Optimizing the Generalized Gini Index in Multi-objective Bandits

Róbert Busa-Fekete[1] and **Paul Weng**[2] and **Orkun Karabasoglu**[2] and **Balázs Szörényi**[3 4]

Abstract. We study the multi-armed bandit (MAB) problem where the agent receives a vectorial feedback that encodes many possibly competing objectives to be optimized. The goal of the agent is to find a policy, which can optimize these objectives simultaneously in a fair way. This multi-objective online optimization problem is formalized by using the Generalized Gini Index (GGI) aggregation function. We propose two learning algorithms to tackle this task: one is a simple UCB-style algorithm, the other is a gradient descent-based algorithm that exploits the convexity of the GGI aggregation function. We test our algorithms on synthetic data as well as on an electric battery control problem where the goal is to trade off the use of the different cells of a battery in order to balance their respective degradation rates.

1 Introduction

The multi-armed bandit (MAB) problem (or bandit problem) refers to an iterative decision making problem in which an agent repeatedly chooses among K options, metaphorically corresponding to pulling one of K arms of a bandit machine. In each round, the agent receives a random payoff, which is a reward or a cost that depends on the arm being selected. The agent's goal is to optimize an evaluation metric, e.g., the *error rate* (expected percentage of times a suboptimal arm is played) or the *cumulative regret* (difference between the sum of payoffs obtained and the (expected) payoffs that could have been obtained by selecting the best arm in each round). In the *stochastic* multi-armed bandit setup, the payoffs are assumed to obey fixed distributions that can vary with the arms but do not change with time. To achieve the desired goal, the agent has to tackle the classical exploration/exploitation dilemma: It has to properly balance the pulling of arms that were found to yield low costs in earlier rounds and the selection of arms that have not yet been tested often enough [1, 2].

The bandit setup has become the standard modeling framework for many practical applications, such as online advertisement [3], medical treatment design [4], to name a few. In these tasks, the feedback is formulated as a single real value. However many real-world online learning problems are rather multi-objective. For example, in our motivating example, namely an electric battery control problem, the learner tries to discover a "best" battery controller, which balances the degradation rates of the battery cells (i.e., components of a battery), among a set of controllers while facing a stochastic power demand.

In this paper, we formalize the multi-objective multi-armed bandit setting in which the feedback received by the agent is in the form of a D-dimensional real-valued cost vector. The goal of the learning agent is to be both efficient, i.e., minimize the cumulative cost for each objective, and fair, i.e., balance the different objectives. The "fairness" between the objectives is quantified in terms of Generalized Gini Index (GGI), which is a well-known inequality measure in economics [5]. We propose two algorithms that extend two standard online methods widely used in single-objective optimization problems. In our synthetic and battery-control experiments we test them and demonstrate their versatility.

2 Formal setup

The multi-armed or K-armed bandit problem is specified by real-valued random variables $X_1, \dots, X_K$ associated, respectively, with K arms (that we simply identify by the numbers $1, \dots, K$). In each time step t, the online learner selects one and obtains a random sample of the corresponding distributions. These samples, which are called costs, are assumed to be independent of all previous actions and costs.[5] The goal of the learner can be defined in different ways, such as minimizing the sum of costs over time [2, 1].

In the *multi-objective* multi-armed bandit (MO-MAB) problem, costs are not scalar real values, but real vectors. More specifically, a D-objective K-armed bandit problem ($D \geq 2$, $K \geq 2$) is specified by K real-valued multivariate random variables $\mathbf{X}_1, \dots, \mathbf{X}_K$ over $[0,1]^D$. Let $\boldsymbol{\mu}_k = \mathbb{E}[\mathbf{X}_k]$ denote the expected vectorial cost of arm k where $\boldsymbol{\mu}_k = (\mu_{k,1}, \dots, \mu_{k,D})$. Furthermore, $\boldsymbol{\mu}$ denotes the matrix whose rows are the $\boldsymbol{\mu}_k$'s.

In each time step the learner can select one of the arms and obtain a sample, which is a cost vector, from the corresponding distribution. Sampling an arm is assumed to be independent over time and moreover independence also holds across the arms, but not necessarily across the components of cost vectors.

At time step t, k_t denotes the index of the arm played by a learner and $\mathbf{X}_{k_t}^{(t)} = (X_{k_t,1}^{(t)}, \dots X_{k_t,D}^{(t)})$ the resulting payoff. After playing t time steps, the empirical estimate of the expected cost $\boldsymbol{\mu}_k$ of the kth arm is

$$\widehat{\boldsymbol{\mu}}_k^{(t)} = \frac{1}{T_k(t)} \sum_{\tau=1}^{t} \mathbf{X}_{k_\tau}^{(\tau)} \mathbf{1}(k_\tau = k) \qquad (1)$$

where all operations are meant elementwise, $T_k(t)$ is the number of times the kth arm has been played (i.e., $T_k(t) = \sum_{\tau=1}^{t} \mathbf{1}(k_\tau = k)$) and $\mathbf{1}(\cdot)$ is the indicator function.

[1] Paderborn University, email: busarobi@upb.de
[2] SYSU-CMU JIE, School of Electronics and Information Technology, SYSU-CMU JRI, email: {paweng,karabasoglu}@cmu.edu
[3] Technion Institute of Technology, email: szorenyi@inf.u-szeged.hu
[4] Research Group on AI, Hungarian Acad. Sci. and Univ. of Szeged

[5] Our setup is motivated by a practical application where feedback is more natural to formulate in terms of cost. However the stochastic bandit problem is most often formulated by using the notion of reward, which can be easily turn into cost by using the transformation $x \mapsto 1 - x$ assuming that the rewards are from $[0, 1]$.

In this study, we shall use the elementwise sample mean as an estimator of the mean vector. Nevertheless, it is worth to mention that this estimator is not admissible in general. For example, the James-Stein estimator [6] always achieves lower least square error in expectation than the elementwise sample mean given in (1) for D-dimensional normal distributions where $D \geq 3$ and the covariance matrix is diagonal in the form of σI. But to the best of our best knowledge, there does not exist a James-Stein estimator for such a general class of multivariate distributions like the class of distributions with bounded support that we focus on in this study.

3 Multi-objective optimization

In order to complete the MO-MAB setting, we need to introduce the notion of optimality of the arms. First, we define binary relation $\preceq$ on $\mathbb{R}^D$ as follows, for any $\boldsymbol{v}, \boldsymbol{v}' \in \mathbb{R}^D$:

$$\boldsymbol{v} \preceq \boldsymbol{v}' \Leftrightarrow \forall d = 1, \ldots, D, v_d \leq v'_d \ . \tag{2}$$

Let $\mathcal{O} \subseteq \mathbb{R}^D$ be a set of D-dimension real vectors. The *Pareto front* of $\mathcal{O}$, denoted $\mathcal{O}^*$, is the set of vectors such that:

$$\boldsymbol{v}^* \in \mathcal{O}^* \Leftrightarrow \left(\forall \boldsymbol{v} \in \mathcal{O}, \boldsymbol{v} \preceq \boldsymbol{v}^* \Rightarrow \boldsymbol{v} = \boldsymbol{v}^*\right) \ . \tag{3}$$

In multiobjective optimization, one usually wants to compute the Pareto front, or search for a particular element of the Pareto front. In practice, it may be costly (and even infeasible depending on the size of the solution space) to determine all the solutions of the Pareto front. One may then prefer to directly aim for a particular solution in the Pareto front. This problem is formalized as a mono-objective optimization problem, using an *aggregation function*.

An aggregation (or scalarizing) function, which is a non-decreasing function $\phi : \mathbb{R}^D \to \mathbb{R}$, allows every vector to receive a scalar value to be optimized. The initial multiobjective problem is then rewritten as follows:

$$\min \phi(\boldsymbol{v}) \quad \text{s.t.} \quad \boldsymbol{v} \in \mathcal{O} \ . \tag{4}$$

A solution to this problem yields a particular solution on the Pareto front. Note that if ϕ is not strictly increasing in every component, some care is needed to ensure that the solution of (4) is on the Pareto front.

Different aggregation function can be used depending on the problem at hand, such as sum, weighted sum, min, max, (augmented) weighted Chebyshev norm [7], Ordered Weighted Averages (OWA) [8] or Ordered Weighted Regret (OWR) [9] and its weighted version [10]. In this study, we focus on the Generalized Gini Index (GGI) [5], which is a special case of OWA.

4 Generalized Gini Index

The Generalized Gini Index (GGI) [5] is defined as

$$G(\mathbf{x}; \mathbf{w}) = \sum_{d=1}^{D} w_d x_d^{\downarrow}$$

where $\mathbf{x}^{\downarrow} = (x_1^{\downarrow}, \cdots, x_D^{\downarrow})$ contains the components of vector $\mathbf{x}$ that are sorted in a decreasing order and weights w_i's are non-increasing, i.e., $w_1 \geq w_2 \geq \ldots \geq w_D$. It is well-known that GGI is convex in $\mathbf{x}$ as it can be written as the maximum of linear functions.

GGI was originally introduced for quantifying the inequality of the welfare based on incomes. As an instance of Weighted Average Ordered Sample statistics, it has also been investigated in statistics [11]. The Weighted Average Ordered Sample statistics, also known as OWA [8], do not require that weights are non-increasing and are therefore not necessarily convex.

GGI has been characterized by Weymark [5]. It encodes both efficiency as it is monotone with Pareto dominance and fairness as it is non-increasing with Pigou-Dalton transfers [12, 13]. Informally, in our setting, a Pigou-Dalton transfer amounts to increasing an objective while decreasing another objective by the same quantity such that the order between the two objectives is not reversed. The effect of such a transfer is to balance a cost vector. Formally, GGI satisfies the following property: $\forall \mathbf{x}$ such that $x_i < x_j$,

$$\forall \epsilon \in (0, x_j - x_i), G(\mathbf{x} + \epsilon \boldsymbol{e}_i - \epsilon \boldsymbol{e}_j, \mathbf{w}) \leq G(\mathbf{x}, \mathbf{w})$$

where $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ are two vectors of the canonical basis. As a consequence, among vectors of equal sum, the best cost vector (w.r.t GGI) is the one with equal values in all objectives.

From now on, to simplify the presentation, we focus on GGI with strictly decreasing weights in $[0, 1]^D$, i.e., $d < d'$ implies $w_d > w_{d'}$. This means that GGI is strictly decreasing with Pigou-Dalton transfers. We also introduce a few notations. For a given $n \in \mathbb{N}$, $[n]$ denote the set $\{1, 2, \ldots, n\}$. Let $\mathbf{w}'$ be the vector defined by $\forall d \in [D], w'_d = w_d - w_{d+1}$ with $w_{D+1} = 0$. Note that all the components of $\mathbf{w}'$ are positive as we assume that those of $\mathbf{w}$ are strictly decreasing. Ogryczak and Sliwinski [14] showed that the GGI value of a vector $\mathbf{x}$ can be obtained by solving a linear program. We shall recall their results and define the linear program-based formulation of GGI.

Proposition 1. *The* GGI *score* $G(\mathbf{x}; \mathbf{w})$ *of vector* $\mathbf{x}$ *is the optimal value of the following linear program*

$$\begin{array}{lllll} \textit{minimize} & \displaystyle\sum_{d=1}^{D} w'_d \left(d r_d + \sum_{j=1}^{D} b_{j,d} \right) & & & \\ \textit{subject to} & r_d + b_{j,d} & \geq x_j & \forall j, d \in [D] \\ & b_{j,d} & \geq 0 & \forall j, d \in [D] \end{array}$$

Proof : The proof uses the Lorenz transform $L : \mathbb{R}^D \to \mathbb{R}^D$ which is defined as

$$L(\mathbf{x}) = \left(x_1^{\downarrow}, x_1^{\downarrow} + x_2^{\downarrow}, \cdots, \sum_{j=1}^{D} x_j^{\downarrow} \right) \ ,$$

and its dth component is denoted by $L_d(\mathbf{x})$. Then the GGI can be written as

$$\begin{aligned} G(\mathbf{x}; \mathbf{w}) &= \sum_{d=1}^{D} w_d x_d^{\downarrow} \\ &= \sum_{d=1}^{D} w_d (L_d(\mathbf{x}) - L_{d-1}(\mathbf{x})) \\ &= \sum_{d=1}^{D} (w_d - w_{d+1}) L_d(\mathbf{x}) \\ &= \sum_{d=1}^{D} w'_d L_d(\mathbf{x}) \end{aligned} \tag{5}$$

where we assume that $L_0(\mathbf{x}) = 0$.

For a given $d \in [D]$, $L_d(\mathbf{x})$ can also be thought of as the optimal solution of the linear program $\mathcal{P}_d$:

$$\begin{array}{lll} \text{maximize} & \sum_{j=1}^{D} y_j x_j & (= L_d(\mathbf{x})) \\ \text{subject to} & \sum_{j=1}^{D} y_j = d & \\ & y_j \in [0,1] & \end{array} \tag{6}$$

To see this, first note that $y_1, \ldots, y_D$ represent decision variables. Thus in principle they should be in $\{0,1\}$, i.e., $y_j = 1$ for an optimal solution means that the jth component of $\mathbf{x}$ is among the d biggest values of $\mathbf{x}$. A simple argument by contradiction shows that optimal solutions are attained at extreme points of $[0,1]^D$.

The dual $\mathcal{D}_d$ of the above problem $\mathcal{P}_d$ is:

$$\begin{array}{lllll} \text{minimize} & dr_d + \sum_{j=1}^{D} b_{j,d} & (= L_d(\mathbf{x})) & & \\ \text{subject to} & r_d + b_{j,d} & \geq x_j & & \forall j \in [D] \\ & b_{j,d} & \geq 0 & & \forall j \in [D] \end{array} \tag{7}$$

According to the strong duality theorem, its optimal solution equals to the optimal solution of the primal problem, which is $L_d(\mathbf{x})$ by construction. This, together with Equation (5) yields the proposition's claim. ■

Interestingly, when realizable solution $\mathbf{x}$ belongs to a polytope, using the linear programs of (7) leads to formulate a simple linear program to solve, which consists in adding the polytope constraints to the linear program of Proposition 1. However, the matter is not as simple if one were to use the linear programs of (6). First, the optimization of GGI and that of (7) are in opposite direction. Second, the resulting optimization program would be quadratic.

5 Optimal policy

In the mono-objective case, the arms are compared in terms of their means, which induces a ranking over the arms, and thus the notion of the best arm is also well-defined. In the multi-objective setting, we make use of the notion of the GGI criterion to compare arms. One can compute the GGI score of each arm k as $G(\boldsymbol{\mu}_k; \mathbf{w})$ if its vectorial mean $\boldsymbol{\mu}_k$ is known. Then the optimal arm is the one that minimizes the GGI score, i.e.,

$$k^* = \operatorname*{argmin}_{k \in [K]} G(\boldsymbol{\mu}_k; \mathbf{w}) \ .$$

Since the GGI operator is convex, $G(\boldsymbol{\mu}_k; \mathbf{w}) = G(\mathbb{E}[\mathbf{X}_k]; \mathbf{w}) \leq \mathbb{E}[G(\mathbf{X}_k; \mathbf{w})]$ by Jensen's inequality.

We are going to deal with mixed strategies, which can be defined as $\mathcal{A} = \{\boldsymbol{\alpha} \in \mathbb{R}^K \,|\, \sum_{k=1}^{K} \alpha_k = 1 \wedge 0 \preceq \boldsymbol{\alpha}\}$, because they may allow to reach lower GGI values than any fixed arm. A policy, which is parameterized by $\boldsymbol{\alpha}$ chooses arm k with probability α_k. The optimal mixed policy can be computed as

$$\boldsymbol{\alpha}^* = \operatorname*{argmin}_{\boldsymbol{\alpha} \in A} G\left(\sum_{k=1}^{K} \alpha_k \boldsymbol{\mu}_k; \mathbf{w}\right) \ . \tag{8}$$

In general, $G\left(\sum_{k=1}^{K} \alpha_k^* \boldsymbol{\mu}_k; \mathbf{w}\right) \leq G(\boldsymbol{\mu}_{k^*}; \mathbf{w})$, therefore using mixed strategies is justified in our setting. Recalling Proposition 1, it is clear that solving the following linear program

$$\begin{array}{lll} \text{minimize} & \sum_{d=1}^{D} w'_d \left(dr_d + \sum_{j=1}^{D} b_{j,d} \right) & \\ \text{subject to} & r_d + b_{j,d} \geq \sum_{k=1}^{K} \alpha_k \mu_{k,j} & \forall j, d \in [D] \\ & \boldsymbol{\alpha}^T \mathbf{1} = 1 & \\ & \boldsymbol{\alpha} \geq \mathbf{0} & \\ & b_{j,d} \geq 0 & \forall j, d \in [D] \end{array} \tag{9}$$

amounts to solving (8).

6 Regret

After playing T time steps, the average cost of this learner can be written as

$$\mathbf{X}^{(T)} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{X}_{k_t}^{(t)} \ .$$

Our goal is to minimize the GGI index of this term. Accordingly we expect the learner to collect costs so as their average in terms of GGI, that is, $G\left(\mathbf{X}^{(T)}; \mathbf{w}\right)$ should be as small as possible. As shown in the previous section, for a given bandit instance with arm means $\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K)^\intercal$, the optimal policy $\boldsymbol{\alpha}^*$ achieves $G\left(\sum_{k=1}^{K} \alpha_k^* \boldsymbol{\mu}_k; \mathbf{w}\right) = G\left(\boldsymbol{\alpha}^{*\intercal} \boldsymbol{\mu}; \mathbf{w}\right)$ if the randomness of the costs are not taken into account. We consider the performance of the optimal policy as a reference value, and define the regret of the learner as the difference of the GGI of its average cost and the GGI of the optimal policy:

$$R^{(T)} = G\left(\mathbf{X}^{(T)}; \mathbf{w}\right) - G\left(\boldsymbol{\alpha}^{*\intercal} \boldsymbol{\mu}; \mathbf{w}\right) \ .$$

Note that the GGI is a continuous function, therefore if the learner follows a policy $\boldsymbol{\alpha}^{(T)}$ that is "approaching" to $\boldsymbol{\alpha}^*$ as $T \to \infty$, then the regret is vanishing. In this paper, we are interested in the expected regret $\mathbb{E}\left[R^{(T)}\right]$ where the expectation is meant with respect to the randomization of the learner and on the randomness of the costs.

7 Learning algorithms

In this section we propose two algorithms, which can optimize the regret defined in the previous section. The first one is devised based on the principle of "optimism in the face of uncertainty" [1], that is, the algorithm makes its decision based on the optimistic estimate of the arm means. The second algorithm exploits the convexity of the GGI operator and formalizes the policy search problem as an online convex optimization problem, which is solved by a gradient descent algorithm with projection [15].

7.1 Optimistic algorithm

The principle of "optimism in the face of uncertainty" is very general and applies to many bandit problems where the environment is stochastic. Assume a learner who has already observed costs on various arms. Based on the observations, the mean of the arm distributions can be estimated with some precision. Then the learner selects the next arm based on the optimistic estimates, that is, the best possible estimates, which is in our setting the high probability lower confidence bounds.

This principle can be easily adapted to our multi-objective setup as well. As mentioned before, with the knowledge of the arm means $\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K)^\intercal$, the optimal policy can be computed by solving the linear program given in (9). The idea of our algorithm is to solve the same linear program but the arm means are replaced by their optimistic estimates. That is, the learner solves the following linear program in each time step t:

$$\begin{aligned} \text{minimize} \quad & \sum_{d=1}^{D} w'_d \left(d r_d + \sum_{j=1}^{D} b_{j,d} \right) \\ \text{subject to} \quad & r_d + b_{j,d} \geq \sum_{k=1}^{K} \alpha_k \left(\widehat{\mu}_{k,j}^{(t)} - c_k^{(t)} \right) \ \ \forall j, d \in [D] \\ & \boldsymbol{\alpha}^T \mathbf{1} = 1 \\ & \boldsymbol{\alpha} \geq \mathbf{0} \\ & b_{j,d} \geq 0 \qquad \forall j, d \in [D] \end{aligned}$$

where $\widehat{\boldsymbol{\mu}}_k^{(t)} = \left[\widehat{\mu}_{k,j}^{(t)} \right]_{1 \leq j \leq D}$ is the mean estimate of kth arm based on the observed costs up to time t and $c_k^{(t)}$ is the confidence interval defined as

$$c_k^{(t)} = \sqrt{\frac{2 \log t}{T_k(t)}} \ .$$

The confidence interval was motivated by the UCB algorithm [1]. We refer to this algorithm as OPTIMISTIC.

7.2 Gradient descent

The multi-objective regret optimization problem can be viewed as a convex optimization task, since the GGI function $G(\mathbf{x}; \mathbf{w})$ is convex in $\mathbf{x}$. Moreover, with the precise knowledge of the arm means $\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K)^\intercal$, one may compute the gradient of $G(\boldsymbol{\alpha}^\intercal \boldsymbol{\mu}; \mathbf{w})$ with respect of $\boldsymbol{\alpha}$, which allows us to apply some online convex optimization technique [16] to tackle this online learning problem. In this section, we devise an algorithm that is motivated by this observation and that uses the empirical estimates of the arm means.

We shall make use of a well-known online convex optimization algorithm called ADAGRAD [17] that computes the gradient step in an adaptive way by scaling the length of the descent step based on the gradients observed in the previous rounds. In each round, the ADAGRAD algorithm first takes a gradient descent step and then project the new point back to the feasible domain. The domain of our optimization problem is $\mathcal{A} = \{\boldsymbol{\alpha} \in \mathbb{R}^K | \sum_{k=1}^K \alpha_k = 1 \wedge 0 \preceq \boldsymbol{\alpha}\}$, which is a convex set. First, assume a policy $\boldsymbol{\alpha}$ whose GGI is $G(\boldsymbol{\alpha}^\intercal \boldsymbol{\mu}; \mathbf{w})$. The gradient of GGI with respect to α_i can be computed as

$$\frac{\partial G(\boldsymbol{\alpha}^\intercal \boldsymbol{\mu}; \mathbf{w})}{\partial \alpha_k} = \sum_{d=1}^{D} w_d \mu_{k,\pi(d)} \ . \tag{10}$$

where π is the permutation that sorts the components of $\boldsymbol{\alpha}^\intercal \boldsymbol{\mu}$ in a decreasing order. Means $\boldsymbol{\mu}_k$'s are not known but they can be estimated based on the costs observed so far.

The multi-objective gradient descent algorithm is defined in Algorithm 1, which we shall refer to as MO-ADAGRAD. The algorithm first pulls each arm at once as an initialization step. Then in each iteration, it computes an estimate of $\boldsymbol{\alpha}^{(t)\intercal} \boldsymbol{\mu}$, which is the expected cumulative cost of the current policy $\boldsymbol{\alpha}^{(t)}$ (line 6). This estimate is used to sort the objectives in a decreasing order, which we need to compute the gradient estimate of the GGI with respect to the current policy $\boldsymbol{\alpha}^{(t)}$ given in (10). As a next step, the algorithms takes

Algorithm 1 MO-ADAGRAD (η)

1: **for** rounds $t = 1 \to K$ **do**
2: set $k_t = t$ and $T_{k_t} = 1$
3: pull arm k_t and observe sample $\mathbf{X}_{k_t}^{(t)}$
4: Set $\boldsymbol{\alpha}^{(K)} = (1/K, \cdots, 1/K)$ and $\boldsymbol{G}^{(K)} = \mathbf{0}$
5: **for** rounds $t = K+1, K+2, \ldots$ **do**
6: $\mathbf{v}^{(t)} = \boldsymbol{\alpha}^{(t)\intercal} \widehat{\boldsymbol{\mu}}^{(t)}$ ▷ $\mathbf{v}^{(t)} \approx \boldsymbol{\alpha}^{(t)\intercal} \boldsymbol{\mu}$
7: $\pi = \text{argsort}\left(\mathbf{v}^{(t)}\right)$ ▷ Permutation for gradient estimate
8: **for** $k = 1 \to K$ **do** ▷ Compute the gradient estimate
9: $g_k^{(t)} = 0$
10: **for** $d = 1 \to D$ **do**
11: $g_k^{(t)} = g_k^{(t)} + w_d \widehat{\boldsymbol{\mu}}_{k,\pi(d)}^{(t)}$
12: $\boldsymbol{G}^{(t)} = \boldsymbol{G}^{(t-1)} + \boldsymbol{g}^{(t)\intercal} \boldsymbol{g}^{(t)}$ ▷ Gradient step
13: $\hat{\boldsymbol{\alpha}}^{(t)} = \boldsymbol{\alpha}^{(t-1)} - \eta {\boldsymbol{G}^{(t)}}^{-1/2} \mathbf{g}^{(t)}$
14: $\boldsymbol{\alpha}^{(t)} = \text{argmin}_{\boldsymbol{\alpha} \in \mathcal{A}} \|\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}^{(t)}\|_{G^{(t)1/2}}$ ▷ Projection
15: Choose an arm k_t according to $\boldsymbol{\alpha}^{(t)}$
16: Set $T_{k_t} = T_{k_t} + 1$
17: Pull arm k_t, and observe sample $\mathbf{X}_{k_t}^{(t)}$

the gradient step (line 12) according to the ADAGRAD algorithm and carries out the projection step where

$$\|\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}^{(t)}\|_{\boldsymbol{G}} = \left(\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}^{(t)}\right)^\intercal \boldsymbol{G} \left(\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}^{(t)}\right) \ .$$

The rationale behind this gradient step and projection is that the value of the multi-variate function to be optimized might be changing faster in some directions resulting in larger step concerning this direction, and the step size, of course, depends on the value of the gradient. This kind of diversity among the step size of various directions is taken into account in the projection step by scaling the distance by the inverse of the cumulative gradient ${G^{(t)}}^{1/2}$. Finally, the MO-ADAGRAD chooses an arm according to $\boldsymbol{\alpha}^{(t)}$ and observes the vectorial cost.

The algorithm has only one hyperparameter, which is the step size η. We found this algorithm robust to this hyperparameter, which might be explained by its adaptive nature. We always set the value of η to 0.1 in our experiments.

8 Experiments

To test our algorithms, we carried out two sets of experiments. In the first we generated synthetic data from multi-objective bandit instances with known parameters. In this way, we could compute the regret defined in Section 6 and, thus investigate the empirical performance of the algorithms. In the second set of experiments, we run our algorithm on a complex multi-objective online optimization problem, namely an electric battery control problem.

8.1 Synthetic Experiments

We generated random multi-objective bandit instances for which each component of the multivariate cost distributions obeys Bernoulli distribution with various parameters. The parameters of each Bernoulli distributions are drawn uniformly at random from $[0, 1]$ independently from each other. The number of arms K was set to 10 and the dimension of the reward distribution was taken from $D \in \{2, 5, 10, 20\}$. The weight vector $\mathbf{w}$ of GGI was set to $w_d = 1/2^{d-1}$. Since the parameters of the bandit instance are known, we could

compute the regret defined in Section 6. We ran the MO-ADAGRAD and OPTIMISTIC algorithms with 20 repetitions. The multi-objective bandit instance were regenerated after each run. The regrets of the two algorithms, which are averaged out over the repetitions, are plotted in Figure 1. The results reveal some general trends. First, the average regrets of both algorithms converge to zero. Second the OPTIMISTIC algorithm outperforms the gradient descent algorithm for small number of round, typically $T < 5000$. This fact might be explained by the fact that the optimistic algorithm solves a linear program for estimating α^* whereas the MO-ADAGRAD minimizes the same objective but using a gradient descent approach with projection, which might achieve slower convergence in terms of regret, but its computational time is significantly decreased compared to the optimistic algorithm.

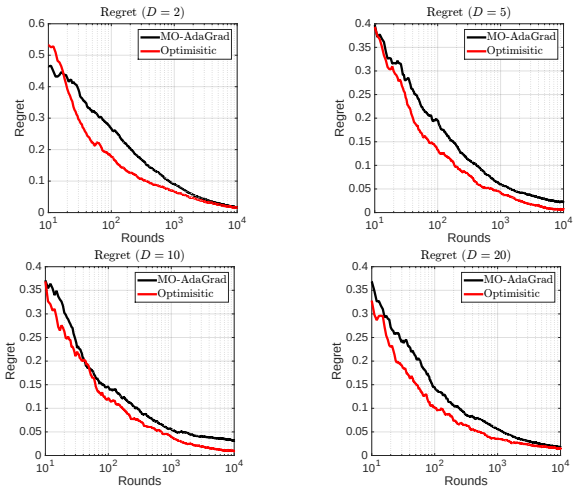

Figure 1. The regret of the OPTIMISTIC and MO-ADAGRAD. The regret is averaged over 20 repetitions and plotted in terms of the number of rounds. The dimension of the arm distributions was set to $D \in \{2, 5, 10, 20\}$, which is indicated in the title of the panels.

8.2 Battery control task

Efficient management of electric batteries leads to better performance and longer battery life, which is important for instance for electric vehicles whose range is still limited compared to those with petrol or diesel engines. An electric battery is composed of several cells whose capacity varies extensively due to inconsistencies in weight and volume of the active material in their individual components, different internal resistance and higher temperatures leading to different aging rates. As a consequence, at any instant, the energy output is different from each cell, which ultimately results in faster rates of decay and ultimately leads to the failure of the battery. To address this problem, a control strategy called cell balancing is utilized, which aims at maintaining a constant energy level — mainly state-of-charge (SOC) — in each cell, while controlling for temperature and aging. Many cell-balancing controllers can be defined, depending on the importance given to the three objectives: SOC, temperature and aging. The values of those objectives should be balanced between the cells because a balanced use of all cells leads to a long lasting system. Moreover, those objectives values should also be balanced within a cell, because for example, a cell can have higher capacity on a higher temperature, but at the same time it has a higher risk to explode.

Our battery control task consists in learning and selecting the "best" cell balancing in an online fashion, so that it can dynamically adapt to the consumption profile and environment, such as outdoor temperature. In case of electric cars, this means that the controller needs to be able to adapt to the driving habits of the driver and to the terrain, such as hilly roads or desert roads. In this experiment, our goal was more specifically to test our multi-objective online learning algorithms in the battery control task, and verify that our online learning algorithms can indeed find a policy for this control task which leads to a balanced parameter values of the cells.

The battery is modeled using the internal resistance (Rint) model [18]. The estimation of SOC is based on the Ampere Hour Counting method [19]. The variation of temperature in the system is determined according to the dissipative heat loss due to the internal resistance and thermal convection [20]. Cell degradation or aging is a function of temperature, charging/discharging rates and cumulative charge [21]. Moreover, the battery model is complemented with 10 different cell-balancing controllers. The whole model is implemented in the Matlab/Simulink software package and can emulate any virtual situation whose electric consumption is described by a time series, which is given as input to the model. For a chosen controller, the output of the model comprises of the objective values of each battery cell at the end of the simulation. Note that the output of the battery model is a multivariate random vector since the power demand is randomized, therefore this control task can be readily accommodated into our setup. In our experiments, the battery consists of 4 cells, thus $D = 12$ in this case. The cell-balancing controllers correspond to the arms, thus $K = 10$.

The online learning task consists of selecting among these controllers/arms so that the final objective values are as balanced as possible. The experiment was carried out as follows: in each iteration, the learner selects a controller according to its policy, then the battery model is run with the selected controller by using a random consumption time series. At the end of the simulation, the learner receives the objective values of each cell output by the battery model as feedback and updates its policy. The goal of the learner is to find a policy over the controllers, which leads to a balanced state of cells in terms of cumulative value.

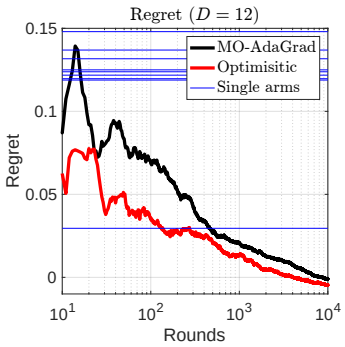

Figure 2. The regret of the OPTIMISTIC and MO-ADAGRAD on the battery control task. The regret is averaged over 10 repetitions and plotted in terms of the number of rounds. The dimension of the arm distributions was $D = 12$.

The results are shown in Figure 2. In this experiments we do not know the means of the arms, but we estimated them based on 30 runs. These mean estimates were used for computing the optimal policy and the regret. We run the MO-ADAGRAD and OPTIMISTIC

over 10 repetitions. Their average regret exhibits the same trend like in the synthetic experiments: the OPTIMISTIC achieved faster convergence. The blue lines shows the regret of the pure policies, which selects always the same arm, i.e., the performance of single strategies. It is important to mention that the optimal mixed controller has lower GGI value since the regret of any arm is positive, and more importantly, the two learners converge to optimal mixed policies in terms of regret. Note that the regret for larger number of time steps is slightly negative, which stems from the fact that we estimated the means of the arms (because their true values are not known).

9 Related work

Multi-armed bandit problems have generated significant theoretical interest, and they have been applied to many real applications [22, 23]. The single-objective MAB problem has been intensively studied especially in recent years, nevertheless there is only a very limited number of work concerning the multi-objective setting. To the best of our best knowledge, Drugan and Nowé [24] considered first the multi-objective multi-armed problem in a regret optimization framework with a stochastic assumption. Their work consists of extending the UCB algorithm [1] so as to be able to handle multi-dimensional feedback vectors with the goal of determining all arms on the Pareto front. Their algorithm makes use of the optimistic estimate of the arm means like our algorithm, but it chooses an arm uniformly at random from the set of arms whose optimistic estimate is on the Pareto front. Their regret definition is based on the distance between the mean of the arms and the Pareto front. Therefore contrary to our regret notion, their regret does not encode fairness.

Azar et al. [25] investigated a sequential decision making problem with vectorial feedback. In their setup the agent is allowed to choose from a finite set of actions and then it observes the vectorial feedback for each action, thus it is a full information setup whereas our setup is a bandit information setting because the agent observes only the feedback corresponding to the chosen arm. Moreover, the feedback is non-stochastic in their setup, as it is chosen by an adversary. They propose an algorithm which can handle a general class of aggregation function, such as the set of bounded domain, continuous, Lipschitz and quasi-concave functions.

10 Conclusion and future work

We introduced a new problem in the context of multi-objective multi-armed bandit (MOMAB). Contrary to most previously proposed approaches in MOMAB, we do not search for the Pareto front, instead we aim for a fair solution, which is important for instance when each objective corresponds to the payoff of a different agent. To encode fairness, we use the well-known generalized Gini index (GGI), a criterion developed in economics. To optimize this criterion, we proposed two algorithms, one based on the principle of "optimisim in the face of uncertainty" and the other exploiting the convexity of GGI. We evaluated both algorithms on two domains and obtained promising experimental results. First, we validated our propositions on random instances of MOMAB. Then, we tried the two algorithms on a more realistic task, which is an electric battery control problem.

As future work, a theoretical analysis is needed for the two algorithms we proposed. Indeed, we plan to prove regret bounds for them and obtain matching lower bounds. Moreover, while we focused in this paper on the stochastic setting, it would be worthwhile to extend our work to the adversarial setting. Finally, it would also be interesting to extend this work to the contextual multi-armed bandit and/or the reinforcement learning setting, which would be useful to solve the electric battery control problem even more finely.

REFERENCES

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.

[2] T. L. Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.

[3] A. Slivkins. Contextual bandits with similarity information. *J. Mach. Learn. Res.*, 15:2533–2568, 2014.

[4] W.H. Press. Bandit solutions provide unified ethical models for randomized clinical trials and comparative effectiveness research. In *Proceedings of the National Academy of Sciences*, volume 106, pages 22398–22392, 2009.

[5] John A. Weymark. Generalized gini inequality indices. *Mathematical Social Sciences*, 1(4):409 – 430, 1981.

[6] William James and Charles Stein. Estimation with quadratic loss. In *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 361–379, 1961.

[7] R.E. Steuer and E.-U. Choo. An interactive weighted Tchebycheff procedure for multiple objective programming. *Mathematical Programming*, 26:326–344, 1983.

[8] R.R. Yager. On ordered weighted averaging aggregation operators in multi-criteria decision making. *IEEE Trans. on Syst., Man and Cyb.*, 18:183–190, 1988.

[9] W. Ogryczak, P. Perny, and P. Weng. On minimizing ordered weighted regrets in multiobjective Markov decision processes. In *International Conference on Algorithmic Decision Theory (ADT)*, Lecture Notes in Artificial Intelligence. Springer, 2011.

[10] W. Ogryczak, P. Perny, and P. Weng. A compromise programming approach to multiobjective Markov decision processes. *International Journal of Information Technology & Decision Making*, 12:1021–1053, 2013.

[11] Zoltán Buczolich and Gábor J. Székely. When is a weighted average of ordered sample elements a maximum likelihood estimator of the location parameter? *Advances in Applied Mathematics*, 10(4):439 – 456, 1989.

[12] A. Pigou. *Wealth and Welfare*. Macmillan, 1912.

[13] H. Dalton. The measurement of inequality of incomes. *Economic J.*, 30(348–361), 1920.

[14] W. Ogryczak and T. Sliwinski. On solving linear programs with the ordered weighted averaging objective. *Eur. J. Operational Research*, 148:80–91, 2003.

[15] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML 2003*, 2003.

[16] Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2012.

[17] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011.

[18] V.H. Johnson. Battery performance models in ADVISOR. *Journal of Power Sources*, 110:312–329, 2002.

[19] J. Dambrowski. Review on methods of state-of-charge estimation with viewpoint to the modern $LiFePO_4$/$Li_4Ti_5O_{12}$ lithium-ion systems. In *International Telecommunication Energy Conference*, 2013.

[20] Lijun Gao, Shenyi Chen, and Roger A. Dougal. Dynamic lithium-ion battery model for system simulation. *IEEE Trans. on Components and Packaging Technologies*, 25(3):495–505, 2002.

[21] Gao Tao. Research on $LiMn_2O_4$ battery's state of charge estimation with the consideration of degradation. Technical report, Tsinghua University, 2012.

[22] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge Univ Pr, 2006.

[23] S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

[24] M. M. Drugan and A. Nowé. Designing multi-objective multi-armed bandits algorithms: A study. In *IJCNN*, pages 1–8, 2013.

[25] Yossi Azar, Uriel Feige, Michal Feldman, and Moshe Tennenholtz. Sequential decision making with vector outcomes. In *ITCS*, pages 195–206, 2014.