PARAMETER TUNING BY SIMPLE REGRET ALGORITHMS AND MULTIPLE SIMULTANEOUS HYPOTHESIS TESTING

Amine Bourki**, Matthieu Coulm**, Philippe Rolet*, Olivier Teytaud* and Paul Vayssière**
*TAO, Inria, Umr CNRS 8623, Univ. Paris-Sud, 91405 Orsay, France
**EPITA, 16 rue Voltaire, 94270 Le Kremlin-Bicêtre, France

Keywords: Simple regret, Automatic parameter tuning, Monte-Carlo tree search.

Abstract: "Simple regret" algorithms are designed for noisy optimization in unstructured domains. In particular, this literature has shown that the uniform algorithm is indeed optimal asymptotically and suboptimal non-asymptotically. We investigate theoretically and experimentally the application of these algorithms, for automatic parameter tuning, in particular from the point of view of the number of samples required for "uniform" to be relevant and from the point of view of statistical guarantees. We see that for moderate numbers of arms, the possible improvement in terms of computational power required for statistical validation can't be more than linear as a function of the number of arms and provide a simple rule to check if the simple uniform algorithm (trivially parallel) is relevant. Our experiments are performed on the tuning of a Monte-Carlo Tree Search algorithm, a great recent tool for high-dimensional planning with particularly impressive results for difficult games and in particular the game of Go.

1 INTRODUCTION

We consider the automatic tuning of new modules. It is quite usual, in artificial intelligence, to design a module, for which there are several free parameters. This is natural in supervised learning, optimization (Nannen and Eiben, 2007b; Nannen and Eiben, 2007a), control (Lee et al., 2009; Chaslot et al., 2009). We will here consider the particular case of Monte-Carlo Tree Search (Chaslot et al., 2006; Coulom, 2006; Kocsis and Szepesvari, 2006; Lee et al., 2009).

Consider a program, in which a new module with parameter $\theta \in \{1,\dots,K\}$ has been added. In the bandit literature, $\{1,\dots,K\}$ is referred to as the set of arms. Then, we're looking for the best parameter $\theta \in \{1,\dots,K\}$ for some performance criterion; the performance criterion $L(\theta)$ is stochastic. We have a finite time budget T (also termed horizon), we can have access to T realizations of $L(\theta_1), L_(\theta_2), \dots, L(\theta_T)$ and we then choose some $\hat{\theta}$. The game is as follows:

- The algorithm chooses $\theta_1 \in \{1,\dots,K\}$.
- The algorithm gets a realization r_1 distributed as $L(\theta_1)$.
- The algorithm chooses $\theta_2 \in \{1,\dots,K\}$.
- The algorithm gets a realization r_2 distributed as $L(\theta_2)$.
- ...
- The algorithm chooses $\theta_T \in \{1,\dots,K\}$.
- The algorithm gets a realization r_T distributed as $L(\theta_T)$.
- The algorithm chooses $\hat{\theta}$.
- The loss is $r_T = \max_\theta \mathbb{E}L(\theta) - \mathbb{E}L(\hat{\theta})$.

The performance measure is the simple regret(Bubeck et al., 2009), i.e. $r_T = \max_\theta \mathbb{E}L(\theta) - \mathbb{E}L(\hat{\theta})$, and we want to minimize it. Then main difference with noisy nonlinear optimization is that we don't use any structure on the domain.

We point out the link with No Free Lunch theorems (NFL (Wolpert and Macready, 1997)), which claim that all algorithms are equivalent when no prior knowledge can be explored. Yet, there are some differences in the framework: NFL considers deterministic optimization, in which testing several times the same point is meaningless. We here consider noisy optimization, with a small search space: all the difficulty is in the statistical validation, for choosing which points in the search space should be tested more intensively.

Bourki A., Coulm M., Rolet P., Teytaud O. and Vayssière P. (2010).
PARAMETER TUNING BY SIMPLE REGRET ALGORITHMS AND MULTIPLE SIMULTANEOUS HYPOTHESIS TESTING.
In *Proceedings of the 7th International Conference on Informatics in Control, Automation and Robotics*, pages 169-173
DOI: 10.5220/0002949901690173

Useful notations:

- $\#E$ is the cardinal of the set E;
- $N_t(i)$ is the number of times the parameter i has been tested at iteration t, i.e.

$$N_t(i) = \#\{j \leq t; \theta_j = i\}.$$

- $\hat{L}_t(i)$ is the average reward for parameter i at iteration t, i.e.

$$\hat{L}_t(i) \frac{1}{N_t(i)} \sum_{j \leq t; \theta_j = i} r_j.$$

(well defined if $N_t(i) > 0$)

Section 2 recalls the terminology of simple regret and discusses the relevance for Automatic Parameter Tuning (APT). Section 3 mathematically considers the statistical validation, which was not yet, to the best of our knowledge, considered for simple regret algorithms; we will in particular show that the dependency of the computational cost as a function of the number of tested parameter values is at best linear, and therefore it is not possible to do better than this linear improvement in terms of statistical validation - we will then switch to experimental analysis, and we'll show that the improvement is indeed improved by far less than a linear factor in our real world setting (section 4).

2 SIMPLE REGRET: STATE OF THE ART AND RELEVANCE FOR AUTOMATIC PARAMETER TUNING

We consider the case in which $L(\theta)$ is, for all θ, a Bernoulli distribution. (Bubeck et al., 2009) states that (i) the naive algorithm distributing θ_i uniformly among the possible parameters, i.e. $\theta_i = \text{mod}(i,K)+1$ with mod the modulo operator, with $\hat{\theta} = \arg\max_i \hat{L}(i)$, has simple regret

$$\mathbb{E}r_T = O(\exp(-c \cdot T)) \quad (1)$$

for some constant c depending on the Bernoulli parameters (more precisely, on the difference between the parameters of the best arm and of the other arms). This is for $\hat{\theta}$ maximizing the empirical reward, *i.e.*

$$\hat{\theta} \in \arg\min_{\theta} \hat{L}_T(\theta)$$

and this is proved optimal.

If we consider distribution-free bounds (i.e. for a fixed T, we consider the supremum of $\mathbb{E}r_T$ for all Bernoulli parameters), then (Bubeck et al., 2009) shows that, with the same algorithm,

$$\sup_{distribution} \mathbb{E}r_T = O(\sqrt{K \log K/T}), \quad (2)$$

where the constant in the $O(.)$ is a universal constant; Eq. 2 is tight within logarithmic factors of K; there's a lower bound for all algorithms of the form.

$$\sup_{distribution} \mathbb{E}r_T = \Omega(\sqrt{K/T}).$$

Importantly, the best known upper bounds for variants of UCB(Auer et al., 2002) are significantly worse than Eq. 1 (the simple regret is then only polynomially decreasing) and significantly worse than Eq. 2 (by a logarithmic factor of T) - see (Bubeck et al., 2009) for more on this.

However, it is clearly shown also in (Bubeck et al., 2009) that for small values of T, using a variant of UCB for choosing the θ_i and $\hat{\theta}$ is indeed much better than uniform sampling. The variant of UCB is as follows, for some parameter $\alpha > 1$:

$$\hat{\Theta}_t = \arg\max_i N_t(i).$$

$$\Theta_i = mod(i,K)+1 \text{ if } i \leq K$$

$$\Theta_i = \arg\max_i \hat{L}_t(i) + \sqrt{\alpha \log(t-1)/N_{t-1}(i)} \text{ otherwise.}$$

Simple regret is a natural criterion when working on automatic parameter tuning. However, the theoretical investigations on simple regret did not answer the following question: how can we validate an arm selected by a simple regret algorithm when a baseline is present ? In usual cases, for the application to parameter tuning, we know the score before a modification, and then we tune the parameters of the optimization: we don't only tune, we validate the tuned modification; this question is nonetheless central in many applications in particular when modifications are included automatically by the tuning algorithm (Nannen and Eiben, 2007b; Nannen and Eiben, 2007a; Hoock and Teytaud, 2010). We'll see in next sections that the naive solution, consisting in testing separately each arm, is not so far from being optimal.

3 MULTIPLE SIMULTANEOUS HYPOTHESIS TESTING IN AUTOMATIC PARAMETER TUNING

As pointed out above, a goal different from minimizing the simple regret consists in finding a good arm could be (i) finding a good arm if any (ii)

avoiding selecting a bad arm if there's no good arm (no arm which outperforms the baseline). We'll briefly show how to apply Multiple Simultaneous Hypothesis Testing (MSHT), and in particular its simplest and most well known variant termed the Bonferroni correction, to Automatic Parameter Tuning. MSHT(Holm, 1979; Hsu, 1996) is very classical in neuro-imagery(Pantazis et al., 2005), bioinformatics, tuning of optimizers(Nannen and Eiben, 2007b; Nannen and Eiben, 2007a).

MSHT consists in statistically testing several hypothesis in same time: for example, when 100 sets of parameters are tested simultaneously, then, whenever each set is tested with confidence 95%, and whenever all sets of parameters have no impact on the result, then with probability $1-(1-0.05^1 00) \simeq 99.4\%$ at least one set of parameters will be validated. MSHT is aimed at correcting this effect, so that taking into account the multiplicity of tests we can have modified tests so that the overall risk remains lower than 5%.

Assume that we expect arms with standard deviation σ (we'll see that for our applications, σ is usually nearly known in advance; it can also be estimated dynamically during the process). Then, the standard Gaussian approximation says that with probability 90%[1], the difference between $\hat{L}_t(\theta)$ and $L_t(\theta)$ for arm θ is lower than $1.645\sigma/\sqrt{N_t(i)}$:
with probability 90%,

$$|\hat{L}_t(\theta) - L_t(\theta)| \leq 1.645\sigma/\sqrt{N_t(i)}. \quad (3)$$

The constant 1.645 directly corresponds to the Gaussian probability distribution (the precise value is $\Phi^{-1}((1+0.9)/2) = 1.645$); a Gaussian standard distribution is ≤ 1.645 in absolute value with probability 90%. If we consider several tests simultaneously, i.e. we consider K arms, then Eq. 3 becomes Eq. 4:
with probability 90%,

$$\forall \theta \in \{1,2,\ldots K\} |\hat{L}_t(\theta) - L_t(\theta)| \leq t_K \sigma/\sqrt{N_t(i)} \quad (4)$$

where, with the so-called Bonferroni correction, $t_K = -\Phi^{-1}(0.05/K)$ where Φ is the normal cumulative distribution function[2]. This is usually estimated with

$$\frac{\exp(-t_K^2)}{t_K\sqrt{2\pi}} = 0.05/K \quad (5)$$

and therefore if we expect improvements of size δ, we can only validate a modification with confidence 90% with n experiments per arm if t_K solving Eq. 5 verifies $t_K\sigma/\sqrt{n} \leq \delta$; a succinct equation for this is

$$s = \delta\sqrt{n}/\sigma \quad (6)$$

$$\frac{\exp(-s^2)}{s\sqrt{2\pi}} \leq 0.05/K \quad (7)$$

This shows that for other quantities fixed, n has a logarithmic dependency as a function of K.

A numerical application for $\delta = 0.02$, $K = 49$ and $\sigma = \frac{1}{2}$ is

$$s = 0.04\sqrt{n}, \quad \frac{\exp(-s^2)}{s\sqrt{2\pi}} = 0.05/49.$$

which implies $n \geq 3219$; this implies that for our confidence interval, we require 3219 runs per arm (i.e. $\inf_\theta N_T(\theta) \geq 3219$). We'll see that this number is consistent with our numerical experiments later. Interestingly, with only one arm, *i.e.* $K = 1$, we get $n \geq 1117$; this is not so much better, and suggests that whatever we do, it will be difficult to get significant results with subtle techniques for pruning the set of arms: if there is only one arm, we can only divide the computational cost for this arm by $O(\log(K))$. In case of perfect pruning, n is also naturally multiplied by K (as all the computational power is spent on only one arm instead of K arms); this provides an additional linear factor, leading to a roughly linear improvement in terms of computational power as a function of the number of arms, in case of perfect pruning.

Bernstein Races

This paper is devoted to the use of simple regret algorithms to APT, compared to the most simple APT algorithm, namely uniform sampling (which is known asymptotically optimal for simple regret); Bernstein races are therefore beyond the scope of this paper. Nonetheless, as our results emphasize the success of uniform sampling (at least in some cases), we briefly discuss Bernstein races. In (Mnih et al., 2008; Hoock and Teytaud, 2010), Bernstein races were considered as tools for discarding statistically bad arms: this is equivalent to *Uniform*, except that tests as above are applied periodically, and statistically bad arms are discarded. This discards arms earlier than the uniform algorithm above which just checks the result at the end, but increases the quantity K involved in tests (as in Eqs. 6 and 7), *even if no arm can be rejected*. The fact that testing arms for discarding on the fly has a cost, whenever no arm is discarded, might be surprising at first view - it is a known effect that when multiple tests are performed, then the number of samples required for a same confidence rate on the result is much higher. This approach can therefore at

[1] The constant 90% is arbitrary; it means that we decide that results are guaranteed within risk 10%.

[2] Note that a tighter formula is $t_K = -\Phi^{-1}(1-(1-0.05)^K)$; this holds thanks to independence of the different arms.

most divide the computational power by $K\log(K)$ before an arm is validated, and the computational power is indeed increases when no early discarding is possible. Nonetheless, this sound approach is probably the best candidate when the visualization is not crucial - *Uniform* can provide nice graphs as shown in the experimental section from http://hal.inria.fr/inria-00467796/.

4 EXPERIMENTAL VALIDATION: THE TUNING OF MOGO

Due to length constraints, the experimental section is reported to http://hal.inria.fr/inria-00467796/.

5 DISCUSSION

We have surveyed simple regret algorithms. They are noisy optimization algorithms, and they don't assume any structure on the domain. We compared *Uniform* (known as optimal for sufficiently large horizon, i.e. sufficiently large time budget) and *UCB* for automatic parameter tuning. Our results are as follows:

- **MSHT (even the Simple Bonferroni Correction) is relevant for Automatic Parameter Tuning.** It predicts how many computational power is required for *Uniform*; when the number K of tested sets of parameters depends on a discretization, MSHT can be applied for choosing the grain of the discretization. The *Uniform* approach combined with MSHT by Bonferroni correction might be the best approach when the computational power is large in front of K, thanks to its statistical guarantees, the easy visualization, the optimality in terms of simple regret. However, non-asymptotically, it is not optimal and the rule below is here for deciding the relevance of *Uniform* when K and T are known.
- **Choosing between the Naive Solution (*Uniform* sampling) and Sophisticated Algorithms.** The naive *Uniform* algorithm is provably optimal for large values of the horizon. We propose the following simple rule for choosing if it is worth using something else than the simple uniform sampling:
  - Compute
    $$s = \delta\sqrt{n}/\sigma.$$
    where
    * δ is the amplitude of the expected change in reward;
    * σ is the expected standard deviation;
    * n is the number of experiments you can perform for each arm with your computational power.
  - Test if $\frac{\exp(-s^2)}{s\sqrt{2\pi}} \leq 0.05/K$ where K is the number of arms.
  - If yes, then uniform sampling is ok. Otherwise, you can try UCB-like algorithms (but, in that case, there's no statistical guarantee), or Bernstein races. At first view, our choice would be Bernstein races for an implementation aimed at automatically tuning and validating several modifications (as in (Hoock and Teytaud, 2010)) as soon as conditions above are not met by the computational power available; if the computational power available is strong enough, *Uniform* has nice visualization properties.
  - **What if Uniform Algorithms can't do it?** If K is not large, nothing can be much better than uniform; at most the required horizon can be divided by $K\log(K)$. What if K is large ? *UCB* is probably much better when K is large. A drawback is that it does not include any statistical validation, and is not trivially parallel; therefore, classical algorithms derived far from the field of simple regret, like Bernstein races(Bernstein, 1924), might be more relevant. Bernstein races are close to the *Uniform* algorithm, except that they discard arms as early as possible (Mnih et al., 2008; Hoock and Teytaud, 2010) by performing statistical tests on the fly. A drawback is that Bernstein races do not provide a complete picture of the search space and of the fitness landscape as *Uniform*; also, if no arm can be discarded early, the horizon required for statistical validation is bigger than for *Uniform* as tests are performed during the run. Yet, Bernstein races might be the most elegant tool for doing better than *Uniform* as they adapt to various frameworks(Hoock and Teytaud, 2010): when many arms can be discarded easily, they will save up a lot of computational power.
- **Results on our Application to MCTS.** For the specific application, the results were significant but moderate; however, it can be pointed out that many handcrafted modifications around Monte-Carlo Tree Search provide such small improvements of a few percents each. Moreover, as shown in (Hoock and Teytaud, 2010), improvements performed automatically by bandits can be applied incrementally, leading to huge improve-

ments once they are cumulated.

- **Comparing Recommendation Techniques: Most Played Arm is Better.** The empirically best arm and the most played arm in UCB are usually the same (this is not the case for various other bandit algorithms), and are much better than the "empirical distribution of play" technique. The most played arm and the empirical distribution of play obviously do not make sense for $Uniform$. Please note that it is known in other settings (see (Wang and Gelly, 2007)) that the most played arm is better(Wang and Gelly, 2007). MPA is seemingly a reliable tool in many settings.

A next experimental step is the automatic use of the algorithm for more parameters, or e.g. by extending automatically the neural network used in the Monte-Carlo Tree Search so that it takes into account more inputs: instead of performing one big modification, apply several modifications the one after the other, and tune them sequentially so that all the modifications can be visualized and checked independently. The fact that the small constant 0.1 was better in UCB is consistant with the known fact that tuned version of UCB (with p related to the variance) provides better results; using tuned-UCB might provide further improvements(Audibert et al., 2006).

ACKNOWLEDGEMENTS

This work has been supported by French National Research Agency (ANR) through COSINUS program (project EXPLO-RA No ANR-08-COSI-004), and grant No. ANR-08-COSI-007-12 (OMD project). It benefited from the help of Grid5000 for parallel experiments.

REFERENCES

Audibert, J.-Y., Munos, R., and Szepesvari, C. (2006). Use of variance estimation in the multi-armed bandit problem. In *NIPS 2006 Workshop on On-line Trading of Exploration and Exploitation*.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2/3):235–256.

Bernstein, S. (1924). On a modification of chebyshev's inequality and of the error formula of laplace. *Original publication: Ann. Sci. Inst. Sav. Ukraine, Sect. Math. 1*, 3(1):38–49.

Bubeck, S., Munos, R., and Stoltz, G. (2009). Pure exploration in multi-armed bandits problems. In *ALT*, pages 23–37.

Chaslot, G., Hoock, J.-B., Teytaud, F., and Teytaud, O. (2009). On the huge benefit of quasi-random mutations for multimodal optimization with application to grid-based tuning of neurocontrollers. In *ESANN*, Bruges Belgium.

Chaslot, G., Saito, J.-T., Bouzy, B., Uiterwijk, J. W. H. M., and van den Herik, H. J. (2006). Monte-Carlo Strategies for Computer Go. In Schobbens, P.-Y., Vanhoof, W., and Schwanen, G., editors, *Proceedings of the 18th BeNeLux Conference on Artificial Intelligence, Namur, Belgium*, pages 83–91.

Coulom, R. (2006). Efficient selectivity and backup operators in monte-carlo tree search. *In P. Ciancarini and H. J. van den Herik, editors, Proceedings of the 5th International Conference on Computers and Games, Turin, Italy*.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. scand. j. statistic., 6:65-70.

Hoock, J.-B. and Teytaud, O. (2010). Bandit-based genetic programming. In *Accepted in EuroGP 2010*, LLNCS. Springer.

Hsu, J. (1996). Multiple comparisons, theory and methods, chapman & hall/crc.

Kocsis, L. and Szepesvari, C. (2006). Bandit based monte-carlo planning. In *15th European Conference on Machine Learning (ECML)*, pages 282–293.

Lee, C.-S., Wang, M.-H., Chaslot, G., Hoock, J.-B., Rimmel, A., Teytaud, O., Tsai, S.-R., Hsu, S.-C., and Hong, T.-P. (2009). The Computational Intelligence of MoGo Revealed in Taiwan's Computer Go Tournaments. *IEEE Transactions on Computational Intelligence and AI in games*.

Mnih, V., Szepesvári, C., and Audibert, J.-Y. (2008). Empirical Bernstein stopping. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 672–679, New York, NY, USA. ACM.

Nannen, V. and Eiben, A. E. (2007a). Relevance estimation and value calibration of evolutionary algorithm par ameters. In *International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 975–980.

Nannen, V. and Eiben, A. E. (2007b). Variance reduction in meta-eda. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 627–627, New York, NY, USA. ACM.

Pantazis, D., Nichols, T. E., Baillet, S., and Leahy, R. (2005). A comparison of random field theory and permutation methods for the statistical analysis of MEG data. *Neuroimage*, 25:355–368.

Wang, Y. and Gelly, S. (2007). Modifications of UCT and sequence-like simulations for Monte-Carlo Go. In *IEEE Symposium on Computational Intelligence and Games, Honolulu, Hawaii*, pages 175–182.

Wolpert, D. and Macready, W. (1997). No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.