Aggregation of Multi-Armed Bandits Learning Algorithms for Opportunistic Spectrum Access

Lilian Besson[1,2], Emilie Kaufmann[2] and Christophe Moy[3]

[1] CentraleSupélec, IETR–SCEE, Cesson-Sévigné, France, Lilian.Besson@CentraleSupelec.fr
[2] CNRS & CRIStAL, Univ. Lille 1, Inria SequeL, Lille, France, Emilie.Kaufmann@Univ-Lille1.fr
[3] University Rennes, CNRS, IETR – UMR 6164, Rennes, France, Christophe.Moy@Univ-Rennes1.fr

***Abstract*—Multi-armed bandit algorithms have been recently studied and evaluated for Cognitive Radio (CR), especially in the context of Opportunistic Spectrum Access (OSA). Several solutions have been explored based on various models, but it is hard to exactly predict which could be the best for real-world conditions at every instants. Hence, expert aggregation algorithms can be useful to select *on the run* the best algorithm for a specific situation. Aggregation algorithms, such as Exp4 dating back from 2002, have never been used for OSA learning, and we show that it appears empirically sub-efficient when applied to simple stochastic problems. In this article, we present an improved variant, called *Aggregator*. For synthetic OSA problems modeled as Multi-Armed Bandit (MAB) problems, simulation results are presented to demonstrate its empirical efficiency. We combine classical algorithms, such as Thompson sampling, Upper-Confidence Bounds algorithms (UCB and variants), and Bayesian or Kullback-Leibler UCB. Our algorithm offers good performance compared to state-of-the-art algorithms (Exp4, CORRAL or LearnExp), and appears as a robust approach to select on the run the best algorithm for any stochastic MAB problem, being more realistic to real-world radio settings than any tuning-based approach.**

***Index Terms*—cognitive radio, learning theory, robust aggregation algorithms, multi-armed bandits, reinforcement learning.**

I. Introduction

Cognitive Radio (CR), introduced in 1999 [1], states that a radio, by collecting information about its environment, can dynamically reconfigure itself in order to improve its functionality regarding various metrics. One of the main direction of research, called Dynamic Spectrum Access [2], is focused on the spectrum access when devices reconfigure themselves by simply changing the frequency of their wireless communication. The model of Opportunistic Spectrum Access (OSA) for CR considers one Secondary User (SU) trying to use a licensed radio network occupied by Primary Users (PU). The network usage from the PU determines the availability patterns of the radio channels, and the goal of the SU is to communicate as efficiently as possible, without interfering with the PU. Thus at each step, a SU first senses one channel, and only transmits if this channel is unoccupied by a PU.

This work is supported by the French National Research Agency (ANR) with the project BADASS, (N ANR-16-CE40-0002), the CNRS with the project PEPS8 BIO, the French Ministry of Research (MENESR) and ENS Paris-Saclay.

A common simple model in the literature is to describe the PU impact on the availability of the K channels in the following way: channels are independent and identically distributed (*i.i.d.*), and their qualities follow parametric distributions, *e.g.*, Bernoulli of means $\mu_1, \dots, \mu_K \in [0,1]$ for availabilities when dealing with binary sensing feedback. The SU has to select the best expected channel each time to maximize its throughput: if successful communications are seen as rewards, the SU has to maximize its cumulative rewards, as in the Multi-Armed Bandit (MAB) problem [3], [4].

MAB learning algorithms have been shown to be useful for the OSA setting [5], [6], and UCB algorithms and other variants (e.g., kl-UCBor Bayes-UCB, [7], [8], [9], [10]) have been successfully applied to both numerical and physically simulated CR problems [11]. The performance of such learning algorithm $\mathcal{A}$ can be measured by different criteria. For example, it is common in the bandit literature to study the regret [10] $\widetilde{R_T^{\mathcal{A}}} = \mu^* T - \sum_{t=1}^{T} r(t)$ which compares the loss in rewards between the algorithm $\mathcal{A}$ and the full-knowledge strategy which always picks the best arm, *i.e.*, the most available of mean μ^*. Good algorithms are expected to have slow-growing expected regret, but other criterion include the best arm pull frequency, or the throughput of the SU.

Many different learning algorithms have been proposed by the machine learning community, and most of them depend on several parameters, for instance α for UCB, the prior for Thompson sampling or BayesUCB, the kl function for kl-UCB etc. Every time a new MAB algorithm $\mathcal{A}$ is introduced, it is compared and benchmarked on some bandit instance, parameterized by $\boldsymbol{\mu} = (\mu_1, \dots, \mu_K)$, usually by focusing on its expected regret $R_T = \mathbb{E}_{\boldsymbol{\mu}}[\widetilde{R_T}]$. For a known and specific instance, simulations help to select the best algorithm in a pool of algorithms. But when one wants to tackle an *unknown* real-world problem, one expects to be efficient against *any* problem, of any kind, size and complexity: ideally one would like to use an algorithm that can be applied identically against any problem. To choose the best algorithm, two approaches can be followed: either extensive benchmarks are done beforehand – if this is possible – to select the algorithm and its optimal parameters, or an adaptive algorithm is used to learn *on the fly* its parameters. We present a simple adaptive solution, that aggregates several learning algorithms in parallel and

adaptively chooses which one to trust the most.

This paper is organized as follows: our OSA model is described in Section II, and MAB learning algorithms are briefly presented in Section III. We explain in Section IV how to combine such algorithms for aggregation. Our proposed algorithm, called **Aggregator**, is detailed in Section IV-A, with numerical experiments presented in Section V, comparing the regret of several algorithms against different aggregation algorithms. Theoretical guarantees are shortly discussed in Section VI, and Section VII concludes.

II. OSA Model for Cognitive Radio

We consider $K \geq 1$ radio channels, also called arms, of different characteristics, unknown to the user. The radio protocol is slotted in both time and frequency, meaning that at each time step $t \in \mathbb{N}$, the Secondary User (SU) *tries to* communicate in a channel $A(t) \in \{1, \dots, K\}$. In the OSA model, the SU first senses one channel k at a time, and can use it to communicate *only* if it was sensed free from any PU (they have full priority over the SU).

In the stochastic model considered in this paper, after choosing the arm k, it is assumed that the sensing provides a *reward* $r_k(t)$, randomly drawn from a certain distribution depending on the arm index. Rewards are assumed to be bounded in $[0, 1]$, and generally they follow one-parameter exponential families. We present our algorithm by restricting to Bernoulli distributions[1], for sake of simplicity, meaning that arm k has a parameter $\mu_k \in [0, 1]$ and rewards are drawn from $B(\mu_k)$, $r_k(t) \sim B(\mu_k)$, which can be simply interpreted by the SU: it is 1 if the channel k is not used by any PU during the time slot t, and is 0 otherwise. Aggregation algorithms usually deal with losses rather rewards [12], so we also introduce the quantity $\ell_k(t) := 1 - r_k(t) \in [0, 1]$.

III. Classical MAB Algorithms : UCB, kl-UCB, TS

An algorithm $\mathcal{A}$ has to maximize its cumulative rewards, by choosing the arm $A(t) \in \{1, \dots, K\}$ at time t, or, equivalently, to minimize its pseudo-regret $\widetilde{R_T^{\mathcal{A}}}$ is defined as

$$\widetilde{R_T^{\mathcal{A}}} := \sum_{t=1}^{T} \big(\mu^* - r_{A(t)}(t)\big) = T\mu^* - \sum_{t=1}^{T} r_{A(t)}(t), \quad (1)$$

where μ^* is the mean of the best arm: $\mu^* = \max_{k=1}^{k} \mu_k = \max_{k=1}^{k} \mathbb{E}_{\boldsymbol{\mu}}[r_k(t)]$, for an expectation taken on the arm distributions. This pseudo-regret is random so we prefer to focus on the *expected* regret, $R_T^{\mathcal{A}} := \mathbb{E}_{\boldsymbol{\mu}}[\widetilde{R_T^{\mathcal{A}}}]$.

The UCB algorithm [7] selects the arm with highest index, where each index is an Upper Confidence Bound on the unknown mean, computed as the sum of the empirical mean of each arm $\hat{\mu_k}(t) = X_k(t)/N_k(t)$ (if $X_k(t) := \sum_{t=1}^{T} r_{A(t)}(t)\mathbb{1}(A(t) = k)$ and $N_k(t) := \sum_{t=1}^{T} \mathbb{1}(A(t) = k)$), and an exploration term defined by $\sqrt{\alpha \log(t)/N_k(t)}$. $\alpha = 1/2$ is known to yield logarithmic regret on all problems, but on some specific instance $\boldsymbol{\mu}$ a better value of α may be found empirically [10].

The kl-UCB algorithm is similar, but instead it uses a Kullback-Leibler divergence function to compute a statistically better UCB [8]. As a different KL function exists for each different exponential family, this algorithm also requires a prior knowledge of the problem to be efficient.

The Thompson sampling (TS) [13] algorithm is Bayesian: it maintains a posterior distribution on each means (*e.g.*, Beta posteriors for Bernoulli arms), updated after each observation, and chooses an arm by sampling a random mean from each posterior and playing the arm with highest mean. The posterior distribution has to be chosen according to the exponential family as the conjugated posterior.

Both UCB, kl-UCB and TS have been proved to have logarithmic regrets [14], [15], [16], meaning that $R_T^{\mathcal{A}} = \mathcal{O}(\log T)$, in Bernoulli bandit problems and also under more general assumptions. The constant in the big-$\mathcal{O}$ is important, and [4] showed that in this setting, the regret of any (uniformly efficient) algorithm is at least $C(\mu) \log T$ when T is large, for a constant $C(\mu)$ depending only on the problem instance $\boldsymbol{\mu}$: $C(\boldsymbol{\mu}) = \sum_{\mu_k \neq \mu^*} (\mu^* - \mu_k)/\mathrm{kl}(\mu_k, \mu^*)$ (with a unique best arm), where $\mathrm{kl}(x, b)$ is the binary KL divergence between two Bernoulli distributions of parameters x and y.

IV. Aggregating Bandit Algorithms

We assume to have $N \geq 2$ MAB algorithms, $\mathcal{A}_1, \dots, \mathcal{A}_N$, and let $\mathcal{A}_{\mathrm{aggr}}$ be an aggregation algorithm, which runs the N algorithms in parallel (with the same slotted time), and use them to choose its channels based on a voting from their N decisions. $\mathcal{A}_{\mathrm{aggr}}$ depends on a pool of algorithms and a set of parameters. We would like that $\mathcal{A}_{\mathrm{aggr}}$ performs almost as well as the best of the $\mathcal{A}_a$, with a good choice of its parameters, independently of the MAB problem. Ideally $\mathcal{A}_{\mathrm{aggr}}$ should perform similarly to the best of the $\mathcal{A}_a$. To simplify the presentation, we only aggregate bandit algorithms that give deterministic recommendations: one arm is chosen with probability 1 and the others with probability 0. However, both **Exp4** and **Aggregator** can be adapted to aggregate randomized bandit algorithms, *i.e.*, algorithms that output a probability distribution ξ_t over the arms $\{1, \dots, K\}$ at each time step, and draw the next selected arm according to this distribution.

The aggregation algorithm maintains a probability distribution π^t on the N algorithms $\mathcal{A}_a$, starting from a uniform distribution: π_a^t is the probability of trusting the decision made by algorithm $\mathcal{A}_a$ at time t. $\mathcal{A}_{\mathrm{aggr}}$ then simply performs a weighted vote on its algorithms: it decides whom to trust by sampling $a \in \{1, \dots, N\}$ from π^t, then follows $\mathcal{A}_a$'s decision. The main questions are then to know what observations (*i.e.*, arms and rewards) should be given as feedback to which algorithms, and how to update the trusts at each step, and our proposal **Aggregator** differs from **Exp4** on these very points.

[1] The model is similar for other distributions, and we also experimented and tested our proposal **Aggregator** with Gaussian, exponential and Poisson distributions, with unbounded or finite in $[0, 1]$ support, and similar conclusions were observed. Non-discrete rewards $r_k(t)$ are interpreted as a relative communication efficiency, but we do not cover this aspect here.

A. The *Aggregator* algorithm

Our proposed **Aggregator** is detailed in Algorithm 1.

Input: N bandit algorithms, $\mathcal{A}_1, \dots, \mathcal{A}_N$, with $N \geq 2$
Input: Number of arms, $K \geq 1$
Input: Time horizon, $T \geq 1$, **not** used for the learning
Input: A sequence of learning rates, $(\eta_t)_{t \geq 1}$
Data: Initial uniform distribution, $\pi^0 = U(\{1, \dots, N\})$
Result: $\mathcal{A}_{\text{aggr}} =$ **Aggregator** $[\mathcal{A}_1, \dots, \mathcal{A}_N]$
for $t = 1, \dots, T$ **do**
  for $a = 1, \dots, N$ **do** `// Can be parallel`
    $\mathcal{A}_a$ updates its indexes (e.g., UCB indexes);
    It chooses $A_a^{t+1} \in \{1, \dots, K\}$.
  end
  Let $p_j^{t+1} := \sum_{a=1}^{N} \pi_a^t \times \mathbb{1}(\{A_a^{t+1} = j\})$, $\forall 1 \leq j \leq K$;
  Then $\mathcal{A}_{\text{aggr}}$ chooses arm $A^{t+1} \sim p^{t+1}$;
  Give *original* reward $(A^{t+1}, 1 - \ell_{A^{t+1}}(t+1))$ to *each* $\mathcal{A}_a$ (maybe *not* on its chosen arm);
  Compute an *unbiased* estimate of the loss of the trusted algorithms,

$$\ell^{t+1} = \ell_{A^{t+1}}(t+1) / p_{A^{t+1}}^{t+1} \ .$$

  for $a = 1, \dots, N$ **do**
    if $\mathcal{A}_a$ *was trusted,* i.e., $A_a^{t+1} = A^{t+1}$ **then**
      $\pi_a^{t+1} = \exp(\eta_t \ell^{t+1}) \times \pi_a^t$
    end
  end
  Renormalize the new π: $\pi^{t+1} := \pi^{t+1} / \sum_{a=1}^{N} \pi_a^{t+1}$.
end

Algorithm 1: Our aggregation algorithm **Aggregator**.

At every time step, after having observed a loss $\ell_{A^{t+1}}(t+1)$ for its chosen action A^{t+1}, the algorithm updates the trust probabilities from π^t to π^{t+1} by a multiplicative exponential factor (using the learning rate and the *unbiased* loss). Only the algorithms $\mathcal{A}_a$ who advised the last decision get their trust updated, in order to trust more the "reliable" algorithms. The loss estimate is unbiased in the following sense. If one had access to the rewards $r_k(t+1)$ (or the losses $\ell_k(t+1)$) for all arms k, the loss incurred by algorithm a at time $t+1$ would be $\tilde{\ell}_{a,t+1} = \ell_{A_a^{t+1}}(t+1)$. This quantity can only be observed for those algorithms for which $A_a^{t+1} = A^{t+1}$. However, by dividing by the probability of observing this recommendation, one obtains an unbiased estimate of $\tilde{\ell}_{a,t}$. More precisely,

$$\hat{\ell}_{a,t+1} := \frac{\ell_{A_a^{t+1}}(t+1)}{p_{A_a^{t+1}}^{t+1}} \mathbb{1}(A_a^{t+1} = A^{t+1})$$

satisfies $\mathbb{E}[\hat{\ell}_{a,t+1} | \mathcal{H}_t] = \tilde{\ell}_{a,t+1}$, for all a, where the expectation is taken conditionally to the history of observations up to round t, $\mathcal{H}_t$. Observe that $\tilde{\ell}_{a,t+1} = \ell^{t+1}$ for all algorithms a such that $A_a^{t+1} = A^{t+1}$, and $\tilde{\ell}_{a,t+1} = 0$ otherwise.

An important feature of **Aggregator** is the feedback provided to each underlying bandit algorithm, upon the observation of arm A^{t+1}. Rather than updating only the trusted algorithms (that is the algorithms which would have drawn arm A^{t+1}) with the observed reward $r_{A^{t+1}}(t+1) = 1 - \ell_{A^{t+1}}(t+1)$, we found that updating each algorithm with the (original) loss observed for arm A^{t+1} improves the performance drastically. As expected, the more feedback they get, the faster the underlying algorithms learn, and the better the aggregation algorithm will be [12].

Regarding the update of π^t, one can note that the trust probabilities are not all updated before the normalization step, and an alternative would be to increase π_a if $A_a^{t+1} = A^{t+1}$ and to decrease it otherwise. It would not be so different, as there is a final renormalization step, and empirically this variation has little impact on the performance of **Aggregator**.

B. *Aggregator versus Exp4*

The **Exp4** algorithm (see, e.g. [10, Section 4.2]) is similar to **Aggregator**, presented in Algorithm 1, but differs in the two following points. First, $a \sim \pi^t$ is sampled first and the arm chosen by $\mathcal{A}_a$ is trusted, whereas **Aggregator** needs to listen to the N decisions to perform the updates on π^{t+1}, and **Exp4** gives back an observation (arm, reward) only to the last trusted algorithm whereas **Aggregator** gives it to all algorithms. Second, after having computed the loss estimate ℓ, **Exp4** updates the estimated cumulative loss for each algorithm, $\widetilde{L}_a(t) = \sum_{s=1}^{t} \ell_{A_a^s}(s) \times \mathbb{1}(A_a^s = A_{\text{aggr}}^s)$. Instead of updating π^t multiplicatively as we do for our proposal, **Exp4** recomputes it, proportionally to $\exp(-\eta_t \widetilde{L}_a(t))$.

The sequence of non-negative learning rates $(\eta_t)_{t \geq 1}$ used by **Exp4** can be arbitrary, it can be constant but should be non-increasing [10, Theorem 4.2]. If the horizon T is known (and fixed), the best choice is given by $\eta_t = \eta := 2\log(N)/(TK)$. However, for real-world communication problems, it is unrealistic to assume a fixed and known time horizon, so we prefer the alternative horizon-free choice of learning rates, $\eta_t := \log(N)/(tK)$ suggested by [10]. We compare both approaches empirically, and the second one usually performs better. We also stick to this choice of $(\eta_t)_{t \geq 1}$ for **Aggregator**.

V. Experiments on Simulated MAB Problems

We focus on *i.i.d.* MAB problems, with $K = 9$ channels[2]. For Bernoulli problem, the first one uses $\boldsymbol{\mu} = [0.1, \dots, 0.9]$, and the second one is divided in three groups: 2 very bad arms ($\mu = 0.01, 0.02$), 5 average arms ($\mu = 0.3$ to 0.6) and 3 very good arms ($\mu = 0.78, 0.8, 0.82$). The horizon is set to $T = 20000$ (but its value is unknown to all algorithms), and simulations are repeated 1000 times, to estimate the expected regret. This empirical estimation of the expected regret R_T is plotted below, as a function of T, comparing some algorithms $\mathcal{A}_1, \dots, \mathcal{A}_N$ (for $N = 6$), and their aggregation with **Aggregator** (displayed in orange bold). The Lai & Robbins'

[2] Similar behaviors are observed for any not-too-large values of K, we tried up-to $K = 100$ and the same results were obtained.

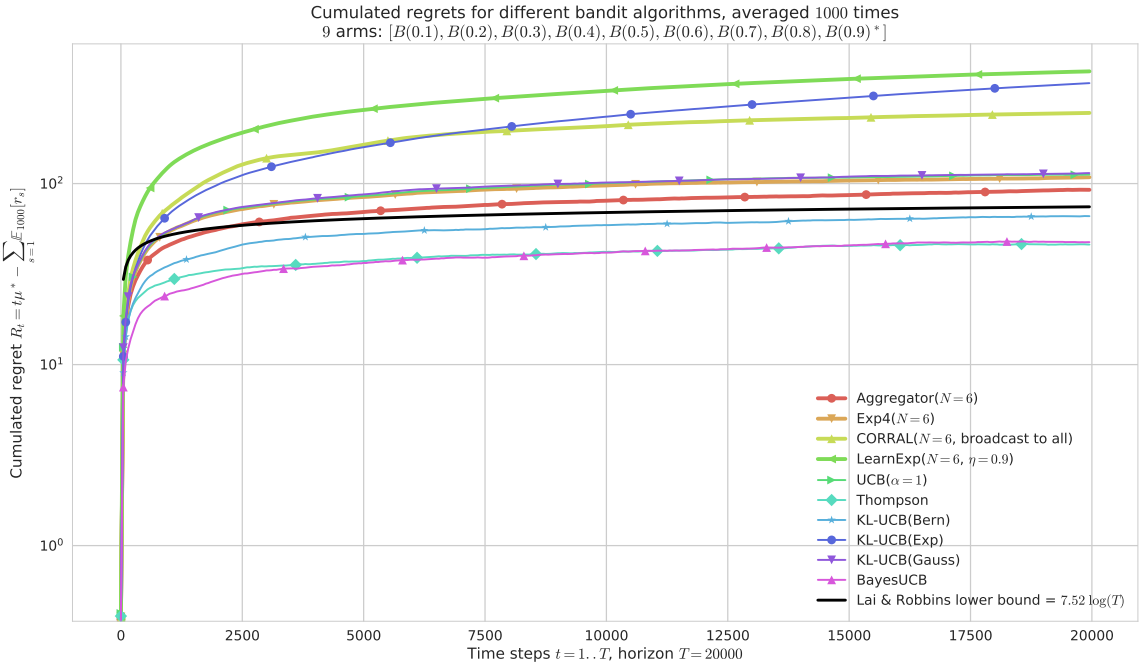

Fig. 1. On a "simple" Bernoulli problem (semilog-y scale).

logarithmic lower-bound [4] is also plotted, and it is crucial to note that it is only asymptotic and to not be surprised by having regret curves smaller than the lower-bound (*e.g.*, for the easier Bernoulli problem). Note that for each of the 1000 simulations, we choose to generate all the rewards beforehand, *i.e.*, one full matrix $(r_k(t))_{1 \leq k \leq K, 1 \leq t \leq T}$ for every repetition, in order to compare the algorithms on the same realizations of the MAB problem.

We compare our **Aggregator** algorithm, as well as other aggregation algorithms, **Exp4**, **CORRAL** and **LearnExp** (both with default parameters) [10], [17], [18]. The aggregated algorithms consist in a naive uniform exploration (to have at least one algorithm with bad performances, *i.e.* linear regret, but it is not included in the plots), UCB with $\alpha = 1/2$, three kl-UCB with Bernoulli, Gaussian and exponential kl functions, and BayesUCB and Thompson sampling with uniform prior.

Figures 1 and 4 are in semilog-y scale, this helps to see that the best algorithms can be an *order of magnitude* more efficient than the worst, and the **Aggregator** performs similarly to the best ones, when the other aggregation algorithms are usually amongst the worst. Figure 5 is in semilog-x scale to show that the regret of efficient algorithms are indeed logarithmic.

For Bernoulli problems (Figures 1 and 2), UCB with $\alpha = 1/2$, Thompson sampling, BayesUCB and kl-UCB$^+$ (with the

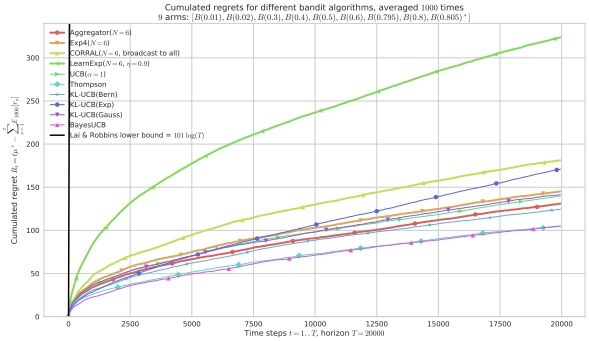

Fig. 2. On a "harder" Bernoulli problem, they all have similar performances, except **LearnExp**.

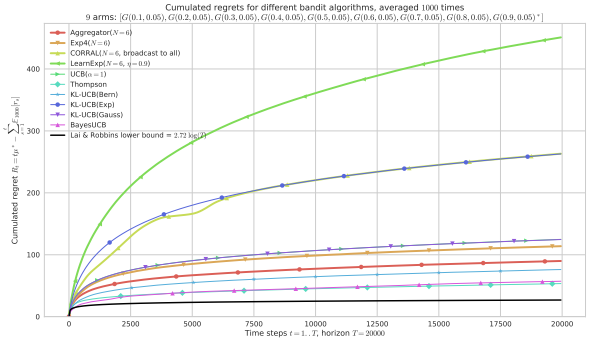

Fig. 3. On an "easy" Gaussian problem, only **Aggregator** shows reasonable performances, thanks to BayesUCB and Thompson sampling.

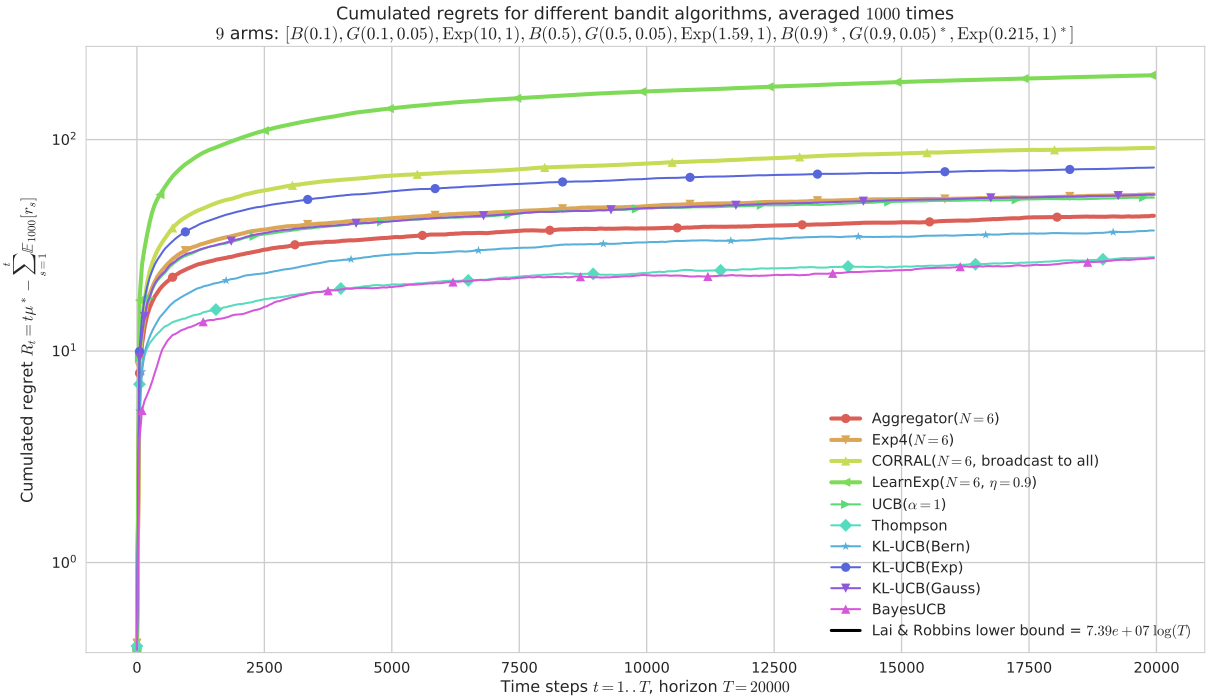

Fig. 4. On a harder problem, mixing Bernoulli, Gaussian, Exponential arms, with 3 arms of each types with the *same mean*.

binary kl function) all perform similarly, and **Aggregator** is found to be as efficient as all of them. For Gaussian and exponential arms, rewards are truncated into $[0, 1]$, and the variance of Gaussian distributions is fixed to $\sigma^2 = 0.05$ for all arms, and can be known to the algorithms (the kl function is adapted to this one-dimensional exponential family). Figure 3 uses only Gaussian arms, with a large gap between their means and a relatively small variance, giving an "easy" problem. And Figure 4 shows a considerably harder "mixed" problem, when the distributions are no longer in the same one-dimensional exponential family and so the Lai & Robbins' lower-bound no longer holds (even if there still exists a lower-bound).

For each of the 4 problems considered, the **Aggregator** algorithm with default option (broadcast loss to all players) is the best of all the aggregation algorithms, and its regret is very close to the best of the aggregated algorithms. Especially in difficult problems with mixed or unknown distributions, **Aggregator** showed to be more efficient that **Exp4** and orders of magnitude better than the other reference aggregation algorithms **LearnExp** and **CORRAL** (see Figures 4 and 5).

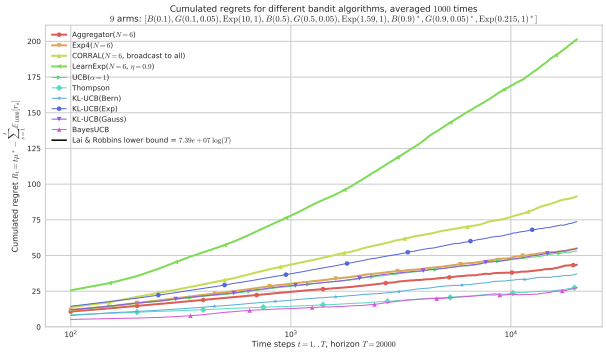

Fig. 5. The semilog-x scale clearly shows the logarithmic growth of the regret for the best algorithms and our proposal **Aggregator**, even in a hard "mixed" problem (*cf.* Figure 4).

VI. Theoretical Guarantees

The **Aggregator** does not have satisfying theoretical guarantees in terms of regret R_T yet, unlike many bandit algorithms. Another notion, the *adversarial regret*, denoted by $\overline{R_T}$, measures the difference in terms of rewards, between the aggregation algorithm $\mathcal{A}_{\mathrm{aggr}}$ and the best aggregated algorithm $\mathcal{A}_a$. This is in contrast with the (classical) regret, which measure the difference with the best fixed-arm strategy (Eq. (1)). Thus, even if the aggregated algorithms have logarithmic (classical) regret, having an adversarial regret scaling as $\sqrt{T}$ does not permit to exhibit a logarithmic (classical) regret for the aggregation algorithm. Under some additional hypotheses, [10, Theorem 4.2] proves that **Exp4** satisfies a bound on adversarial regret, $\overline{R_T} \leq 2\sqrt{TN \log k}$, with the good choice of the learning rate sequence $(\eta_t)_{t \geq 1}$. Our proposed algorithm

follows quite closely the architecture of **Exp4**, and a similar bound for **Aggregator** is expected to hold.

This would be a first theoretical guarantee, but not satisfactory as simple algorithms like UCB have regrets scaling as $\log T$ [7], [10], not $\sqrt{T}$. Regret bounds in several different settings are proved for the **CORRAL** algorithm [17], but no logarithmic upper-bound can be obtained from their technique, even in the simplest setting of stochastic bandits. However, **Aggregator** always seems to have a (finite-horizon) logarithmic regret in all the experiments we performed, for both Bernoulli and non-Bernoulli problems (*e.g.*, Gaussian, exponential and Poisson distributions). Further theoretical developments are left as future work.

VII. Conclusion

We presented the use of aggregation algorithms in the context of Opportunistic Spectrum Access for Cognitive Radio, especially for the real-world setting of unknown problem instances, when tuning parameters before-hand is no longer possible and an adaptive algorithm is preferable. Our proposed **Aggregator** was presented in details, and we also highlighted its differences with **Exp4**.

We realized experiments on simple MAB problems already used in the community of bandit algorithms for OSA [6], and the simulations results showed that **Aggregator** works as expected, being able to identify on the fly the best algorithm to trust for a specific problem. Experiments on problems mixing different families of distributions were also presented, with similar conclusions in favor of **Aggregator**. It is not presented in this article, but our proposed algorithm also works well in dynamic scenarios, in which the distribution of the arms can change abruptly at some time, and appears to be more robust than simple non-aggregated algorithms.

Exp4 has theoretical guarantees in terms of adversarial regret, and even if the same result could hold for **Aggregator**, results in terms of classical regret are yet to be proved. Empirically, **Aggregator** showed to always have a logarithmic regret R_T if it aggregates algorithms with logarithmic regrets (like UCB, kl-UCB, Thompson sampling, BayesUCB etc). It usually succeeds to be close to the best of the aggregated algorithms, both in term of regret and best arm pull frequency. As expected, the **Aggregator** is never able to outperform any of the aggregated algorithms, but this was an over-optimistic goal. What matters the most is that, empirically, **Aggregator** is able to quickly discover *on the fly* the best algorithms to trust, and then performs almost as well as if it was following it from the beginning.

Our **Aggregator** algorithm can probably be rewritten as an Online Mirror Descent, as **Exp4** and **CORRAL**, but this does not appear useful as in the case of **CORRAL** the analysis cannot bring a logarithmic bound on the regret even by aggregating asymptotically optimal algorithms. We will continue investigating regret bounds for **Aggregator**, and other directions include possible applications to the non-stochastic case (*e.g.*, rested or restless Markovian problems, like it was very recently studied in [19]).

VIII. Acknowledgements

The authors would like to thank Odalric-Ambrym Maillard at Inria Lille, Rémi Bonnefoi and Quentin Bodinier at CentraleSupélec Rennes, for fruitful discussions.

References

[1] J. Mitola and G. Q. Maguire, "Cognitive Radio: making software radios more personal," *IEEE Personal Communications*, vol. 6, no. 4, pp. 13–18, 1999.

[2] S. Haykin, "Cognitive Radio: Brain-Empowered Wireless Communications," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 2, pp. 201–220, 2005.

[3] H. Robbins, "Some aspects of the sequential design of experiments," *Bulletin of the American Mathematical Society*, vol. 58, no. 5, pp. 527–535, 1952.

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

[5] Q. Zhao and B. M. Sadler, "A Survey of Dynamic Spectrum Access," *IEEE Signal Processing magazine*, vol. 24, no. 3, pp. 79–89, 2007.

[6] W. Jouini, D. Ernst, C. Moy, and J. Palicot, "Upper Confidence Bound based decision making strategies and Dynamic Spectrum Access," in *IEEE International Conference on Communications (ICC)*, pp. 1–5, IEEE, 2010.

[7] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The Non-Stochastic Multi-Armed Bandit Problem," *SIAM Journal on Computing*, vol. 32, no. 1, pp. 48–77, 2002.

[8] A. Garivier and O. Cappé, "The KL-UCB Algorithm for Bounded Stochastic Bandits and Beyond," in *COLT*, pp. 359–376, 2011.

[9] E. Kaufmann, O. Cappé, and A. Garivier, "On Bayesian Upper Confidence Bounds for Bandit Problems," in *AISTATS*, pp. 592–600, 2012.

[10] S. Bubeck, N. Cesa-Bianchi, *et al.*, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Foundations and Trends® in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.

[11] W. Jouini, C. Moy, and J. Palicot, "Decision Making for Cognitive Radio equipment: analysis of the first 10 years of exploration," *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, no. 1, pp. 1–16, 2012.

[12] O.-A. Maillard and R. Munos, "Adaptive Bandits: Towards the best history-dependent strategy," in *AISTATS*, pp. 570–578, 2011.

[13] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3/4, pp. 285–294, 1933.

[14] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-Time Analysis of the Multi-Armed Bandit Problem," *Machine Learning*, vol. 47, no. 2, pp. 235–256, 2002.

[15] S. Agrawal and N. Goyal, "Analysis of Thompson sampling for the Multi-Armed Bandit problem," in *JMLR, Conference On Learning Theory*, pp. 39–1, 2012.

[16] E. Kaufmann, N. Korda, and R. Munos, *Thompson Sampling: an Asymptotically Optimal Finite-Time Analysis*, pp. 199–213. Springer, Berlin Heidelberg, 2012.

[17] A. Agarwal, H. Luo, B. Neyshabur, and R. E. Schapire, "Corralling a Band of Bandit Algorithms," *arXiv preprint arXiv:1612.06246*, 2016.

[18] A. Singla, H. Hassani, and A. Krause, "Learning to Use Learners' Advice," *arXiv preprint arXiv:1702.04825*, 2017.

[19] H. Luo, A. Agarwal, and J. Langford, "Efficient Contextual Bandits in Non-stationary Worlds," *arXiv preprint arXiv:1708.01799*, 2017.

Note: the source code (Python 2 or 3) used for the simulations and the figures is open-sourced at `https://GitHub.com/SMPyBandits/SMPyBandits` and a full documentation is available at and documented at `https://SMPyBandits.GitHub.`.