{N}$ be the set of the positive integers. For $l \in \mathbb{N}$ define $[l] = \{1, 2, 3, ..., l\}$. Given an event E let $\neg E$ be the event that E does *not* occur. Let $\mathbb{P}(E)$ be the probability that event E occurs. For a random variable Y, let $\mathbb{E}(Y)$ be the expected value of Y. Given a differentiable function $h : \mathbb{R}^n \to \mathbb{R}$ and a vector $\boldsymbol{x} \in \mathbb{R}^n$ let $\nabla h(\boldsymbol{x})$ be the derivative of h evaluated at $\boldsymbol{x}$. Let $\partial_i h(\boldsymbol{x})$ be the i-th component of $\nabla h(\boldsymbol{x})$.

2.2 Problem Setup

In this section we formally define our problem. We have a set of n actions enumerated from 1 to n. Each action i has an *energy* $z_i \in [0, \beta]$ for some $\beta < 1$, and on each trial t each action i has a *cost* $c_i^t \in \mathbb{R}$ (which can be negative) and a *reward* $r_i^t \in \mathbb{R}^+$. The learner knows $\boldsymbol{z}$, but $\boldsymbol{c}^t$ and $\boldsymbol{r}^t$ are revealed to the learner only at the end of trial t. On each trial t the learner has to select a set $X^t \subseteq [n]$ of actions, such that the total energy $\sum_{i \in X^t} z_i$ of the selected actions is no greater than 1. In selecting the set X^t, the learner pays a cost equal to $\sum_{i \in X^t} c_i^t$. On each trial t the learner then receives the maximum reward $\max_{i \in X^t} r_i^t$, over all actions selected (defined as equal to zero if X^t is empty). Hence, the *profit* obtained by the learner on trial t is equal to $\max_{i \in X^t} r_i^t - \sum_{i \in X^t} c_i^t$.

Formally, this online problem can be defined as follows: We have a vector $\boldsymbol{z} \in \mathcal{P}$ known to the learner. On trial t:

1. Nature selects vectors $\boldsymbol{c}^t \in \mathbb{R}^n$ and $\boldsymbol{r}^t \in \mathcal{P}$ (but does not reveal these vectors to learner)
2. Learner selects a set $X^t \subseteq [n]$ with $\sum_{i \in X^t} z_i \leq 1$
3. Learner obtains profit:
$$\mu^t(X^t) := \max_{i \in X^t} r_i^t - \sum_{i \in X^t} c_i^t$$
4. $\boldsymbol{c}^t$ and $\boldsymbol{r}^t$ are revealed to the learner.

In this paper we write, for a trial t, the cost vector $\boldsymbol{c}^t$ as the sum $\boldsymbol{c}^{t+} + \boldsymbol{c}^{t-}$ where $c_i^{t+} := \max\{0, c_i^t\}$ and $c_i^{t-} := \min\{0, c_i^t\}$, i.e. $\boldsymbol{c}^{t+}$ and $\boldsymbol{c}^{t-}$ are the positive and negative parts of the cost vector, respectively.

In order to bound the cumulative profit of our algorithm we, for some $\alpha, \delta \in [0,1]$, some set $S \subseteq [n]$ and some trial t, define the (α,δ)-*discounted* profit $\hat{\mu}^t_{\alpha,\delta}(S)$ as:

$$\hat{\mu}^t_{\alpha,\delta}(S) := \alpha \max_{i \in S} r_i^t - \alpha \sum_{i \in S} c_i^{t-} - \delta \sum_{i \in S} c_i^{t+}$$

which would be the profit obtained on trial t if we selected the subset of actions S, and all the rewards and negative costs were multiplied by α and all positive costs were multiplied by δ.

In this paper we provide a randomised quasi-linear time (per trial) algorithm MaxHedge that, for any set $S \subseteq [n]$ with $\sum_{i \in S} z_i \leq 1$, obtains an expected cumulative profit bounded below by:

$$\mathbb{E}\left(\sum_{t=1}^{T} \mu^t(X^t)\right) \geq \sum_{t=1}^{T} \hat{\mu}^t_{\alpha,\delta}(S) - n\sqrt{2T}\delta(\hat{r} + \hat{c})$$

where $\delta := \left(1 - \sqrt{\beta}\right)^2$, $\alpha := 1 - \exp\left(-\left(1 - \sqrt{\beta}\right)^2\right)$, $\hat{r} := \max_{t \in [T], i \in [n]} r_i^t$ and $\hat{c} := \max_{t \in [T], i \in [n]} |c_i^t|$.

This paper is structured as follows. In Section 3 we introduce the algorithms that define MaxHedge. In Section 4 we prove that the sets of actions selected by MaxHedge are feasible. In Section 5 we prove the above bound on the cumulative profit. In Section 6 we give special cases of the general problem.

3 Algorithms

We now present our learning strategy MaxHedge, describing the two subroutines "Algorithm 1" and "Algorithm 2" (see the pseudocode below). MaxHedge maintains a vector $\boldsymbol{\omega} \in C$ where $C := \{\boldsymbol{x} \in [0,1]^n : \langle \boldsymbol{x}, \boldsymbol{z} \rangle \leq 1\}$. We define $\boldsymbol{\omega}^t$ to be the vector $\boldsymbol{\omega}$ at the start of trial t. We initialise $\boldsymbol{\omega}^1 \leftarrow \boldsymbol{0}$. On trial t MaxHedge (randomly) constructs X^t from $\boldsymbol{\omega}^t$ using Algorithm 1. After receiving $\boldsymbol{r}^t$ and $\boldsymbol{c}^t$ MaxHedge updates $\boldsymbol{\omega}$ (from $\boldsymbol{\omega}^t$ to $\boldsymbol{\omega}^{t+1}$) using Algorithm 2. Algorithm 2 also uses a *"learning rate"* $\hat{\eta}^t$ which is defined from $\hat{\eta}^{t-1}$. We define $\hat{\eta}^0 := \infty$.

Algorithm 1 operates with a partition of all possible actions and, for each set in this partition, Algorithm 1 draws a certain subset of actions from it. Given a set in the partition, the number of actions drawn from it and the probability distribution governing the draws depends on β and $\boldsymbol{\omega}^t$. The subset, X^t, of actions selected by Algorithm 1 satisfies the following three crucial properties (proved in sections 4 and 5):

- The total energy of all actions selected is no greater than 1.
- Given an arbitrary set $Z \subseteq [n]$, the probability that X^t and Z intersect is lower bounded by $1 - \exp\left(-\delta \sum_{i \in Z} \omega_i^t\right)$.
- Given an action i, the probability that it is selected on trial t is upper bounded by $\delta \omega_i^t$.

In the analysis we shall construct, from $\boldsymbol{r}^t$ and $\boldsymbol{c}^t$, a convex function $h^t : C \to \mathbb{R}$. Using the second and third properties (given above) of Algorithm 1, we show that $h^t(\boldsymbol{\omega}^t)$ is an upper bound on the negative of the expected profit on trial t. Algorithm 2 computes the gradient $\boldsymbol{g}^t := \nabla h^t(\boldsymbol{\omega}^t)$ and updates $\boldsymbol{\omega}$ using online gradient descent on C.

The last line of Algorithm 2 requires us to project (with Euclidean distance) the vector $\boldsymbol{y}^t$ onto the set C, i.e. we must compute the $\boldsymbol{x}$ that minimises the value $\|\boldsymbol{x} - \boldsymbol{y}^t\|$ subject to $\boldsymbol{x} \in C$. Note that minimising $\|\boldsymbol{x} - \boldsymbol{y}^t\|$ is equivalent to minimising $\|\boldsymbol{x} - \boldsymbol{y}^t\|^2 = \langle \boldsymbol{x}, \boldsymbol{x} \rangle - 2\langle \boldsymbol{y}^t, \boldsymbol{x} \rangle + \langle \boldsymbol{y}^t, \boldsymbol{y}^t \rangle$, which is in turn equivalent to minimising $\langle \boldsymbol{x}, \boldsymbol{x} \rangle - 2\langle \boldsymbol{y}^t, \boldsymbol{x} \rangle$. The constraints defining the set C then imply that this projection is a case of the continuous bounded quadratic knapsack problem which can be solved in linear time (see, e.g., [14]).

The bottleneck of the algorithms is hence the ordering step in Algorithm 2 which takes a time of $\mathcal{O}(n \log(n))$

Algorithm 1 Constructing X^t

```
1: $C \leftarrow \{\boldsymbol{x} \in [0,1]^n : \langle \boldsymbol{x}, \boldsymbol{z} \rangle \leq 1\}$
2: $\beta \leftarrow \max_{i \in [n]} z_i$
3: $\tau \leftarrow 1 - \sqrt{\beta}$
4: $\delta \leftarrow \left(1 - \sqrt{\beta}\right)^2$.
5: $\Gamma \leftarrow \{q \in \mathbb{N} : \exists i \in [n] \text{ with } \tau^q \beta < z_i \leq \tau^{q-1}\beta\}$
6: For all $q \in \Gamma$ set $\Omega_q \leftarrow \{i \in [n] : \tau^q \beta < z_i \leq \tau^{q-1}\beta\}$.

7: Input: $\boldsymbol{\omega}^t \in C$

8: For all $q \in \Gamma$ set $\pi_q^t \leftarrow \sum_{i \in \Omega_q} \omega_i^t$
9: For all $q \in \Gamma$ set $\zeta_q^t \leftarrow \lfloor \delta \pi_q^t \rfloor$
10: For all $q \in \Gamma$ and for all $k \leq \zeta_q^t$ draw $\xi_{q,k}^t$ randomly
    from $\Omega_q$ such that $\xi_{q,k}^t \leftarrow i$ with probability $\omega_i^t / \pi_q^t$
11: For all $q \in \Gamma$ and for $k := \zeta_q^t + 1$ draw $\xi_{q,k}^t$ randomly
    from $\Omega_q \cup \{0\}$ such that, for $i \in \Omega_q$ we have
    $\xi_{q,k}^t \leftarrow i$ with probability $(\delta \pi_q^t - \lfloor \delta \pi_q^t \rfloor)\omega_i^t / \pi_q^t$, and
    we have $\xi_{q,k}^t \leftarrow 0$ with probability $\lfloor \delta \pi_q^t \rfloor + 1 - \delta \pi_q^t$.
    NB: In the case that $\pi_q^t = 0$ we define $0/0 = 0$

12: Output: $X^t \leftarrow \{\xi_{q,k}^t : q \in \Gamma, k \leq \zeta_q^t + 1\} \setminus \{0\}$
```

Algorithm 2 Computing $\boldsymbol{\omega}^{t+1}$

```
1: $C \leftarrow \{\boldsymbol{x} \in [0,1]^n : \langle \boldsymbol{x}, \boldsymbol{z} \rangle \leq 1\}$
2: $\beta \leftarrow \max_{i \in [n]} z_i$
3: $\delta \leftarrow \left(1 - \sqrt{\beta}\right)^2$

4: Input: $\boldsymbol{\omega}^t \in C$, $\boldsymbol{c}^t \in \mathbb{R}^n$ $\boldsymbol{r}^t \in \mathcal{P}$

5: Order $[n]$ as $[n] = \{\sigma(t,1), \sigma(t,2), ...\sigma(t,n)\}$ such
   that $r^t_{\sigma(t,j)} \geq r^t_{\sigma(t,j+1)}$ for all $j \in [n-1]$
6: For all $j \in [n]$ set $\epsilon^t_j \leftarrow \exp\left(-\delta \sum_{k=1}^{j} \omega^t_{\sigma(t,k)}\right)$
7: For all $j \in [n]$ set:
   $\lambda^t_j \leftarrow r^t_{\sigma(t,n)} \epsilon^t_n + \sum_{k=j}^{n-1} \left(r^t_{\sigma(t,k)} - r^t_{\sigma(t,k+1)}\right) \epsilon^t_k$
8: For all $j \in [n]$ set:
   $g^t_{\sigma(t,j)} \leftarrow \delta \left(c^{t+}_{\sigma(t,j)} + c^{t-}_{\sigma(t,j)} \exp\left(-\delta \omega^t_{\sigma(t,j)}\right) - \lambda^t_j\right)$
9: $\hat{\eta}^t \leftarrow \min\left\{\hat{\eta}^{t-1}, \sqrt{n}/\|\boldsymbol{g}^t\|\right\}$
10: $\eta^t \leftarrow \hat{\eta}^t / \sqrt{2t}$
11: $\boldsymbol{y}^t \leftarrow \boldsymbol{\omega}^t - \eta^t \boldsymbol{g}^t$

12: Output: $\boldsymbol{\omega}^{t+1} \leftarrow \Pi_C(\boldsymbol{y}^t)$
```

4 The Feasibility of X^t

In this section we show that the the total energy of the actions selected by Algorithm 1 is no greater than 1, as required in our problem definition (see Section 2.2). We first introduce the sets and quantities used in the selection of the actions.

Definition 4.1. *We define the following:*

- $\tau := 1 - \sqrt{\beta}$ *and* $\delta := \left(1 - \sqrt{\beta}\right)^2$
- *For all* $q \in \mathbb{N}$ *we define* $\Omega_q := \{i \in [n] : \tau^q \beta < z_i \leq \tau^{q-1}\beta\}$
- $\Gamma := \{q \in \mathbb{N} : \Omega_q \neq \emptyset\}$. *Note that* $\{\Omega_q : q \in \Gamma\}$ *is a partition of* $[n]$
- *On trial t, for all* $q \in \Gamma$ *define* $\pi^t_q := \sum_{i \in \Omega_q} \omega^t_i$ *and* $\zeta^t_q := \lfloor \delta \pi^t_q \rfloor$.

Algorithm 1 works by drawing actions $\{\xi^t_{q,k} : q \in \Gamma, k \in [\zeta^t_q + 1]\}$ randomly, where the number of actions (including a "null action" 0) drawn (with replacement) from Ω_q is equal to $\zeta^t_q + 1$ and the probability distribution of the draws is dependent on $\boldsymbol{\omega}^t$.

The following theorem ensures that the choice of X^t made by our method satisfies our problem's energy constraint.

Theorem 4.2. *On trial t we have* $\sum_{i \in X^t} z_i \leq 1$.

Proof. See Appendix A □

5 Bounding the Cumulative Profit

In this section we bound the cumulative profit of MaxHedge.

5.1 The Probability of Intersection

In this subsection we first bound below the probability that, on a trial t, an arbitrary set Z intersects with X^t. We start with the following lemma:

Lemma 5.1. *Given a set D of independent draws from |D|-many probability distributions, such that the probability of an event E happening on draw* $d \in D$ *is* ρ_d*, then the probability of event E happening on either of the draws is lower bounded by:*

$$1 - \exp\left(-\sum_{d \in D} \rho_d\right)$$

Proof. See Appendix A □

We now bound the probability of intersection:

Theorem 5.2. *For any trial t and any subset* $Z \subseteq [n]$ *we have* $\mathbb{P}(Z \cap X^t \neq \emptyset) \geq 1 - \exp\left(-\delta \sum_{i \in Z} \omega^t_i\right)$.

Proof. See Appendix A □

We now bound the probability that some arbitrary action is selected on trial t:

Theorem 5.3. *Given some action* $i \in [n]$ *and some trial* $t \in [T]$ *we have* $1 - \exp(-\delta\omega^t_i) \leq \mathbb{P}(i \in X^t) \leq \delta\omega^t_i$.

Proof. See Appendix A □

5.2 Approximating the Expected Profit

In this subsection we define a convex function $h^t : C \to \mathbb{R}$ and show that the expected profit on trial t is bounded below by $-h^t(\boldsymbol{\omega}^t)$.

Definition 5.4. *For each trial t we order* $[n]$ *as* $[n] = \{\sigma(t,1), \sigma(t,2), \ldots, \sigma(t,n)\}$ *where* $r^t_{\sigma(t,j)} \geq r^t_{\sigma(t,j+1)}$ *for all* $j \in [n-1]$. *We also define* $\sigma(t, n+1) := 0$ *and* $r^t_0 := 0$.

Definition 5.5. *Given a trial t and a number* $j \in [n]$ *we define the function* $f^t_j : \mathcal{P} \to \mathbb{R}$ *by:*

$$f^t_j(\boldsymbol{\gamma}) := \left(r^t_{\sigma(t,j)} - r^t_{\sigma(t,j+1)}\right)\left(1 - \exp\left(-\delta \sum_{k=1}^{j} \gamma_{\sigma(t,k)}\right)\right)$$

We also define the function $h^t : \mathcal{P} \to \mathbb{R}$ *as:*

$$h^t(\boldsymbol{\gamma}) = \langle \boldsymbol{c}^{t+}, \boldsymbol{\gamma} \rangle + \sum_{i=1}^{n} c^{t-}_i (1 - \exp(-\delta\gamma_i)) - \sum_{j=1}^{n} f^t_j(\boldsymbol{\gamma})$$

Theorem 5.6. *For all $t \in [T]$, the function h^t is convex.*

Proof. See Appendix A □

The rest of this subsection proves that the expected profit on trial t is bounded below by $-h^t(\boldsymbol{\omega}^t)$.

Lemma 5.7. *On trial t we have $\max_{i \in X^t} r_i^t = \sum_{j=1}^n \left(r^t_{\sigma(t,j)} - r^t_{\sigma(t,j+1)}\right) \mathcal{I}\left(\exists k \le j : \sigma(t,k) \in X^t\right)$.*

Proof. See Appendix A □

Lemma 5.8. *On trial t we have $\mathbb{E}\left(\max_{i \in X^t} r_i^t\right) = \sum_{j=1}^n \left(r^t_{\sigma(t,j)} - r^t_{\sigma(t,j+1)}\right) \mathbb{P}\left(\exists k \le j : \sigma(t,k) \in X^t\right)$.*

Proof. Direct from lemma 5.7 using linearity of expectation. □

Theorem 5.9. *On trial t we have $\mathbb{E}\left(\mu^t(X^t)\right) \ge -h^t(\boldsymbol{\omega}^t)$.*

Proof. See Appendix A □

5.3 The Gradient

In this subsection we show how to construct the gradient of h^t and bound its magnitude. We start with the following definitions.

Definition 5.10. *On any trial t and for any $j \in [n]$ we define:*

- $\epsilon_j^t := \exp\left(-\delta \sum_{k=1}^{j} \omega^t_{\sigma(t,k)}\right)$
- $\lambda_j^t := \sum_{k=j}^n \left(r^t_{\sigma(t,k)} - r^t_{\sigma(t,k+1)}\right) \epsilon_k^t$
- $g^t_{\sigma(t,j)} := \delta\left(c^{t+}_{\sigma(t,j)} + c^{t-}_{\sigma(t,j)} \exp\left(-\delta \omega^t_{\sigma(t,j)}\right) - \lambda_j^t\right)$

We first show that $\boldsymbol{g}^t$ is the gradient of h^t evaluated at $\boldsymbol{\omega}^t$.

Theorem 5.11. *On any trial t we have $\boldsymbol{g}^t = \nabla h^t(\boldsymbol{\omega}^t)$.*

Proof. See Appendix A □

We now bound the magnitude of the gradient.

Lemma 5.12. *For any trial t we have $\|\boldsymbol{g}^t\|^2 \le n\delta^2(\hat{r} + \hat{c})^2$.*

Proof. Since $\omega^t_{\sigma(t,k)} \ge 0$ for all $k \in [n]$ we have $\epsilon_j^t \in [0,1]$ for all $j \in [n]$. This gives us, for all $j \in [n]$, that $\delta\lambda_j^t \le \delta r^t_{\sigma(t,n)} + \delta \sum_{k=j}^{n-1}(r^t_{\sigma(t,k)} - r^t_{\sigma(t,k+1)}) = \delta r^t_{\sigma(t,j)} \le \delta\hat{r}$. Since also $\lambda_j^t \ge 0$ this implies that $-g_i^t \le \delta\hat{r} + \delta\hat{c}$ and that $g_i^t \le \delta\hat{c}$ so $(g_i^t)^2 \le (\delta\hat{r} + \delta\hat{c})^2$. This then implies the result. □

5.4 Online Gradient Descent

In this subsection we show that Algorithm 2 corresponds to the use of online gradient descent over C with convex functions $\{h^t : t \in [T]\}$ and we use the standard analysis of online gradient descent to derive a lower bound on the cumulative profit.

From here on we compare the performance of our algorithm against any fixed set S of actions such that $\sum_{i \in S} z_i \le 1$. We define $\boldsymbol{\phi}$ as the vector in $\mathbb{R}^n$ such that, for all $i \in [n]$, we have $\phi_i := 0$ if $i \notin S$ and $\phi_i := 1$ if $i \in S$. It is clear that $\boldsymbol{\phi} \in C$.

Definition 5.13. *Our learning rates are defined as follows:*

- $\hat{\eta}^t := \min_{t' \le t}(\sqrt{n}/\|\boldsymbol{g}^{t'}\|)$
- $\eta^t := \hat{\eta}^t/\sqrt{2t}$

The next result follows from the standard analysis of online gradient descent.

Theorem 5.14. *We have:*

$$\sum_{t=1}^{T}(h^t(\boldsymbol{\omega}^t) - h^t(\boldsymbol{\phi})) \le \frac{R^2}{2\eta^T} + \frac{1}{2}\sum_{t=1}^{T}\eta^t\|\boldsymbol{g}^t\|^2$$

where $R := \max_{x,y \in C}\|x - y\|$.

Proof. For all trials t: From Theorem 5.6 we have that h^t is a convex function. From Theorem 5.11 we have that $\boldsymbol{g}^t = \nabla h^t(\boldsymbol{\omega}^t)$. We also have that $\eta^{t+1} \le \eta^t$ so since, by Algorithm 2, we have $\boldsymbol{\omega}^{t+1} = \Pi_C(\boldsymbol{\omega}^t - \eta^t \boldsymbol{g}^t)$ and $\boldsymbol{\phi} \in C$, the standard analysis of online gradient descent (see, e.g., [24]) gives us the result. □

We now bound the right hand side of the equation in Theorem 5.14.

Definition 5.15. *Define $s := \max_{t \in [T]}\|\boldsymbol{g}^t\|^2/n$.*

Lemma 5.16. *For any trial t, $\eta^t\|\boldsymbol{g}^t\|^2 \le n\sqrt{s/(2t)}$.*

Proof. We have $\hat{\eta}^t = \min_{t' \le t}(\sqrt{n}/\|\boldsymbol{g}^{t'}\|) \le \sqrt{n}/\|\boldsymbol{g}^t\|)$. This implies that that $\eta^t = \hat{\eta}^t/\sqrt{2t} \le \sqrt{n}/(\|\boldsymbol{g}^t\|\sqrt{2t})$ so $\eta^t\|\boldsymbol{g}^t\|^2 \le \sqrt{n}\|\boldsymbol{g}^t\|/\sqrt{2t}$ which, by definition of s, is bounded above by $n\sqrt{s/(2t)}$. □

Lemma 5.17. $R^2/\eta^T \le n\sqrt{2sT}$

Proof. We have $\hat{\eta}^T = \min_{t \in [T]}\sqrt{n}/\|\boldsymbol{g}^t\| = 1/\sqrt{s}$ so $\eta^T := \hat{\eta}^T/\sqrt{2T} \le 1/\sqrt{2sT}$. Since $C \subseteq [0,1]^n$, if $\boldsymbol{x}, \boldsymbol{y} \in C$ then each component of $\boldsymbol{x} - \boldsymbol{y}$ has magnitude bounded above by 1 which implies that $R^2 \le n$. Putting together gives the result. □

Theorem 5.18. *We have:*

$$\sum_{t=1}^{T}(h^t(\boldsymbol{\omega}^t)-h^t(\boldsymbol{\phi}))\leq n\sqrt{2T}\delta(\hat{r}+\hat{c})$$

Proof. See Appendix A □

The next result bounds $h^t(\boldsymbol{\phi})$.

Lemma 5.19. *On trial t we have $\hat{\mu}^t_{\alpha,\delta}(S)\leq -h^t(\boldsymbol{\phi})$ where $\alpha:=1-e^{-\delta}$.*

Proof. Let $j'=\operatorname{argmax}_{j\in[n]:\sigma(t,j)\in S}r^t_{\sigma(t,j)}$ which is equal to the minimum j such that $\sigma(t,j)\in S$. Note that for all $j\geq j'$ we have $\sum_{k=1}^{j}\phi_{\sigma(t,j)}\geq 1$ and hence $f^t_j(\boldsymbol{\phi})\geq\left(r^t_{\sigma(t,j)}-r^t_{\sigma(t,j+1)}\right)(1-e^{-\delta})$ so $\sum_{j=1}^{n}f^t_j(\boldsymbol{\phi})\geq\sum_{j=j'}^{n}f^t_j(\boldsymbol{\phi})\geq\sum_{j=j'}^{n}\left(r^t_{\sigma(t,j)}-r^t_{\sigma(t,j+1)}\right)(1-e^{-\delta})=r^t_{\sigma(t,j')}(1-e^{-\delta})=(1-e^{-\delta})\max_{i\in S}r^t_i$.

Also note that $\delta\langle\boldsymbol{c}^{t+},\boldsymbol{\phi}\rangle=\delta\sum_{i\in S}c^{t+}_i$ and that $\sum_{i=1}^{n}c^{t-}_i(1-\exp(-\delta\phi_i))=\sum_{i\in S}c^{t-}_i(1-e^{-\delta})$. Combining with the above gives us the result. □

Putting together we obtain the main result:

Theorem 5.20. *We have:*

$$\mathbb{E}\left(\sum_{t=1}^{T}\mu^t(X^t)\right)\geq\sum_{t=1}^{T}\hat{\mu}^t_{\alpha,\delta}(S)-n\sqrt{2T}\delta(\hat{r}+\hat{c})$$

where $\delta:=\left(1-\sqrt{\beta}\right)^2$, $\alpha:=1-\exp\left(-\left(1-\sqrt{\beta}\right)^2\right)$, $\hat{r}:=\max_{t\in[T],i\in[n]}r^t_i$ and $\hat{c}:=\max_{t\in[T],i\in[n]}|c^t_i|$.

Proof. Let $\alpha:=1-e^{-\delta}$. By Theorem 5.9 we have, for all $t\in[T]$, that $\mathbb{E}\left(\mu^t(X^t)\right)\geq -h^t(\boldsymbol{\omega}^t)$. By Lemma 5.19 we have, for all $t\in[T]$, that $\hat{\mu}^t_{\alpha,\delta}(S)\leq -h^t(\boldsymbol{\phi})$. Hence we have that $\hat{\mu}^t_{\alpha,\delta}(S)-\mathbb{E}\left(\mu^t(X^t)\right)\leq h^t(\boldsymbol{\omega}^t)-h^t(\boldsymbol{\phi})$. By Theorem 5.18 we than have:

$$\begin{aligned}&\sum_{t=1}^{T}\left(\hat{\mu}^t_{\alpha,\delta}(S)-\mathbb{E}\left(\mu^t(X^t)\right)\right)\\ \leq&\sum_{t=1}^{T}(h^t(\boldsymbol{\omega}^t)-h^t(\boldsymbol{\phi}))\\ \leq&n\sqrt{2T}\delta(\hat{r}+\hat{c})\end{aligned}$$

Rearranging gives us:

$$\sum_{t=1}^{T}\mathbb{E}\left(\mu^t(X^t)\right)\geq\sum_{t=1}^{T}\hat{\mu}^t_{\alpha,\delta}(S)-n\sqrt{2T}\delta(\hat{r}+\hat{c})$$

□

6 Special Cases

The following online variants of classic computer science problems are special cases of the general problem.

6.1 Facility Location Problem

The (inverted) facility location problem is defined by a vector $\boldsymbol{c}\in\mathcal{P}$ and vectors $\boldsymbol{r}^1,\boldsymbol{r}^2,\cdots,\boldsymbol{r}^T\in\mathcal{P}$. A feasible solution is any $X\subseteq[n]$. The aim is to maximise the objective function:

$$\sum_{t=1}^{T}\max_{i\in X}r^t_i-\sum_{i\in X}c_i$$

An example of the problem is as follows. There are n sites and T users, all located in some metric space. We have to choose a set X of sites to open a facility on. Opening a facility on site i costs us c_i. Each user pays us a reward based on how near it is to the closest open facility. If the nearest open facility to user t is at site i then user t rewards us r^t_i. The objective is to maximise the total profit.

In our online variant of the (inverted) facility location problem, learning proceeds in trials. On trial t:

1. For all sites i, the cost, c^t_i of opening a facility on site i is revealed to the learner.
2. The learner chooses a set X^t of sites in which to open facilities on.
3. User t requests the use of a facility, revealing $\boldsymbol{r}^t$ to the learner.
4. Learner incurs profit: $\max_{i\in X^t}r^t_i-\sum_{i\in X^t}c^t_i$

The objective is to maximise the cumulative profit. Note that this is the special case of our problem when, for all $i\in[n]$ and $t\in[T]$ we have $z_i=0$ and $c^t_i\geq 0$. Given some set S the expected cumulative profit of MaxHedge is then bounded below by:

$$\sum_{t=1}^{T}\left((1-1/e)\max_{i\in S}r^t_i-\sum_{i\in S}c^t_i\right)-n\sqrt{2T}(\hat{r}+\hat{c})$$

6.2 Knapsack Median Problem

The (inverted) knapsack median problem is defined by a vector $\boldsymbol{z}\in\mathcal{P}$ and vectors $\boldsymbol{r}^1,\boldsymbol{r}^2,\cdots,\boldsymbol{r}^T\in\mathcal{P}$. A feasible solution is any $X\subseteq[n]$ with $\sum_{i\in X}z_i\leq 1$. The aim is to maximise the objective function $\sum_{t=1}^{T}\max_{i\in X}r^t_i$.

An example of the problem is as follows. There are n sites and T users, all located in some metric space. We have to choose a set X of sites to open a facility on. Opening a facility on site i has a fee of z_i and we have a budget of 1 to spend on opening facilities. Each user pays us a reward based on how near it is to the closest open facility. If the nearest open facility to user t is at site i then user t rewards us r_i^t. The objective is to maximise the total reward.

In our online variant of the (inverted) knapsack median problem, learning proceeds in trials. The learner has knowledge of the fee z_i for every site i. On trial t:

1. The learner chooses a set X^t of sites in which to open facilities on. The total fee, $\sum_{i \in X^t} z_i$, can't exceed 1.
2. User t requests the use of a facility, revealing $\boldsymbol{r}^t$ to the learner.
3. Learner incurs reward: $\max_{i \in X^t} r_i^t$

The objective is to maximise the cumulative reward. Note that this is the special case of our problem when, for all $i \in [n]$ and $t \in [T]$ we have $c_i^t = 0$. Given some set S with $\sum_{i \in S} z_i \leq 1$ the expected cumulative reward of MaxHedge is then bounded below by:

$$(1 - \exp(-\delta)) \sum_{t=1}^{T} \max_{i \in S} r_i^t - \delta n \sqrt{2T} \hat{r}$$

where $\delta := (1 - \sqrt{\max_{i \in [n]} z_i})^2$

6.3 0-1 Knapsack Problem

The knapsack problem is defined by a vector $\boldsymbol{z} \in \mathcal{P}$ and a vector $\boldsymbol{v} \in \mathcal{P}$. A feasible solution is any $X \subseteq [n]$ with $\sum_{i \in X} z_i \leq 1$. The aim is to maximise the objective function $\sum_{i \in X} v_i$.

An example of the problem is as follows. We have n items. Item i has a value v_i and a weight z_i. The objective is to place a set $X \subseteq [n]$ of items in the knapsack that maximises the total value of all items in the knapsack subject to their total weight being no greater than 1.

In our online variant of the knapsack problem, learning proceeds in trials. The learner has knowledge of the weight z_i for every item i. On trial t:

1. The learner chooses a set X^t of items to place in the knapsack. The total weight, $\sum_{i \in X^t} z_i$, can't exceed 1.
2. For each item i, the value v_i^t, of item i on this trial is revealed to the learner
3. Learner incurs profit: $\sum_{i \in X^t} v_i^t$

The objective is to maximise the cumulative profit. Note that this is the special case of our problem when, for all $i \in [n]$ and $t \in [T]$ we have $r_i^t = 0$ and $c_i^t \leq 0$ (noting that $c_i^t = -v_i^t$). Given some set S with $\sum_{i \in S} z_i \leq 1$ the expected cumulative profit of MaxHedge is then bounded below by:

$$(1 - \exp(-\delta)) \sum_{t=1}^{T} \sum_{i \in S} v_i^t - \delta n \sqrt{2T} \hat{c}$$

where $\delta := (1 - \sqrt{\max_{i \in [n]} z_i})^2$

7 Conclusions and Ongoing Research

We presented and investigated in depth a novel online framework, capable of encompassing several online learning problems and capturing many practical problems in the real-world. The main challenge of the general version of this problem lies in the fact that the learner's profit depends on the maximum reward of the selected actions, instead of the sum of all their rewards. We proposed and rigorously analysed a very scalable and efficient learning strategy MaxHedge.

Current ongoing research includes:

- Deriving a lower bound on the achievable profit.
- Complementing our results with a set of experiments on synthetic and real-world datasets.
- Several real systems usually have a switching cost for turning on/off services, which translates in our framework to the cost incurred whenever an action selected at any given trial is not selected in the preceding one. This represents an interesting direction for further research, which is certainly motivated by practical problems.

Acknowledgements. This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon. Fabio Vitale acknowledges support from the ERC Starting Grant "DMAP 680153", the Google Focused Award "ALL4AI", and grant "Dipartimenti di Eccellenza 2018-2022", awarded to the Department of Computer Science of Sapienza University.

References

[1] M. Charikar, S. Guha, E. Tardos, D.B. Shmoys. A constant–factor approximation algorithm for the k–median problem. In *ACM Symposium on Theory of Computing* (STOC), pp. 1–10, ACM (1999).

[2] M. Charikar and S. Guha. Improved combinatorial algorithms for the facility location and k–median problems. In *IEEE Foundations of Computer Science*, pages 378–388, 1999.

[3] L. Chen, H. Hassani, A. Karbasi. Online Continuous Submodular Maximization. In *International Conference on Artificial Intelligence and Statistics*, AISTATS 2018.

[4] G. Cornuejols, G. L. Nemhauser, and L. A. Wolsey. The uncapacitated facility location problem. In *Pitu B. Mirchandani and Richard L. Francis, editors, Discrete Location Theory*, pages 119–171. John Wiley and Son, Inc., New York, 1990.

[5] M. Cygan, A. Czumaj, M. Mucha, P. Sankowski. Online Facility Location with Deletions. In *Annual European Symposium on Algorithms*, ESA 2018.

[6] G.B. Dantzig. Discrete variable extremum problems, In *Operations Research 5* (1957) 266–277.

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

[8] T. Fujita, K. Hatano, E Takimoto Combinatorial Online Prediction via Metarounding. In *Algorithmic Learning Theory* (2013), 68-82,

[9] D. Golovin, A. Krause, M. Streeter. Online Submodular Maximization under a Matroid Constraint with Application to Learning Assignments In *Technical report, arXiv, 2014.*

[10] K. Jain and V. Vazirani. Approximation algorithms for metric facility location and k–median problems using the primal–dual schema and Lagrangian relaxation. In *J. ACM*, 48(2):274–296, 2001.

[11] A. Kalai and S. Vempala. Efficient algorithms for online decision problems, In *Journal of Computer and System Sciences*, vol. 71, no. 3, pp. 291–307, 2005.

[12] S. Kakade, A. Kalai, and K. Ligett. Playing games with approximation algorithms. In *ACM Symposium on the Theory of Computing*, pages 546-555, STOC 2007.

[13] W.M. Koolen, M.K. Warmuth, J. Kivinen. Hedging structured concepts. In *Conference on Learning Theory*, Omnipress, 2010, pp. 239-254.

[14] K. C. Kiwiel. Breakpoint searching algorithms for the continuous quadratic knapsack problem. In *Math. Program*, 112(2): 473-491 (2008).

[15] A. Kumar. Constant factor approximation algorithm for the knapsack median problem. In *ACM–SIAM Symposium on Discrete Algorithms* (SODA), pp. 824–832, SIAM (2012).

[16] E. Hazan, and S. Kale. Online submodular minimization. In *Journal of Machine Learning Research (JMLR)*, 2012.

[17] D. P. Helmbold and S. V. N. Vishwanathan. Online Learning of Combinatorial Objects via Extended Formulation. In *International Conference on Algorithmic Learning Theory*, ALT 2018.

[18] N. Laoutaris, G. Smaragdakis, K. Oikonomou, I. Stavrakakis, A. Bestavros. Distributed Placement of Service Facilities in Large–Scale Networks. In *IEEE INFOCOM*, pages 2144–2152, 2007.

[19] S. Martello, P. Toth. An upper bound for the zero–one knapsack problem and a branch and bound algorithm, In *European Journal of Operational Research 1* (1977) 169–175.

[20] A. Meyerson. Online Facility Location. In *IEEE Symposium on Foundations of Computer Science*, FOCS 2001.

[21] D. Shmoys, E. Tardos and K. Aardal. Approximation algorithms for facility location problems. In *ACM Symposium on Theory of Computing* (STOC 1997), pages 265–274, ACM Press, 1997.

[22] M. Streeter and D. Golovin. An Online Algorithm for Maximizing Submodular Functions. In *Conference on Neural Information Processing Systems*, NIPS 2008.

[23] H. Yu, M. J. Neely, X. Wei. Online Convex Optimization with Stochastic Constraints. In *Conference on Neural Information Processing Systems*, NIPS 2017.

[24] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *International Conference on Machine Learning*, 2003.

[25] S. Pasteris, S. Wang, M. Herbster, T. He. Service Placement with Provable Guarantees in Heterogeneous Edge Computing Systems. To appear in *IEEE INFOCOM*, 2019.