# Regret Minimization and Job Scheduling

Yishay Mansour[⋆]

Blavatnik School of Computer Science
Tel Aviv University, Tel Aviv, Israel
`mansour@tau.ac.il`

**Abstract.** Regret minimization has proven to be a very powerful tool in both computational learning theory and online algorithms. Regret minimization algorithms can guarantee, for a single decision maker, a near optimal behavior under fairly adversarial assumptions. I will discuss a recent extensions of the classical regret minimization model, which enable to handle many different settings related to job scheduling, and guarantee the near optimal online behavior.

## 1 Regret Minimization

Consider a single decision maker attempting to optimize it performance in face of an uncertain environment. This simple online setting has attracted attention from multiple disciplines, including operations research, game theory, and computer science. In computer science, computational learning theory and online algorithms both focus on this task from different perspectives. I will concentrate only on a certain facet of this general issue of decision making, and consider settings related to *regret minimization*, where the performance of the online decision maker is compared to a benchmark based on a class of comparison policies.

Regret minimization has its roots in computational learning theory and game theory. While the motivation for the research in the two fields have been very different, the basic model that both fields has studied have been very similar. They both consider an online setting, where an agent needs to select actions, while having only information about the past performance and having no (or very limited) information regarding the future. Many natural computer science problems give rise to such online settings; typical examples include scheduling problems, paging, routing protocols, and many more. Online regret minimization learning algorithms have been introduced and studied in the computational learning community [27,19,2,13,24] and also in the game theory community [20,17,16,21]. (See [11] for an excellent book on the topic.)

The online model studied has the following general structure. In each time step, the online algorithm needs to select an action, and it can select a specific

⋆ This work was supported in part by the IST Programme of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886, by a grant from the Israel Science Foundation (grant No. 709/09) and grant No. 2008-321 from the United States-Israel Binational Science Foundation (BSF). This publication reflects the authors' views only.

action or a convex combination of some of the actions. After the online algorithm performs its action, it observes the loss of all the actions and receives a loss for the action it chose. The aim is to minimize the total cumulative loss. In general, one would like to prove guarantees that hold for arbitrary loss sequences; that is, one imagines there is a powerful adversary that might generates the worse loss sequence for the online learning algorithm.

When evaluating the performance of online algorithms, it is important to select the "right" benchmark. On the one hand we like it to be sufficiently challenging, so that it will encourage innovative algorithms. On the other hand we need to keep the expectation realistic, which will allow to introduce algorithms and not only impossibility results. One way to think about the benchmark is a set of algorithms that we like to match the performance of the *best* of them. In the above online setting it is clear that no online algorithm can hope to compete well against *any* algorithm, simply consider the case of predicting a random coin, any online algorithm will succeed in the prediction only half of the times (in expectation) while someone who first views the coins outcome will be able to predict perfectly. External regret bounds the *difference* between the online algorithm loss and the best algorithm in a comparison class of algorithms, and generally one can achieve a regret bound of the form $O(\sqrt{T \log N})$, where $T$ is the number of time steps, $N$ are the number of algorithms in the comparison class and assuming that the losses are from $[0, 1]$ (see [13,14]). This implies that the per step regret is vanishing at the rate of $O(\sqrt{(\log N)/T})$. From an online algorithm perspective, the external regret can be viewed comparing the online algorithm to the best *static* solution. In the *multi-arm bandit* setting [28,25] the decision maker observes only the payoff of the action it selected (and does not get any information regrading the other actions). In this setting the average regret vanishes at the rate of $O(\sqrt{(N \log N)/T})$ for the adversarial setting [2] and $O((\log T)/T)$ for the stochastic setting [1].[1]

## 2 Regret Minimization and Job Scheduling

In an online job scheduling setting, at each time step a job arrives and the online algorithm needs to schedule it on one of the machines. At the end of the run, the online algorithm has a certain load on each machine (depending on the jobs it scheduled on it) and the global loss function can be either makespan (the load on the most loaded machine) or some norm of the loads (e.g., the sum of the square of the loads). Such objective functions are very different from the additive objective function usually used in regret minimization.

It is worth first discussing the differences between the regret minimization model, suggested here, and the classical job scheduling model. We have a very different information model, where the decision maker discovers the actions outcome (e.g., in job scheduling this is the load of the job on each machine) only

[1] The $O$-notation in the stochastic setting hides dependency on the stochastic parameters.

*after* it selects an action (e.g., a machine where to schedule the job) and not before (e.g., when the job arrives). In contrast, in the classical online job scheduling model, when a job arrives, the decision maker, first observes the load of the job on each machine (the outcome of the actions) and only then selects an action.

Our information model is the "right" abstraction in many applications. For example, consider a *network load balancer* (an *online algorithm*) which has to select an outgoing link for each session (the *action*). The load balancer goal is to minimize the load on the most loaded link (known as *makespan*). When a session arrives, the load balancer needs to be scheduled on an output link before we learn the load of the session. After we scheduled the session we can observe its load, but then we can not change the link on which it is already scheduled.

It is worthwhile to note the difference between our setting and that of online job scheduling in the competitive analysis literature [10]. The main difference is in the *information* that the decision maker observes about the online tasks that arrive. In the online job scheduling setting the "standard" assumption is that the decision maker first observes the "load" of the job on each machine, and only then decides on which machine to schedule it. In our model, much like the regret minimization model, we have a different information model. First the decision maker selects how to schedule the job on the machines, and only then he observes the "load" of the job on each machine. This different information model is a very important distinction between the two models. The other important distinction is regarding the "benchmark class", which in the competitive analysis is the optimal hindsight assignment of jobs to machines, and in our case it is a significantly more limited class. Finally, there is a very significant difference in the results we would like to derive. We are aiming at getting a bounded regret, which is the *difference* between the online cost and the minimal cost policy in the benchmark class, while the competitive analysis is satisfied with bounding the *ratio*.

## 3 Model and Results

In this section we sketch the model and the results of [15], which would be our main source.

We are interested in studying the following extension of the regret minimization model. For each action we will maintain the cumulative loss of the online algorithm resulting from this action. The online algorithm *global loss function* would be a given (convex) function of the cumulative loss of the actions. (The makespan is a perfect example of such a global loss function.) The online algorithm objective is to minimize the loss of the global loss function. For example, assume that $\ell_i^t \in [0,1]$ is the loss at time $t$ from action $i$, and let $L_i^T = \sum_{t=1}^T \ell_i^t$ be the cumulative loss of action $i$. At time $t$, first, the online algorithm specifies a distribution $p_i^t$ over the actions, and then it observes the losses of the different actions. Its loss from action $i$ at time $t$ is $p_i^t \ell_i^t$, and the cumulative loss of action $i$ is $L_i^{ON,T} = \sum_{t=1}^T p_i^t \ell_i^t$. Unlike the usual regret minimization model, we will not sum the losses of the online algorithm for different action, but consider the global loss function over the $L_i^{ON,T}$. For example, the makespan cost of the online algorithm is $\max_i\{L_i^{ON,T}\}$.

We first need to select an appropriate benchmark, and a reasonable benchmark class is the class of policies that *statically* distributes the weight between actions (and receives the proportional loss in each action). Specifically, given loads $L_1, \ldots, L_N$, for the makespan the best weights are $p_i^* = \frac{L_i^{-1}}{\sum_{j=1}^N L_j^{-1}}$, and the makespan is $\frac{1}{\sum_{j=1}^N L_j^{-1}}$. As before, the regret is the difference between the global loss function (e.g., makespan) of the online algorithm and that of the best set of weights, and the average regret per step is the regret divided by the number of steps.

In the talk we will discuss both an adversarial model (described above) and a stochastic model of losses. For the adversarial mode, the main results appear in [15] and include a general online algorithm whose average regret is vanishing at the rate of $O(\sqrt{N/T})$.

## 4 Other Extensions of the Regret Minimization Model

Unfortunately, we can not give a comprehensive review of the large body of research on regret minimization algorithm, and we refer the interested reader to [12]. In the following we highlight a few more relevant research directions.

There has been an ongoing interest in extending the basic comparison class for the regret minimization algorithm, for example by introducing *shifting experts* [18], *time selection functions* [5], and *wide range regret* [26]. Still, all those works assume that the loss is additive between time steps.

A different research direction has been to improve the computational complexity of the regret minimization algorithms, especially in the case that the comparison class is exponential in size. General computationally efficient transformation where given by [23], in the case that the cost function is linear and the optimization oracle can be computed in polynomial time, and extended by [22], to the case of an approximate-optimization oracle.

There has been a sequence of works establishing the connection between online competitive algorithms [9] and online learning algorithm [12]. One issue is that online learning algorithms are stateless, while many of the problems address in the competitive analysis literature have a state (see, [6]). For many problems one can use the online learning algorithms and guarantee a near-optimal static solution, however a straightforward application requires both exponential time and space. Computationally efficient solutions have been given to specific problems including, paging [7], data-structures [8], and routing [4,3].

We remark that all the above works concentrate on the case where the global cost function is additive between time steps.

## References

1. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-Time Analysis of the Multi-Armed Bandit Problem. Machine Learning 47(2-3), 235–256 (2002)
2. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: The Nonstochastic Multiarmed Bandit Problem. SIAM Journal on Computing 32(1), 48–77 (2002); (A preliminary version appeared in FOCS 1995 as Gambling in a Rigged Casino: The Adversarial Multi-Armed Bandit Problem)

3. Awerbuch, B., Kleinberg, R.: Online Linear Optimization and Adaptive Routing. J. Comput. Syst. Sci. 74(1), 97–114 (2008)
4. Awerbuch, B., Mansour, Y.: Adapting to a Reliable Network Path. In: PODC, pp. 360–367 (2003)
5. Blum, A., Mansour, Y.: From External to Internal Regret. Journal of Machine Learning Research 8, 1307–1324 (2007)
6. Blum, A., Burch, C.: On-Line Learning and the Metrical Task System Problem. In: COLT, pp. 45–53 (1997)
7. Blum, A., Burch, C., Kalai, A.: Finely-Competitive Paging. In: FOCS, pp. 450–458 (1999)
8. Blum, A., Chawla, S., Kalai, A.: Static Optimality and Dynamic Search-Optimality in Lists and Trees. Algorithmica 36(3), 249–260 (2003)
9. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press, Cambridge (1998)
10. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press, Cambridge (1998)
11. Cesa-Bianchi, N., Lugosi, G.: Prediction, Learning and Games. Cambridge University Press, Cambridge (2006)
12. Cesa-Bianchi, N., Lugosi, G.: Prediction, Learning, and Games. Cambridge University Press, New York (2006)
13. Cesa-Bianchi, N., Freund, Y., Helmbold, D.P., Haussler, D., Schapire, R.E., Warmuth, M.K.: How to Use Expert Advice. Journal of the ACM 44(3), 427–485 (1997); (A preliminary version appeared in STOC 1993)
14. Cesa-Bianchi, N., Lugosi, G.: Potential-Based Algorithms in On-Line Prediction and Game Theory. Machine Learning 51(3), 239–261 (2003)
15. Even-Dar, E., Kleinberg, R., Mannor, S., Mansour, Y.: Online Learning for Global Cost Functions. In: COLT (2009)
16. Foster, D., Vohra, R.: Regret in the On-Line Decision Problem. Games and Economic Behavior 21, 40–55 (1997)
17. Foster, D.P., Vohra, R.V.: A Randomization Rule for Selecting Forecasts. Operations Research 41(4), 704–709 (1993)
18. Freund, Y., Schapire, R.E., Singer, Y., Warmuth, M.K.: Using and Combining Predictors that Specialize. In: STOC, pp. 334–343 (1997)
19. Freund, Y., Schapire, R.E.: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. In: Euro-COLT, pp. 23–37. Springer, Heidelberg (1995)
20. Hannan, J.: Approximation to Bayes Risk in Repeated Plays. In: Dresher, M., Tucker, A., Wolfe, P. (eds.) Contributions to the Theory of Games, vol. 3, pp. 97–139. Princeton University Press, Princeton (1957)
21. Hart, S., Mas-Colell, A.: A Simple Adaptive Procedure Leading to Correlated Equilibrium. Econometrica 68, 1127–1150 (2000)
22. Kakade, S.M., Tauman-Kalai, A., Ligett, K.: Playing Games with Approximation Algorithms. In: STOC, pp. 546–555 (2007)
23. Kalai, A., Vempala, S.: Efficient Algorithms for Online Decision Problems. Journal of Computer and System Sciences 71(3), 291–307 (2005); An earlier version appeared in COLT (2003)
24. Kalai, A., Vempala, S.: Efficient Algorithms for On-Line Optimization. J. of Computer Systems and Science (JCSS) 71(3), 291–307 (2005); (A preliminary version appeared in COLT 2003)

25. Lai, T.L., Robbins, H.: Asymptotically Efficient Adaptive Allocations Rules. Advances in Applied Mathmatics 6, 4–22 (1985)
26. Lehrer, E.: A Wide Range No-Regret Theorem. Games and Economic Behavior 42, 101–115 (2003)
27. Littlestone, N., Warmuth, M.K.: The Weighted Majority Algorithm. Information and Computation 108, 212–261 (1994)
28. Robbins, H.: Some Aspects of the Sequential Design of Experiments. Bulletin of the American Mathematical Society 58, 527–535 (1952)