Online Learning with Expert Advice and Finite-Horizon Constraints

Branislav Kveton
Intel Research
Santa Clara, CA
branislav.kveton@intel.com

Jia Yuan Yu
Department of Electrical and
Computer Engineering
McGill University
jia.yu@mcgill.ca

Georgios Theocharous
Intel Research
Santa Clara, CA
georgios.theocharous@intel.com

Shie Mannor
Department of Electrical and
Computer Engineering
McGill University
shie@ece.mcgill.ca

Abstract

In this paper, we study a sequential decision making problem. The objective is to maximize the average reward accumulated over time subject to temporal cost constraints. The novelty of our setup is that the rewards and constraints are controlled by an adverse opponent. To solve our problem in a practical way, we propose an expert algorithm that guarantees both a vanishing regret and a sublinear number of violated constraints. The quality of this solution is demonstrated on a real-world power management problem. Our results support the hypothesis that online learning with convex cost constraints can be performed successfully in practice.

Introduction

Online learning with expert advice (Cesa-Bianchi & Lugosi 2006) has been studied extensively by the machine learning community. The framework has been also successfully used to solve many real-world problems, such as adaptive caching (Gramacy *et al.* 2003) or power management (Helmbold *et al.* 2000; Dhiman & Simunic 2006; Kveton *et al.* 2007). The major advantage of the online setting is that no assumption is made about the environment. As a result, there is no need to build its model and estimate its parameters. In turn, this type of learning is naturally robust to environmental changes and suitable for solving dynamic real-world problems.

In this paper, we study online learning problems with side constraints. A similar setup was considered by Mannor and Tsitsiklis (2006). Side constraints are common in real-world domains. For instance, power management problems are often formulated as maximizing power savings subject to some average performance criteria. The criteria usually restrict the rate of bad power management actions and can be naturally represented by constraints.

Our work makes two contributions. First, we show how to apply prediction with expert advice to solve online optimization problems with temporal cost constraints efficiently. Our solution is based on the mixing of expert policies that violate only a bounded number of constraints. This approach is both practical and sound. Based on our knowledge, this is the first online solution to a general class of constrained optimization problems with these properties. We believe that the solution is suitable for real-world optimization problems, which may need adaptation over time without making any statistical assumptions on the dynamics of data. To support the claim, we demonstrate the quality of our solution in a real-world power management (PM) domain. This is our second contribution.

The paper is structured as follows. First, we formulate our optimization problem and relate it to the existing work. Second, we propose and analyze a practical solution to the problem based on prediction with expert advice. Third, we evaluate the quality of our solution on a real-world PM problem. Finally, we summarize our work and suggest future research directions.

Online constrained optimization

In this paper, we study an online learning problem, where an agent wants to maximize its total reward subject to temporal cost constraints. At every time t, the agent takes some action θ_t from the action set $\mathcal{A}$, and then receives a reward $r_t(\theta_t) \in [0, 1]$ and a cost $c_t(\theta_t) \in [0, 1]$. We assume that our agent has no prior knowledge on reward and cost functions except that they are bounded. Therefore, they can be generated in a non-stationary or even adverse way. The agent may consider only the past rewards $r_1, \ldots, r_{t-1}$ and the past costs $c_1, \ldots, c_{t-1}$ when deciding what action θ_t to take.

To clarify our online learning problem and its challenges, we first define an *offline* version of the problem. This offline version simply assumes that our agents knows all reward and cost terms in advance. In such a setting, the optimal strategy of the agent is a solution to the optimization problem:

$$\begin{aligned} \text{maximize}_{\boldsymbol{\theta}} \quad & \frac{1}{T} \sum_{t=1}^{T} r_t(\theta_t) \\ \text{subject to:} \quad & g_t(\boldsymbol{\theta}) \leq c_0 \quad \forall\, t \in \mathcal{G}; \end{aligned} \tag{1}$$

where the sequence of actions $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_T)$ is optimized to maximize the average reward over T time steps subject to a set of constraints $\mathcal{G}$. We assume that the constraint function $g_t(\boldsymbol{\theta})$ is a temporal function of instantaneous costs:

$$g_t : c_{t-\tau+1}(\theta_{t-\tau+1}), \ldots, c_t(\theta_t) \to \mathbb{R} \tag{2}$$

defined on a time interval that spans τ time steps and ends at the time t. In practice, the function is typically convex:

$$g_t(\boldsymbol{\theta}) = \frac{1}{\tau} \sum_{\ell=t-\tau+1}^{t} c_\ell(\theta_\ell). \tag{3}$$

Note that our optimization problem has a very general form, and g_t, τ, c_0, and $\mathcal{G}$ are its parameters. How to choose these parameters to represent a problem is a design issue.

Instances of our problem are common in the field of engineering. For example, most of power management problems are formulated as maximizing power savings subject to some performance criteria. The criteria are usually represented by side constraints $g_t(\boldsymbol{\theta}) \leq c_0$, which are defined at regular time steps $k\tau$, where $k \geq 1$ is an integer. The function $g_t(\boldsymbol{\theta})$ is often convex and aggregates recent costs of the system, such as the frequency of taking bad power management actions. The scalar c_0 determines the acceptable level of bad PM actions.

The offline version of our online optimization problem (1) can be solved by standard techniques for nonlinear programming (Bertsekas 1999). In this work, we attempt to solve the problem online. The challenges of the online setting are that the horizon T is unknown, and no statistical assumptions on the rewards r_t and costs c_t are made in advance. As a result, it may be too ambitious to expect to learn as good policies as in the offline setting. Therefore, the goal of our work is more modest. We want to learn a policy that yields as high rewards as the best solution to our problem chosen offline from some set of experts. This setup is known as prediction with expert advice (Littlestone & Warmuth 1994). At the same time, our solution should satisfy all but a small number of constraints.

Existing work

This section positions our work with respect to the literature on online convex programming (Zinkevich 2003) and online learning with constraints (Mannor & Tsitsiklis 2006).

Online convex programming (Zinkevich 2003) involves a convex feasible set $\mathcal{F} \subset \mathbb{R}^n$ and a sequence of convex functions $f_t : \mathcal{F} \to \mathbb{R}$. At every time t, a decision maker chooses some action $\theta_t \in \mathcal{F}$ based on the past functions $f_1, \ldots, f_{t-1}$ and actions $\theta_1, \ldots, \theta_{t-1}$. The goal is to minimize the regret:

$$\sum_{t=1}^{T} f_t(\theta_t) - \min_{\theta \in \mathcal{F}} \sum_{t=1}^{T} f_t(\theta). \tag{4}$$

In contrast to online convex programming, the feasible set $\mathcal{F}$ in our problem (1) may change arbitrarily over time because our constraint functions (Equation 2) depend on the costs c_t. A useful interpretation is that the feasible set $\mathcal{F}$ is controlled by an adverse opponent. This is the main source of difficulty addressed in this paper.

Mannor and Tsitsiklis (2006) investigated online learning in the context of the constrained optimization problem:

$$\begin{aligned} \text{maximize}_{\theta \in \Delta(\mathcal{A})} \quad & \frac{1}{T} \sum_{t=1}^{T} r_t(\theta) \\ \text{subject to:} \quad & \frac{1}{T} \sum_{t=1}^{T} c_t(\theta) \leq c_0; \end{aligned} \tag{5}$$

where $\Delta(\mathcal{A})$ denotes the simplex of probability distributions over the set of actions $\mathcal{A}$. Based on their work, the reward in hindsight, which corresponds to the solution of the optimization problem (5), is generally unattainable online. Results in our paper do not contradict to this claim. The main reason is that we study a different optimization problem. In particular, our goal is to satisfy a set of finite-horizon constraints rather than a single terminal constraint. The repetitive nature of the problem allows for solving it efficiently by online learning.

Online learning with constraints

In this section, we show how to learn an online policy $\boldsymbol{\theta}$ from a set of experts $\xi_1, \ldots, \xi_N$ that guarantees a sublinear *regret*:

$$\max_{n=1,\ldots,N} \sum_{t=1}^{T} r_t(\xi_n(t)) - \sum_{t=1}^{T} \mathrm{E}\left[r_t(\theta_t)\right] \tag{6}$$

with respect to the best expert, and a sublinear bound on the total number of *constraint violations*[1]:

$$\sum_{t \in \mathcal{G}} \mathbf{1}_{[g_t(\boldsymbol{\theta}) > c_0]}. \tag{7}$$

In other words, we want to achieve close-to-optimal rewards as $T \to \infty$ while violating a vanishing number of constraints. These two objectives are optimized independently instead of being combined. This allows us to provide separate guarantees on the regret and constraint violation of learned policies. Therefore, our approach is suitable for constrained optimization problems, where the tradeoff between the two objectives is hard to quantify, or it cannot be established at all.

The online policy $\boldsymbol{\theta}$ is learned by a modified version of the standard exponentially weighted forecaster (Cesa-Bianchi & Lugosi 2006). Our experts $\xi_1, \ldots, \xi_N$ can be viewed as suboptimal solutions to a constrained optimization problem (1), which satisfy most of its constraints. These experts are often available in practice. For instance, in the power management domain, the experts are PM heuristics that do not exceed the acceptable level of bad PM actions.

Non-overlapping constraints

First, let us consider an online learning setup where we have access to a pool of experts $\xi_1, \ldots, \xi_N$ that never violate constraints. We assume that these constraints do not *overlap*. In other words, $\mathrm{Dom}(g_t) \cap \mathrm{Dom}(g_{t'}) = \emptyset$ for every $t' \neq t$. An example of such a constraint space is given in Figure 1a. The constraints span τ time steps and are defined at the times $k\tau$, where $k \geq 1$ is an integer. In the rest of this section, we use the constraint space to illustrate our online learning solution. This can be done without loss of generality.

Recall that our objective is to learn an online policy from a set of experts $\xi_1, \ldots, \xi_N$ that yields a sublinear regret with respect to the best expert. Since our constraints do not overlap, and they are satisfied by the experts, a regret minimizing policy that satisfies all constraints can be easily computed as follows. First, we partition the time steps $1, \ldots, T$ into T/L *segments* $\mathcal{P}_1, \ldots, \mathcal{P}_{T/L}$ of the length L such that every constraint belongs to a single segment (Figure 1a). The length L is a multiple of the constraint span τ. Second, we modify the

[1] Our work can be extended to other constraint violation metrics, such as the magnitude of violated constraints $\sum_{t \in \mathcal{G}} [g_t(\boldsymbol{\theta}) - c_0]^+$.

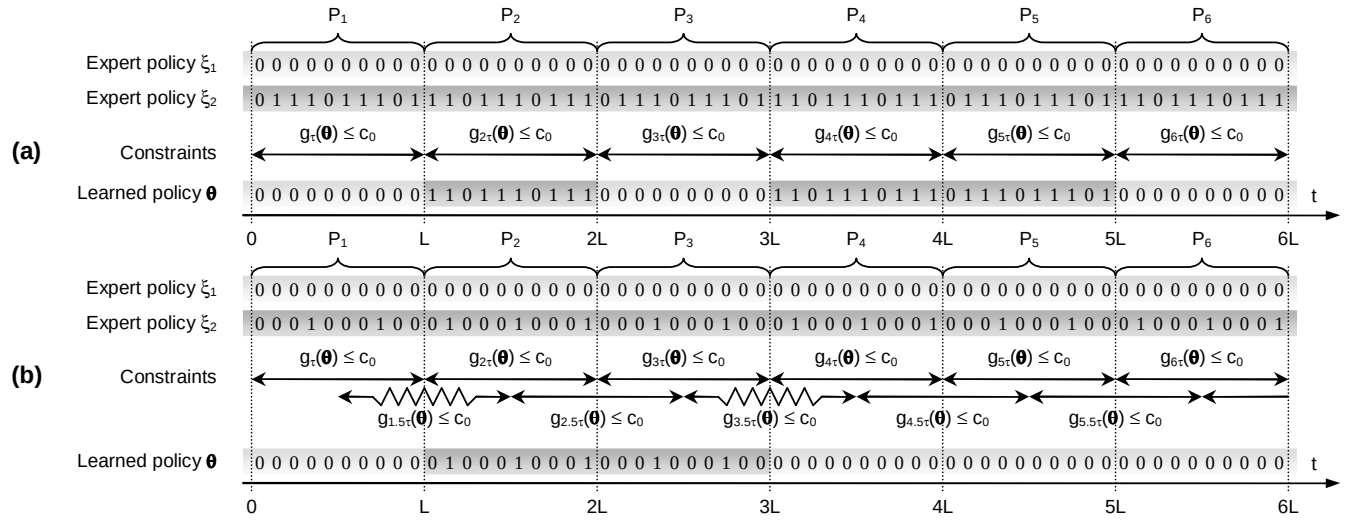

Figure 1: **a.** An online learning problem with non-overlapping constraints. The example involves expert policies ξ_1 and ξ_2, side constraints, and a policy $\boldsymbol{\theta}$ generated by the lazy learner (Figure 2). The policies are depicted as functions of time. The temporal span of the constraints is illustrated by arrows. Time steps at which the lazy learner may switch between the experts are denoted by dotted lines. **b.** An online learning problem with overlapping constraints. Constraints that may be violated due to switching between the experts are depicted as jagged arrows.

standard exponentially weighted forecaster (Cesa-Bianchi & Lugosi 2006) such that expert switches are permitted only at the beginning of each segment $\mathcal{P}$.

The modified forecaster is described in Figure 2. We refer to it as a *lazy learner* because it commits to one expert for L time steps. The *learning window* L is the main parameter of the forecaster. The following proposition relates this parameter to the quality of lazily learned policies.

Proposition 1. *Let $\xi_1, \ldots, \xi_N$ be expert policies that satisfy all constraints. Then the regret of the lazy learner (Figure 2) is bounded as:*

$$\max_{n=1,\ldots,N} \sum_{t=1}^{T} r_t(\xi_n(t)) - \sum_{t=1}^{T} \mathrm{E}\left[r_t(\theta_t)\right] \leq \frac{\log(N)}{\eta} + \frac{\eta T L}{2}.$$

The constraint violation of its policies is 0.

Proof: Our first claim is proved by interpreting lazy learning as an online learning problem with a convex reward function $\mathrm{E}\left[r_t(\theta_t)\right] = \sum_{e_t} P(e_t) r_t(\xi_{e_t}(t))$:

$$\begin{aligned}
&\max_{n=1,\ldots,N} \sum_{t=1}^{T} r_t(\xi_n(t)) - \sum_{t=1}^{T} \mathrm{E}\left[r_t(\theta_t)\right] \\
&\quad= \max_{n=1,\ldots,N} \sum_{m=0}^{T/L-1} \sum_{t=mL+1}^{(m+1)L} \left(r_t(\xi_n(t)) - \mathrm{E}\left[r_t(\theta_t)\right]\right) \\
&\quad\leq \frac{\log(N)}{\eta} + \frac{\eta}{2} \sum_{m=0}^{T/L-1} \\
&\qquad \max_{n=1,\ldots,N} \left(\sum_{t=mL+1}^{(m+1)L} \left(r_t(\xi_n(t)) - \mathrm{E}\left[r_t(\theta_t)\right]\right) \right)^2 \\
&\quad\leq \frac{\log(N)}{\eta} + \frac{\eta}{2} \frac{T}{L} L^2 \\
&\quad= \frac{\log(N)}{\eta} + \frac{\eta T L}{2}.
\end{aligned}$$

The first step of the proof follows by algebra, the second step is based on Theorem 2.1 (Cesa-Bianchi & Lugosi 2006), and the third step results from the reward terms r_t being bounded on the interval $[0, 1]$.

Our second claim follows from two assumptions. First, no constraint is violated by the experts. Second, no constraint is violated due to switching between the experts. ■

The parameters η and L can be set such that the regret bound in Proposition 1 is sublinear in T. For instance, for $\eta = T^{-\frac{3}{4}}$ and $L = T^{\frac{1}{2}}$, the bound is on the order of $O(T^{\frac{3}{4}})$. Therefore, the average regret of the lazy learner vanishes as T increases. Furthermore, note that when the learning window L is given, the regret bound is tightest for $\eta = \sqrt{2\log(N)/(TL)}$. From now on, we adopt this setting of the learning rate η.

In the rest of the paper, we study the lazy learner in a more general context. In particular, we relax the assumptions that constraints do not overlap and that the experts do not violate them. It turns out that the parameter L has an important role in this new setting. It trades off the regret of learned policies for their constraint violation.

Imperfect expert policies

Building of expert policies that satisfy all constraints may be too restrictive or hard in practice. In this section, we attempt to relax this assumption. We still assume that the constraints do not overlap.

One way of restricting the constraint violation of individual experts is with respect to each segment $\mathcal{P}$. If this quantity is bounded, we can bound the regret and constraint violation of lazily learned policies as follows.

Proposition 2. *Let $\xi_1, \ldots, \xi_N$ be expert policies whose con-*

straint violation is bounded as:

$$\max_{n=1,\ldots,N} \sum_{t \in \mathcal{P} \cap \mathcal{G}} \mathbf{1}_{[g_t(\xi_n) > c_0]} \leq \varepsilon$$

for some scalar ε and all segments $\mathcal{P}$. Then the regret of the lazy learner (Figure 2) is bounded as suggested by Proposition 1. The constraint violation of its policies is:

$$\sum_{t \in \mathcal{G}} \mathbf{1}_{[g_t(\boldsymbol{\theta}) > c_0]} \leq \frac{T}{L} \varepsilon.$$

Proof: The regret bound is proved as in Proposition 1. The constraint violation bound follows from the observation that the number of segments $\mathcal{P}$ is T/L. ■

The parameters η and L can be set such that both our regret and constraint violation bounds (Proposition 2) are sublinear in T. For instance, for $\eta = T^{-\frac{3}{4}}$ and $L = T^{\frac{1}{2}}$, the bounds are on the order of $O(T^{\frac{3}{4}})$ and $O(T^{\frac{1}{2}})$. In turn, both the average regret and constraint violation of the lazy learner vanish as T increases.

Finally, note that our bounds suggest that the parameter L can trade off the regret of learned policies for their constraint violation. For instance, as the learning window L decreases, the regret bound becomes tighter. At the same time, the constraint violation bound becomes looser. This phenomenon is studied empirically in the experimental section.

Overlapping constraints

The assumption of non-overlapping constraints helped us to learn online policies up to this point. In this section, we relax this assumption.

A simple example of a constraint space where constraints overlap is depicted in Figure 1b. The constraints span τ time steps and are defined at the times $k(\tau/2)$, where $k \geq 1$ is an integer. When the lazy learner is applied to these constraints, it may violate a constraint whenever it switches between the experts. The reason is that constraint functions $g_t(\boldsymbol{\theta})$, which are affected by policy transitions, involve actions of multiple experts. Therefore, even if the constraints $g_{1.5\tau}(\xi_n) \leq c_0$ and $g_{3.5\tau}(\xi_n) \leq c_0$ are satisfied by both expert policies ξ_1 and ξ_2, they may be violated by the learned policy $\boldsymbol{\theta}$.

Fortunately, the maximum number of constraints overlapping between any two consecutive segments $\mathcal{P}_\ell$ and $\mathcal{P}_{\ell+1}$ is often bounded in practice. The reason is that side constraints are usually enforced with some periodicity, such as minutes, hours, or days. Based on this observation, we can generalize Proposition 2 as follows.

Proposition 3. *Let $\xi_1, \ldots, \xi_N$ be expert policies whose constraint violation is bounded as in Proposition 2. In addition, let s be the maximum number of constraints that overlap between any two segments $\mathcal{P}_\ell$ and $\mathcal{P}_{\ell+1}$. Then the regret of the lazy learner (Figure 2) is bounded as suggested by Proposition 1. The constraint violation of its policies is:*

$$\sum_{t \in \mathcal{G}} \mathbf{1}_{[g_t(\boldsymbol{\theta}) > c_0]} \leq \frac{T}{L}(\varepsilon + s).$$

Proof: The regret bound is proved as in Proposition 1. The first term in the constraint violation bound is borrowed from Proposition 2. The second term accounts for constraint violations due to switching between the experts. ■

Inputs:
a learning window L
a learning rate η
expert policies $\xi_1, \ldots, \xi_N$
expert weights $w_{t-1}(1), \ldots, w_{t-1}(N)$
an expert e_{t-1} followed at the time step $t-1$
a reward function r_t

Algorithm:
if $(t \pmod L) \equiv 1)$
  randomly choose an expert e_t according to the distribution:
    $P(e_t) = \frac{w_{t-1}(e_t)}{\sum_{n=1}^{N} w_{t-1}(n)}$
else
  $e_t = e_{t-1}$

play an action $\theta_t = \xi_{e_t}(t)$

for every $n = 1, \ldots, N$
  $w_t(n) = w_{t-1}(n) \exp[\eta r_t(\xi_n(t))]$

Outputs:
an action θ_t played at the time step t
updated expert weights $w_t(1), \ldots, w_t(N)$
an expert e_t followed at the time step t

Figure 2: An exponentially weighted forecaster that permits one expert switch per L time steps.

Similarly to Proposition 2, the parameters η and L can be set such that both bounds in Proposition 3 are sublinear in T. In addition, the parameter L allows for trading off the regret of learned policies for their constraint violation.

Convex constraint functions

From the theoretical point of view, lazy learning can provide sublinear guarantees on the regret and constraint violation of learned policies without making any assumption on the form of side constraints. However, in this general setting, building of expert policies that violate only a bounded number of constraints is hard in practice. This type of experts is required as an input to the lazy learner (Figure 2). In the rest of this section, we show how to build these experts efficiently when the constraint function $g_t(\boldsymbol{\theta})$ is convex (Equation 3).

Our solution assumes that we have a special action θ^0 that guarantees $c_t(\theta^0) \ll c_0$ for all time steps t. The key is to play the action θ^0 when the original expert may violate more than an acceptable fraction of constraints. We refer to this process as *arbitration*.

Arbitration can be carried out in most real-world domains. For instance, it corresponds to taking no power management actions in the power management domain. It is conceptually equivalent to having a non-empty feasible set in a traditional optimization setting. Unfortunately, the arbitrating action θ^0 often yields zero rewards. Therefore, it should be taken only when necessary because it conflicts with our main objective of maximizing rewards.

In the rest of the paper, we evaluate the performance of the lazy learner on a power management problem. The nature of the problem is not adversarial as typically assumed in online learning. Therefore, although our approach learns almost as

good policies as the best experts, our bounds are a little loose to justify its performance.

Package power management

Our online solution is evaluated on a challenging real-world problem. We look at the power management of the complete processing unit including multi-core processors, L1 and L2 caches, and associated circuitry. Solving this PM problem is important because the complete processing unit may account for as much as 40 percent of the power consumed by mobile computers. In the rest of this paper, we use the term *package* to refer to the complete processing unit.

The primary goal of package PM is to minimize the power consumption of the package without impacting its perceived performance. This performance objective can be restated as maximizing the *residency* of the package in low power states while minimizing the *latency* in serving hardware interrupts. The latency is a delay caused by waking up the package from low power states. The central component of package PM is a prediction module, which predicts idle CPU periods that are sufficiently long to power down the package. This prediction is made at every OS interrupt. Under normal circumstances, Microsoft Windows generates OS interrupts every 15.6 ms.

A state-of-the-art solution to package PM are static timeout policies. A *static timeout policy* (Karlin *et al.* 1994) is a simple power management strategy, which is parameterized by the timeout parameter T. When the package remains idle for more than T ms, the policy puts it into a low power state. When an unpredicted hardware interrupt occurs, the package must wake up to serve it. Due to the delay in performing this task, the package incurs a 1 ms latency penalty. The package wakes up ahead of the OS interrupts because these interrupts are predictable. This setting is suggested by domain experts.

In the experimental section, we consider a pool of experts, which are adaptive timeout policies. The policies adapt their timeout parameters at every OS interrupt with respect to the current workload (Kveton *et al.* 2007).

Experiments

The main goal of the experimental section is to demonstrate online learning with constraints in practice. Our experiments are performed on the package PM problem. We simulate the package in MATLAB on two CPU activity traces.

Experimental setup

The first trace is recorded during running MobileMark 2005 (MM05). MM05 is a performance benchmark that simulates the activity of an average Microsoft Windows user. A corresponding CPU activity trace is 90 minutes long and contains more than 500,000 OS interrupts. The second trace is generated by running Adobe Photoshop, Microsoft Windows Explorer, Microsoft WordPad, and Microsoft Media Player. It reflects 30 minutes of human activity and contains more than 200,000 OS interrupts. In the rest of the section, we refer to it as a heavy workload trace.

Our goal is to maximize the residency of the package subject to latency constraints. This is a constrained optimization problem, where the residency and latency of the package between two consecutive OS interrupts represent instantaneous rewards $r_t(\theta_t)$ and costs $c_t(\theta_t)$, respectively. The variable θ_t denotes the timeout parameter of the package PM module at the time step t. Our latency constraints are averages over 10 second periods, which corresponds to $\tau = 640$. The purpose of the constraints is to restrict the rate of bad PM actions over longer periods of time. Such actions may significantly affect the perceived performance of the computer.

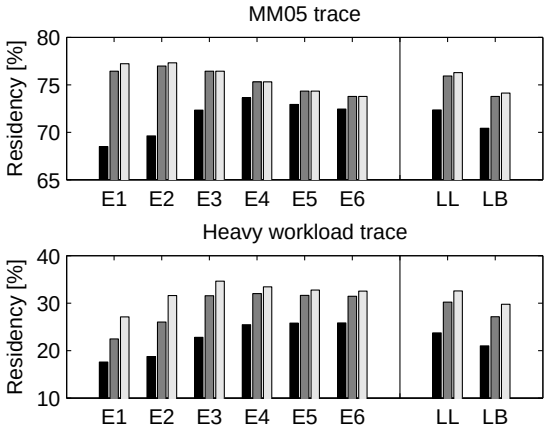

Figure 3: Comparison of lazily learned policies (LL) to their pools of experts (E1, . . . , E6). The policies are compared by their residency for different latency budgets c_0: 0.02 (black bars), 0.04 (dark gray bars), and 0.06 (light gray bars). We also show lower bounds on the residency of the learned policies (LB) as suggested by Proposition 1.

All online solutions to our constrained optimization problem are computed by the lazy learner (Figure 2). The experts $\xi_1, \ldots, \xi_N$ are timeout policies (Kveton *et al.* 2007), whose timeout parameters are adapted by the fixed-share algorithm (Herbster & Warmuth 1995). These policies are additionally arbitrated to satisfy our latency constraints.

Unless specified otherwise, the following parametrization is employed in our experiments. Side constraints are defined at 10 second intervals such that $\mathcal{G} = \{640, 1280, 1920, \ldots\}$, the latency budget c_0 is set to 0.03, and the learning window L is equal to τ. All reported empirical results correspond to averages of 10 stochastic simulations.

Experimental results

The first experiment (Figure 3) demonstrates that lazy learning yields high-quality package PM policies (Proposition 1). We assume non-overlapping constraints (Figure 1a) that are completely satisfied by our experts. Based on our results, the regret of learned policies is always less than 2 percent. This evidence suggests that the policies are almost as good as the best expert in hindsight. The learned policies are also significantly better than worst experts, which may yield as much as 7 percent less residency. On the heavy workload trace, this 7 percent improvement can be viewed as more than 30 percent when measured in relative terms. At the same time, our policies satisfy all latency constraints and their regret is bounded as suggested by Proposition 1.

Finally, the learned policies also outperform static timeout policies, which are state-of-the-art solutions to package PM. On the heavy workload trace, for instance, none of the timeout polices that satisfy latency constraints $g_t(\boldsymbol{\theta}) \leq 0.06$ yield more than 5 percent residency. This is more than 6 times less

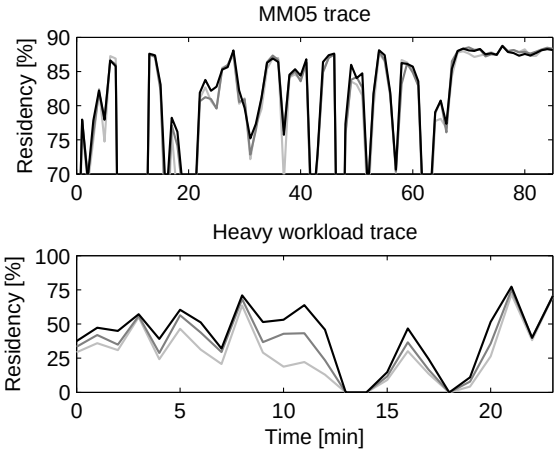

Figure 4: Average residency of three lazily learned policies. The policies are learned from three pools of experts that satisfy 100 (light gray lines), 50 (dark gray lines), and 0 (black lines) percent of latency constraints in every segment $\mathcal{P}$. The residency is depicted as a function of time (in minutes).

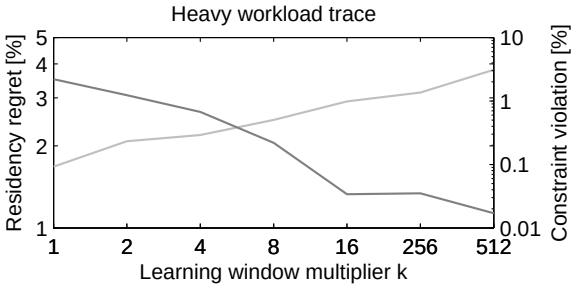

Figure 5: Constraint violation (dark gray line) and residency regret (light gray line) of lazily learned policies as a function of the learning window L. The learning window is computed as $L = k\tau$, where k is an integer multiplier.

than the residency of a corresponding lazily learned policy.

The second experiment (Figure 4) illustrates how the residency of lazily learned policies is impacted by imperfect experts. Similarly to the previous experiment, we assume non-overlapping constraints (Figure 1a). We vary the proportion of the constraints that is satisfied by the experts and study its effect. The learning window L is equal to 10τ. Based on our results, relaxation of the experts yields policies with a higher residency. This is expected since the relaxation increases the residency of individual experts.

The third experiment (Figure 5) shows how the regret and constraint violation of lazily learned policies are affected by the learning window L (Proposition 3). We assume overlapping constraints (Figure 1b) defined at one second intervals. Based on our results, the parameter L trades off the regret of learned policies for their constraint violation. Smaller values of the parameter generally cause a smaller regret and higher constraint violation. Higher values of the parameter have an opposite effect. This behavior is suggested by Proposition 3.

Conclusions

Although online learning has been studied extensively by the machine learning community, solving constrained optimization problems online remains a challenging problem. In this paper, we proposed a practical online solution to constrained optimization problems with temporal constraints. Moreover, we provided guarantees on the quality of this solution in the form of regret and constraint violation bounds. The solution was evaluated on a challenging real-world PM problem. Our experiments support the hypothesis that online learning with side constraints can be carried out successfully in practice.

Results of this paper can be extended in several directions. First, since the lazy learner (Figure 2) is derived based on the standard exponentially weighted forecaster (Cesa-Bianchi & Lugosi 2006), we believe that it can be easily generalized to minimize the regret with respect to tracking the best expert. Second, arbitration is by no means the most efficient way of guaranteeing that online learned policies violate only a small number of constraints. Since arbitration typically yields zero rewards, it is crucial to develop online learning solutions that do not require it. Whether this goal can be achieved by modifying standard techniques for nonlinear programming, such as penalty methods, is an interesting open question.

Acknowledgment

We thank anonymous reviewers for comments that led to the improvement of this paper.

References

Bertsekas, D. 1999. *Nonlinear Programming*. Belmont, MA: Athena Scientific.

Cesa-Bianchi, N., and Lugosi, G. 2006. *Prediction, Learning, and Games*. New York, NY: Cambridge University Press.

Dhiman, G., and Simunic, T. 2006. Dynamic power management using machine learning. In *Proceedings of the 2006 IEEE / ACM International Conference on Computer-Aided Design*.

Gramacy, R.; Warmuth, M.; Brandt, S.; and Ari, I. 2003. Adaptive caching by refetching. In *Advances in Neural Information Processing Systems 15*, 1465–1472.

Helmbold, D.; Long, D.; Sconyers, T.; and Sherrod, B. 2000. Adaptive disk spin-down for mobile computers. *Mobile Networks and Applications* 5(4):285–297.

Herbster, M., and Warmuth, M. 1995. Tracking the best expert. In *Proceedings of the 12th International Conference on Machine Learning*, 286–294.

Karlin, A.; Manasse, M.; McGeoch, L.; and Owicki, S. 1994. Competitive randomized algorithms for nonuniform problems. *Algorithmica* 11(6):542–571.

Kveton, B.; Gandhi, P.; Theocharous, G.; Mannor, S.; Rosario, B.; and Shah, N. 2007. Adaptive timeout policies for fast fine-grained power management. In *Proceedings of the 22nd National Conference on Artificial Intelligence*, 1795–1800.

Littlestone, N., and Warmuth, M. 1994. The weighted majority algorithm. *Information and Computation* 108(2):212–261.

Mannor, S., and Tsitsiklis, J. 2006. Online learning with constraints. In *Proceedings of 19th Annual Conference on Learning Theory*, 529–543.

Zinkevich, M. 2003. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, 928–936.