Anytime Backtrack Unimodal Bandits and Applications to Cloud Computing

Stephan Kunne
Nokia Bell Labs Paris-Saclay
stephan.kunne@gmail.com

Lorenzo Maggi
Nokia Bell Labs Paris-Saclay
lorenzo.maggi@nokia-bell-labs.com

Johanne Cohen
CNRS, Université Paris-Saclay
jcohen@lri.fr

Xinneng Xu
Université Paris Sud
xinneng.xu@u-psud.fr

***Abstract*—Unimodal functions appear ubiquitously in resource allocation problems. They arise whenever the expected value $f(x)$ of the metric to be optimized has a single peak as a function of the available control variable x. The exact shape of f is often not known in advance, hence it can only be learned via sampling. For this purpose, we exploit the formalism of multi-armed bandits to design a learning algorithm that *i)* converges in probability to the optimal arm maximizing f and *ii)* does not require to know the learning horizon in advance, i.e., it is *anytime*. Moreover, it adapts naturally to the scenario where *iii)* prior knowledge on the location of the peak is available, even if the prior is inaccurate. We also present an efficient heuristic that can use explicitly the prior belief on the location of the optimal arm. We demonstrate the applicability of our approaches to a basic resource allocation problem for cloud computing, where the orchestrator adapts the resources allocated to a client to its dynamic requests.**

***Index Terms*—multi-armed bandits, unimodal, online learning, cloud computing, resource allocation**

I. Introduction

Unimodal functions $f(x)$ are univariate functions with a single peak, i.e., increasing for $x \leq x^*$ and decreasing for $x > x^*$, for some (optimal) value x^*. They generalize the concept of concave functions with a single variable, and appear ubiquitously in the realm of communication network control. In fact, unimodal functions often arise when an optimal trade-off must be stricken between two opposite forces. In such cases, the unimodal function $f(x)$ describes the *expected* value of the metric of interest to be maximized, and x denotes the control variable. However, in a number of cases the function f is unknown, and only the *realizations* of the metric of interest can be observed.

In this general scenario we propose LSE-backtrack, the first learning algorithm that converges in probability to the optimal (in the multi-armed bandit jargon) *arm* x^* and is oblivious to the time horizon (i.e., it is *anytime*). It zooms in and out an interval $[x^L, x^H]$ containing x^* with high probability, and its convergence properties are insensitive to the initialization of $[x^L, x^H]$ itself. Thus, it adapts well to scenarios where a prior belief on the location of x^* is available.

One important application scenario for our approach, working on general stochastic environments and relying on few basic assumptions, is cloud computing. There, the main challenge faced by the cloud orchestrator is to allocate the "right" amount of resources (memory and/or CPU, typically) to each user, without knowing in advance its dynamic needs. To be concrete, one can think of a container orchestration module like Kubernetes [1], promising to scale a cloud infrastructure based on users' demand, by dynamically adding or removing pods on the computing nodes via the so-called *autoscaling* procedure [2]. The orchestrator has two conflicting objectives: avoiding to waste resources and satisfying the clients. More specifically, the trade-off faced by the orchestrator is between the resource running cost, proportional to the amount of allocated resources x, and the expected Quality of Service $\mathbb{E}[\mathrm{QoS}(x)]$ perceived by the user. Typically, $\mathbb{E}[\mathrm{QoS}(x)]$ is a concave function of x. For instance, consider that a small Response Time (RT) is an indicator for good QoS: since queueing theory teaches us that $\mathbb{E}[\mathrm{RT}(x)]$ is convex decreasing in the number of computing resources x (e.g., see Kingman's formula [3]), then we come up with the unimodal objective function $f(x) = -x - \gamma\mathbb{E}[\mathrm{RT}(x)]$, with $\gamma > 0$. As another example, let us consider that the user has a QoS target that is met only if the allocated resources x is not inferior to the user's resource request, as in Section VI. There too, under weak assumptions, the resulting objective function $f(x)$ is unimodal.

However, the objective function f is generally unknown, since the orchestrator can only observe the *realizations* of the instantaneous QoS perceived by the users, but not its *expected* value. For this reason, the orchestrator needs to learn the optimal allocation policy via experience, and quickly converge to the optimal trade-off via a learning algorithm. Most importantly, the time duration of the user connection is not known in advance, from which the need of an *anytime* algorithm stems in this scenario.

Main contributions. We design LSE-backtrack, a bandit algorithm on a continuous arm space that exploits the knowledge of the unimodal structure of the average arm reward to converge to the optimal arm x^*. We prove in Theorem 2 that LSE-backtrack *i)* converges to the optimal arm x^* in probability. Moreover, *ii)* it does not need to be parametrized with respect to the time horizon of interest, thus LSE-backtrack is *anytime*, i.e., it is able to provide performance guarantees at any time step. This is crucial in realistic applications where the algorithm runs indefinitely and still needs to perform provably well at all times. In order to achieve this, LSE-backtrack exploits a dichotomy algorithm that shrinks and expands the interval of search to make up for wrong decisions. Such feature also lets our algorithm *iii)* adapt naturally to scenarios where a prior knowledge on the location of the optimal arm x^* is available. Indeed, one can restrict the initial search

The fourth author received financial support from CDS Center for Data Science (Idex Paris Saclay) for this work.
ISBN 978-3-903176-28-7

interval and still be able to converge to x^* in probability even when the initialization is wrong (see Corollary 2.1). LSE-backtrack extends an elegant existing algorithm called LSE [4] which does not enjoy the three crucial properties above though. Finally, *iv)* LSE-backtrack is characterized by a low computational complexity, as it only requires to compute the average reward at six arms and make simple comparisons at each iteration. We also present the heuristic LSE-weight that naturally exploits the prior distribution on x^*. LSE-weight has weaker theoretical guarantees than LSE-backtrack (see Lemma 1) but it performs well in practice (see Section VI).

Paper organization. We describe our model and the related works in Sections II and III, respectively. We present our LSE-backtrack algorithm and its convergence properties in Section IV. We introduce LSE-weight, a variant that exploits historic data, in Section V. Finally, we show how to apply our algorithms to a cloud computing scenario in Section VI, along with extensive numerical simulations.

II. Our model

We consider a stochastic multi-armed bandit problem where the expected reward is unimodal over partially ordered arms. More specifically, we consider a continuous arm space over the interval of reals $\mathcal{X} = [a, b]$. The environment sets a reward function $\mathcal{X} \to \mathcal{M}(\mathbb{R})$, where $\mathcal{M}(\mathbb{R})$ is the space of probability distributions over the reals. We denote the reward distribution assigned to arm $x \in \mathcal{X}$ as M_x, and we call $f(x)$ the expected reward associated to the arm x.

A decision maker interacts with the environment. At each step $t = 1, 2, \ldots$, she selects an arm $x_t \in \mathcal{X}$ and observes a reward $Y(x_t, t)$ randomly drawn at from the distribution M_{x_t}. The random variables $(Y(x, t))_{x\in[a,b],t>0}$ are independent, and for a given arm x, $(Y(x, t))_{t>0}$ are independent and identically distributed. Our main assumption is that the decision maker knows in advance, before starting interacting with the environment, that the function f is *bounded* and *unimodal*.

Assumption 1 (unimodality)**.** *The function $f : [a, b] \to \mathbb{R}$ is bounded and unimodal, i.e., there exists $x^* \in [a, b]$ such that f is increasing on $[a, x^*]$ and decreasing on $[x^*, b]$.*

We further assume that the bounds of f are known. Hence, without loss of generality, we can suppose $f : [0, 1] \to [0, 1]$. Yet, the maximum and minimum of f are still unknown.

In order to be able to distinguish the average reward of different arms with a finite number of samples and converge to the optimal arm x^*, we need to assume that the function f has a minimum increase rate at both sides of the optimum x^*.

Assumption 2. *There exists $C_L > 0$ such that $|f(x)-f(y)| \geq C_L|x - y|$ for all pairs $x, y \in [0, x^*]$ or $x, y \in [x^*, 1]$.*

Observe that Assumption 2 is different from Assumption 3.2 in [4] since we focus on the convergence rather than on the regret. We now state the main problem that we tackle.

Problem. *Let Assumptions 1, 2 hold. Then, find an algorithm $\mathcal{A}$ that converges to the optimal arm x^* in probability, i.e.,*

$$\lim_{t\to\infty} \Pr\left(|x_t - x^*| < h\right) = 1, \qquad \forall\, h > 0,$$

where x_t is the arm sampled at time t by algorithm $\mathcal{A}$.

III. Related works

This type of repeated decision making problems has been well studied in the literature under the name multi-armed bandits [5], [6]. In the general case of a continuous reward function whose derivative is not accessible but only a noisy version of the function value is, locating the maximum can be achieved through derivative-free approaches as [7]. Also, one can evaluate the gradient by sampling the function at a single (random) point; this is "gradient descent without a gradient" [8]. The *Stochastic Gradient Descent* algorithm (SGD, [9]) is less appropriate, since a number of samples are needed just to estimate the gradient, which affects the convergence speed.

If the reward function is known to be unimodal, then more specialized techniques can be used. *Golden Section Search (GSS, [10])* is a dichotomic algorithm solving the problem in the absence of noise, if the reward function is deterministic. The stochastic setting is trickier, and requires numerous samplings of the reward function during the dichotomy. *Line Search Elimination* (LSE, [4]) is an adaptation of GSS using PAC bounds [11] to determine an appropriate number of samples. LSE is particularly efficient if the reward function is convex or close to convex, and fails at the other extreme, if the reward function presents a Dirac-like shape, with its optimum on top of a sharp peak, surrounded by a flat plateau. *Stochastic Pentachotomy* [12] is another dichotomy scheme, with performance guarantees on all unimodal functions, but with a much higher computational complexity. In fact, it requires solving at each iteration an optimization problem over a class of unimodal functions (see IT_K routine in [12]). Bayesian optimization [13] can be also used in this context.

Up the authors' knowledge, all the algorithms for unimodal bandits necessitate to know the time horizon T in advance to properly tune some key parameters and ensure certain theoretical guarantees on the regret and/or on the convergence. For instance, the parameter δ in algorithm LSE [4] should be proportional to $1/T$ in order to guarantee convergence with high probability; also, the subroutine that decides the number of samplings during one iteration of Stochastic Pentachotomy [12] takes the remaining number of plays as input. Our LSE-backtrack is the first algorithm for unimodal bandits that does not make assumptions on the horizon T and could be executed forever, i.e., it is *anytime* [14]. We remark that the *doubling trick* is a well-known technique allowing to transform a non-anytime bandit algorithm into an anytime one with respect to the performance guarantees on the *regret* in the long run. It works by phases: each phase corresponds to a standard run over a fixed time horizon T. At the end of each phase, it reinitializes the algorithm from scratch and doubles the time horizon [15]. However, the doubling trick does not allow us to achieve convergence in probability, for the reason itself that the algorithm must be restarted periodically from scratch. Also, in practice the doubling trick is not viable because it causes severe performance degradation on a periodic basis.

Finally, if the arms form a discrete space, then Optimal Sampling for Unimodal Bandits (OSUB, [16]) is yet another approach, based on KL-UCB [17], [18]. However, this is not our scenario since we consider a continuous set of arms.

In wireless networks, unimodal bandits have been exploited in [19] for rate adaptation, allowing to maximize the achieved throughput by appropriately tuning the coding rate. The expected throughput can be described as a unimodal function of the coding rate, as it equals the product between the coding rate itself and the probability of correct reception. The approach in [19] is conceived for scenarios where the observations are binary, as at a given time the sender can only know whether the transmission was successful for the employed coding rate.

IV. LSE-Backtrack

In this section we describe our main algorithm for unimodal continuous bandits, called LSE-backtrack, and we prove its convergence properties. Before that, for the reader's convenience, we present the state-of-the-art algorithm that we improve upon, called Line Search Elimination (LSE) [4].

A. Line Search Elimination (LSE)

The Line Search Elimination (LSE) algorithm [4] adapts Golden Section Search (GSS, [10]) to stochastic bandits using the PAC-bounds that can be found in [11]. GSS is a dichotomy algorithm that looks for the extremum of a *deterministic* unimodal function. Instead of evaluating the function once at each point (or *arm*) of interest as GSS does, LSE samples the arms a sufficiently large number of times in order to obtain a good estimate of the average reward of each selected arm.

At every iteration of the algorithm's main loop, LSE focuses on the interval $[x^L, x^H]$ in which the optimal arm x^* lies with high probability. LSE estimates the reward of four arms $x^L < x^A < x^B < x^H$ by sampling. Then, it shrinks the interval by a constant factor $1/\varphi$ (where $\varphi = (1+\sqrt{5})/2$ is the golden ratio) in the direction where the optimal arm x^* is supposed to lie, knowing that the underlying average reward function f is unimodal. Thanks to the special property of the golden ratio φ, shrinking the interval by a factor $1/\varphi$ allows LSE to land on three arms already considered at the previous iteration; hence, only one new arm is added at each iteration. We report the pseudo-code of LSE in Algorithm 1.

Algorithm 1 LSE [4]

Require: Unimodal bandit function $f : [0,1] \to [0,1]$;
$N(n)$: number of samples per arm at iteration n ($n \leftarrow 1$);
Initialize the search interval $[x^L, x^H] \leftarrow [0,1]$
loop
  $x^A \leftarrow (\varphi x^L + x^H)/(1+\varphi)$;
  $x^B \leftarrow (x^L + \varphi x^H)/(1+\varphi)$;
  Sample f at arms x^L, x^A, x^B, x^H a number $N(n)$ of times each and observe the resulting average rewards;
  Let x be the arm with the highest average reward among $\{x^L, x^A, x^B, x^H\}$
  if $x \in \{x^L, x^A\}$ **then**
    $[x^L, x^H] \leftarrow [x^L, x^B]$ {*Shrink to the left*}
  else if $x \in \{x^B, x^H\}$ **then**
    $[x^L, x^H] \leftarrow [x^A, x^H]$ {*Shrink to the right*}
  end if
  Increment the iteration count $n \leftarrow n+1$
end loop

As a terminology remark, the "time step" refers to exactly one arm sample. We define "iteration" as one round through the algorithm's main loop. Thus, one iteration for LSE consists of $4N$ time steps, where N is the number of samples per arm. According to [4] and to [11, Theorem 1], if N is chosen as follows:

$$N = \frac{1}{\varepsilon^2}\log(8/\delta) \tag{1}$$

for some ε and δ, then at each iteration, an ε-optimal arm is correctly identified with probability at least $1-\delta$ among the four sampled arms. Therefore, it is critical for LSE that δ is chosen small, with knowledge of the horizon T; more specifically, in [4] δ is chosen proportional to $1/T$. Note that a small value of δ directly impacts the number N of samples per iteration, which is proportional to $\log(1/\delta)$, according to (1). However, even if we choose a "small" value of δ, there is always a positive probability ($< \delta$) at each iteration of shrinking the interval in the wrong direction. This would cause LSE to fail to frame the optimal value subsequently, since it cannot catch up with previous mistakes. On the other hand, ε should be chosen small enough to be able to distinguish the rewards at different arms. We now provide the main PAC result that holds for LSE and that we aim at improving upon.

Theorem 1. *Run LSE algorithm with $\varepsilon(n) = C_L/\varphi^{n+3}$, $N(n) = 1/\varepsilon(n)^2 \log(8/\delta)$ and $\delta \in (0,1)$. After n iterations, LSE successfully frames optimal arm x^* in an interval of length at most φ^{-n} with probability at least $(1-\delta)^n$.*

The proof of Theorem 1 directly stems from the proof of Theorem 4.4 in [20]. Note that, if T is unknown and δ is fixed, then the probability that LSE frames the optimal arm tends to 0 as the number of iterations $n \to \infty$.

B. LSE-backtrack: Algorithm description

Let us first review the drawbacks of the existing algorithms for unimodal bandit optimization that we wish to overcome, by improving upon the elegant solution offered by LSE [4]. The works in [12], [4] require to know the time horizon in advance to ensure theoretical guarantees, i.e., they are not *anytime*. In particular, LSE [4] has positive probability of deviating from the optimal arm indefinitely when the horizon is unknown. Stochastic Pentachotomy [12] has high computational complexity, as it requires to solve an optimization problem over the class of unimodal functions at each iteration. Moreover, [12] and [4] do not adapt to the situation where a prior belief over the location of the optimal arm is available. Finally, the algorithms in [8], [21] do fully not exploit the unimodal structure of the reward function as they are designed for more general scenarios.

Our LSE-backtrack algorithm is inspired to the dichotomy procedure proposed by LSE, which prescribes to appoint an interval $[x^L, x^H]$ where the optimal arm x^* is supposed to lie, and to iteratively modify $[x^L, x^H]$ according to the samples collected on a small set of arms. However, at each iteration LSE can only shrink the designated interval $[x^L, x^H]$, while LSE-backtrack can stretch it when it is not believed to contain x^*. More specifically, at each iteration LSE-backtrack samples two additional arms called x^{LL} and x^{HH} outside the designated interval $[x^L, x^H]$. If the average sampled reward is higher at either of the exterior arms x^{LL} and x^{HH}, then the optimal reward cannot be located inside of the search

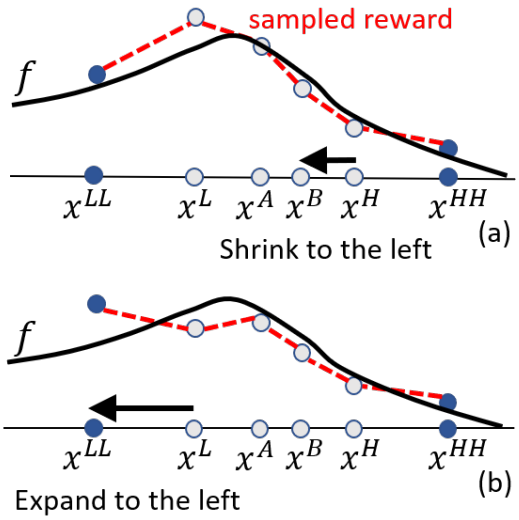


Fig. 1: **LSE-backtrack.** At each iteration, LSE-backtrack samples six arms $x^{LL}, x^{L}, x^{A}, x^{B}, x^{H}, x^{HH}$. According to whether the highest sampled reward is *(a)* at the interior or *(b)* at the border of the sampled points, LSE-backtrack *(a)* shrinks or *(b)* expands the search interval $[x^L, x^H]$ that is supposed to contain the optimum x^*.

interval $[x^L, x^H]$; this hints that $[x^L, x^H]$ has been shrank in the wrong direction at a previous step (see Figure 1(b)). Then, our algorithm "backtracks": it stretches the interval $[x^L, x^H]$ to the right or to the left by the golden ratio factor φ, whether the sampled reward is the highest at x^{HH} or x^{LL}, respectively. This allows LSE-backtrack to cater for all the issues of state-of-the-art approaches listed above, as we show next.
We report the pseudo-code of LSE-backtrack in Algorithm 2.

Algorithm 2 LSE-backtrack

Require: Unimodal bandit function $f : [0,1] \to [0,1]$;
$N(n)$: number of samples per arm for iteration n ($n \leftarrow 1$);
Choose initial search interval $[x^L, x^H]$ ($\leftarrow [0,1]$ by default)
loop
  $x^A \leftarrow (\varphi x^L + x^H)/(1+\varphi)$;
  $x^B \leftarrow (x^L + \varphi x^H)/(1+\varphi)$;
  Let $x^{LL} = \max(\bar{x}^{LL}, 0)$ such that $x^L = (\varphi \bar{x}^{LL} + x^H)/(1+\varphi)$;
  Let $x^{HH} = \min(\bar{x}^{HH}, 1)$ such that $x^H = (x^L + \varphi \bar{x}^{HH})/(1+\varphi)$;
  Sample arms $x^{LL}, x^L, x^A, x^B, x^H, x^{HH}$, a number of times $N(n)$ each and observe the resulting average rewards;
  Let x be the arm with the highest average reward among $\{x^{LL}, x^L, x^A, x^B, x^H, x^{HH}\}$
  if $x = x^{LL}$ **then**
    $[x^L, x^H] \leftarrow [x^{LL}, x^H]$ {*Expand to the left*}
  else if $x \in \{x^L, x^A\}$ **then**
    $[x^L, x^H] \leftarrow [x^L, x^B]$ {*Shrink to the left*}
  else if $x \in \{x^B, x^H\}$ **then**
    $[x^L, x^H] \leftarrow [x^A, x^H]$ {*Shrink to the right*}
  else if $x = x^{HH}$ **then**
    $[x^L, x^H] \leftarrow [x^L, x^{HH}]$ {*Expand to the right*}
  end if
  Increment the iteration count $n \leftarrow n + 1$
end loop

We remark that the theoretical analysis in our work (as well as in [4] for LSE) assumes that all the samples are independent across different iterations. For this reason, we prescribe to resample each arm $N(n)$ times at iteration n. However, the use of the golden ratio ensures that at least three of the four arms in LSE and four of the six arms in LSE-backtrack remain the same at the next iteration. Hence, in practice, it is possible to reuse samples from previous iterations for such arms in order to accelerate the convergence. However, for LSE-backtrack we suggest to reuse previous samples only if the interval is shrank, since an interval expansion hints that previous observations failed to correctly estimate the average reward at some arms due to the presence of outliers.

C. LSE-backtrack: High probability PAC bounds

After presenting the details of LSE-backtrack for the optimization of bandit unimodal functions, we now deliver our main theoretical results on its convergence properties.

We start by remarking the Theorem 1, that was proven for LSE, still holds for LSE-backtrack; however, the reader should note that $(1-\delta)^n \xrightarrow{n \to +\infty} 0$ if the time horizon T is unknown and δ is constant. In other words, Theorem 1 does not allow us to prove that LSE-backtrack converges in probability to the optimal arm x^*.

Convergence in probability. Our main result presented in Theorem 2 below shows that the arms sampled by LSE-backtrack converge in probability to the optimal arm $x^* = \operatorname{argmax}_x f(x)$. The proof is in the Appendix.

Theorem 2. *Run LSE-backtrack with $\varepsilon(n) = C_L/\varphi^{n+3}$, $N(n) = 1/\varepsilon(n)^2 \log(8/\delta)$, $\delta \leq 0.074$ and initialize the search interval as $[x^L, x^H] = [0,1]$. With probability at least $1-2^{-n}$, after $3n$ iterations, the search interval $[x^L, x^H]$ has length at most φ^{-n}, and either contains optimal arm x^* or is contained in an interval of length φ^{-n} which also contains x^*.*

We first remark that LSE-backtrack is oblivious to the time horizon of the learning process: indeed, the parameter δ can be kept constant (as long as it is small enough) and $\epsilon := \epsilon(n)$ only depends on the iteration count n. Hence, our algorithm can run indefinitely and still provide performance guarantees at any time. For this reason, it is denoted as an *anytime* algorithm, and this feature has important practical implications described in Section VI.

We observe that, in order to frame the optimal arm x^*, the number of samples per arm $N(n)$ must grow exponentially with the iteration count n. This is due to the following three joint facts: *i)* the interval $[x^L, x^H]$ converges exponentially fast to x^*, *ii)* strong-max Assumption 2 holds, hence the rewards within $[x^L, x^H]$ also converge (at worst) at exponential rate to $f(x^*)$ and *iii)* by [11, Theorem 1], one has to collect a number of samples proportional to the inverse of the (square of the) reward difference of the arms to be able to distinguish them. However, we show via numerical experiments in Section VI that LSE-backtrack is able to converge to x^* even with very few samples per arm and per iteration (e.g., $N(n) := N = 5$).

Adaptation to prior belief. Theorem 2 already caters for the need of *i)* converging in probability to the optimal x^* and *ii)* being oblivious to the time horizon. However, we still need to show that *iii)* LSE-backtrack adapts to the situation

where prior belief over the location of the optimal arm is available. This translates into being able to initialize the search interval $[x^L, x^H]$ to the region where x^* is believed to lie, and still converge to x^* in probability even if the prior is unreliable. Therefore, it remains to prove that the LSE-backtrack converges to x^* in probability even if the initial search interval $[x^L, x^H]$ does not include the optimal arm x^*. For this purpose, a similar proof of Theorem 2 shows that the same result holds for a sufficiently large number of iterations n, needed to catch up with the initial offset. We report in Corollary 2.1 the formal statement of this result.

Corollary 2.1. *Assume that LSE-backtrack is initialized with a search interval $[x^L, x^H] \not\ni x^*$. Then, the result in Theorem 2 still holds for sufficiently large values of n. Hence, LSE-backtrack still converges in probability to x^*.*

In Section VI-B we evaluate numerically the sensitivity of LSE-backtrack to different initialization intervals. Specifically, we show how the evolution of the approximation error $|x_t - x^*|$ over time t is affected by inaccurate initializations of $[x^L, x^H]$.

Yu and Mannor [20] provide an upper bound for the expected regret of LSE in the order of $O(\sqrt{T \log T})$. Unfortunately, we did not succeed to prove any result on the regret of LSE-backtrack, the main technical difficulty residing in the search interval being able to either retract or enlarge, unlike LSE.

V. An efficient heuristic: LSE-weight

In this section we propose another variant of LSE, called LSE-weight, that can be also naturally adapted to the case where a prior belief on the location of the optimal arm x^* is available. We already remarked that LSE is not able to catch up with a wrong initialization that does not frame the optimal arm. On the contrary, LSE-backtrack can use prior information by initializing the algorithm with a smaller interval where the optimal arm x^* is believed to lie, and still converge to x^* in probability even if the prior is inaccurate (see Corollary 2.1).

In turn, LSE-weight relies on a weight function $w : [0, 1] \to \mathbb{R}_+$ that is not necessarily a probability density function (i.e., $\int w(x)dx$ does not necessarily equal 1). However, as intuition will suggest, the weight function w can be naturally initialized to the prior belief on the location of the optimal arm x^*.

Algorithm 3 LSE-weight

Require: Unimodal function; $f : [0, 1] \to [0, 1]$;
N: number of samples per arm;
Dampening coefficient $\beta \in [0; 1)$;
Initial weight distribution $w : [0, 1] \to \mathbb{R}^+$
loop
  Let x^A be such that $\int_0^{x^A} w(x)dx = \varphi^{-2} \int_0^1 w(x)dx$;
  Let x^B be such that $\int_0^{x^B} w(x)dx = \varphi^{-1} \int_0^1 w(x)dx$;
  Sample f at arms x^A, x^B, a number N of times each;
  Let $x \in \{x^A, x^B\}$ be the arm with the highest average reward
  if $x = x^A$ **then**
    $w(x) \leftarrow \beta w(x),\ \forall x \in [x^B, 1]$ {*Dampen right*}
  else
    $w(x) \leftarrow \beta w(x),\ \forall x \in [0, x^A]$ {*Dampen left*}
  end if
end loop

The idea behind LSE-weight is simple. In LSE, the search interval $[x^L, x^H]$ is partitioned into three subintervals $[x^L, x^A], [x^A, x^B]$ and $[x^B, x^H]$. After appropriate sampling, it is decided that either $[x^L, x^A]$ or $[x^B, x^H]$ cannot contain the optimal arm x^* according to the unimodality property, and this subinterval is definitively removed from the search region. In LSE-weight instead, at every iteration, the subinterval that is believed not to contain x^* is not eliminated, but rather its corresponding weight function w is dampened by a factor $\beta < 1$. Then, the new points x^A and x^B are decided according to the newly updated weights, in such a way that if the initial weight distribution is uniform and $\beta = 0$ then LSE-weight is equivalent to LSE.

Setting a dampening factor $\beta > 0$ allows LSE-weight to recover from mistakes, since no subinterval is ever definitively removed from the search region; hence, for LSE-weight it is reasonable in practice to use a number of samplings per iteration being lower than for LSE. On the other hand, in the extreme case where no mistake is ever committed while evaluating the best arm among $\{x^A, x^B\}$, setting $\beta > 0$ slows down the convergence of the algorithm. Thus, the best strategy would prescribe to eliminate forever the suboptimal subintervals, as LSE does. For this reason, in Lemma 1 we provide a worst-case analysis for LSE-weight: we prove that the convergence of the interval $[x^B, x^A]$ to x^* is slower than LSE at worst by a constant factor $\alpha(\beta)$ increasing with β.

Lemma 1. *Assume that algorithms LSE and LSE-weight use the same number of samples per arm at each iteration. Define by ω^{LSE} ($\omega^{\mathrm{LSE-w}}$, resp.) the number of iterations needed by LSE (LSE-weight, resp.) before $x^B - x^A \leq \ell\varphi^{-3}$, if no mistake is made for the selection of the interval at each iteration. Then,*

$$\alpha(\beta) := \frac{\omega^{\mathrm{LSE-w}}}{\omega^{\mathrm{LSE}}} = \frac{\left\lceil \log_{\frac{\varphi+\beta}{\varphi+1}}(\ell) \right\rceil}{\left\lceil -\log_{\varphi}(\ell) \right\rceil} \approx \frac{\ln \varphi}{\ln(\varphi+1) - \ln(\varphi+\beta)}.$$

For instance, $\alpha(0) = 1$ (in fact, we already observed that LSE-weight is equivalent to LSE if $\beta = 0$), $\alpha(0.5) \approx 2.27$ and $\alpha(0.75) \approx 4.8$. However, in practice, setting $\beta > 0$ allows LSE-weight to gain in flexibility, to use fewer samples per iteration, and to outperform LSE and, in some cases, even LSE-backtrack. We show this numerically in Section VI.

VI. Anytime bandits for Cloud computing

In Sections IV, V we described and provided the theoretical guarantees for our two main algorithms, LSE-backtrack and LSE-weight. Here we demonstrate their applicability to a resource allocation problem in cloud computing. In Section VI-B we will provide further intuitions through *in-vitro* numerical experiments with artificial bandit functions.

In cloud computing, one of the main challenges faced by the cloud orchestrator is to match the amount of resources allocated to a user with the user's request. Specifically, at each decision step, the orchestrator must reserve a certain amount of resources, mainly in terms of CPU and memory utilization, to a specific user for a certain time period, without knowing its future needs [22], [23]. Yet, resource over-provisioning is not a viable solution since CPU and memory are scarce resources.

It has been demonstrated in [24] that the profile of user requests in Google clusters is highly uncorrelated over time,

hence its accurate prediction proves to be extremely hard. Thus, it is challenging to keep adapting the reservation policy to the highly dynamic user request pattern. Rather, it is preferable to find the allocation policy that strikes a good medium-term balance between the running cost of the allocated resources and the risk of not meeting the user's request, hence to degrade its Quality of Service (QoS); this may also require the deployment of expensive extra resources [25].

We model such resource allocation problem in cloud as follows. At fixed time steps $t = 1, 2, \dots$ the cloud orchestrator must decide the amount x_t of resources (typically, memory or CPU) allocated to a user over the following time period. Then, the user request r_t materializes over the next period, and the orchestrator receives a penalty γ if $r_t > x_t$, i.e., if the user request cannot be fulfilled by the allocated resources. Then, the operator incurs an instantaneous cost $c_t(x_t) = x_t + \gamma \mathbb{1}(r_t > x_t)$, where $\mathbb{1}$ denotes the indicator function. Note that, once the operator has allocated x_t resources, the observed reward $-c(x_t)$ is stochastic since it depends on the number of requested resources r_t which materializes afterwards. In this scenario, the bandit function f equals the expected value of the instantaneous reward:

$$f(x) = \mathbb{E}[-c(x)] = -x - \gamma \Pr(r > x), \qquad \forall\, x. \quad (2)$$

We can further normalize f between $[0, 1]$ by accounting for the maximum number of resources that a user can ask for.

The goal of the cloud orchestrator is to learn the optimal reservation policy $x^* = \operatorname{argmax} f(x)$ by only observing the instantaneous cost function $c(x_t)$, at each time t. Moreover, the time horizon over which the user will remain connected to the infrastructure is unknown in advance to the orchestrator. Therefore, it is crucial to require that a resource allocation algorithm learning x^* must be *anytime*. We also remark that the penalty γ can be interpreted either as a financial one, described in the service level agreement stipulated with the user, or as an input parameter that tunes the sensitivity of the optimal solution with respect to the QoS degradation.

A. Real dataset experiment

We evaluate our algorithms on the Google cluster dataset [26] by considering a CPU allocation scenario. The estimated violation probability $\Pr(r > x)$ and the corresponding shape of the bandit function f are illustrated in Figure 2. We observe that f can be well approximated as a unimodal function. In fact, the distribution of user requests r is (approximately) decreasing in r, hence $\Pr(r > x)$ is (almost) convex in x.

In order to evaluate numerically our algorithms we use two metrics. The former one is the *approximation error* $|x_t - x^*|$, measuring the distance between the currently sampled arm x_t and the optimal arm x^*. The latter metric is the classic *regret* $R_t = \sum_{i=1}^{t} (f(x^*) - f(x_i))$, measuring the sum of the differences between the expected reward $f(x^*)$ of the best arm x^* and the reward collected by algorithm. Both metrics are averaged over 80 independent trials. We remind that we did not succeed to provide an upper bound for the expected regret of our algorithms.

We benchmark our algorithms against state-of-the-art approaches like LSE [4], SGD [9] and SGD without gradient (W/O gradient) [8]. Stochastic Petachotomy [12] is not adapted to our scenarios where decisions are to be taken at the time scale of seconds, since a hard approximation problem needs to be solved at each iteration. In Figure 3 we show the results of our experiments on Google cluster dataset [26]. We use the same number of samples $N = 25$ per arm for all algorithms. In practice, LSE-backtrack converges to the optimal point even for values of N smaller than those prescribed in Theorem 2, thanks to its ability to make up for previous mistakes.

We observe that all algorithms except for SGD converge quickly to a specific resource allocation value. SGD shows much slower convergence rate since it cannot directly observe a (noisy) version of $f'(x)$, thus it needs to approximate it via the expensive finite difference technique. Remarkably, LSE-backtrack manages to settle itself (on average, over 80 trials) on the closest local maximum to the optimal allocation x^*. LSE-weight follows closely. The exact convergence to x^* is not attained only because the function f shown in Figure 2 is not perfectly unimodal. LSE, instead, excludes x^* from the search interval early on, and by construction it is no longer able to recover from previous mistakes. We finally note that for W/O gradient [8] we use a learning rate of $\Omega(1/t)$ that shows optimized convergence properties.

B. In-vitro experiments

We now show via *in-vitro* experiments the convergence properties of our algorithms when *i)* the number N of samples per arm is small and independent of the iteration count and when *ii)* prior information on the location of the optimal arm x^* is available, but possibly wrong.

Figure 4 accounts for case *i)* and compares different algorithms using the same number N of samples per arm. LSE-backtrack and LSE-weight manage to converge to the optimal arm x^* even for values of N ($N = 5, 10$) being independent of the iteration n and much smaller than the values prescribed by Theorem 2 (there, $N(n) = \varphi^{2(n+3)}/(C_L)^2 \log(8/\delta)$). On the contrary, LSE deviates forever from x^* at the first interval shrinkage mistake. W/O gradient [8] can converge to x^* but at a slower rate. Finally, SGD is never able to converge with so few samples per arm, hence we omit to show it. For a fair comparison, LSE-weights is initialized with uniform weight function (i.e., no prior information is available on x^*).

We remark that, the number of samples N being equal, LSE-weight and LSE have an edge over LSE-backtrack in early iterations, since only two and fours arms respectively (instead of six for LSE-backtrack) need to be explored at each iteration. However, LSE-backtrack is more data-efficient than LSE since it can revisit its decisions, hence it could afford to collect fewer samples for each arm, especially at the first iterations when arms are easier to distinguish since farther from the peak.

The case *ii)* is covered by Figure 5 showing the approximation error evolution of LSE-backtrack for different initial search intervals $[x^L, x^H]$. The reader should appreciate how the convergence speed depends on the initial interval length $x^H - x^L$ and on whether the interval already frames the optimal arm, i.e., whether $x^* \in [x^L, x^H]$. Remarkably, LSE-backtracks always manage to converge to x^* in all the trials, even when the prior turns out to be inaccurate.

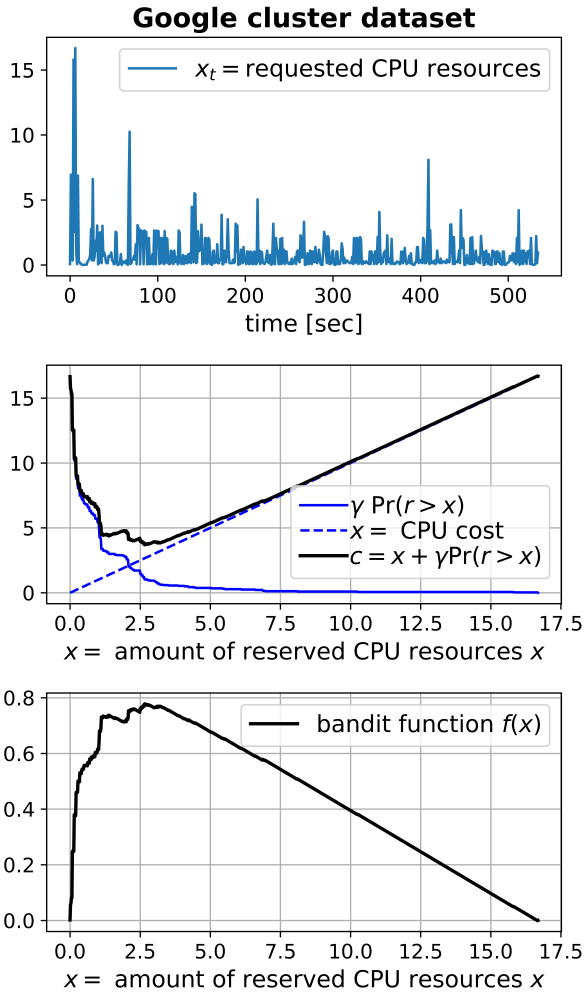


Fig. 2: **Google cluster data measurements**. *(top)*: Time series sample from [26], on the number of CPU resources requested by a cloud user. *(middle)*: Probability that the orchestrator does not meet the user's request r by allocating x CPU resources, and related cost function c. *(bottom)*: Corresponding (approximately) unimodal bandit reward function $f(x)$.

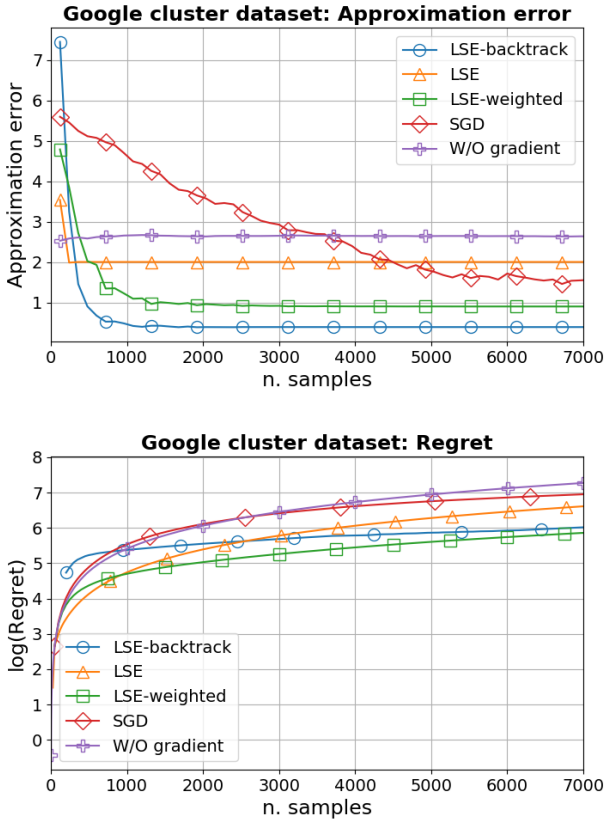


Fig. 3: **Numerical experiments on the Google cluster dataset** [26]. The algorithms are evaluated with respect to *(top)* the approximation error $|x_t - x^*|$ and *(bottom)* the regret, averaged over 80 independent trials. The (approximately) unimodal bandit function is computed as in (2), and it is illustrated in Fig. 2.

VII. Conclusion

We proposed LSE-backtrack, the first anytime algorithm for unimodal multi-armed bandit problems with continuous arm space. It converges in probability to the optimal arm by exploiting the underlying unimodal structure of the reward function. It has low complexity per iteration and it is adapted to situations where prior information on the likely location of the optimal arm is available. Indeed, it converges even when the prior is inaccurate. We also presented LSE-weight, a heuristic that naturally exploits the prior distribution on the optimal arm and performs well in practice. We showed that our algorithms can be successfully used to solve a basic problem in cloud computing. There, the main challenge faced by the orchestrator is allocating to a user the number of resources striking the optimal trade-off between the running cost of the allocated resources and the risk of degrading the user's Quality of Service. In this scenario, the anytime property of LSE-backtrack is especially appealing, since the orchestrator is not aware in advance of the duration of the user connection.

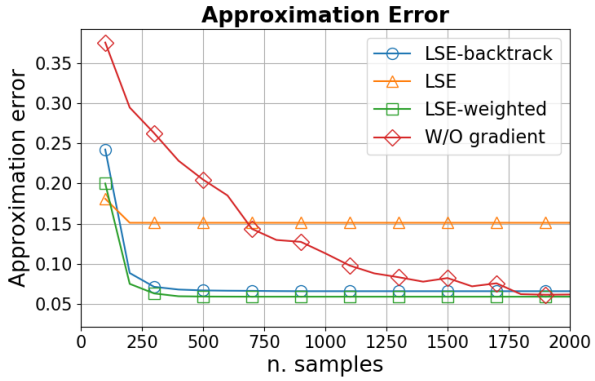
Extensive simulations showed that LSE-backtrack converges even when the number of samples per arm is exiguous, which makes our approach even more attractive in practice.

VIII. Appendix

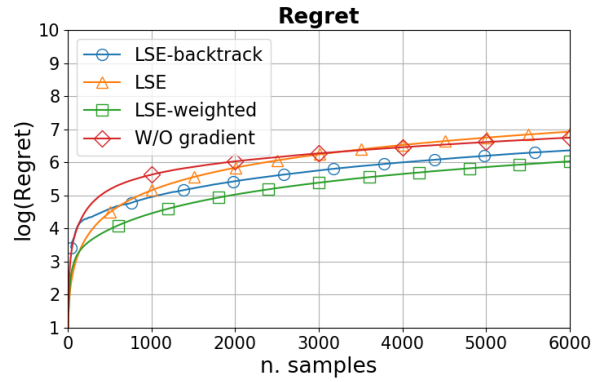
A. Proof of Theorem 2

Proof. We will exploit a Markov chain type of argument to show that the interval $[x_n^L, x_n^H]$ zooms in on an ϵ-optimal arm exponentially quickly with high probability. Let us introduce two variables to track the quality of the interval $[x_n^L, x_n^H]$ provided by LSE-backtrack at iteration n. The former variable is $\ell_n = x_n^H - x_n^L$ which denotes the length of interval $[x_n^L, x_n^H]$ during the n-th iteration. The latter variable $\xi_n \geq 0$ describes the "distance" of the interval of interest $[x^L, x^H]$ from the optimal arm x^* at iteration n. To be more specific, *i)* if $\xi_n = 0$, then a ε-optimal arm x lies in the interval $[x_n^L, x_n^H]$; *ii)* if $\xi_n > 0$, then there exists an ε-optimal arm x such that, by expanding ξ_n times the interval $[x_n^L, x_n^H]$ by a factor φ in the direction of x, it would result that $x \in [x_n^L, x_n^H]$.

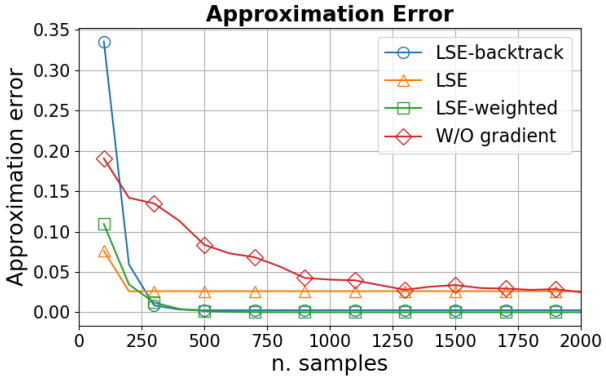
We then wish to prove that, at the n-th iteration, LSE-backtrack provides an interval $[x_n^L, x_n^H]$ of length $\ell_n = 1/\varphi^n$ in which an ε-optimal arm lies ($\xi_n = 0$) with probability at least $1 - 2^{-n}$. Then, by Assumption 2 we will conclude that the



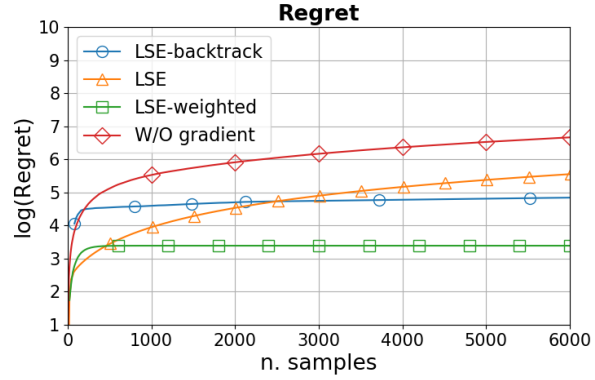
(a) Approximation error $|x_t - x^*|$ for $N = 5$ samples per arm



(b) Regret for $N = 5$ samples per arm

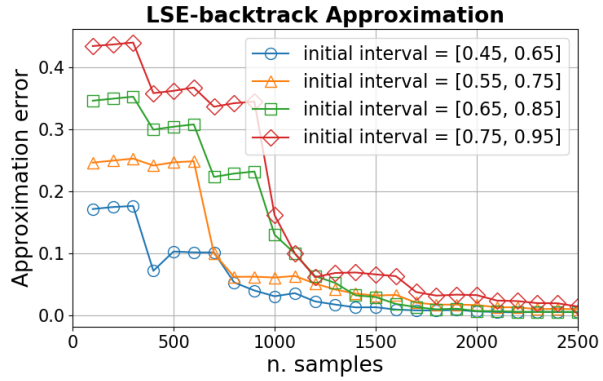


(c) Approximation error $|x_t - x^*|$ for $N = 10$ samples per arm



(d) Regret for $N = 10$ samples per arm

Fig. 4: **Small number N of samples.** Unimodal triangle function: $f(x) = x/x^*$ for $x \in [0, x^*]$ and $f(x) = 1 - (x - x^*)/(1 - x^*)$ for $x \in [x^*, 1]$, with $x^* = 0.3$ Variance of the observed reward: $|0.2 + f(x)/2|$. Results are averaged across 80 independent trials.



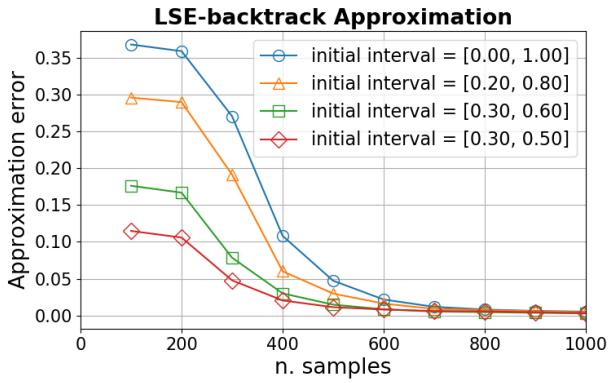


Fig. 5: **Importance of choosing a good prior** (=initial interval). LSE-backtrack approximation error $|x_t - x^*|$, averaged over 80 independent trials. Each arm is sampled $N = 30$ times, irrespective of the iteration number. We employed unimodal quadratic-wise functions with optimal arm $x^* = 0.4$. Please note how the convergence time depends on the length of the initial interval $[x^L, x^H]$, as well as on whether it contains the optimal arm x^*. Remarkably, the exact convergence to x^* is reached in all cases.

optimal arm actually lies in the interval $[x_n^L, x_n^H]$. We remark that the value of δ must not depend on the time horizon, that is unknown and potentially infinite. For this purpose, we now describe our algorithm by the following (pessimistic version of the) Markov chain on the states (ℓ, ξ). Assume that at iteration n the chain is at state (ℓ, ξ), meaning that the current interval $[x^L, x^H]$ has length $\ell_n = \ell$ and is at distance $\xi_n = \xi$ from the optimal arm x^*. Using a result in [11, Theorem 1], sampling $N = (1/\varepsilon^2)\log(8/\delta)$ times an arm approximates its expected reward within ε with probability $1-\delta$. Thus, at each iteration n, by setting $N(n) = (1/\varepsilon(n)^2)\log(8/\delta)$, an $\varepsilon(n)$-optimal arm is correctly identified with probability at least $1-\delta$. Therefore, if $\xi = 0$, then the new state is $(\ell/\varphi, 0)$ with probability at least $(1-\delta)$, while it can be $(\ell\varphi, 0)$ or $(\ell/\varphi, 1)$ with probability at most $(1-\delta)$. Otherwise, if $\xi \neq 0$, then the new state is $(\ell\varphi, \xi - 1)$ with probability at least $(1-\delta)$, while it can be $(\ell\varphi, \xi)$ or $(\ell/\varphi, \xi+1)$ with probability at most $(1-\delta)$. We illustrate this Markov chain on (ℓ, ξ) in Figure 6. We shall show that on this chain, with high probability, the algorithm does not stray too far from the left straight branch, corresponding to $\xi = 0$ and with progressively exponentially smaller distance from the optimal arm.

It is convenient to denote by E_n the number of times a "wrong" branch, i.e., a right-hand branch with probability at

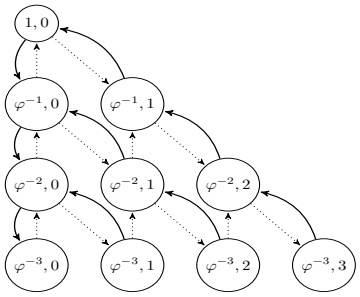


Fig. 6: **Infinite Markov chain** (ℓ, ε)**.** (the states with $\ell \leq \varphi^{-4}$ are not displayed.) Plain edges represent transitions with probability $\geq 1-\delta$ while the dotted edges are transitions with probability $\leq \delta$.

most δ, is taken by the stochastic process during the first n steps (read, iterations). Then for all $n, k \geq 0$:

$$\Pr\left(\ell_{n+2k} \leq \varphi^{-n}, \xi_{n+2k} = 0\right) \geq 1 - \Pr\left(E_{n+2k} \geq k\right).$$

In fact, if the number of wrong decisions is bounded by k, then the algorithm is able to reach state $(l = \varphi^{-n}, \epsilon = 0)$ within $n + 2k$ iterations. Next, we want to upper bound the value of $\Pr\left(E_{n+2k} \geq k\right)$. For $n \geq 1, k \geq 0$, we can write:

$$\Pr\left(E_{n+2k} \geq k\right) \leq \binom{n+2k}{k} \max_i \delta_i^k \leq \binom{n+2k}{k} \delta_1^k. \quad (3)$$

Then, by using Stirling's formula, we obtain:

$$\Pr\left(E_{n+2k} \geq k\right) \leq \sqrt{\frac{2}{\pi}} \sqrt{\frac{n+2k}{k(n+k)}} \frac{(n+2k)^{n+2k}}{k^k (n+k)^{n+k}} \delta^k.$$

In particular, by setting $k = n$ we obtain $\Pr\left(E_{3n} \geq n\right) \leq \sqrt{\frac{1}{n}} \left(\frac{27\delta}{4}\right)^n$. If $\delta \leq 0.074 < \frac{2}{27}$, then the stochastic process is at node φ^{-n} or one of its descendants with probability greater than $1 - 2^{-n}$, after $3n$ steps. In other words, after $3n$ iterations, the algorithm is sampling arms at distance at most φ^{-n} from an $\varepsilon(3n)$-optimal arm. Following Assumption 2, $|f(x_{3n}^L) - f(x_{3n}^H)| \geq C_L |x_{3n}^L - x_{3n}^H| \geq \frac{C_L}{\varphi^{-3n}}$, and an $\varepsilon(3n)$-optimal arm is at distance at most $\varepsilon(3n)/C_L = \varphi^{-3n}$ from the optimal arm x^*, q.e.d.. □

B. Proof of Lemma 1

Proof. Let x^* be the optimum arm. Let $\ell \in [0, 1]$. We start observing that, for both LSE and LSE-weight, the length $x^B - x^A$ of the interval $[x^A, x^B]$ indicates the progression of the algorithm and it measures the quality of the approximation of optimal arm x^*. By assumption, no mistake has been made when retracting the interval. Then $\max_{x \in \{x^A, x^B\}} |x^* - x| \leq \varphi^2 (x^B - x^A)$. At every iteration in algorithm LSE, the length between arms x^A and x^B in LSE is divided by φ. Initially, that length is φ^{-3}. Hence, it takes to LSE exactly $\omega^{\mathrm{LSE}} = \left\lceil -\log_\varphi(\ell) \right\rceil$ iterations until $x^B - x^A \leq \ell \varphi^{-3}$.

At each iteration of LSE-weight, the total weight is multiplied by $1 - (1-\beta)\frac{1}{\varphi^2} = \frac{\varphi+\beta}{\varphi+1}$. By definition of x^A and x^B, the weight in $[x^A, x^B]$ remains proportional to the total weight. Hence the length $|x^B - x^A|$ of interval between x^A and x^B must be multiplied accordingly by $\frac{\varphi+\beta}{\varphi+1}$. The ratio $\frac{\varphi+\beta}{\varphi+1}$ is affine in β, and interpolates between $1/\varphi \approx 0.618$ (when $\beta = 0$) and 1 (when $\beta \to 1$). Thus, for LSE-weight, $\omega^{\mathrm{LSE-w}} = \left\lceil \log_{\frac{\varphi+\beta}{\varphi+1}}(\ell) \right\rceil$ iterations are needed until $x^B - x^A \leq \ell \varphi^{-3}$. The thesis follows.

□

REFERENCES

[1] "Kubernetes description." https://kubernetes.io/.

[2] "Kubernetes autoscaling algorithm." https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/.

[3] J. Kingman, "The single server queue in heavy traffic," in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 57, pp. 902–904, Cambridge University Press, 1961.

[4] J. Y. Yu and S. Mannor, "Unimodal bandits," in *Proceedings of 28th International Conference on Machine Learning, ICML*, pp. 41–48, 2011.

[5] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in applied mathematics*, vol. 6, no. 1, pp. 4–22, 1985.

[6] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[7] A. S. Berahas, R. H. Byrd, and J. Nocedal, "Derivative-free optimization of noisy functions via quasi-newton methods," *SIAM Journal on Optimization*, vol. 29, no. 2, pp. 965–993, 2019.

[8] A. D. Flaxman, A. T. Kalai, A. T. Kalai, and H. B. McMahan, "Online convex optimization in the bandit setting: gradient descent without a gradient," in *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 385–394, 2005.

[9] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Twentieth International Conference on Machine Learning*, pp. 928–936, 2003.

[10] J. Kiefer, "Sequential minimax search for a maximum," *Proceedings of the American mathematical society*, vol. 4, no. 3, pp. 502–506, 1953.

[11] E. Even-Dar, S. Mannor, and Y. Mansour, "PAC Bounds for Multi-armed Bandit and Markov Decision Processes," in *Computational Learning Theory, 15th Annual Conference on Computational Learning Theory, COLT 2002*, vol. 2375, pp. 255–270, Springer, 2002.

[12] R. Combes and A. Proutière, "Unimodal bandits without smoothness," *CoRR*, vol. abs/1406.7447, 2014.

[13] M. R. Andersen, E. Siivola, and A. Vehtari, "Bayesian optimization of unimodal functions," in *NIPS workshop on Bayesian optimization*, 2017.

[14] R. Kleinberg, "Anytime algorithms for multi-armed bandit problems," in *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pp. 928–936, 2006.

[15] L. Besson and E. Kaufmann, "What Doubling Tricks Can and Can't Do for Multi-Armed Bandits," *arXiv preprint arXiv:1803.06971*, 2018.

[16] R. Combes and A. Proutière, "Unimodal bandits: Regret lower bounds and optimal algorithms," in *Proceedings of 31th International Conference on Machine Learning, ICML 2014*, vol. 32, pp. 521–529, 2014.

[17] A. Garivier and O. Cappé, "The KL-UCB algorithm for bounded stochastic bandits and beyond," in *Proceedings of the 24th annual Conference On Learning Theory*, pp. 359–376, 2011.

[18] O. Cappé, A. Garivier, O.-A. Maillard, R. Munos, G. Stoltz, *et al.*, "Kullback–Leibler upper confidence bounds for optimal sequential allocation," *The Annals of Statistics*, vol. 41, no. 3, pp. 1516–1541, 2013.

[19] R. Combes, A. Proutiere, D. Yun, J. Ok, and Y. Yi, "Optimal rate sampling in 802.11 systems," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pp. 2760–2767, IEEE, 2014.

[20] J. Y. Yu and S. Mannor, "Unimodal bandits," 2011. Available at https://www.win.tue.nl/~aboer/sdt/unimodal.pdf.

[21] A. Locatelli and A. Carpentier, "Adaptivity to Smoothness in X-armed bandits," in *Conference On Learning Theory, COLT 2018*, vol. 75, pp. 1463–1492, PMLR, 2018.

[22] N. Liakopoulos, G. Paschos, and T. Spyropoulos, "No regret in cloud resources reservation with violation guarantees," in *IEEE INFOCOM 2019-Conference on Computer Communications*, pp. 1747–1755, 2019.

[23] Q. Zhang, M. F. Zhani, R. Boutaba, and J. L. Hellerstein, "Dynamic heterogeneity-aware resource provisioning in the cloud," *IEEE transactions on cloud computing*, vol. 2, no. 1, pp. 14–28, 2014.

[24] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proceedings of the Third ACM Symposium on Cloud Computing*, p. 7, ACM, 2012.

[25] Amazon, "How AWS Pricing Works." Amazon, https://aws.amazon.com/whitepapers/,2018, 2018.

[26] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format + schema," *Google Inc., White Paper. Dataset available at https://github.com/google/cluster-data*, pp. 1–14, 2011.