MVA Reinforcement Learning
Optimization of very difficult functions

Nathan de Lara, Florian Tilquin

January 12, 2016

1 Introduction

1.1 Problem statement

The goal of this paper is to test and compare recently developed algorithms for the optimization of *very difficult functions*. This work is based on the papers by Bubeck [2], Grill [4], Lazaric [1], Bull [3] and Valko [5]. Each one of this paper has a specific definition of *difficult* but the general idea is that the function to optimize has many local maxima and only one global maximum, it has very fast variations and is not necessarily differentiable such that a gradient-based approach to find the optimum should not be successful. All functions are assumed to be bounded and to have a compact support which, up to scaling can be fixed to be $[0,1]^p$. In this paper, we only consider $p = 1$. In the end, the general formulation of the problem is:

$$\text{maximize } f(x) \text{ for } x \in [0,1] \tag{1}$$

1.2 About the multi-armed bandit

As previously mentioned, a gradient-based approach is not likely to perform well on the considered functions. Thus, the idea is to use a multi-armed bandit with theoretically an infinite number of arms, sometimes called *continuum-armed bandit*. The algorithms progressively defines a sequence of evaluation points x_t and observes a reward $y_t = f(x_t) + \xi_t$ where ξ_t is a noise term. The choice of x_{t+1} depends on the sequence of $(y_{t'})_{t' \leq t}$ and is meant to converge to x^* while controlling the cumulative regret:

$$R_T = \sum_{t=1}^{T} f(x^*) - f(x_t) \tag{2}$$

In practice, for the algorithms considered, $x_t = \mathcal{U}(I_t)$. This means that x_t is pulled uniformly at random in an interval I_t that is chosen by the algorithm. The set of intervals defines a tree structure such that the set of leaves is a partition of $[0,1]$. In the end, the original problem is transformed into a particular case of regular multi-armed bandit. Without any prior information on the function to optimize, we only consider here the family of *dyadic trees*. As mentioned in [3], the dyadic tree on [0, 1) is the tree with root node [0, 1), and where each node [a, b) has children $[a, \frac{1}{2}(a+b), [\frac{1}{2}(a+b), b)$. An example of dyadic tree is displayed in 1.2.

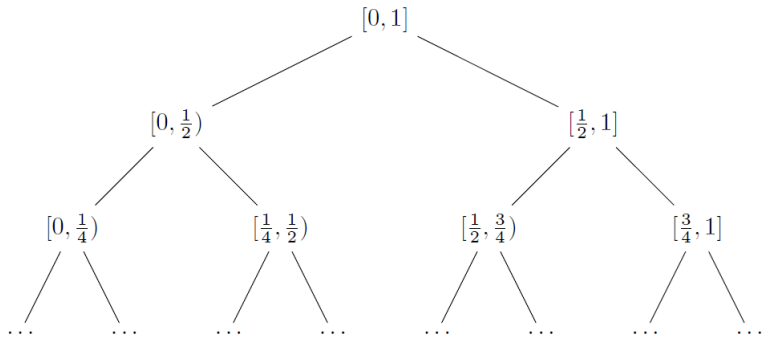

Figure 1: A dyadic tree from [3]

1.3 Notations

In the rest of this paper, we note:

- h, i the coordinates of a node in the dyadic tree such that h is the depth of the node and i its width
- $\widehat{\mu}_{h,i}$ the empirical estimate of the reward associated to the node h, i
- $n_{h,i}$ the number of times the node h, i has been hit during the evaluations up to the current time
- T_{max} the total number of evaluations of the function allowed or *budget*

2 Algorithms

In this section, we list the algorithms to be compared and briefly present their respective behaviors. The reader is welcome to consult the original papers for more detailed descriptions and proofs on theoretical performances. As previously mentioned, each algorithm defines a sequence of intervals I_t from which x_t is sampled. Typically, the selected interval maximizes a certain selection function of $(x_{t'}, y_{t'})_{t'<t}$ among a subset of considered intervals. The main differences in the following algorithms is the definition of the selection function, which is most of the time an upper bound on the expected reward of the arm and the choice of the considered intervals at each time step, which is sometimes called "expansion rule".

2.1 Hierarchical and Parallel Optimistic Optimization

This algorithm is called *Optimistic* because its idea is to sequentially building a tree and try the most promising children.

Upper bounds $B_{h,i} = min(U_{h,i}, max(B_{h+1,2i-1}, B_{h,2i}))$ where:

$$U_{h,i} = \widehat{\mu}_{h,i} + \nu\rho^h + \sqrt{\frac{2\log(T_{max})}{n_{h,i}}} \tag{3}$$

Update rule At each time step, the most promising child with respect to B is added to the tree.

Optimistic Optimization When there is no prior knowledge on the function to optimize, the tuning of the parameters ν and ρ of equation 3 can be difficult. Thus, the idea of this algorithm is to sequentially launch several HOOs that keep running in parallel in order to get the best parameters. Note that this algorithm can be run with other algorithms than HOO such as HCT.

2.2 High-Confidence Tree

This algorithm exists in two different versions: with correlated and uncorrelated feedback. For the purpose of performance comparison with other algorithms, we only consider the uncorrelated feedback, which is close to HOO algorithm. In order to be pulled, an arm must maximize B among the arms that have not been pulled enough yet. In order to reduce computational cost, U is refreshed only when t is a power of 2.

Upper bounds $B_{h,i} = min(U_{h,i}, max(B_{h+1,2i-1}, B_{h,2i}))$ where:

$$U_{h,i} = \widehat{\mu}_{h,i} + \nu\rho^h + \sqrt{\frac{c^2 \log(1/min(1, \frac{c_1\delta}{2^{\lfloor \log(t) \rfloor + 1}}))}{n_{h,i}}} \tag{4}$$

Update rule Only the leaves of the current covering tree that have not been pulled enough with respect to a certain threshold $\tau_h(t)$ are expanded.

2.3 Stochastic Simultaneous Optimistic Optimization

This algorithm is called *simultaneous* because it can perform at each time steps as many evaluations as the depth of its current covering tree. Indeed, the idea is that, in order not to rush into a local maximum, it is good to keep sampling from small depths until the estimation of the local reward is better known. Each node a its own evaluation budget k in order no to spend to much budget on the first nodes.

Upper bounds In order to be pulled, a leaf must both have a positive remaining individual budget and maximize B among the leaves of the same depth:

$$B_{h,i} = \widehat{\mu}_{h,i} + \sqrt{\frac{\log(\frac{T_{max}k}{\delta})}{2n_{h,i}}} \tag{5}$$

Update rule Once a node that maximizes B has used its entire budget, it is expanded.

2.4 Adaptive-Treed Bandits

This algorithm has a slightly different spirit than the others. The tree is not sequentially built but given as an input and the evaluations are performed among

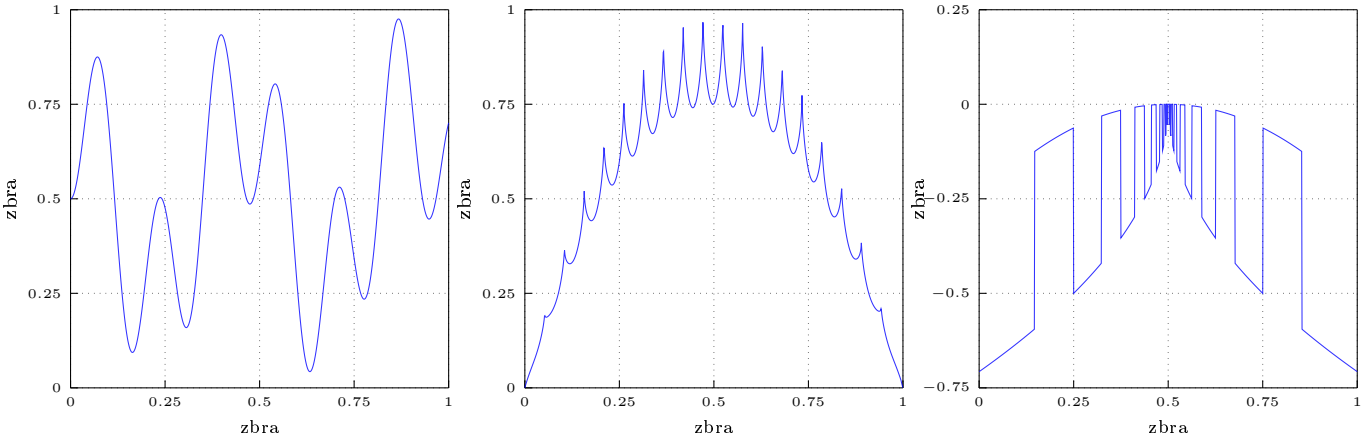

Figure 2: From the left to the right: Two-sine product, Garlang, Grill.

a set of *active boxes* or *active nodes* (which in dimension 1 are simply intervals). Knowing the entire tree allows to update the statistics of the children of active nodes each time an arm is pulled, while in the other algorithms, only the statistics of the parents could be updated.

Upper bounds At each time step, the arm pulled is chosen among the active nodes and maximizes:

$$B_{h,i} = \widehat{\mu}_{h,i} + (1+2\nu)r_{h,i} \tag{6}$$

where $r_{h,i} = 2\sqrt{\frac{\log[2^{h+1}(\tau+n_{h,i})]}{n_{h,i}}}$ is the confidence radius of the interval.

Update rule If an active node has a radius small enough compared to the ones of its children, it is removed and replaced by them.

3 Results

3.1 Experimental Setup

Objective functions We test the algorithms on different reference functions from [5] and [4]:

1. Two-sine product function: $f_1(x) = \frac{1}{2}(\sin(13x).\sin(27x)) + 0.5$.
2. Garland function: $f_2(x) = 4x(1-x).(\frac{3}{4} + \frac{1}{4}(1 - \sqrt{|\sin(60x)|}))$.
3. Grill function: $f_3(x) = s(\log_2(|x-0.5|).(\sqrt{|x-0.5|} - (x-0.5)^2) - \sqrt{|x-0.5|}$ where $s(x) = \mathbf{1}(x - \lfloor x \rfloor \in [0, 0.5])$.

The associated plots are displayed in 3.1.

Success rates and average cumulative regrets of the algorithms

Algorithm	f_1	f_2	f_3	R_1	R_2	R_3
HOO						
POO						
HCT						
StoSOO						
ATB						

Figure 3: These results are obtained for $\epsilon =$, $T =$ and $N =$.

Algorithms setup In order to compare the performances of the different algorithms, we set a desired precision ϵ and a total number of function evaluations T and a number of runs N. Then we compute for each and each algorithm run the best value returned $\widehat{x}^*$ and the cumulative regret defined in 2. The run is considered a success if $|\widehat{x}^* - x^*| \leq \epsilon$. The success rates are average cumulative regrets are displayed in 3.1. The tuning of the parameters specific to each algorithm is performed manually.

3.2 Analysis

A S'more random shit in the appendix

A.1 HOO

A.2 POO

A.3 SOO

A.4 HCT

A.5 ATB

References

[1] Mohammad Gheshlaghi Azar, Alessandro Lazaric, and Emma Brunskill. Online stochastic optimization under correlated bandit feedback. *arXiv preprint arXiv:1402.0562*, 2014.

[2] Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvari. X-armed bandits. *The Journal of Machine Learning Research*, 12:1655–1695, 2011.

[3] Adam D Bull. Adaptive-treed bandits. *arXiv preprint arXiv:1302.2489*, 2013.

[4] Jean-Bastien Grill, Michal Valko, and Rémi Munos. Black-box optimization of noisy functions with unknown smoothness. In *Neural Information Processing Systems*, 2015.

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

Algorithm 1 the hierarchical optimistic optimization algorithm

for n = 1 .. Nev **do**
  $P \leftarrow [0,1]$
  $h, i \leftarrow 0, 1$
  while $[h,i] \in \mathcal{T}$ **do**
    if $B_{h+1,2i-1} > B_{h+1,2i}$ **then**
      $h, i \leftarrow h+1, 2i-1$
    else if $B_{h+1,2i-1} < B_{h+1,2i}$ **then**
      $h, i \leftarrow h+1, 2i$
    else
      $Z \sim Ber(0.5)$
      $h, i \leftarrow h+1, 2i-Z$
    end if
    $P \leftarrow P \cup [h,i]$
  end while
  $H, I \leftarrow h, i$
  $X \sim \mathcal{U}(\mathcal{P}_{H,I})$
  reward: $Y = f(X)$
  $\mathcal{T} \leftarrow \mathcal{T} \cup [H,I]$
  for all $[h,i] \in P$ **do**
    $T_{h,i} \leftarrow T_{h,i} + 1$
    $\mu_{h,i} \leftarrow (1 - \frac{1}{T_{h,i}})\mu_{h,i} + \frac{Y}{T_{h,i}}$
  end for
  for all $[h,i] \in \mathcal{T}$ **do**
    $U_{h,i} \leftarrow \mu_{h,i} + \sqrt{\frac{2 \ln n}{T_{h,i}}} + \mu_1 \rho^h$
  end for
  $B_{H+1,2I-1}, B_{H+1,2I} \leftarrow +\infty, +\infty$
  $\mathcal{T}' \leftarrow \mathcal{T}$
  while $\mathcal{T}' \neq \{[0,1]\}$ **do**
    $[h,i] \leftarrow \text{Leaf}(\mathcal{T}')$
    $B_{h,i} \leftarrow \min\{U_{h,i}, B_{h+1,2i-1}, B_{h+1,2i}\}$
    $\mathcal{T}' \leftarrow \mathcal{T}' \{[h,i]\}$
  end while
end for

Algorithm 2 the hierarchical optimistic optimization algorithm

for n = 1 .. Nev **do**
  $P \leftarrow [0,1]$
  $h, i \leftarrow 0, 1$
  while $[h,i] \in \mathcal{T}$ **do**
    if $B_{h+1,2i-1} > B_{h+1,2i}$ **then**
      $h, i \leftarrow h+1, 2i-1$
    else if $B_{h+1,2i-1} < B_{h+1,2i}$ **then**
      $h, i \leftarrow h+1, 2i$
    else
      $Z \sim Ber(0.5)$
      $h, i \leftarrow h+1, 2i-Z$
    end if
    $P \leftarrow P \cup [h,i]$
  end while
  $H, I \leftarrow h, i$
  $X \sim \mathcal{U}(\mathcal{P}_{H,I})$
  reward: $Y = f(X)$
  $\mathcal{T} \leftarrow \mathcal{T} \cup [H,I]$
  for all $[h,i] \in P$ **do**
    $T_{h,i} \leftarrow T_{h,i} + 1$
    $\mu_{h,i} \leftarrow (1 - \frac{1}{T_{h,i}})\mu_{h,i} + \frac{Y}{T_{h,i}}$
  end for
  for all $[h,i] \in \mathcal{T}$ **do**
    $U_{h,i} \leftarrow \mu_{h,i} + \sqrt{\frac{2 \ln n}{T_{h,i}}} + \mu_1 \rho^h$
  end for
  $B_{H+1,2I-1}, B_{H+1,2I} \leftarrow +\infty, +\infty$
  $\mathcal{T}' \leftarrow \mathcal{T}$
  while $\mathcal{T}' \neq \{[0,1]\}$ **do**
    $[h,i] \leftarrow \text{Leaf}(\mathcal{T}')$
    $B_{h,i} \leftarrow \min\{U_{h,i}, B_{h+1,2i-1}, B_{h+1,2i}\}$
    $\mathcal{T}' \leftarrow \mathcal{T}'\{[h,i]\}$
  end while
end for

Algorithm 3 the hierarchical optimistic optimization algorithm

for n = 1 .. Nev **do**
  $P \leftarrow [0,1]$
  $h, i \leftarrow 0, 1$
  while $[h,i] \in \mathcal{T}$ **do**
    if $B_{h+1,2i-1} > B_{h+1,2i}$ **then**
      $h, i \leftarrow h+1, 2i-1$
    else if $B_{h+1,2i-1} < B_{h+1,2i}$ **then**
      $h, i \leftarrow h+1, 2i$
    else
      $Z \sim Ber(0.5)$
      $h, i \leftarrow h+1, 2i-Z$
    end if
    $P \leftarrow P \cup [h,i]$
  end while
  $H, I \leftarrow h, i$
  $X \sim \mathcal{U}(\mathcal{P}_{H,I})$
  reward: $Y = f(X)$
  $\mathcal{T} \leftarrow \mathcal{T} \cup [H,I]$
  for all $[h,i] \in P$ **do**
    $T_{h,i} \leftarrow T_{h,i} + 1$
    $\mu_{h,i} \leftarrow (1 - \frac{1}{T_{h,i}})\mu_{h,i} + \frac{Y}{T_{h,i}}$
  end for
  for all $[h,i] \in \mathcal{T}$ **do**
    $U_{h,i} \leftarrow \mu_{h,i} + \sqrt{\frac{2 \ln n}{T_{h,i}}} + \mu_1 \rho^h$
  end for
  $B_{H+1,2I-1}, B_{H+1,2I} \leftarrow +\infty, +\infty$
  $\mathcal{T}' \leftarrow \mathcal{T}$
  while $\mathcal{T}' \neq \{[0,1]\}$ **do**
    $[h,i] \leftarrow \text{Leaf}(\mathcal{T}')$
    $B_{h,i} \leftarrow \min\{U_{h,i}, B_{h+1,2i-1}, B_{h+1,2i}\}$
    $\mathcal{T}' \leftarrow \mathcal{T}' \{[h,i]\}$
  end while
end for

Algorithm 4 the hierarchical optimistic optimization algorithm

for n = 1 .. Nev **do**
  $P \leftarrow [0,1]$
  $h, i \leftarrow 0, 1$
  while $[h,i] \in \mathcal{T}$ **do**
    if $B_{h+1,2i-1} > B_{h+1,2i}$ **then**
      $h, i \leftarrow h+1, 2i-1$
    else if $B_{h+1,2i-1} < B_{h+1,2i}$ **then**
      $h, i \leftarrow h+1, 2i$
    else
      $Z \sim Ber(0.5)$
      $h, i \leftarrow h+1, 2i-Z$
    end if
    $P \leftarrow P \cup [h,i]$
  end while
  $H, I \leftarrow h, i$
  $X \sim \mathcal{U}(\mathcal{P}_{H,I})$
  reward: $Y = f(X)$
  $\mathcal{T} \leftarrow \mathcal{T} \cup [H,I]$
  for all $[h,i] \in P$ **do**
    $T_{h,i} \leftarrow T_{h,i} + 1$
    $\mu_{h,i} \leftarrow (1 - \frac{1}{T_{h,i}})\mu_{h,i} + \frac{Y}{T_{h,i}}$
  end for
  for all $[h,i] \in \mathcal{T}$ **do**
    $U_{h,i} \leftarrow \mu_{h,i} + \sqrt{\frac{2 \ln n}{T_{h,i}}} + \mu_1 \rho^h$
  end for
  $B_{H+1,2I-1}, B_{H+1,2I} \leftarrow +\infty, +\infty$
  $\mathcal{T}' \leftarrow \mathcal{T}$
  while $\mathcal{T}' \neq \{[0,1]\}$ **do**
    $[h,i] \leftarrow \text{Leaf}(\mathcal{T}')$
    $B_{h,i} \leftarrow \min\{U_{h,i}, B_{h+1,2i-1}, B_{h+1,2i}\}$
    $\mathcal{T}' \leftarrow \mathcal{T}'\{[h,i]\}$
  end while
end for