

Supplementary reading S6

Phi-regret minimization

Instructor: Max Fishelson (maxfish@mit.edu)

We saw in earlier lectures the definition of Φ -regret. As we have seen, external regret minimization is a very narrow instantiation of Φ -regret minimization—perhaps the smallest sensible instantiation. Then, clearly, the problem of coming up with a regret minimizer for a set X cannot be harder than the problem of coming up with a Φ -regret minimizer for X for richer sets of transformation functions Φ . It might then seem surprising that there exists a construction that reduces Φ -regret minimization to regret minimization. We see this general construction in Section S6.2. First, we start with an example of this principle, whose discovery predates the general construction of Section S6.2, and which happens to be a special case of it.

S6.1 Blum-Mansour's swap regret minimization algorithm

Let's start by considering the task of constructing a swap regret minimizer for probability simplex $X = \Delta^n$ over n actions $A = \{1, \dots, n\}$. As we mentioned when discussing learning, when all players in a normal-form game play according to the strategies generated by such an algorithm, the average product strategy converges to the set of correlated equilibria of the game. The swap regret minimization algorithm of [Blum, A., & Mansour, Y. \[BM07\]](#) is a simple and elegant algorithm that minimizes *swap* regret by starting from multiple copies of an *external* regret minimizer.

Recall that in swap regret minimization, the learner wants to minimize the Φ -regret with respect to the set of strategy transformations Φ represented by stochastic matrices

$$\Phi := \{P = (p_1 \mid \dots \mid p_n) : p_1, \dots, p_n \in \Delta^n\}.$$

In order to construct a swap regret minimizer for $X = \Delta^n$, we start with $|A|$ copies of an external regret minimizer for Δ^n , denoted by R_i .

- To compute the `NextStrategy()` of the swap regret minimizer, we first call the `NextStrategy()` of each R_i to obtain a distribution $p_i^{(t)} \in \Delta^n$. We will then assemble all $p_i^{(t)}$ into a stochastic matrix $P^{(t)}$, and will output a fixed point

$$x^{(t)} = P^{(t)}x^{(t)} = \sum_{i=1}^n x_i^{(t)} p_i^{(t)} \in \Delta^n.$$

- To compute the $\text{ObserveUtility}(u^{(t)})$ of the swap regret minimizer, we first call the ObserveUtility of each R_i with the linear utility function

$$v_i^{(t)} := x_i^{(t)} u^{(t)}$$

that is obtained by rescaling $u^{(t)}$ by the component $x_i^{(t)}$ of the last-output strategy $x^{(t)}$.

The process can be depicted pictorially as in Figure 1.

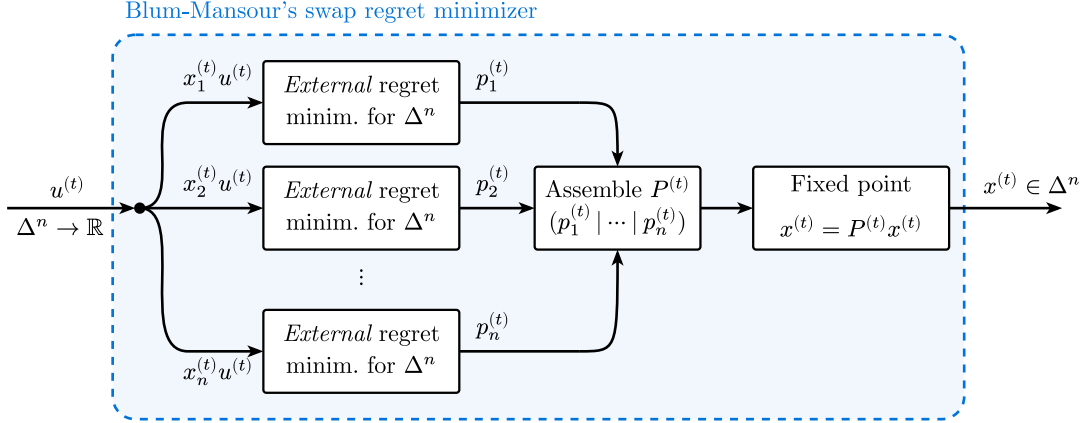


Figure 1: Blum-Mansour's swap regret minimization algorithm for Δ^n .

We now claim that the algorithm described above is a swap regret minimizer for Δ^n .

Theorem S6.1. Let $\text{Reg}_i^{(T)}$ denote the regret incurred by each external regret minimizer R_i for Δ^n in the Blum-Mansour construction described above. Then, the swap regret cumulated by the algorithm satisfies:

$$\text{SwapReg}^{(T)} = \sum_{i=1}^n \text{Reg}_i^{(T)}.$$

In particular, the swap regret grows sublinearly in T whenever the external regret minimizers R_i guarantee sublinear regret.

Proof. By construction, the external regret incurred by each R_i is

$$\text{Reg}_i^{(T)} = \max_{\tilde{p}_i \in \Delta^n} \sum_{t=1}^T \left(v^{(t)}(\tilde{p}_i) - v^{(t)}(p_i^{(t)}) \right). \quad (1)$$

Pick any $\hat{P} = (\hat{p}_1 | \dots | \hat{p}_n) \in \Phi$. Then, the swap regret cumulated compared to always transforming strategies according to $\hat{P}$ is, by definition,

$$\begin{aligned}
& \sum_{t=1}^T u^{(t)}(\hat{P}x^{(t)}) - u^{(t)}(x^{(t)}) \\
&= \sum_{t=1}^T u^{(t)}(\hat{P}x^{(t)}) - u^{(t)}(P^{(t)}x^{(t)}) && (x^{(t)} = P^{(t)}x^{(t)}) \\
&= \sum_{t=1}^T \left[\left(\sum_{i=1}^n x_i^{(t)} u^{(t)}(\hat{p}_i) \right) - \left(\sum_{i=1}^n x_i^{(t)} u^{(t)}(p_i^{(t)}) \right) \right] && (\text{linearity of } u^{(t)}) \\
&= \sum_{t=1}^T \left(\sum_{i=1}^n v_i^{(t)}(\hat{p}_i) - v_i^{(t)}(p_i^{(t)}) \right) && (\text{definition of } v_i^{(t)}) \\
&= \sum_{i=1}^n \left(\sum_{t=1}^T v_i^{(t)}(\hat{p}_i) - v_i^{(t)}(p_i^{(t)}) \right) && (\text{switching summation order}) \\
&\leq \sum_{i=1}^n \text{Reg}_i^{(T)}. && (\text{from (1)})
\end{aligned}$$

Taking a maximum over all $\hat{P} \in \Phi$ gives equality: each column $\hat{p}_i$ can be chosen independently in Δ^n to attain the corresponding external-regret maximum. This concludes the proof. $\square$

S6.2 The general case: Gordon-Greenwald-Marks's reduction

Blum-Mansour's swap regret minimization algorithm is a special case of a much more general construction. [Gordon, G. J., Greenwald, A., & Marks, C. \[GGM08\]](#) show that Φ -regret minimization for a strategy set X can be constructed starting from the following two ingredients:

1. an *external* regret minimization for the set Φ ; and
2. a *fixed point oracle* Φ , that is, an algorithm that given any $\phi \in \Phi$ outputs a fixed point $\phi(x) = x \in X$.

Intuitively, the external regret minimizer for Φ has the role of tracking which transformation ϕ the decision maker should focus on at each time. The linear utility function $U^{(t)} : \Phi \rightarrow \mathbb{R}$ observed by the external regret minimizer is constructed from the last-output strategy $x^{(t)}$ and the utility function $u^{(t)}$ observed at time t , according to the formula

$$U^{(t)}(\phi) = u^{(t)}(\phi(x^{(t)})), \quad (2)$$

where $x^{(t)}$ is the last-output strategy. The final construction is as follows:

- Each call to `NextStrategy` first calls `R.NextStrategy` to obtain the next transformation $\phi^{(t)}$. Then, a fixed point $x^{(t)} = \phi^{(t)}(x^{(t)}) \in X$ is computed and output.
- Each call to `ObserveUtility`($u^{(t)}$) with linear utility function $u^{(t)}$ constructs the linear utility function $U^{(t)} : \phi \mapsto u^{(t)}(\phi(x^{(t)}))$ given in (2), and passes it to `R` via `R.ObserveUtility`($U^{(t)}$).

Graphically, we can summarize the process as in the following block diagram.

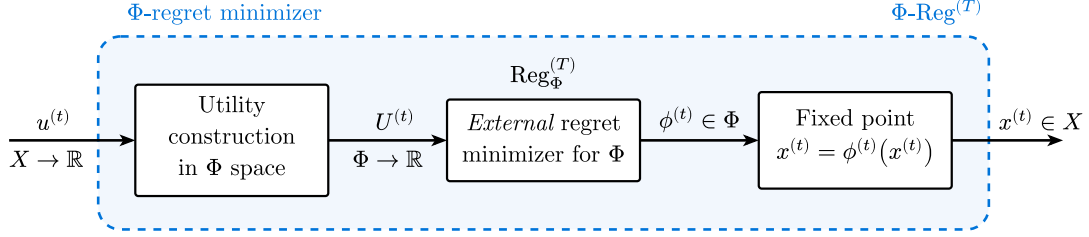


Figure 2: Gordon-Greenwald-Marks’s reduction from Φ -regret minimization to external regret minimization for Φ plus a fixed point oracle for Φ .

Theorem S6.2 ([GGM08]). The Φ -regret $\Phi\text{-Reg}^{(T)}$ cumulated up to time T by the algorithm we just defined is exactly equal to the (external) cumulative regret $\text{Reg}_{\Phi}^{(T)}$ cumulated by R :

$$\Phi\text{-Reg}^{(T)} = \text{Reg}_{\Phi}^{(T)} \quad \forall T = 1, 2, \dots$$

Because the regret cumulated by R grows sublinearly by hypothesis of it being a regret minimizer, then so does the Φ -regret of the Φ -regret minimization algorithm defined above.

Proof. The proof of correctness of the above construction is deceptively simple. Since R outputs transformations $\phi^{(1)}, \phi^{(2)}, \dots \in \Phi$ and receives utilities $\phi \mapsto u^{(1)}(\phi(x^{(1)})), \phi \mapsto u^{(2)}(\phi(x^{(2)})), \dots$, its cumulative regret $R^{(T)}$ is by definition

$$\text{Reg}_{\Phi}^{(T)} = \max_{\hat{\phi} \in \Phi} \left\{ \sum_{t=1}^T (u^{(t)}(\hat{\phi}(x^{(t)})) - u^{(t)}(\phi^{(t)}(x^{(t)}))) \right\}.$$

Now, since by construction $x^{(t)}$ is a fixed point of $\phi^{(t)}$, $\phi^{(t)}(x^{(t)}) = x^{(t)}$, and therefore we can write

$$\text{Reg}_{\Phi}^{(T)} = \max_{\hat{\phi} \in \Phi} \left\{ \sum_{t=1}^T (u^{(t)}(\hat{\phi}(x^{(t)})) - u^{(t)}(x^{(t)})) \right\},$$

where the right-hand side is exactly the cumulative Φ -regret $\Phi\text{-Reg}^{(T)}$ incurred by R_{Φ} . $\square$

Bibliography

- [BM07] A. Blum and Y. Mansour, “From external to internal regret.,” *Journal of Machine Learning Research*, vol. 8, no. 6, 2007.
- [GGM08] G. J. Gordon, A. Greenwald, and C. Marks, “No-regret learning in convex games,” in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 360–367.