Online Stochastic Optimization under Correlated Bandit Feedback

Mohammad Gheshlaghi Azar MOHAMMAD.AZAR@NORTHWESTERN.EDU
Rehabilitation Institute of Chicago, Northwestern University

Alessandro Lazaric ALESSANDRO.LAZARIC@INRIA.FR
Team SequeL, INRIA Nord Europe

Emma Brunskill EBRUN@CS.CMU.EDU
School of Computer Science, CMU

Abstract

In this paper we consider the problem of online stochastic optimization of a locally smooth function under bandit feedback. We introduce the high-confidence tree (HCT) algorithm, a novel anytime $\mathcal{X}$-armed bandit algorithm, and derive regret bounds matching the performance of state-of-the-art algorithms in terms of the dependency on number of steps and the near-optimality dimension. The main advantage of HCT is that it handles the challenging case of correlated bandit feedback (reward), whereas existing methods require rewards to be conditionally independent. HCT also improves on the state-of-the-art in terms of the memory requirement, as well as requiring a weaker smoothness assumption on the mean-reward function in comparison with the existing anytime algorithms. Finally, we discuss how HCT can be applied to the problem of policy search in reinforcement learning and we report preliminary empirical results.

1. Introduction

We consider the problem of maximizing the sum of the rewards obtained by sequentially evaluating an unknown stochastic function. This problem is known as stochastic optimization under bandit feedback or *$\mathcal{X}$-armed bandit*, since each function evaluation can be viewed as pulling one of the arms in a generic arm space $\mathcal{X}$. Our objective is to minimize the cumulative regret relative to selecting at each step the global maximum of the function. In particular, we focus on the case where the reward obtained by pulling an arm (i.e., evaluating the function in a point) may depend on prior history of evaluations and outcomes. This implies that the reward, conditioned on its corresponding arm, is not an independent and identically distributed (iid) random variable, in contrast to prior work on $\mathcal{X}$-armed bandits (see e.g., Munos, 2013; Kleinberg et al., 2013; Bubeck et al., 2011a). $\mathcal{X}$-armed bandit with correlated reward is relevant to many real-world applications, including internet auctions, adaptive routing, and online games. As one important example, we show that the problem of policy search in an ergodic Markov Decision Process (MDP), a popular setting for learning in unknown MDPs, can be framed as an instance of the setting we consider in this paper (Sect. 5).

Our approach builds on recent advances in $\mathcal{X}$-armed bandits for iid settings (Bull, 2013; Djolonga et al., 2013; Bubeck et al., 2011a; Srinivas et al., 2009; Cope, 2009; Kleinberg et al., 2008; Auer et al., 2007). Under regularity assumptions on the mean-reward function (e.g., Lipschitz-smoothness), these methods provide formal guarantees on the cumulative regret, which is proved to scale sub-linearly w.r.t. the number of steps n. To obtain this regret, these methods heavily rely on the iid assumption. To handle non-iid settings, we introduce a new anytime $\mathcal{X}$-armed bandit algorithm, called *high-confidence tree* (HCT) (Sect. 3). Similar to the HOO algorithm of Bubeck et al. (2011a), *HCT* makes use of a covering binary tree to explore the arm space. The tree is constructed incrementally in an optimistic fashion, where the exploration of the arm space is guided by upper bounds on the largest reward of the arms covered by a particular node. Our key insight is that to achieve small regret it is enough to expand an optimistic node only when the estimate of its mean-reward has become sufficiently accurate. Under mild ergodicity and mixing assumptions, this allows us to obtain an accurate estimate of the reward of a particular arm even in the non-iid setting. Despite handling a more general case of non-iid feedback, our regret bounds matches (Sect. 4.1) that of HOO (Bubeck et al., 2011a) and zooming algorithm

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

(Kleinberg et al., 2008), both of which only apply to iid setting, in terms of dependency on the number of steps n and the near-optimality dimension d (Sect. 2). An important part of the proof of this result is the development of concentration inequalities for non-iid episodic random variables (Sect. 4). In addition to this result, the structure of our HCT approach has a favorable sub-linear space complexity of $O(n^{d/(d+2)}(\log n)^{2/(d+2)})$ and a linearithmic runtime complexity, making it suitable for scaling to *big data* scenarios. These results meet or improve the space and time complexity of prior work designed for iid data (Sect. 4.2). Finally, we demonstrate the benefit in simulations (Sect. 6).

2. Preliminaries

The optimization problem. Let $\mathcal{X}$ be a measurable space of arms. We formalize the optimization problem as an interaction between the learner and the environment. At each time step t, the learner pulls an arm x_t in $\mathcal{X}$ and the environment returns a reward $r_t \in [0,1]$ and possibly a context $y_t \in \mathcal{Y}$, with $\mathcal{Y}$ a measurable space (e.g., the state space of a Markov decision process). Whenever needed, we relate r_t to the arm pulled by using the notation $r_t(x)$. The context y_t and the reward r_t may depend on the history of all previous rewards, pulls, contexts and the current pull x_t. For any time step $t > 0$, the space of histories $\mathcal{H}_t := ([0,1] \times \mathcal{X} \times \mathcal{Y})^t$ is defined as the space of past rewards, arms,and observations (with $\mathcal{H}_0 = \emptyset$). An environment M corresponds to an infinite sequence of time-dependent probability measures $M = (Q_1, Q_2, \dots)$, such that each $Q_t : \mathcal{H}_{t-1} \times \mathcal{X} \to \mathcal{M}([0,1] \times \mathcal{Y})$ is a mapping from the history $\mathcal{H}_{t-1}$ and the arm space $\mathcal{X}$ to the space of probability measures on rewards and contexts. Let $\mathcal{Z} = ([0,1] \times \mathcal{X} \times \mathcal{Y})$, at each step t we define the random variable $z_t = (r_t, x_t, y_t) \in \mathcal{Z}$ and we introduce the filtration $\mathcal{F}_t$ as a σ-algebra generated by $(z_1, z_2, \dots, z_t)$. At each step t, the arm x_t is $\mathcal{F}_{t-1}$-measurable since it is based on all the information available up to time $t-1$. The pulling strategy of the learner can be expressed as an infinite sequence of measurable mappings $(\psi_1, \psi_2, \dots)$, where $\psi_t : \mathcal{H}_{t-1} \to \mathcal{M}(\mathcal{X})$ maps $\mathcal{H}_{t-1}$ to the space of probability measures on arms. We refine this general setting with two assumptions on the reward-generating process.

Definition 1 (Time average reward)**.** *For any $x \in \mathcal{X}$, $S > 0$ and $0 < s \leq S$, the time average reward is $\bar{r}_{s\to S}(x) := 1/(S-s+1)\sum_{s'=s}^{S} r_{s'}(x)$.*

We now state our first assumption which guarantees that the mean of the process is well defined (ergodicity).

Assumption 1 (Ergodicity)**.** *For any $x \in \mathcal{X}$, any $s > 0$ and any sequence of prior pulls $(x_1, x_2, \dots, x_{s-1})$, the process $(z_t)_{t>0}$ is such that the mean-reward function $f(x) := \lim_{S\to\infty} \mathbb{E}(\bar{r}_{s\to S}(x)|\mathcal{F}_{s-1})$ exists.*

This assumption implies that, regardless of the history of prior observations, if arm x is pulled infinitely many times from time s, then the time average reward converges in expectation to a fixed point which only depends on arm x and is independent from the past history. We also make the following mixing assumption (see e.g., Levin et al., 2006).

Assumption 2 (Finite mixing time)**.** *There exists a constant $\Gamma \geq 0$ (mixing time) such that for any $x \in \mathcal{X}$, any $S > 0$, any $0 < s \leq S$ and any sequence of prior pulls $(x_1, x_2, \dots, x_{s-1})$, the process $(z_t)_{t>0}$ is such that we have that $|\mathbb{E}[\sum_{s'=s}^{S}(r_{s'}(x) - f(x))\big|\mathcal{F}_{s-1}]| \leq \Gamma$.*

This assumption implies that the stochastic reward process induced by pulling arm x can not substantially deviate from $f(x)$ in expectation for more than Γ *transient* steps. Note that both assumptions trivially hold if each arm is an iid process: in this case $f(x)$ is the mean of x and $\Gamma = 0$.

Given the mean-reward f, we assume that the maximizer $x^* = \arg\max_x f(x)$ exists and we denote the corresponding maximum $f(x^*)$ by f^*. We measure the performance of the learner over n steps by its regret R_n w.r.t. the f^*, defined as $R_n := nf^* - \sum_{t=1}^{n} r_t$. The goal of learner, at every $0 \leq t \leq n$, is to choose a strategy ψ_t such that the regret $\mathcal{R}_n$ is as small as possible.

Related models. Although the learner observes a context y_t at each time t, this problem differs from the contextual bandit setting (see e.g., Slivkins, 2009). In contextual bandits, the reward r_t is random realization of a function $r(x_t, y_t)$ of the selected arm and input context y_t. The contextual bandit objective is typically to minimize the regret against the optimal arm in the context provided at each step, y_t, i.e. $x_t^* = \arg\max r(x, y_t)$. A key difference is that in our model the reward, and next context, may depend on the entire history of rewards, arms pulled, and contexts, instead of only the current context and arm, and we define $f(x)$ only as the average reward obtained by pulling arm x. In this sense, our model is related to the reinforcement learning (RL) problem of trying to find a policy that maximizes the long run reward. Among prior work in RL our setting is similar to the general reinforcement learning model of Lattimore et al. (2013) which also considers arbitrary temporal dependence between rewards and observations. The main difference is that here we consider the regret in undiscounted reward scenario, whereas the focus of Lattimore et al. (2013) is on proving PAC-bounds in the discounted case. Another difference is that in our model, unlike that of Lattimore et al. (2013), the observation and action spaces need not to be finite (see further discussion in Sect. 5).

The cover tree. Similar to recent optimization methods (e.g., Bubeck et al., 2011a), our approach seeks to minimize the regret by building an estimate of f using an infinite binary *covering tree* $\mathcal{T}$, in which each node covers a subset of $\mathcal{X}$. We denote by (h, i) the node at depth h and index i among the nodes at the same depth (e.g., the root

node which covers $\mathcal{X}$ is indexed by $(0,1)$). By convention $(h+1,2i-1)$ and $(h+1,2i)$ refer to the two children of the node (h,i). The area corresponding to each node (h,i) is denoted by $\mathcal{P}_{h,i} \subset \mathcal{X}$. These regions must be measurable and, at each depth, they partition $\mathcal{X}$ with no overlap, i.e.,

$$\mathcal{P}_{0,1} = \mathcal{X}$$
$$\mathcal{P}_{h,i} = \mathcal{P}_{h+1,2i-1} \cup \mathcal{P}_{h,2i} \quad \forall h \geq 0 \text{ and } 1 \leq i \leq 2^h.$$

For each node (h,i), we define an arm $x_{h,i} \in \mathcal{P}_{h,i}$, which is pulled whenever the node (h,i) is selected.

We now state a few additional geometrical assumptions.

Assumption 3 (Dissimilarity)**.** *The space $\mathcal{X}$ is equipped with a dissimilarity function $\ell : \mathcal{X}^2 \to \mathbb{R}$ such that $\ell(x,x') \geq 0$ for all $(x,x') \in \mathcal{X}^2$ and $\ell(x,x) = 0$.*

Given a dissimilarity ℓ, the diameter of a subset $A \subseteq \mathcal{X}$ is defined as $\text{diam}(A) := \sup_{x,y \in A} \ell(x,y)$, while an ℓ-ball of radius $\varepsilon > 0$ and center $x \in \mathcal{X}$ is defined as $\mathcal{B}(x,\varepsilon) := \{x' \in \mathcal{X} : \ell(x,x') \leq \varepsilon\}$.

Assumption 4 (Local smoothness)**.** *We assume that there exist constants $\nu_2, \nu_1 > 0$ and $0 < \rho < 1$ such that for all nodes (h,i):*

(a) diam$(\mathcal{P}_{h,i}) \leq \nu_1 \rho^h$

(b) $\exists\, x^o_{h,i} \in \mathcal{P}_{h,i}$ s.t. $\mathcal{B}_{h,i} := \mathcal{B}(x^o_{h,i}, \nu_2 \rho^h) \subset \mathcal{P}_{h,i}$,

(c) $\mathcal{B}_{h,i} \cap \mathcal{B}_{h,j} = \emptyset$,

(d) For all $x \in \mathcal{X}$, $f^* - f(x) \leq \ell(x^*, x)$.

These assumptions coincide with those in (Bubeck et al., 2011a), except for the local smoothness (Asm. 4.d), where the function is assumed to be Lipschitz between any two arms x, x' *close* to the maximum x^* (i.e., $|f(x) - f(x')| \leq \ell(x,x')$), while here we only require the function to be Lipschitz w.r.t. the maximum. Finally, we characterize the *complexity* of the problem using the near-optimality dimension, which defines how *large* is the set of ϵ-optimal arms in $\mathcal{X}$. For brevity, we consider a slightly simplified definition of near-optimality dimension w.r.t. (Bubeck et al., 2011a).

Assumption 5 (Near-optimality dimension)**.** *Let $\epsilon = 3\nu_1\rho^h$ and $\epsilon' = \nu_2\rho^h < \epsilon$, for any subset of ϵ-optimal nodes $\mathcal{X}_\epsilon = \{x \in \mathcal{X} : f^* - f(x) \leq \epsilon\}$, there exists a constant C such that $\mathcal{N}(\mathcal{X}_\epsilon, \ell, \epsilon') \leq C(\epsilon')^{-d}$, where d is the near-optimality dimension of f and $\mathcal{N}(\mathcal{X}_\epsilon, \ell, \epsilon')$ is the ϵ'-cover number of $\mathcal{X}_\epsilon$ w.r.t. the dissimilarity measure ℓ.*

3. The High Confidence Tree algorithm

We now introduce the High Confidence Tree (HCT) algorithm. Throughout this discussion, a function evaluation corresponds to the reward received from pulling an arm.

Algorithm 1 The *HCT* algorithm.

Require: Parameters $\nu_1 > 0$, $\rho \in (0,1)$, $c > 0$, tree structure $(\mathcal{P}_{h,i})_{h \geq 0, 1 \leq i \leq 2^i}$ and confidence δ.
Initialize $t = 1$, $\mathcal{T}_t = \{(0,1),(1,1),(1,2)\}$, $H(t) = 1$, $U_{1,1}(t) = U_{1,2}(t) = +\infty$,
loop
  if $t = t^+$ **then** ▷ Refresh phase
    for all $(h,i) \in \mathcal{T}_t$ **do**
      $U_{h,i}(t) \leftarrow \widehat{\mu}_{h,i}(t) + \nu_1\rho^h + \sqrt{\frac{c^2 \log(1/\tilde{\delta}(t^+))}{T_{h,i}(t)}}$
    end for;
    for all $(h,i) \in \mathcal{T}_t$ Backward from $H(t)$ **do**
      if $(h,i) \in$ leaf$(\mathcal{T}_t)$ **then**
        $B_{h,i}(t) \leftarrow U_{h,i}(t)$
      else
        $B_{h,i}(t) \leftarrow \min\left[U_{h,i}(t), \max_{j \in \{2i-1, 2i\}} B_{h+1,j}(t)\right]$
      end if
    end for
  end if;
  $\{(h_t, i_t), P_t\} \leftarrow$ OptTraverse$(\mathcal{T}_t)$
  if Algorithm ***HCT*-iid** **then**
    Pull arm x_{h_t,i_t} and observe r_t
    $t = t + 1$
  else if Algorithm ***HCT*-Γ** **then**
    $T_{cur} = T_{h_t,i_t}(t)$
    while $T_{h_t,i_t}(t) < 2T_{cur}$ **AND** $t < t^+$ **do**
      Pull arm $x_{h,i}$ and observe r_t
      $(h_{t+1}, i_{t+1}) = (h_t, i_t)$
      $t = t + 1$
    end while
  end if
  Update counter $T_{h_t,i_t}(t)$ and empirical average $\widehat{\mu}_{h_t,i_t}(t)$
  $U_{h_t,i_t}(t) \leftarrow \widehat{\mu}_{h_t,i_t}(t) + \nu_1\rho^h + \sqrt{\frac{c^2 \log(1/\tilde{\delta}(t^+))}{T_{h_t,i_t}(t)}}$
  UpdateB$(\mathcal{T}_t, P_t, (h_t, i_t))$
  $\tau_h(t) = \frac{c^2 \log(1/\tilde{\delta}(t^+))}{\nu_1^2} \rho^{-2h_t}$
  if $T_{h_t,i_t}(t) \geq \tau_{h_t}(t)$ AND $(h_t, i_t) =$leaf$(\mathcal{T})$ **then**
    $\mathcal{I}_t = \{(h_t + 1, 2i_t - 1), (h_t + 1, 2i_t)\}$
    $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{I}_t$
    $U_{h_t+1,2i_t-1}(t) = U_{h_t+1,2i_t}(t) = +\infty$
  end if
end loop

We first describe the general structure of HCT, before discussing two particular variants: *HCT*-iid, designed for the case when arm rewards are iid, and *HCT*-Γ which handles the correlated feedback case, where the reward from pulling an arm may depend on all prior arms pulled and resulting outcomes. Alg. 1 shows the structure of the algorithm for *HCT*-iid and *HCT*-Γ and their minor differences.

The general structure. The *HCT* algorithm relies on a binary covering tree $\mathcal{T}$ provided as input to construct a hierarchical approximation of the mean-reward function f. At each node (h,i) of the tree, the algorithm keeps track of some statistics regarding the arm $x_{h,i}$ corresponding to node (h,i). These include the empirical estimate $\widehat{\mu}_{h,i}(t)$ of

Algorithm 2 The *OptTraverse* function.

Require: Tree $\mathcal{T}$
  $(h,i) \leftarrow (0,1)$, $P \leftarrow (0,1)$
  $T_{0,1} = \tau_0(t) = 1$;
  while $(h,i) \notin$ Leaf$(\mathcal{T})$ **AND** $T_{h,i}(t) \geq \tau_h(t)$ **do**
    if $B_{h+1,2i-1} \geq B_{h+1,2i}$ **then**
      $(h,i) \leftarrow (h+1, 2i-1)$
    else
      $(h,i) \leftarrow (h+1, 2i)$
    end if
    $P \leftarrow P \cup \{(h,i)\}$
  end while
  return (h,i) and P

Algorithm 3 The *UpdateB* function.

Require: Tree $\mathcal{T}$, the path P_t, selected node (h_t, i_t)
  if $(h_t, i_t) \in$ Leaf$(\mathcal{T})$ **then**
    $B_{h_t,i_t}(t) = U_{h_t,i_t}(t)$
  else
    $B_{h_t,i_t}(t) = \min\left[U_{h_t,i_t}(t), \max_{j \in \{2i_t-1, 2i_t\}} B_{h_t+1,j}(t)\right]$
  end if;
  for all $(h,i) \in P_t - (h_t, i_t)$ backward **do**
    $B_{h,i}(t) = \min\left[U_{h,i}(t), \max_{j \in \{2i-1, 2i\}} B_{h+1,j}(t)\right]$
  end for

the mean-reward of $x_{h,i}$ computed as

$$\widehat{\mu}_{h,i}(t) := (1/T_{h,i}(t)) \sum_{s=1}^{T_{h,i}(t)} r^s(x_{h,i}), \qquad (1)$$

where $T_{h,i}(t)$ is the number of times node (h,i) has been selected in the past and $r^s(x_{h,i})$ denotes the s-th reward observed after pulling $x_{h,i}$ (while we previously used r_t to denote the t-th sample of the overall process). As explained in Sect. 2, although a node is associated to a single arm $x_{h,i}$, it also covers a full portion of the input space $\mathcal{X}$, i.e., the subset $\mathcal{P}_{h,i}$. Thus, similar to the HOO algorithm (Bubeck et al., 2011a), *HCT* also maintains two upper-bounds, $U_{h,i}$ and $B_{h,i}$, which are meant to bound the mean-reward $f(x)$ of all the arms $x \in \mathcal{P}_{h,i}$. For any node (h,i), the upper-bound $U_{h,i}$ is computed as

$$U_{h,i}(t) := \widehat{\mu}_{h,i}(t) + \nu_1 \rho^h + c\sqrt{\log(1/\tilde{\delta}(t^+))/T_{h,i}(t)}, \quad (2)$$

where $t^+ = 2^{\lfloor \log(t) \rfloor + 1}$ and $\tilde{\delta}(t) := \min\{c_1 \delta / t, 1\}$. Intuitively speaking, the second term is related to the *resolution* of node (h,i) and the third term accounts for the *uncertainty* of $\widehat{\mu}_{h,i}(t)$ in estimating $f(x_{h,i})$. The B-values are designed to have a tighter upper bound on $f(x)$ by taking the minimum between $U_{h,i}$ for the current node, and the maximum upper bound of the node's two child nodes, if present.[1] More precisely,

$$B_{h,i}(t) = \begin{cases} U_{h,i}(t) & (h,i) \in \text{leaf}(\mathcal{T}_t) \\ \min[U_{h,i}(t), \max_{j \in \{2i-1, 2i\}} B_{h+1,j}(t)] & \text{otherwise.} \end{cases} \qquad (3)$$

To identify which arm to pull, the algorithm traverses the tree along a path P_t obtained by selecting nodes with maximum $B_{h,i}$ until it reaches an optimistic node (h_t, i_t), which is either a leaf or a node which is not pulled enough w.r.t. to a given threshold $\tau_h(t)$, i.e., $T_{h,i}(t) \leq \tau_h(t)$ (see function *OptTraverse* in Alg. 2). Then the arm $x_{h_t,i_t} \in \mathcal{P}_{h_t,i_t}$ corresponding to selected node (h_t, i_t) is pulled.

The key element of *HCT* is the condition to decide when to expand the tree. We expand a leaf node only if we have pulled its corresponding arm a sufficient number of times such that the uncertainty over the maximum value of the arms contained within that node is dominated by the size of the subset of $\mathcal{X}$ it covers. Recall from Eq. 2 that the upper bound $U_{h,i}$ is composed of two terms beside the empirical average reward. The first ($\nu_1 \rho^h$) is a constant that depends only on the node depth and from assumptions 3 and 4 it follows that it bounds the possible difference in the mean-reward function between the representative arm for this node and all other arms also contained in this node, i.e., the difference between $f(x_{h,i})$ and $f(x)$ for any other $x \in \mathcal{P}_{h,i}$. The second term depends only on t and decreases with the number of pulls. At some point, the second term will become smaller than the first term, implying that the uncertainty over the rewards in $\mathcal{P}_{h,i}$ becomes dominated by the potential difference in the mean-reward of the arms in the node. This means that the domain $\mathcal{P}_{h,i}$ is too large, and thus the resolution of the current approximation of f in that region needs to be increased. Therefore *HCT* waits until these two terms become of the same magnitude before expanding a node. This happens when the number of pulls $T_{h_t,i_t}(t)$ exceeds a threshold

$$\tau_h(t) := c^2 \log(1/\tilde{\delta}(t^+)) \rho^{-2h_t} / \nu_1^2. \qquad (4)$$

(see Sect. A of the supplement for further discussion). It is at this point that expanding the node to two children can increase the accuracy of the approximation of $f(x)$, since $\nu_1 \rho^{h+1} \leq \nu_1 \rho^h$. Therefore if $T_{h_t,i_t}(t) \geq \tau_h(t)$, the algorithm expands the leaf, creates both children leaves, and set their U-values to $+\infty$. Furthermore, notice that this expansion only occurs for nodes which are likely to contain x^*. In fact, *OptTraverse* does select nodes with big B-value, which in turn receive more pulls and are thus expanded first. The selected arm x_{h_t,i_t} is pulled either for a single time step (in *HCT*-iid) or for a full episode (in *HCT*-Γ), and then the statistics of all the nodes along the optimistic path P_t are updated backwards. The statistics of all the nodes outside the optimistic path remain unchanged.

[1] Since the node's children together contain the same input space as the node (i.e., $\mathcal{P}_{h,i} = \mathcal{P}_{h+1,2i-1} \cup \mathcal{P}_{h,2i}$), the node's maximum cannot be greater than the maximum of its children.

As *HCT* is an anytime algorithm, we periodically need to recalculate the node upper bounds to guarantee their validity with *enough* probability (see supplementary material for a more precise discussion). To do so, at the beginning of each step t, the algorithm verifies whether the B and U values need to be refreshed or not. In fact, in the definition of U in Eq. 2, the uncertainty term depends on the confidence $\tilde{\delta}(t^+)$, which changes at $t = 1, 2, 4, 8, \ldots$. Refreshing the U and B values triggers a "resampling phase" of the internal nodes of the tree $\mathcal{T}_t$ along the optimistic path. In fact, the second condition in the *OptTraverse* function (Alg. 2) forces *HCT* to pull arms that belong to the current optimistic path P_t until the number of pulls $T_{h,i}(t)$ becomes greater than $\tau_h(t)$ again. Notice that the choice of the confidence term $\tilde{\delta}$ is particularly critical. For instance, choosing a more natural $\tilde{\delta}(t)$ would tend to trigger the refresh (and the resampling) phase too often thus increasing the computational complexity of the algorithm and seriously affecting its theoretical properties. On the other hand, the choice of $\tilde{\delta}(t^+)$ limits the need to refresh the U and B values to only $O(\log(n))$ times over n rounds and guarantees that U and B are valid upper bounds with high probability.

***HCT*-iid and *HCT*-Γ.** The main difference between the two implementations of *HCT* is that, while *HCT*-iid pulls the selected arm for only one step before re-traversing the tree from the root to again find another optimistic node, *HCT*-Γ pulls the the representative arm of the optimistic node for an episode of T_{cur} steps, where T_{cur} is the number of pulls of arm $x_{h,i}$ at the beginning of episode. In other words, the algorithm doubles the number of pulls of each arm throughout the episode. Notice that a similar approach has been used before in other methods working with ergodic processes, such as the UCRL algorithm for ergodic MDPs (Jaksch et al., 2010). The additional stopping condition in the loop is such that not all the episodes may actually finish after T_{cur} steps and double the number of pulls: The algorithm may interrupt the episode when the confidence bounds of B and U are not valid anymore (i.e., $t \geq t^+$) and perform a refresh phase. The reason for this change is that in order to accurately estimate the mean-reward given correlated bandit feedback, it is necessary to pull an arm for a series of pulls rather than a single pull. Due to our assumption on the mixing time (Asm. 2), pulling an arm for a sufficiently long consecutive number of steps will provide an accurate estimate of the mean-reward even in the correlated setting, thus ensuring that the empirical average $\widehat{\mu}_{h,i}$ actually concentrates towards their mean value (see Lem. 2). It is this mechanism, coupled with only expanding the nodes after obtaining a good estimate of their mean reward, that allows us to handle the correlated feedback setting. Although in this sense *HCT*-Γ is more general, we do however include the *HCT*-iid variant because whenever the rewards are iid it performs better than *HCT*-Γ. This is due to the fact that, unlike *HCT*-iid, *HCT*-Γ has to keep pulling an arm for a full episode even when there is evidence that another arm could be better. We also notice that there is a small difference in the constants c_1 and c: in *HCT*-iid $c_1 := \sqrt[8]{\rho/(3\nu_1)}$ and $c := 2\sqrt{1/(1-\rho)}$, whereas *HCT*-Γ uses $c_1 := \sqrt[9]{\rho/(4\nu_1)}$ and $c := 3(3\Gamma+1)\sqrt{1/(1-\rho)}$.

4. Theoretical Analysis

In this section we analyze the regret and the complexity of *HCT*. All the proofs are reported in the supplement.

4.1. Regret Analysis

We start by reporting a bound on the maximum depth of the trees generated by *HCT*.

Lemma 1. *Given the threshold $\tau_h(t)$ in Eq. 4, the depth $H(n)$ of the tree $\mathcal{T}_n$ is bounded as*

$$H(n) \leq H_{\max}(n) = 1/(1-\rho)\log(n\nu_1^2/(2(c\rho)^2)). \quad (5)$$

This bound guarantees that *HCT* never expands trees beyond depth $O(\log n)$. This is ensured by the fact the *HCT* waits until the mean-reward of a node is sufficiently well estimated before expanding it and this implies that the number of pulls exponentially grows with the depth of tree, thus preventing the depth to grow linearly as in HOO.

We report regret bounds in high probability, bounds in expectation can be obtained using standard techniques.

Theorem 1 (Regret bound of *HCT*-iid)**.** *Let Assumptions 3–5 hold and at each step t, the reward r_t is independent of all prior random events. Then the regret of HCT-iid in n steps is, with probability $1-\delta$,*[2]

$$R_n \leq O\big(\big(\log(n/\delta)\big)^{1/(d+2)} n^{(d+1)/(d+2)}\big).$$

Remark (the bound). We notice that the bound perfectly matches the bound for HOO up to constants (see Thm. 6 in (Bubeck et al., 2011a)). This represents a first sanity check w.r.t. the structure of *HCT*, since it shows that changing the structure of HOO and expanding nodes only when they are pulled enough, preserves the regret properties of the algorithm. Furthermore, this result holds under milder assumptions than HOO. In fact, Asm. 4-(d) only requires f to be Lipschitz w.r.t. to the maximum x^*. Other advantages of *HCT*-iid are discussed in the Sect. 4.2 and 6.

Although the proof is mostly based on standard techniques and tools from bandit literature, *HCT* has a different structure from HOO (and similar algorithms) and moving from iid to correlated arms calls for the development of a significantly different proof technique. The main technical issue

[2]Constants are provided in Sect. A of the supplement.

is to show that the empirical average $\widehat{\mu}_{h,i}$ computed by averaging rewards obtained across different episodes actually converges to $f(x_{h,i})$. In particular, we prove the following high-probability concentration inequality (see Lem. 6 in the supplement for further details).

Lemma 2. *Under Assumptions 1 and 2, for any fixed node (h,i) and step t, we have that, w.p. $1-\delta$,*

$$|\widehat{\mu}_{h,i}(t) - f(x_{h,i})| \leq (3\Gamma+1)\sqrt{\frac{2\log(5/\delta)}{T_{h,i}(t)}} + \frac{\Gamma\log(t)}{T_{h,i}(t)}.$$

Furthermore $K_{h,i}(t)$, the number of episodes in which (h,i) is selected, is bounded by $\log_2(4T_{h,i}(t)) + \log_2(t)$.

This technical lemma is at the basis of the derivation of the following regret bound for *HCT*-Γ.

Theorem 2 (Regret bound of *HCT*-Γ)**.** *Let Assumptions 1–5 hold and that rewards are generated according to the general model defined in Sect. 2. Then the regret of HCT-Γ after n steps is, w.p. $1-\delta$, $R_n \leq O\big((\log(n/\delta))^{1/(d+2)} n^{(d+1)/(d+2)}\big)$.*

Remark (the bound). The most interesting aspect of this bound is that *HCT*-Γ achieves the same regret as *HCT*-iid when samples are non-iid. This represents a major step forward w.r.t. the existing algorithms, since it shows that the very general case of correlated rewards can be managed as well as the simpler iid case. In Sect. 5 we discuss how this result can be used in policy search for MDPs.

4.2. Complexity

Time complexity. The run time complexity of both versions of *HCT* is $O(n\log(n))$. This is due to the boundedness of the depth $H(n)$ and by the structure of the refresh phase. By Lem. 1, we have that the maximum depth is $O(\log(n))$. As a result, at each step t, the cost of traversing the tree to select a node is at most $O(\log n)$, which also coincides with the cost of updating the B and U values of the nodes in the optimistic path P_t. Thus, the total cost of selecting, pulling, and updating nodes is no larger than $O(n\log n)$. Notice that in case of *HCT*-Γ, once a node is selected is pulled for an entire episode, which further reduces the total selection cost. Another computational cost is represented by the refresh phase where all the nodes in the tree are actually updated. Since the refresh is performed only when $t = t^+$, then the number of times all the nodes are refreshed is of order of $O(\log n)$ and the boundedness of the depth guarantees that the number of nodes to update cannot be larger than $O(2^{\log n})$, which still corresponds to a total cost of $O(n\log n)$. This implies that *HCT* achieves the same run time as *T-HOO* (Bubeck et al., 2011a). Though unlike *T-HOO*, our algorithm is fully anytime and it does not suffer from the extra regret incurred due to the truncation and the doubling trick.

Space complexity. The following theorem provides bound on space complexity of the *HCT* algorithm.

Theorem 3. *Under the same conditions of Thm. 2, let $\mathcal{N}_n$ denote the space complexity of HCT-Γ, then we have that $\mathbb{E}(\mathcal{N}_n) = O(\log(n)^{2/(d+2)} n^{d/(d+2)})$.*

This result guarantees that the space complexity of *HCT*-Γ scales sub-linearly w.r.t. n. An important observation is that the space complexity of *HCT*-Γ increases slower, by a factor of $\widetilde{O}(n^{1/(d+2)})$, than its regret. This implies that, for small values of d, HCT does not require to use a large memory space to achieve a good performance. An interesting special case is the class of problem with near-optimality dimension $d = 0$. For this class of problems the bound translates to a space complexity of $O(\log(n))$, whereas the space complexity of alternative algorithms may be as large as n (see e.g., HOO). The fact that *HCT*-Γ solves the optimization problem using only a relatively small memory space makes it a suitable choice for *big-data* applications, where the algorithms with linear space complexity can not be used due to very large size of the dataset.

Switching frequency. Finally, we also remark another interesting feature of *HCT*-Γ. Since an arm is pulled for an entire episode before another arm could be selected, this drastically reduces the number of switches between arms. In many applications, notably in reinforcement learning (see next section), this can be a significant advantage since pulling an arm may correspond to the actual implementation of a complex solution (e.g., a position in a portfolio management problem) and continuously switch between different arms might not be feasible. More formally, since each node has a number of episodes bounded by $O(\log n)$ (Lem. 2), then the number of switches can be derived from the number of nodes in Thm. 3 multiplied by $O(\log n)$, which leads to $O(\log(n)^{(d+4)/(d+2)} n^{d/(d+2)})$.

5. Application to Policy Search in MDPs

As discussed in Sect. 2, *HCT* is designed to handle the very general case of optimization in problems with strong correlation among the rewards, arm pulls, and contexts, at different time steps. An important subset of this general class is represented by the problem of policy search in infinite-horizon *ergodic* Markov decision processes.

A MDP M is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, P\rangle$ where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $P : \mathcal{S}\times\mathcal{A} \to \mathcal{M}(\mathcal{S}\times[0,1])$ is the transition kernel mapping each state-action pair to a distribution over states and rewards. A (stochastic) policy $\pi : \mathcal{S} \to \mathcal{M}(\mathcal{A})$ is a mapping from states to distribution over actions. Policy search algorithms (Scherrer & Geist, 2013; Azar et al., 2013; Kober & Peters, 2011) aim at finding the policy in a given policy set which maximizes the long-term performance. Formally, a policy search algorithm receives as input a set of policies $\mathcal{G} = \{\pi_\theta; \theta \in \Theta\}$,

each of them parameterized by a parameter vector θ in a given set $\Theta \subset \Re^d$. Any policy $\pi_\theta \in \mathcal{G}$ induces a state-reward transition kernel $T : \mathcal{S} \times \Theta \to \mathcal{M}(\mathcal{S} \times [0,1])$. T relates to the state-reward-action transition kernel P and the policy kernel π_θ as follows $T(ds', dr|s, \theta) := \int_{u \in \mathcal{A}} P(ds', dr|s, u)\pi_\theta(du|s)$. For any $\pi_\theta \in \mathcal{G}$ and initial state $s_0 \in S$, the time-average reward over n steps is $\mu^{\pi_\theta}(s_0, n) := 1/n\mathbb{E}[\sum_{t=1}^{n} r_t]$, where $r_1, r_2, \ldots, r_n$ is the sequence of rewards observed by running π_θ for n steps staring at s_0. If the Markov reward process induced by π_θ is ergodic, $\mu^{\pi_\theta}(s_0, n)$ converges to a fixed point independent of the initial state s_0. The average reward of π_θ is thus defined as $\mu(\theta) := \lim_{n\to\infty} \mu^{\pi_\theta}(s_0, n)$. The goal of policy search is to find the best $\theta^* = \arg\max_{\theta\in\Theta} \mu(\theta)$.[3]

It is straightforward now to match the MDP scenario to the general setting in Sect. 2, notably mapping Θ to $\mathcal{X}$ and $\mu(\theta)$ to $f(x)$ (further details are provided in Sect. D of the supplement). This allows us to directly apply *HCT*-Γ to the problem of policy search. The advantage of *HCT*-Γ algorithm w.r.t. prior work is that, to the best of our knowledge, it is the first policy search algorithm which provides finite sample guarantees in the form of regret bounds on the performance loss of policy search in MDPs (see Thm. 2), which guarantee that *HCT*-Γ suffers from a small sub-linear regret w.r.t. π_{θ^*}. Also, it is possible to prove that the policy induced by *HCT*-Γ has a small simple regret, that is, the average reward of the policy chosen by *HCT*-Γ converges to $\mu(\theta^*)$ with a polynomial rate.[4] Another interesting feature of *HCT*-Γ is that it can be used in large (continuous) state-action problems since it does not make any restrictive assumption on the size of state-action space.

Related work. A related work to *HCT*-Γ is the UCCRL algorithm by Ortner & Ryabko (2012), which extends the original UCRL algorithm (Jaksch et al., 2010) to continuous state spaces. Although a direct comparison between the two methods is not possible, it is interesting to notice that the assumptions used in UCCRL are stronger than for *HCT*-Γ, since they require both the dynamics and the reward function to be globally Lipschitz. Furthermore, UCCRL requires the action space to be finite, while *HCT*-Γ can deal with any continuous policy space. Finally, while *HCT*-Γ is guaranteed to minimize the regret against the best policy in the policy class $\mathcal{G}$, UCCRL targets the performance of the actual optimal policy of the MDP at hand. Another relevant work is the OMDP algorithm of Abbasi et al. (2013) which deals with the problem of RL in continuous state-action MDPs with adversarial rewards. OMDP achieves a sub-linear regret under the assumption that the space of policies is finite.

[3]Note that π_{θ^*} is optimal in the policy class $\mathcal{G}$ and it may not coincide with the optimal policy π^* of the MDP.

[4]Refer to Bubeck et al. (2011a); Munos (2013) for how to transform cumulative regret bounds to simple regret bounds.

6. Numerical Results

While our primary contribution is the definition of *HCT* and its technical analysis, we also give preliminary simulation results to demonstrate some of its properties.

Setup. We focus on minimizing the regret across repeated noisy evaluations of the garland function $f(x) = x(1-x)(4-\sqrt{|\sin(60x)|})$ relative to repeatedly selecting its global optima.[5] We evaluate the performance of each algorithm in terms of the per-step regret, $\widetilde{R}_n = R_n/n$. Each run is $n = 10^5$ steps and we average the performance on 10 runs. For all the algorithms compared in the following, parameters[6] are optimized to maximize their performance.

I.i.d. setting. In the first experiment we compare *HCT*-iid to the truncated hierarchical optimistic optimization (T-HOO) algorithm (Bubeck et al., 2011a). T-HOO is a state-of-the-art $\mathcal{X}$-armed bandit algorithm, developed as a computationally-efficient alternative of HOO. In Fig. 1 we show the per-step regret, the runtime, and the space requirements of each approach. As predicted by the theoretical bounds, the per-step regret $\widetilde{R}_n$ of both *HCT*-iid and *T-HOO* decreases rapidly with number of steps. Though the big-O bounds are identical for both approaches, empirically we observe that in this setting *HCT*-iid outperforms *T-HOO* by a large margin. Similarly, though the computational complexity of both approaches matches in the dependence on the number of time steps, empirically we observe that our approach outperforms *T-HOO* (Fig. 1). Perhaps the most significant expected advantage of HCT-iid over T-HOO for iid settings is in the space requirements. HCT-iid has a space requirement for this domain that scales logarithmically with the time step n, as predicted by Thm. 3. In contrast, in this domain we observe a polynomial growth of memory usage for T-HOO. These patterns mean that *HCT*-iid can achieve a very small regret using a sparse cover tree with only few hundred nodes, whereas *T-HOO* requires orders of magnitude more nodes than *HCT*-iid.

Correlated setting. In this setting, we compare *HCT*-Γ to PoWER, a standard RL policy search algorithm (Kober & Peters, 2011), on a continuous-state-action MDP constructed out of the garland function.[7] PoWER uses an Expectation Maximization approach to optimize the policy parameters. We also compare our algorithm with T-HOO, although this algorithm is designed for the iid setting and it may fail to converge to the global optimum under correlated bandit feedback. Fig. 2 shows per-step regret of the 3 approaches in the MDP. Only *HCT*-Γ suc-

[5]We discuss some properties of the garland function in Sect. C of the supplement where the function is illustrated in Fig. 3.

[6]For both HCT and T-HOO we introduce a tuning parameter used to multiply the upper bounds, while for PoWER we optimize the window for computing the weighted average.

[7]See Sect. C of the supplement for details.

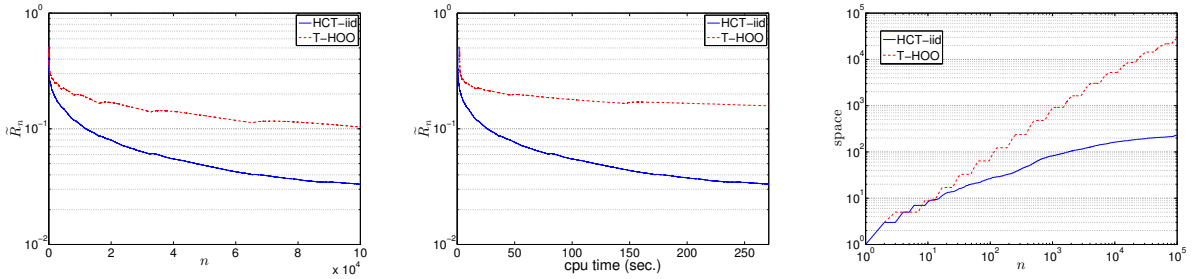

Figure 1. Comparison of the Performance of HCT-iid and the Previous Methods under the iid Bandit Feedback.

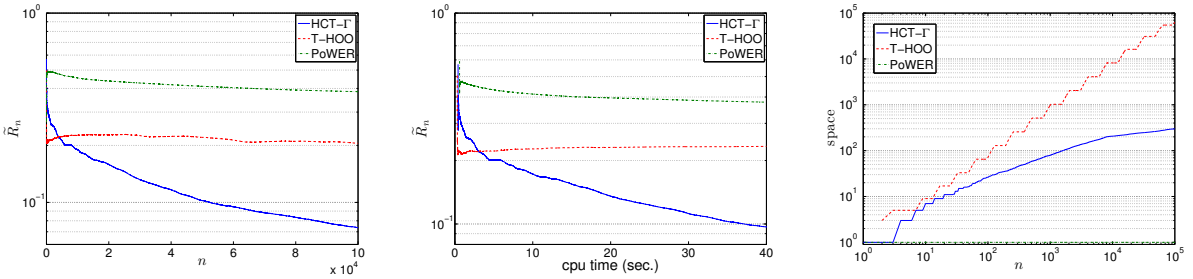

Figure 2. Comparison of the Performance of HCT $-$ Γ and the Previous Methods under Correlated Bandit Feedback (MDP setting)

ceeds in finding the globally optimal policy, since only for *HCT*-Γ the average regret tends to converge to zero (as predicted by Thm. 2). The PoWER method finds worse solutions than both stochastic optimization approaches for the same amount of computational time, likely due to using EM which is known to be susceptible to local optima. On the other hand, its primary advantage is that it has a very small memory requirement. Overall this illustrates the benefit of *HCT* for online MDP policy search, since it can quickly (as a function of samples and runtime) find a global optima, and is, to our knowledge, one of the only policy search methods guaranteed to do so.

7. Discussion and Future Work

In the current version of *HCT* we assume that the learner has access to the information regarding the smoothness of function $f(x)$ and the mixing time Γ. In many problems those information are not available to the learner. In the future it would be interesting to build on prior work that handles unknown smoothness in iid settings and extend it to correlated feedback. For example, Bubeck et al. (2011b) require a stronger global Lipschitz assumption and propose an algorithm to estimate the Lipschitz constant. Other work on the iid setting include Valko et al. (2013) and Munos (2011), which are limited to the simple regret scenario, but who only use the mild local smoothness assumption we define in Asm. 4, and do not require knowledge of the dissimilarity measure ℓ. On the other hand, Slivkins (2011) and Bull (2013) study the cumulative regret but consider a different definition of smoothness related to the zooming concept introduced by Kleinberg et al. (2008). Finally, we notice that to deal with unknown mixing time, one may rely on data-dependent tail's inequalities, such as empirical Bernstein inequality (Tolstikhin & Seldin, 2013; Maurer & Pontil, 2009), replacing the mixing time with the empirical variance of the rewards.

In the future we also wish to explore using *HCT* in other problems that can be modeled as optimization with correlated bandit feedback. For example, *HCT* may be used for policy search in partially observable MDPs (Vlassis & Toussaint, 2009; Baxter & Bartlett, 2000), as long as the POMDP is ergodic.

To conclude, in this paper we introduce a new $\mathcal{X}$-armed bandit algorithm, called *HCT*, for optimization under bandit feedback and prove regret bounds and simulation results for it. Our approach improves on existing results to handle the important case of correlated bandit feedback. This allows *HCT* to be applied to a broader range of problems than prior $\mathcal{X}$-armed bandit algorithms, such as policy search in continuous MDPs.

Acknowledgements

This work was supported in part by the NSF Award SBE-0836012 to the Pittsburgh Sciences of Learning Center. We would like to thank R. Munos, A. Farahmand and A. Slivkins for valuable discussions. A. Lazaric would like to acknowledge the support of the Ministry of Higher Education and Research, Nord-Pas- de-Calais Regional Council and FEDER through the Contrat de Projets Etat Region (CPER) 2007-2013, and the European Communitys Seventh Framework Programme (FP7/2007-2013) under grant agreement 231495 (project CompLACS).

References

Abbasi, Yasin, Bartlett, Peter, Kanade, Varun, Seldin, Yevgeny, and Szepesvari, Csaba. Online learning in markov decision processes with adversarially chosen transition probability distributions. In Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K.Q. (eds.), *Advances in Neural Information Processing Systems 26*, pp. 2508–2516. 2013.

Auer, Peter, Ortner, Ronald, and Szepesvári, Csaba. Improved rates for the stochastic continuum-armed bandit problem. In *COLT*, pp. 454–468, 2007.

Azar, Mohammad Gheshlaghi, Lazaric, Alessandro, and Brunskill, Emma. Regret bounds for reinforcement learning with policy advice. In *ECML/PKDD*, pp. 97–112, 2013.

Baxter, Jonathan and Bartlett, Peter L. Reinforcement learning in pomdp's via direct gradient ascent. In *ICML*, pp. 41–48, 2000.

Bubeck, Sébastien, Munos, Rémi, Stoltz, Gilles, and Szepesvári, Csaba. *X*-armed bandits. *Journal of Machine Learning Research*, 12:1655–1695, 2011a.

Bubeck, Sébastien, Stoltz, Gilles, and Yu, Jia Yuan. Lipschitz bandits without the lipschitz constant. In *ALT*, pp. 144–158, 2011b.

Bull, Adam. Adaptive-tree bandits. *arXiv preprint arXiv:1302.2489*, 2013.

Cope, Eric. Regret and convergence bounds for immediate-reward reinforcement learning with continuous action spaces. *IEEE Transactions on Automatic Control*, 54(6): 1243–1253, 2009.

Djolonga, Josip, Krause, Andreas, and Cevher, Volkan. High dimensional gaussian process bandits. In *Neural Information Processing Systems (NIPS)*, 2013.

Jaksch, Thomas, Ortner, Ronald, and Auer, Peter. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11:1563–1600, 2010.

Kleinberg, Robert, Slivkins, Aleksandrs, and Upfal, Eli. Multi-armed bandits in metric spaces. In *STOC*, pp. 681–690, 2008.

Kleinberg, Robert, Slivkins, Aleksandrs, and Upfal, Eli. Bandits and experts in metric spaces. *arXiv preprint arXiv:1312.1277*, 2013.

Kober, Jens and Peters, Jan. Policy search for motor primitives in robotics. *Machine Learning*, 84(1-2):171–203, 2011.

Lattimore, Tor, Hutter, Marcus, and Sunehag, Peter. The sample-complexity of general reinforcement learning. In *Proceedings of Thirtieth International Conference on Machine Learning (ICML)*, 2013.

Levin, David A., Peres, Yuval, and Wilmer, Elizabeth L. *Markov chains and mixing times*. American Mathematical Society, 2006.

Maurer, Andreas and Pontil, Massimiliano. Empirical bernstein bounds and sample variance penalization. *arXiv preprint arXiv:0907.3740*, 2009.

Munos, Rémi. Optimistic optimization of a deterministic function without the knowledge of its smoothness. In *NIPS*, pp. 783–791, 2011.

Munos, Rémi. From bandits to monte-carlo tree search: The optimistic principle applied to optimization and planning. *Foundations and Trends in Machine Learning*, 2013.

Ortner, Ronald and Ryabko, Daniil. Online regret bounds for undiscounted continuous reinforcement learning. In Bartlett, P., Pereira, F.c.n., Burges, C.j.c., Bottou, L., and Weinberger, K.q. (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1772–1780, 2012.

Scherrer, Bruno and Geist, Matthieu. Policy search: Any local optimum enjoys a global performance guarantee. *arXiv preprint arXiv:1306.1520*, 2013.

Slivkins, Aleksandrs. Contextual bandits with similarity information. *CoRR*, abs/0907.3986, 2009.

Slivkins, Aleksandrs. Multi-armed bandits on implicit metric spaces. In *Advances in Neural Information Processing Systems*, pp. 1602–1610, 2011.

Srinivas, Niranjan, Krause, Andreas, Kakade, Sham M., and Seeger, Matthias. Gaussian process bandits without regret: An experimental design approach. *CoRR*, abs/0912.3995, 2009.

Tolstikhin, Ilya O and Seldin, Yevgeny. PAC-bayes-empirical-bernstein inequality. In *Advances in Neural Information Processing Systems*, pp. 109–117, 2013.

Valko, Michal, Carpentier, Alexandra, and Munos, Rémi. Stochastic simultaneous optimistic optimization. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 19–27, 2013.

Vlassis, Nikos and Toussaint, Marc. Model-free reinforcement learning as mixture learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1081–1088, 2009.