Reduction of collisions and regret in time sharing schemes for opportunistic spectrum access

Dimitris Giatsios Thanasis Korakis Leandros Tassiulas
University of Thessaly and CERTH

Iordanis Koutsopoulos
Athens University of Economics and Business and CERTH

***Abstract*—We examine decentralized learning and access algorithms for opportunistic spectrum access with multiple users. Several distributed algorithms have been proposed for this problem, mainly as an application of corresponding algorithms for the multiarmed bandit problem, which are provably order optimal in terms of regret. However, none of them pays particular attention to reducing collisions among users caused by lack of message exchanges. The effect of such collisions becomes more observable as the number of users increases, causing a considerable amount of added regret, despite retaining the optimal order. Focusing on time division fair sharing schemes based on the idea of orthogonal offsets, we propose a simple algorithm for detecting offset collisions and trying to resolve them as quickly as possible, inspired from persistent distributed schemes for multiple access. We demonstrate the improved performance achieved by our algorithm by means of simulations.**

I. Introduction

Channel selection in opportunistic spectrum access environments with multiple secondary users (SUs) is a challenging problem, as it entails two parallel objectives: learning the channels with the highest average availabilities and efficiently coordinating access among users to avoid collisions. The challenge becomes even greater if we restrict our attention to fully distributed schemes, that is, if there is no message exchange among the users, neither at the beginning nor during the learning and access process.

The learning objective has been studied extensively, mainly viewed as an extension to the relative single-user problem. It is typically modeled as a multiarmed bandit (MAB) problem, which, in its simplest form, can be described as follows: There are N independent arms. Playing arm i yields an i.i.d. random reward with a distribution parameterized by an unknown θ_i. At each timeslot, the player chooses one arm to play, aiming to maximize the total expected reward in the long run. In the analogy with the opportunistic spectrum access schemes, the N arms correspond to N primary channels and rewards correspond to channel availabilities for secondary users at each slot.

The model we focus on comprises N channels of equal bandwidth but different and unknown expected availabilities and M secondary users where $M < N$. That is, we consider scenarios where the available bandwidth is sliced in a way that channels outnumber the population of secondary users. We are interested in schemes both socially optimal and fair, so clearly the goal is to have the users equally share the rewards offered by the M channels with the highest availabilities.

Our main goal is to devise efficient algorithms for reducing collisions due to lack of coordination. Focusing on time division fair sharing schemes where the target is to have each user sequentially access the M best channels with a unique schedule offset with respect to other users, we propose a strategy for detecting offset collisions and trying to resolve them as quickly as possible. For the second objective, that of quick offset collision resolution, we revisit some of the work on persistent multiple access schemes proposed as enhancements to CSMA/CA.

As the rate at which the users learn the true ranking of primary channels in terms of availabilities can not be improved without explicit cooperation in terms of message exchanges, intuitively what we are trying to achieve is to minimize the added regret caused by the lack of offset coordination, even when all users have accurate estimates of the channel ranking. Although even a trivial randomized policy will guarantee that this added regret remain finite (eventually the users settle in orthogonal offsets), restricting its magnitude for a finite time horizon T by means of fast convergence requires more careful planning. Given that for a network with more than a few users this added regret can be shown to form the greatest part of the overall regret, we claim that it is important to place the required focus on this problem.

The remainder of this paper is organized as follows. Section 2 outlines related work on the MAB problem for a single user, on decentralized algorithms for MAB with multiple users, and on multiple access schemes with persistence properties which converge to collision-free schedules. The basic network model is presented in Section 3, along with the assumptions made. Our proposed decentralized algorithm with fast convergence to collision-free operation is explained in Section 4. In Section 5, we evaluate the performance of our algorithms by means of simulations and demonstrate the significant improvements with respect to existing algorithms. Section 6 gathers various remarks regarding the design choices we made and provides hints for future extensions of this work. The paper is concluded with Section 7, which summarizes the results obtained.

II. Related Work

A. Background on single player multiarmed bandit

Due to the prohibitive computational complexity associated with Bayesian approaches to solving the MAB problem, non-Bayesian algorithms have gained particular attention in this context. These algorithms typically try to minimize the expected regret, which is the reward loss with respect to a so-called genie-aided scenario where the expected arm rewards are known in advance.

In [1] the minimum possible asymptotic regret of any uniformly good policy was established and shown to be $O(logT)$, with a leading constant that depends on the Kullback-Leibler distances between the best arm and suboptimal arms. The authors in [1] also constructed an optimal policy, i.e. a policy that achieves the lower bound, for several reward distributions including Bernoulli, Poisson, Gaussian and Laplace, under the assumption that the distribution type is known.

In [2], the results of [1] were extended to the case of multiple plays, that is, the player is allowed to play M arms in each slot instead of one. A straightforward extension of the results in [1] is possible, and the only thing that changes is the leading constant in the logarithmic regret. The result of [2] is equivalent to a centralized MAB with M players each playing one arm per slot. If all players exchange observations at each slot and make decisions jointly (which means they avoid selecting common channels - they're always orthogonalized), then they act as a single player who has the ability to choose M arms simultaneously. As a consequence, the lower bound of [2] is a lower bound for any decentralized MAB where players don't exchange information and make decisions independently. It is also a lower bound for any potential cooperative policy, where the players exchange information but perhaps not on a per slot basis.

In [3], Agrawal proposes an order optimal index-type policy, where each arm is assigned with an index that is a function of its sample mean and the arm with the greatest index will be played in the next slot. The proposed policy is simpler than [1], however it does not achieve the best leading constant.

In [4], the authors establish a simple order-optimal sample mean based index policy based on [3], called UCB1 (UCB stands for Upper Confidence Bound). For an arm i with sample mean $\hat{\theta}_i$ which has been played n_i times, the index at timeslot t is given by:

$$I(\hat{\theta}_i, n_i) = \hat{\theta}_i + \sqrt{\frac{2logt}{n_i}} \tag{1}$$

This policy yields a larger leading constant than [3]. The main difference with the previous works is that [4] proves that the logarithmic order of regret is not only achievable asymptotically, but also uniformly over time. UCB1 requires reward distributions of known finite support, but the type of the distribution is not required. Looking at the expression, we see that it corresponds to an "increased" reward estimate. The second term is increasing while an arm is not played, but it does so with a rate that guarantees that the optimal arm will be played exponentially more often than the suboptimal arms.

In [5] a policy using deterministic sequencing of exploration and exploitation sequences (DSEE) is proposed. In this policy, purely explorative periods where all the channels are probed are interpolated with exploitation periods where the estimated best channel is consistently targeted. Its advantage over the previous approaches is that does not require any assumption on the distribution type or its support. The disadvantage is that it requires knowledge of a lower bound on the difference between the expected rewards of the best and the second best arm, a potentially demanding requirement.

B. Work on distributed algorithms for multiple user multi-armed bandit

Both game-theoretic and non-game-theoretic approaches have been adopted in the multiple user case of MAB problems. To remain within the scope of our work, here we only list some non-game-theoretic work on MAB with i.i.d. rewards, which has been performed specifically focusing on the opportunistic spectrum access application.

Non-game theoretic approaches focus on minimizing the total system regret, which is the aggregate reward loss with respect to the genie-aided case where the users are aware of channel statistics and always avoid collisions among them. Apart from the balance between exploration and exploitation, the additional challenge lies in the construction of decentralized algorithms aiming at orthogonalizing the users, i.e. having them access distinct channels at each slot. These approaches typically involve implicit cooperation, in the sense that the secondary users agree to adopt a policy that achieves system regret optimality. [6], [7] and [8] are examples of such approaches. These papers focus on the case of a known number of users, equal to M, and assume that the number of channels N is greater than M. They all achieve logarithmic order of the regret, which is the optimal order.

Note that from a policy perspective, what happens when two or more SUs sense the same channel and find it to be available is not so crucial, provided that the expected reward for each user in this case is less than the expected reward if it accessed the channel by itself. For instance, different models might assume collision or CSMA contention or perhaps time sharing. In any case, the outcome is suboptimal from a system perspective.

In [6] each of the M SUs settles on a particular priority rank, say w_j for user j, and targets the channel with the w_j-th best availability. The proposed algorithm converges to a socially optimal user-channel configuration, but does not achieve fairness among users.

In [7], on the other hand, fairness among users is achieved through the use of a time division fair sharing (TDFS) structure featuring M subsequences, where a user targets a channel of different rank in each subsequence. In order to avoid collisions, users must try not to target channels of the same rank during the same timeslot. An orthogonal set of offsets is required for this purpose. If offsets are not preallocated by some mechanism, users must employ feedback from collisions experienced to converge to such an offset configuration. Rapid convergence to orthogonal offsets is the primary object of our work in the present paper.

TDFS policies are also used in [8], where they are combined with a variant of UCB1 suitable for targeting channels with any rank in descending reward order (not only the best channel), called SL(k). The resulting algorithm is named DLF (Distributed Learning with Fairness). SL(k) was initially proposed in the context of prioritized user ranking in this work, where it is crucial to use it instead of UCB1 because a user always targets the same rank. In DLF each user targets all M best channels, hence UCB1 can also be used as is.

In [10] an analysis of the problem of allocation of multiple users to channels is performed under a general interference function and considering various levels of knowledge of users and levels of cooperation. The model considered includes the case where the number of users is greater than the number of channels. Randomized algorithms with sublinear but not logarithmic regret are proposed for settings with i.i.d. channel rewards.

In [5] a scheme using DSEE is also proposed for the case of multiple users. The drawback of the scheme is that it requires knowledge of a lower bound on the expected reward difference between the $M-th$ and the $(M+1)-th$ arm. The scheme uses the sampling results from the exploration periods to distinguish the M best channels and uses common arm indices for fair time sharing in the exploitation periods. User offsets are considered predetermined, however it is easy to see that in schemes with random offsets the clear separation of exploration and exploitation intervals has the added benefit of simplifying the distributed convergence to orthogonal time division fair schedules. Indeed, offset collisions can be detected and resolved quickly during the exploration intervals, when there is no channel ranking ambiguity.

The above works generally assume perfect sensing for primary activity. Taking imperfect sensing into consideration complicates the decentralized MAB problem if users cannot distinguish between collisions with primary and with other secondary users. See [9] for an approach to decentralized MAB under potentially corrupted sensing samples based on a TDFS structure with preallocated offsets, where the main issue is synchronization of channel access orders between transmitter and receiver.

C. Alternative MAC schemes for increased efficiency

Part of the results of this work is inspired by multiple access schemes proposed in the literature as alternatives to CSMA/CA. The main goal of these schemes is to increase medium utilization through minimizing medium idle time. The idea is that the stations try to settle in a fixed backoff configuration and essentially schedule their transmissions into a TDMA-like round robin fashion.

In [11] the Learning-BEB algorithm was proposed, also referred to as CSMA/ECA. It is essentially a variation of 802.11 DCF using a fixed backoff V after a successful transmission and a uniformly random backoff after a colliding transmission. This is essentially the algorithm used in [7] to reach a collision-free offset configuration. Convergence to a collision-free schedule with Learning-BEB can take a lot of time, particularly when the number of stations is close to the number of slots [13].

In the ZC-MAC algorithm proposed in [12], each station observes all timeslots, not just the ones it transmits in. Upon a successful transmission, it "reserves" the selected timeslot, provided it does not leave it idle for more than a predetermined number of timeslots. This is achieved by stations only trying to transmit to non-reserved slots. Upon a failure, the station looks at the slot occupancy in the previous schedule and makes a uniform selection in the set consisting of idle slots and its previously selected slot.

In the L-ZC algorithm proposed in [13] each station keeps using the same slot while its transmissions are successful. In case of transmission failure, it selects the same slot with probability γ or one of the n_i idle slots with probability $(1-\gamma)/n_i$. For C slots and N users, γ is chosen to be equal to $1/(C-N+2)$. In the L-MAC algorithm proposed in the same work, inspired from [14], the stations retain a probability vector for the C slots initialized with a uniform distribution. Upon a successful transmission, a station updates the probability of selecting the same slot to 1 (and 0 for the rest of the slots), while upon a failure the probability of selecting the last slot $s(n)$ is updated as $p_{s(n)}(n+1) = \beta p_{s(n)}(n)$ and the probability of selecting another slot $j \neq s(n)$ is updated as $p_j(n+1) = \beta p_j(n) + \frac{1-\beta}{C-1}$. In agreement to intuition, choosing $\beta = 1$ (persistence) minimizes mean convergence time, however taking into account factors such as transient fairness and throughput under oversubscription motivates a somewhat lower value $\beta = 0.95$.

The L-MAC algorithm is the one more suited to our problem, because it does not assume that the stations can sense idle slots. Indeed, in our framework the equivalent of idle slots would be unused offsets, and users can only observe activity at their selected offsets.

D. Our Contribution

Our contribution falls into the region of decentralized algorithms for multiple user opportunistic spectrum access under the assumption that the number of channels is greater than that of secondary users. It specifically focuses on the case where there are no preallocated user ranks or priorities and builds upon prior existing work, by further incorporating an offset collision detection functionality and a persistent algorithm for multiple access.

III. Basic Model

We have N independent primary channels and slotted time indexed by t. The availability of channel i is a Bernoulli random process $X_i(t)$ with expected value θ_i.[1] We denote the set of channel average availabilities as $\Theta = \{\theta_i, 1 \leq i \leq N\}$. There are M decentralized secondary users (SUs), where by the term users we refer to disjoint transmitter-receiver pairs. M is smaller than N and the users are infinitely backlogged. We do not examine the case where M is greater than N, because then there is no learning dimension in the problem, since the socially optimal strategy entails exploitation of all the channels.

At each timeslot t, each user independently selects one channel for its transmission. It senses the selected channel for primary transmissions at the beginning of the timeslot. If it is found idle, it transmits in the remainder of the timeslot. Else, it stays silent until the next timeslot. Sensing for primary user activity is assumed to be error-free, even though a relaxation of this assumption can be easily incorporated into our algorithm, along the lines of [9]. All users perceive the same channel availabilities.

When multiple users transmit in the same channel at the same timeslot, a collision occurs and none of the users

[1]Our proposed algorithms are also applicable to MAB problems with other types of reward distributions. The only requirement is that collision events are observable.

gets any reward. We assume that in the event of a collision a user can not distinguish the number of users that were involved. The only feedback is the presence or not of an ACK from the receiver side, which determines if a collision took place. If a user is the only one to transmit in its selected channel, it receives a reward equal to 1. Note that the learning ability of a secondary user is not affected by collisions, since sensing for primary activity takes place before any secondary transmissions.

We denote by $\pi_j(t)$ the local policy for user j at timeslot t, which may be deterministic or randomized, and by $\pi = \{\pi_j(t), 1 \leq j \leq M\}$ the set of policies of all users. The total regret of a decentralized policy π after t timeslots and for a set of availabilities Θ is the expected aggregate reward loss of the policy when compared to a genie aided scheme where the best M channels are known in advance and the users are always orthogonalized. If we define $I_{i,j}(t)$ to be 1 when user j is the sole user of channel i at timeslot t and 0 otherwise, and O_M^* to be the set of channels with the M largest availabilities, then the expected total regret after t timeslots is expressed as

$$R^{\pi}(\Theta;t) = t\sum_{i\in O_M^*}\theta_i - E^{\pi}[\sum_{s=1}^{t}\sum_{i=1}^{N}\sum_{j=1}^{M}X_i(s)I_{i,j}(s)] \quad (2)$$

In the above expression, the expectation is taken with respect to both the random process of channel availabilities and the possibly randomized channel selection policies of the users.

In order to efficiently learn the best M channels and their order, we assume that each user is employing the SL(k) algorithm, developed in [8], which is a variation of UCB1 algorithm targeted at detecting the k-th best channel, while UCB1 only targets the best one. We briefly present the algorithm here again, for ease of reference. Consider any random user and assume that at timeslot t it is targetting the k-th best channel and has sensed the i-th channel $n_i(t)$ times, observing a sample mean of $\hat{\theta}_i(t)$. Then, SL(k) works as shown in Algorithm 1.

Algorithm 1 The SL(k) algorithm

```
Notation
$\hat{\theta}_i(t)$: Sample mean of channel $i$ at slot $t$
$n_i(t)$: Number of samples of channel $i$ at slot $t$
$X_i(t)$: Availability (0 or 1) of channel $i$ at slot $t$
$c(k)$: Channel ranked $k$ according to the current state of learning

Initialization: Play each arm once

Main Loop:
$t = t + 1$
Let $O_k$ contain the $k$ arms with the $k$ largest values in: $\hat{\theta}_i(t-1) + \sqrt{\frac{2lnt}{n_i(t-1)}}$
Select channel $c(k)$ in $O_k$ such that:
$c(k) = \arg\min_{i\in O_k}\{\hat{\theta}_i(t-1) - \sqrt{\frac{2lnt}{n_i(t-1)}}\}$
$\hat{\theta}_{c(k)}(t) = \frac{\hat{\theta}_{c(k)}(t-1)n_{c(k)}(t-1)+X_{c(k)}(t)}{n_{c(k)}(t-1))+1}$
$n_{c(k)}(t) = n_{c(k)}(t-1) + 1$
```

In order to avoid collisions while retaining fairness among users, a time division fair sharing (TDFS) policy is employed.

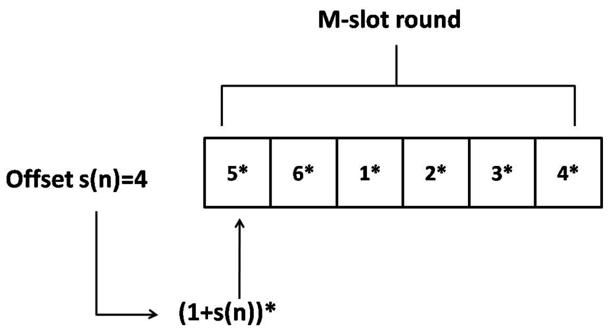

Fig. 1. Example of the structure of a M-slot round in DLF, where the notion of user offset is depicted

We assume that each user is aware of the total number of users in the network (M). After the initialization period, where all N channels are sensed by all users, time is divided into rounds of M timeslots. In a given round, a user sequentially targets the M best channels according to the result of SL(k) running locally, with a given offset in $\{0, 1, ..., M-1\}$, starting from the channel with ranking equal to 1 plus the offset. This is the TDFS variant employed in [8], DLF. Note that in DLF, as the SL(k) statistics of the channels are recalculated after each timeslot rather than after each round, and no channel is excluded from candidacy to be selected at a given timeslot, it may be possible for a channel to be selected more than once within a round. However, in practice this scenario takes place so sparsely that we can safely neglect it. We index the rounds with n and denote the selected offset of a given user at round n as $s(n)$.

As an example, consider a network with 6 users, where time is divided into periods of 6 timeslots. Then, a user with an offset equal to 4 will target sequentially $5^*, 6^*, 1^*, 2^*, 3^*, 4^*$, where by k^* we refer to the user's estimate of the k-th best channel based on the order of SL(k) local metrics. We depict this scenario in figure 1, where the notion of user offset can be better comprehended.

IV. Reduction of offset collisions under channel ranking uncertainty

In a fully distributed setting with no message exchanges between secondary users, it makes no sense to assume a predetermined user ranking, which would translate in known offsets. Thus settling into a configuration of orthogonal offsets needs to take place through local offset readjustments triggered by evaluation of collisions experienced in past timeslots. Offset readjustment may be decided at regular intervals, conveniently before each new round of M timeslots, or on the fly after an event associated with one or multiple collisions occurs in the user's history. We adopt the former case, as on the fly adjustment of offsets complicates synchronization of decisions.

The randomization policies used in [7] and [8] for offset collision resolution are based on the principle "if you collided -even once- in the previous round of timeslots, then randomize your selection among the M possible offsets for the next round". As we already mentioned, this is the equivalent of the Learning-BEB algorithm [11] for settling stations into distinct transmission slots. While this indeed ultimately leads to an absorbing state where the users are settled orthogonally in the M best channels, it does not yield the optimal convergence rate [13].

Note that the efficiency of an offset collision resolution has two facets. The first is detecting an offset collision with the highest accuracy possible. Uncertainty arises from the fact that a series of experienced collisions in the recent history may not be due to actual offset collision with another user, but due to wrong channel ranking estimates by the user itself or by other users. As time advances and estimates become more and more accurate, the occurrence of such events diminishes and collisions are increasingly due to offset collisions.

The second facet of collision resolution is to apply a smart randomization policy when readjusting local offsets. In general, after a decision to adjust the local offset, a user must make a selection for the next round given by a probability mass function over all possible M offsets. It makes sense that users which did not experience a collision in the previous slot should retain their selection. The offset selection algorithm for users which detected an offset collision will determine the speed of convergence to an orthogonal offset configuration.

A. A threshold rule for offset collision detection

In an attempt to detect offset collisions with high accuracy, while keeping the required computation overhead low, we decided to employ a threshold rule based on the number of collisions a user experiences in a round of M timeslots. Our decision was motivated by an effort to balance efficiency with complexity. More sophisticated and computationally complex alternatives are briefly discussed in Section VI.

Intuitively, it makes sense that few or no collisions are an indication that the user's selected offset is unique and experienced collisions are due to erroneous estimates of channel ranking. A lot of collisions are an indication of an offset collision. Thus, if we set an appropriate threshold, we can limit unnecessary offset adjustments. After extensive simulations using a great variety of parameters (number of channels, number of users, channel availabilities), we found a good threshold to be approximately $M/2$, that is, half the number of users. As any choice of the threshold involves both false alarms and missed detections[2], and those can have different direct and indirect effects on the regret, we based our choice on minimizing the mean regret observed.

We express the threshold as a multiplicative factor on the number of users M and denote it as Thr, so our choice corresponds to $Thr = 0.5$, while the value used in the randomized algorithms in [7] and [8] is $Thr = 0$.

B. Persistent algorithms for fast offset collision resolution

In our study of the problem of smart resolution of offset collisions, we revisited results from multiple access protocols proposed for wireless as enhancements to CSMA/CA, especially those which focus on persistent algorithms. In our case, imagine the extreme where the channel ranking is already known by all the users and the only problem is for them to arrive at an orthogonal offset configuration. Then, as offset collisions can be detected without errors, users could employ the following simple algorithm. Initially, each user picks an offset uniformly at random. Upon completion of a round with no collisions, a user "locks" in the selected offset for all future rounds, irrespectively of potential offset collisions in the future. On the other hand, a user that keeps colliding will keep selecting an offset uniformly at random until it stops colliding, at which time it "locks" into the selected offset. This is essentially the L-MAC algorithm from [13], with parameter $\beta = 1$.

It is easy to understand that this simple distributed algorithm leads to an orthogonal schedule very fast (see [13]). The problem is that, as we argued, offset collision cannot be detected with certainty. Therefore, we need to come up with a less rigid scheme, where users are not permanently locked into an offset irrespectively of future collision events. With this requirement in mind, we examine two alternative policies.

The first one is essentially the L-MAC algorithm with $\beta < 1$. If the parameter β is large enough, the effect of "stickiness" to offsets used successfully in the past remains, while keeping β strictly smaller than 1 ensures that repeated offset collisions will trigger readjustment of offset. We tested this approach and found values between 0.5 and 0.9 to yield very good performance (with differences among them smaller than 5%). For smaller values of β the per user regret was found to be fairly larger. Note that the necessity to use a value of β smaller than 1 does not derive from some transient fairness motivation, as in [13], but is essential to avoid multiple users from locking in the same offset.

The second approach we examined is more focused to the structure of the problem. A user locks into its selected offset after a round in which it does not detect an offset collision and records its estimate of M best channels and their rank during this round. In future rounds, if it detects an offset collision, it cross-examines its current ordered list of M best channels with the one it keeps in its records since the timeslot when it locked into its current offset. If the lists differ, then the user releases its offset lock. If the lists are the same, then the user retains its offset, despite the offset collision detection. In this way, we assure that once the users learn the M best channels, settling into orthogonal offsets is relatively fast. Also, we avoid having a nonzero probability $(1 - \beta)$ to switch from a previously successful offset after a single offset collision round.

Intuitively, we expect both algorithms to work significantly better than non-persistent algorithms, due to the effect of "stickiness" that they exploit. The results from the simulations we conducted showed that they exhibit similar performance. Therefore, we select the former one for our algorithm, as it is fairly more simple. The combination of offset collision detection, offset collision resolution and the DLF time sharing structure with SL(k) learning constitute our proposed algorithm, which we refer to as DLF-persistent. Its function can be viewed in Algorithm 2.

V. Performance evaluation

We conducted simulations to evaluate the performance of our algorithms. We fixed the number of channels to 18. The

[2]Please note that false alarms and missed detections in this context refer to detection of offset collisions and should not be confused with sensing for primary activity.

Algorithm 2 The DLF-persistent algorithm

```
Notation:
$N$: number of channels, $M$: number of users
$ncoll(n)$: Number of collisions during $n$-th $M$-slot round
$s(n)$: Offset selected during $n$-th $M$-slot round
$s(0)$: Offset employed during initialization phase
$p(n,m)$: Probability of selecting offset $m$ ($m \in \{0,1,...,M-1\}$) during $n$-th $M$-slot round
$Thr$: Multiplicative threshold for offset collision detection, selected to be 0.5
$\beta$: Parameter controlling the degree of persistence, selected to be 0.9
$c(k)$: Channel ranked $k$ according to the current state of learning
$\hat{\theta}_i(t)$: Sample mean of channel $i$ at slot $t$
$n_i(t)$: Number of samples of channel $i$ at slot $t$
$X_i(t)$: Availability (0 or 1) of channel $i$ at slot $t$

Initialization:
Select $s(0)$ uniformly at random
for $t = 1$ to $N$ do
   Select channel $k$ such that $k = ((t - s(0)) \mod N) + 1$
end for
$p(1,k) = 1/M$
Select $s(1)$ according to distribution $p(1,k)$

Main Loop:
-At slot t during $n$-th $M$-slot round:
Using SL($k$) specified in Algorithm 1, select $c(k)$, where $k = ((t - s(n)) \mod M) + 1$
$\hat{\theta}_{c(k)}(t) = \frac{\hat{\theta}_{c(k)}(t-1)n_{c(k)}(t-1)+X_{c(k)}(t))}{n_{c(k)}(t-1)+1}$
$n_{c(k)}(t) = n_{c(k)}(t-1) + 1$

-At the end of $n$-th $M$-slot round:
if $ncoll(n) > \lfloor Thr * M \rfloor$ then
   $p(n+1, s(n)) = \beta p(n, s(n))$
   $p(n+1, m) = \beta p(n,m) + \frac{1-\beta}{M-1}$, for all $k \neq s(n)$
else
   $p(n+1, s(n)) = 1$
   $p(n+1, m) = 0$, for all $k \neq s(n)$
end if
Select $s(n+1)$ according to distribution $p(n+1, m)$
```

average channel availabilities were chosen equally spaced from 0.26 to 0.94 with 0.04 spacing. We varied the number of users from 8 to 16 with a step of 2 and compared the per user regret of different algorithms after 500000 timeslots, taking the mean value of 50 simulations.

In particular, we compared the regret performance of four algorithms: i) DLF with predetermined offsets (as a benchmark) ii) DLF with random initial offsets and the offset updating policy of [7], which we denote as DLF-rand, with threshold $Thr = 0$, iii) DLF-rand with threshold $Thr = 0.5$ and iv) our proposed algorithm, DLF-persistent, with threshold $Thr = 0.5$. The value of 500000 was selected so that even in the worst policy (DLF-rand with threshold zero) the network has enough time to converge in a collision-free operation where regret follows the typical logarithmic pattern. Alternatively we could also plot the leading constants of the asymptotically logarithmic regret for the four algorithms, but we believe a straightforward comparison of aggregate regret values is more intuitive and useful for our purpose.

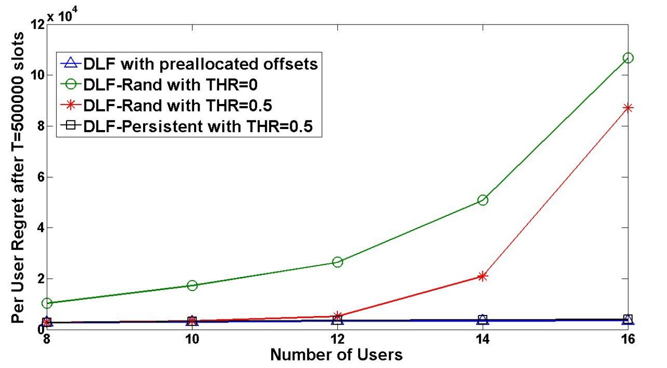

Fig. 2. Comparison of Per User Regret after 500000 timeslots for: i) preallocated offsets, ii) DLF-Rand with Thr=0, iii) DLF-Rand with Thr=0.5 and iv) DLF-Persistent with Thr=0.5. The number of channels is fixed to 18 with equally spaced average availabilities between 0.26 and 0.94, and we vary the number of users from 8 to 16

The results are depicted in figure 2. As we expected, DLF-rand with $Thr = 0$ exhibits the worst performance. Its regret for 16 users is about 30 times the regret of DLF with preallocated offsets, with considerably more regret also for fewer users. DLF-rand with $Thr = 0.5$ exhibits relatively good behavior up to a certain number of users, but its performance for 14 and 16 users is significantly higher than with preallocated offsets. This shows that for a number of users up to approximately 12, appropriate selection of the Thr parameter can by itself yield significant benefits, however it is not enough for larger numbers of users. Our proposed scheme, which is DLF-Persistent with $Thr = 0.5$ exhibits very good behavior independent of the number of users, with only slightly more regret than that with preallocated offsets.

In figure 3 we get a closer look at the regret performance of our proposed algorithm, DLF-persistent, in comparison with DLF with preallocated offsets. In the same figure, we also plot the performance of the second candidate algorithm for offset collision resolution we mentioned in subsection IV-B. As we can see, DLF-persistent performs slightly better than the other candidate, besides being simpler to implement, thus justifying our choice. It also performs very close to DLF with preallocated offsets, essentially minimizing the effect of their lack.

We must stress the fact that all the algorithms simulated exhibit the optimal logarithmic order of regret. However, as the number of users increases, and especially above 10 users, the rapid convergence to an orthogonal offset configuration achieved by our proposed algorithm translates into a very significant reduction in per user regret.

VI. Discussion

A. Alternatives to threshold rule for offset collision detection

The threshold rule based on the number of collisions experienced during a round of M slots is clearly not the best possible way to detect offset collisions. A more sophisticated method would examine each collision event and estimate the

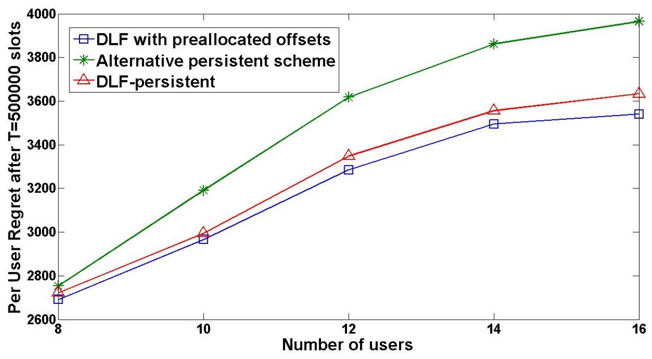

Fig. 3. Comparison of Per User Regret after 500000 timeslots for i) preallocated offsets, ii) DLF-Persistent with Thr=0.5 and iii) the alternative persistent scheme proposed in subsection IV-B. The number of channels is fixed to 18 with equally spaced average availabilities between 0.26 and 0.94, and we vary the number of users from 8 to 16

probability that the channel where the collision took place is misranked, or some approximation of it, such as the probability that the relative ranking between this channel and the one with the closest sample mean is wrong. Bounds of these probabilities, based for instance on Gaussian confidence bounds for difference of means, could be used. Such methods correctly assign a different weight on each collision event as to its likelihood of being due to an offset collision. Additionally, they provide robustness, as they can better detect offset collisions, even in cases where the majority of the actual channel availabilities are small and hence actual collisions might not be detected due to refrainment from access. However, this is a fairly more complex method, and simulation results show that using the simple threshold rule we employed combined with the persistence mechanism gives very good results. Therefore, we leave it as future work to quantitatively assess the benefits from using a more sophisticated rule for offset collision detection.

B. *On usage of common indices*

An intuitively obvious extension of our algorithm would be to use common indices for ranking the top M channels at each user. Common indices rely on a globally agreed set of channel identifiers, for instance corresponding to increasing center frequency. The idea is that the user employs the statistics gathered from its local learning procedure to decide which are the best M channels, and then sorts these channels by common index instead of by their statistics. This approach resolves cases with ties in the expected availabilities of the M best channels, and reduces the regret in cases with very close availabilities among subsets of these channels. It is also better suited to the nature of the problem, considering that under a time division sharing policy the crucial objective is to distinguish the M best channels from the rest, while the internal ranking among them based on availability estimates becomes irrelevant.

There is however a caveat in using this approach. In particular, the risk associated with using ranking by common index for the M best arms is that a change in the set of M best arms can cause multiple re-rankings of arms. This can cause multiple collisions and possibly trigger a false alarm for offset collision.

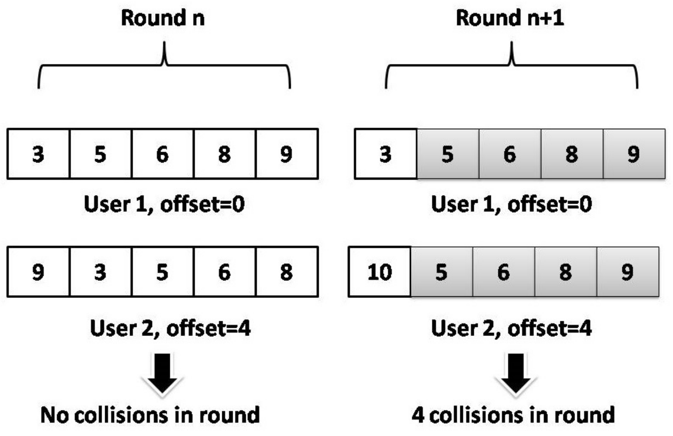

Fig. 4. An example demonstrating the risk of using ranking by common indices. A single change in the M-best set of one user causes multiple collisions despite different offsets

For instance, consider a network with 5 users and 10 channels (indexed $\{1, ..., 10\}$), and focus on two users, one with offset equal to 0 and one with offset equal to 4, whose common estimate of the 5 best arms during round n is the set $\{3, 5, 6, 8, 9\}$. Suppose that at the next round $n + 1$ channel 10 makes it in the second user's M-best set, replacing channel 3. Then, the entire common index based channel ranking for that user changes and, while the two users have different offsets, they may collide in up to four timslots of the 5-slot round (depending on the actual availabilities of the channels in the respective slots), which in turn will likely trigger a false offset collision detection alarm. This scenario is depicted in Figure 4. These chain reaction re-rankings that can be caused by common index ranking are the drawback of using this approach.

We verified through simulations that such scenarios are indeed a cause of increased regret particularly in cases where there are ties or close availabilities involving the actual M-th best arm. On the other hand, if the ties or close availabilities are away from the M-th best channel boundary, ranking by common indices works fine, as expected, and yield the expected performance benefits compared to ranking by sample mean based estimates. There seems to be no straightforward workaround for this problem, even with applying hybrid channel ranking schemes. We are working in further investigating potential solutions that provide good performance for any set of expected channel availabilities. It is possible that there exist a tradeoff between robustness under any scenario and optimal performance for specific scenaria.

C. *Comparison with DSEE*

We are aware that the work in [5], which is based on DSEE, also addresses the problem of collisions among users, when there is no prior agreement on user ranking. However, this work relies on the assumption of a known lower bound between the availabilities of the M-th best and the $(M + 1)$-th best channel. This is a rather unrealistic assumption in an environment with unknown channel statistics. Essentially, this approach is exploiting the fact that a bound on the required discerning ability of the algorithm is given in advance, so that the extent of exploration does not rely on random reward results.

Furthermore, in [5] learning is performed using only observations from the exploration period, a fact that decreases the learning rate with respect to the case where all observations are utilized. The benefit from using only observations from the exploration period is that all users share exactly the same estimations of the channel rankings, hence they share the same estimated set of M best channels. Therefore, they can safely adopt ranking by common indices in the exploitation period without the caveat we discussed in the previous subsection. It is clear that the additional requirement for simultaneous commencement of the learning process by all users is particularly crucial in this case.

D. Future directions

It should be clear that our proposed algorithm is applicable not only in DLF with $\mathrm{SL}(k)$, but also to similar TDFS-based distributed policies using other learning algorithms, such as those from [1] and [3]. In fact, it applies to any learning algorithm with sublinear regret, even if it is not order-optimal. Indeed, we have not made any assumption regarding the learning mechanism, but have only focused on detecting and resolving offset collisions. Since the Lai-Robbins learning algorithm [1] achieves the optimal logarithmic constant of regret, we expect that applying our algorithm onto the framework of [7], which builds around this algorithm, will actually yield even better regret performance.

An interesting direction is to see whether the requirement for all stations to commence their learning and access operations concurrently, implicit in this paper and all relevant past work, is crucial for the performance of the relevant algorithms. In this context, dropping the assumption of known number of users M and instead having to estimate it on the fly seems the first step for accommodating a varying user population. This is a topic we intend to investigate further in our future studies.

VII. Conclusions

We examined fully decentralized algorithms for an opportunistic spectrum access network with M secondary users and $N > M$ channels with i.i.d availabilities. We enhanced the performance of offset-based time division fair sharing policies previously proposed in the literature, by focusing on limiting offset collisions among users, based on feedback from the collision history. We proposed a threshold rule for detecting offset collisions and a persistent algorithm for accelerating convergence to an orthogonal offset configuration. The combination of these two techniques yields regret performance close to the setting with preallocated offsets, as shown by simulations with Bernoulli distributed channel rewards.

Acknowledgements

D. Giatsios acknowledges the support of the European Commission under Marie Curie IRSES grant PIRSES-GA-2010-269132 AGILENet.

I. Koutsopoulos acknowledges the support of ERC08-RECITAL project, co-financed by Greece and the European Union (European Social Fund) through the Operational Program Education and Lifelong Learning NSRF 2007-2013.

References

[1] L. Lai and Robbins, “Asymptotically efficient adaptive allocation rules”, Adv. Appl. Math., vol.6, no.1, p.4C22, 1985

[2] V. Anantharam, Varaiya and Walrand, “Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays - Part I: IID rewards”, IEEE Trans. Autom. Control, vol.32, no.11, pp.968-976, 1987

[3] R. Agrawal, “Sample mean based index policies with O(logn) regret for the multiarmed bandit problem”, Adv. Appl. Probab., vol.27, no.4, pp.1054-1078, Dec.1995

[4] Auer, Cesa-Bianchi and Fischer, “Finite-time analysis of the multiarmed bandit problem”, Mach. Learn., vol.47, pp.235-256, 2002

[5] S. Vakili, K. Liu and Q. Zhao, ”Deterministic Sequencing of Exploration and Exploitation for Mutli-Armed Bandit Problems”, IEEE Journal of Selected Topics in Signal Processing, vol.7, no.5, October 2013

[6] A. Anandkumar, N. Michael, A. Tang and A. Swami, ”Distributed Algorithms for Learning and Cognitive Medium Access with Logarithmic Regret”, IEEE JSAC, vol.29, no.4, April 2011

[7] K. Liu and Q. Zhao, ”Distributed Learning in Multi-Armed Bandit With Multiple Players”, IEEE Trans. Signal Process., vol.58, no.11, pp.5667-5681, Nov.2010

[8] Y. Gai and B. Krishnamachari, ”Decentralized Online Learning Algorithms for Opportunistic Spectrum Access”, IEEE Global Communications Conference (GLOBECOM 2011), Houston, USA, December, 2011

[9] K. Liu and Q. Zhao, ”Cooperative Game in Dynamic Spectrum Access with Unknown Model and Imperfect Sensing”, IEEE Transactions on Wireless Communications, vol.11, no.4, April 2012

[10] C. Tekin and M. Liu, ”Performance and convergence of multiuser online learning”, in Proc. Int. Conf. Game Theory Netw. (GAMENETS), April 2011

[11] J. Barcelo, B. Bellalta, C. Cano and M. Oliver, ”Learning-BEB: Avoiding Collisions in WLAN”, Stuttgart: Eunice Summer School, 2008

[12] J. Lee and J. Walrand ”Design and analysis of an asynchronous zero collision MAC protocol”, Arxiv Preprint 0806.3542, 2008

[13] M. Fang, D. Malone, K.R. Duffy, D.J. Leith, Decentralised learning MACs for collision-free access in WLANs”, Wireless Networks, v.19 n.1, p.83-98, January 2013

[14] D. Leith and P. Clifford, ”A self-managed distributed channel selection algorithm for WLANs”, in proc. of International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, pp.1-9, 2006