IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 3, 2021, accepted September 18, 2021, date of publication September 22, 2021, date of current version October 5, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3114430

Multi-Armed Bandits for Spectrum Allocation in Multi-Agent Channel Bonding WLANs

SERGIO BARRACHINA-MUÑOZ[1], ALESSANDRO CHIUMENTO[2], (Member, IEEE), AND BORIS BELLALTA[3], (Senior Member, IEEE)

[1]Centre Tecnològic de Telecomunicacions de Catalunya, 08860 Barcelona, Spain
[2]Pervasive Systems, EEMCS Faculty, University of Twente, 7522 NB Enschede, The Netherlands
[3]Department of Information and Communications Technologies, Universitat Pompeu Fabra, 08018 Barcelona, Spain

Corresponding author: Sergio Barrachina-Muñoz (sbarrachina@cttc.es)

The work of Sergio Barrachina-Muñoz and Boris Bellalta was supported in part by Cisco, Machine Learning for Wireless Networking in Highly Dynamic Scenarios (WINDMAL), under Grant PGC2018-099959-B-I00 [Ministerio de Ciencia e Innovación (MCIU)/Agencia Estatal de Innovación (AEI)/Fondo Europeo de Desarrollo Regional (FEDER), Union Europea (UE)] and Grant SGR-2017-1188. The work of Alessandro Chiumento was supported in part by the ECSEL Joint Undertaking (JU) through the Intelligent Secure Trustable Things (InSecTT) Project (https://www.insectt.eu/) under Grant 876038.

ABSTRACT While dynamic channel bonding (DCB) is proven to boost the capacity of wireless local area networks (WLANs) by adapting the bandwidth on a per-frame basis, its performance is tied to the primary and secondary channel selection. Unfortunately, in uncoordinated high-density deployments where multiple basic service sets (BSSs) may potentially overlap, hand-crafted spectrum management techniques perform poorly given the complex hidden/exposed nodes interactions. To cope with such challenging Wi-Fi environments, in this paper, we first identify machine learning (ML) approaches applicable to the problem at hand and justify why model-free RL suits it the most. We then design a complete RL framework and call into question whether the use of complex RL algorithms helps the quest for rapid learning in realistic scenarios. Through extensive simulations, we derive that stateless RL in the form of lightweight multi-armed-bandits (MABs) is an efficient solution for rapid adaptation avoiding the definition of broad and/or meaningless states. In contrast to most current trends, we envision lightweight MABs as an appropriate alternative to the cumbersome and slowly convergent methods such as Q-learning, and especially, deep reinforcement learning.

INDEX TERMS Channel bonding, spectrum allocation, multi-agent, reinforcement learning, multi-armed bandit.

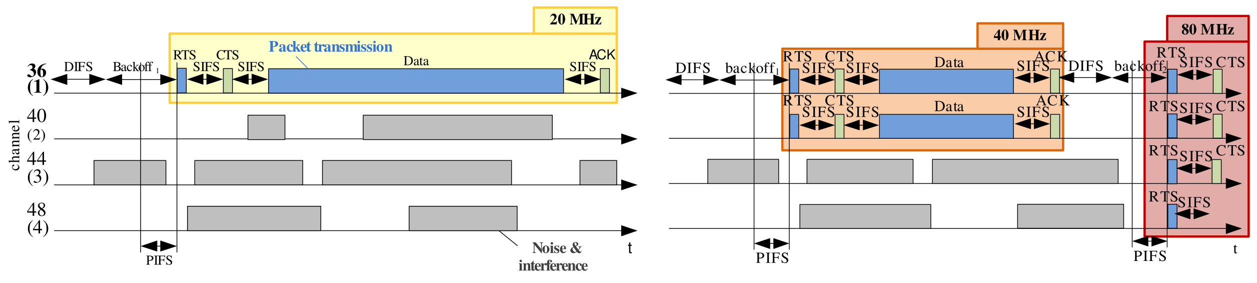
I. INTRODUCTION

State-of-the-art wireless applications like augmented reality, virtual reality, or 8K video streaming are urging next-generation wireless local area networks (WLANs) to support ever-increasing demands on performance. Among the different approaches to enhance spectrum efficiency in WLANs, channel bonding was introduced in 802.11n-2009 [2] for bonding up to 40 MHz and further extended in 802.11ac/ax [3], [4] and 802.11be [5] to bond up to 160 and 320 MHz, respectively, in the 5-GHz band.

Up to this date, the standards allow static and dynamic channel bonding (DCB). While the former is detrimental for high-density WLANs since only bonded transmissions are permitted, the latter adapts on a per-frame basis, so it can easily switch, e.g., from 160 MHz to 20 MHz from one frame to the next [6]. Accordingly, in this work, we study DCB as a preeminent technique of spectrum management in WLANs. Figure 1 illustrates the temporal evolution of single-channel and DCB.

In brief, WLAN spectrum management entails channel allocation and channel bonding. The main objective of channel allocation is to avoid interference between potentially overlapping nodes, and the main objective of channel bonding is to maximize the network capacity. Unfortunately, it is unavoidable per definition to transmit in higher bandwidths while reducing the interference with neighboring nodes: the objectives are, in principle, in contradiction. However, the trade-off between lowering interference and maximizing capacity has aspects that we can leverage. For instance, BSS's are not always saturated, which means that two overlapping BSS's may share a channel and support moderate

The associate editor coordinating the review of this manuscript and approving it for publication was Yiming Tang.



(a) Single-channel.

(b) Dynamic channel bonding (DCB).

FIGURE 1. Temporal evolution of (a) single-channel, and (b) DCB in IEEE 802.11ax channelization. While the duration of control packets is kept the same, the data duration is reduced by increasing the transmission bandwidth.

traffic loads. At the same time, by increasing the bandwidth, we decrease the power per unit-Hertz, reducing the absolute interference per channel as well, which may ultimately favor coping with the contention of large bandwidth transmissions.

Implementing efficient spectrum allocation and channel bonding is not straightforward because Wi-Fi networks are often complex systems where WLAN performance depends on a plethora of parameters. Consequently, there is a clear trend in the wireless community towards machine learning (ML), disregarding traditional spectrum management protocols based on pre-fixed rules or heuristics. heuristics [7]–[9].

In this regard, we addressed in [1] why supervised and unsupervised learning are not as suitable for the problem given the need for fast adaptation in uncertain environments, where it is not conceivable to model the nature behind the observed phenomena. In particular, we envisioned stateless reinforcement learning (RL) as the only ML candidate to achieve the intended purpose, which is not aimed to generalize to unknown deployments (i.e., learn the model) but to adapt to particular, most of the times unique, scenarios where neighboring BSSs may also change their spectrum management setups.

To that aim, this paper focuses on multi-armed bandits (MABs), a stateless RL formulation that enables faster *on-line* adaptation than temporal difference algorithms like Q-learning, and especially deep reinforcement learning (DRL). Other papers in the literature leverage RL techniques for spectrum management in wireless networks. However, most of the works rely on a variety of hard assumptions that can result in inaccurate or even misleading conclusions. In contrast, we explore the performance of RL in DCB WLANs by capturing both physical and medium access control (MAC) dynamics and stochastic buffering. By means of simulations through the Komondor wireless network simulator [10], this is the first work assessing the performance of RL algorithms in potentially overlapping WLANs with actual IEEE 802.11ac/ax channel bonding capabilities.

We propose a complete RL framework for the spectrum allocation problem in DCB WLANs and evaluate different RL algorithms in two different settings: a self-contained toy scenario with known optimal configurations [1] and generalized random deployments to benchmark the algorithms under different node densities and traffic load variations. Results show that the simplest MABs (especially exploration-first) outperform the rest both in terms of learning speed and mid/long-term performance. These results suggest that deploying lightweight MAB algorithms in domestic access points (APs) would boost the performance through fast adaptation to the environment while raising fairness.

We highlight the following contributions:

1) We extend the discussion in [1] on why supervised and unsupervised learning are not suitable for the problem and propose stateless RL as the best ML candidate.
2) We propose a general RL framework for spectrum management in uncoordinated DCB WLANs by depicting the actions, attributes, states, and reward functions.
3) We depict a complete MAB taxonomy and its application to the spectrum management problem, including reward, contextual, and temporal categorizations.
4) We study a holistic toy scenario dataset gathered by simulating all the possible system's spectrum management configurations to benchmark different RL approaches against the optimal configuration.
5) We show the effectiveness in terms of rapid learning of stateless RL in the form of *lightweight* MABs against other RL algorithms, even in high-density deployments with constant and varying traffic loads.
6) Finally, we address the tradeoff between fast adaptation and potential long-term performance resulting from reducing the action space.

II. A CHANGE OF PARADIGM TOWARDS REINFORCEMENT LEARNING

A. HEURISTICS LIMITATIONS

Multiple custom channel allocation and channel bonding solutions in wireless networks rely on statistics such as channel occupancy, packet reception rate, or color conflicts (e.g., [11]–[14]). We call these prefixed rule-based solutions as *heuristic* solutions in front of machine learning solutions, which we discuss later. Heuristics are low-complex solutions that work well in steady scenarios. However, in highly dynamic scenarios, their performance is severely undermined since heuristics are tied to the validity of recent observations,

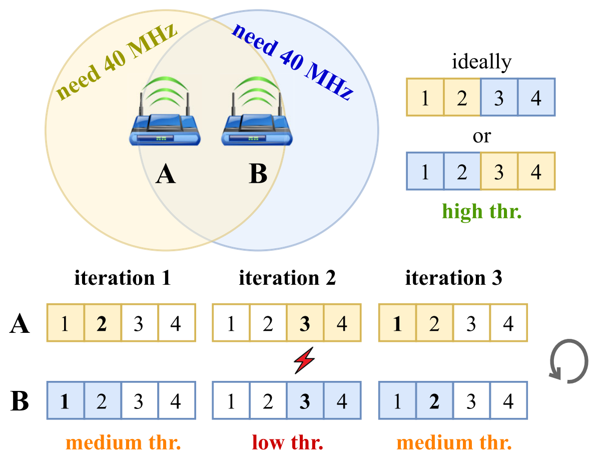


FIGURE 2. A toy example where throughput maximizing heuristics fall into detrimental loops. The primary channel in each iteration is highlighted with bold text for both BSSs.

and such depend on the speed at which the environment changes. Moreover, they do not consider the performance of previous actions, so *they do not learn*.

To illustrate the pitfalls of heuristic approaches in DCB WLANs, let us consider the toy scenario in Figure 2, with two potentially overlapping BSSs with DCB capabilities, A and B, in a system of $\times 4$ 20-MHz channels (so-called basic channels) following the IEEE 802.11ac/ax channelization. Assume also that both BSSs continuously require 40 MHz to satisfy their traffic load demands. Therefore, by allocating channels 1 and 2 to one BSS, and channels 3 and 4 to the other, the problem would be solved. However, in decentralized multi-agent settings, it is up to the BSSs to find satisfactory configurations.

A possible heuristic for this particular case could be to rely on the primary channel selection that is expected to provide the highest throughput given the occupancy of both the target primary and the neighboring secondary channels [15]. In this case, an AP periodically measures all the channels' occupancy. After each monitoring period, the AP switches to another primary if not satisfied with the current performance. For instance, BSS A, starting with primary channel 2 would switch to channel 3 or 4 with the same probability given that both were free during period 1, so they are likely to be good candidates to increase the throughput by bonding them. However, since BSS B also determines that switching from primary channel 1 to 3 would be beneficial, a loop is started. This occurs because observations are outdated from one period to the next, and action selection is never based on the past performance of the actions. It follows that heuristics get even less effective in larger multi-agent settings.

Learning from past experiences represents then a very strong alternative to pre-fixed rules. The capability of ML to go beyond rule of thumb strategies by automatically learning (and adapting) to (un)seen situations can cope with heterogeneous wireless network scenarios [7], [8].

B. TOWARDS RL SPECTRUM MANAGEMENT

Practically, in RL, an agent tries to learn how to behave in an environment by performing actions and observing the collected rewards. At a given iteration t, action a (e.g., switch primary channel) results in a reward observation (e.g., throughput) drawn from a reward distribution $r_t(a) \sim \theta_a$. In the context of WLANs, an agent could be installed into an AP that tries to maximize a certain performance metric by testing different configurations and adapting to the collected reward observations through trial-and-error.

In particular, since system interactions in problems such as decentralized spectrum management are critically complex and generated by multiple actors (i.e., wireless devices), it is often preferable to rely on model-free approaches. That is, after getting rid of strong assumptions like Markovian channels, it would not make sense to have an agent trying to infer what is the model behind the interactions perceived at the varying power levels detected at each of the channels. Moreover, even in the case where such interference model was stationary (and let us also assume simple enough to be accurately captured through a model), in the very moment that the agent-empowered AP initiated a transmission, the contention and interference generated to surrounding nodes could completely change the learned model. Thus, one can easily derive that having a giant model considering all the possibilities or a specific model per each configuration is merely unpractical.

C. WHY RL? ML IN DYNAMIC ENVIRONMENTS

Even though observations can be misleading in RL as well, agents rely on the learning performed through historical state/action-reward pairs, generating action selections policies beyond current observations, thus outperforming heuristics in dynamic scenarios. There are, however, two other main ML alternatives to RL: supervised learning (SL) and unsupervised learning (UL). We next justify why such categories are unpractical for this type of problem.

Supervised Learning (SL): Is the ML task of learning a function that maps an input x to an output y based on example input-output pairs (x_i, y_i). It infers a function from labeled training data consisting of a set of training examples. An SL algorithm then analyzes the training data and produces an inferred function, which maps new examples (i.e., generalizing). One way to use SL in our problem would be to try to learn the general and *true* WLAN behavior, function $f(x) = y$, through an estimate $h_\theta(x) = \hat{y}$, where $h_\theta(x)$ is the learned function. Then, once the hypothesis $h_\theta(x)$ is learned – i.e., the error is sufficiently low with respect to $f(x)$ – one could try to infer the input (or configuration) $\hat{x}^*$ that maximizes the performance $\hat{y}^*$, hoping that $\hat{x}^*$ would also result in the actual y^*.

The main three drawbacks of SL for our problem are *i*) the model design, which should consist of a vast number of attributes (e.g., nodes' locations, configuration parameters, performance metrics, etc), *ii*) the need for a lot of data to fit such a complex model, leading to the need for unmanageable training,[1] and *iii*) the target of SL is to generalize to unknown scenarios, which is not the focus of our problem because we want the agent to adapt to potentially infinite different worlds from scratch.

The only way to overcome such a task with SL would be to measure the performance of plentiful settings in a lot of scenarios (primarily offline), then try to infer some general behavior from the true and unknown function $y = f(x)$, where input x is the representation of the action setting and the environment around the agent, and finally predicting the performance $\hat{y} = h_\theta(x')$ for unlabeled (unknown) input x'. The significant issues are that the input domain of x is multi-dimensional (with even categorical dimensions) and tends to infinity. The function f representing WLANs' *behavior* is so complex that learning an accurate estimate h_θ in reasonable time is simply inconceivable. Moreover, there is another issue when trying to replicate Wi-Fi through (deep) supervised learning: the problem of estimating a function is different from the problem of maximizing an unknown function. That is, even though we could get a good estimator $h_\theta(x)$ of the *true* function, the optimal of such estimated function may be completely different than the *true* optimal of f. In other words, our oracle may work well for most of the inputs x but fail for the input x^* maximizing $f(x)$.

In **unsupervised learning** (UL), there are only inputs x and no corresponding output variables. That is, there is no y. UL's goal is to model the underlying structure or distribution in the observed data to learn more about it. It is called unsupervised because unlike SL there are no correct answers since samples are unlabeled. Algorithms are left on their own to discover potentially meaningful structures in the data. UL problems can be further grouped into clustering problems (to discover the inherent groupings in the data) and association (to discover rules that describe large portions of the data).

As in SL, the underlying structure of the observed data in the problem at issue is expected to be so complex that any attempt to model it through UL will be most likely fruitless. Hence, we believe that following an RL black-box approach is preferable: try to rapidly adapt from scratch to whatever the system's observations are, no matter its intrinsic nature.

III. RELATED WORK

There are many valuable works in the literature dealing with spectrum allocation and channel bonding in wireless networks (e.g., [11]–[14]). However, only a few of them treat the joint problem altogether in the context of WLANs, from which we highlight the following ones. A distributed spectrum assignment for home WLANs without control traffic is proposed in [16]. Continuous time Markov networks are employed in [17] to study a centralized approach for maximizing the network fairness. In addition, a heuristic-based algorithm for primary channel selection based on the bonding direction likelihoods was recently presented in [18]. However, such likelihoods are estimated by assuming a known number of users in each channel, which is normally not feasible in real deployments. It is worth noticing that the aforementioned works consider fully-backlogged traffic, thus missing insights on more realistic patterns with different traffic needs. Conversely, an uncertain traffic channel allocation approach was presented in [19]. Still, a centralized controller in the back-end is required. Recently, authors in [15] formulated a decentralized algorithm, adaptive in the sense that a new primary channel is only adopted when the WLAN performance is below a heuristic based on a throughput satisfaction threshold.

While the aforementioned algorithms show good performance under the studied scenarios, the complex dynamics of multi-agent channel bonding WLANs make it necessary to rely on some sort of learning in order to overcome hand-crafted alternatives. In this regard, different ML solutions for the spectrum management problem, especially in the form of RL, have been conducted in the last years. However, there are certain assumptions these papers have in common that hinder the accurate evaluation of ML solutions. For instance, most papers consider synchronous time slots (e.g., [20]–[25]), which is a strong assumption that does not hold at all in carrier sense multiple access with collision avoidance (CSMA/CA) WLANs. Also, some papers define a binary reward, where actions are simply good or bad (e.g., [20], [24], [26]). This, while easing the RL framework, hinders rewards taking into account continuous-valued performance metrics like throughput or delay. Further, most of the papers consider simple traffic patterns or even fully backlogged regimes (e.g., [27]–[29]), thus overlooking the effects of dynamic traffic loads. Also related to dynamism, most papers do not consider packet aggregation, which is a critical feature affecting the throughput and delay. To ease their analysis, some papers even consider conflict graphs rather than actual carrier sense areas (e.g., [21], [27]), thus disregarding spatial distribution effects. Last, but not least, a majority of the papers provide custom, ad-hoc RL solutions where the design of states, actions, and rewards is not justified, but simply formulated. This leaves short room for extrapolating such solutions to slightly different problems, and it even makes it harder to reproduce the presented results.

In this paper, we methodically introduce an RL framework for the spectrum allocation problem in DCB WLANs by justifying each of its components (e.g., actions). Besides, we study complex Wi-Fi scenarios, where the physical and MAC layers are not abstracted, but simulated with the Komondor wireless simulator [10], from which we derive conclusions on the performance of RL in WLANs otherwise not possible. We do so

[1] We state that generating a dataset rich enough to generalize the spectrum management problem in DCB WLANs is unfeasible since data samples should have large multi-dimensional attributes to be meaningful. So, for instance, simply by moving one STA one meter away, a new scenario (or data sample) would be generated.

in a reproducible way, by providing our code and algorithms will full detail.[2]

IV. MAPPING THE PROBLEM TO RL

This section depicts the modifiable attributes, actions, states, and rewards composing the learning framework for the joint problem of primary channel and maximum bandwidth allocation in DCB BSSs. Notice that by defining the primary channel and maximum bandwidth, we actually determine the secondary channels since we follow the IEEE 802.11ac/ax channelization.

A. SYSTEM MODEL

1) WLAN DEPLOYMENT AND AGENTS

We consider a static deployment of W potentially overlapping BSSs $\mathscr{W} = \{w_1, w_2, \dots, w_W\}$, each of them composed by one AP and one or multiple STAs. The TMB path-loss model is assumed [30] and spatially distributed scenarios are captured. Namely, we cover scenarios where the BSSs may not all be inside the carrier sense range of each other. Therefore, the typical phenomena of home Wi-Fi networks like flow in the middle and hidden/exposed nodes are captured as in [6], [15], [31].

2) CHANNELIZATION

We consider C 20-MHz channels (or *basic* channels), e.g., $C = 8$ from channel 36 to 64 following the IEEE 802.11ac/ax standards. Accordingly, a transmitter may bond up to $C \times 20$ MHz (e.g., 160 MHz if $C = 8$). Note that not any bonding combination is permitted by the IEEE 802.11ac/ax ammendments. For instance, 60 MHz transmissions (3 *basic* channels) are not permitted [3], [4]. As for the channel access, we assume all the BSSs to have DCB capabilities to bond up to C channels on a per-frame basis [6]. In particular, DCB is a standard-compliant channel bonding policy that adapts to the idle secondary channels on a per-frame basis, selecting the largest combination of basic channels available when the backoff terminates. Nevertheless, as explained later in § IV-B, the maximum number of channels to be aggregated is limited by the maximum bandwidth attribute, b. So, for instance, we can easily restrict a BSS to single-channel transmissions by setting such an attribute to $b = 1$. The adaptive RTS/CTS mechanism introduced in the IEEE 802.11ac standard [3] for dynamic bandwidth is considered, meaning that the receiver broadcasts the CTS only on the subset of secondary channels that was found available at the destination.

3) TRAFFIC GENERATION

We consider downlink data traffic from the AP to one STA at each data frame transmission. However, control packets in the uplink (CTS and ACK/BACKs) are also simulated. When multiple STAs are associated to an AP, the latter selects the destination STA uniformly at random. So, we take the performance of the whole BSS by averaging over the performance of its associated STAs. Data packets generated by the AP of a BSS w follow a Poisson process with mean duration between packets given by $1/\ell_w$, where ℓ_w is the mean load in packets per second of w [6], [15], [31]. In contrast to much of the works in the literature, we consider that the mean traffic load of every BSS may vary over time, thus introducing mid/long-term dynamism into the problem.

B. ATTRIBUTES AND ACTIONS

While it is clear that the primary channel is critical to the BSS performance since it is where the backoff procedure is carried, the maximum bandwidth is of significant relevance as well. Indeed, once the backoff expires, the channel bonding policy and maximum allowed bandwidth will determine the *basic* channels to transmit in. For instance, in the example in Figure 1b, if the BSS was allocated 40 MHz (channels 1 and 2) rather than 80 MHz (channels 1 to 4), DCB would bond the first 2 channels at the end of the second backoff, rather than 4. Notice that in some scenarios it is much appropriate to limit the maximum bandwidth to reduce adverse effects like hidden/exposed nodes. Next, we motivate why the set of possible maximum bandwidth values b has a deep impact on the BSS performance. So, the more flexible the allocation of b, i.e., the more values it can take, the higher the potential performance.

1) THE ROLE OF THE MAXIMUM BANDWIDTH

To illustrate why not restricting the maximum bandwidth can be counterproductive, let us consider a toy scenario facing a potential hidden node situation. The BSS in Figure 3a consists of BSSs A and B, each with one AP and one STA to serve. BSS A is allocated 160 MHz, $C_\text{A} = \{1, 2, \dots, 8\}$, whereas BSS B is allocated just 20 MHz, $C_\text{B} = \{3\}$. Figure 3b and Figure 3c represent the feasible system states through continuous Markov chains when restricting and not restricting A's bandwidth, respectively. Notice that STA A is close to AP B, so STA A experiences disturbing interference from AP B. Likewise, AP B is far enough from AP A to avoid contention whenever AP A bonds channel 3 (e.g., when using 80 or 160 MHz). Beyond contention, there is a potential hidden node issue for STA A: if the power of the signal of interest (received from AP A) is not sufficiently high with respect to the interference power (received from AP B) and the background noise, STA A would not be able to decode any packets from AP A. In particular, assuming a constant transmission power, STA A would need AP A to transmit in 20 or 40 MHz at maximum to receive enough power of interest to decode the packets. So, in this example, AP A should not transmit in 80 or 160 MHz whenever AP B is transmitting, even when AP A encounters its full allocated spectrum free at the backoff termination.

With this simple example, we corroborate that, while channel bonding is necessary to improve the satisfaction under

[2]All of the source code used in this paper is open, encouraging sharing of algorithms between contributors and providing the ability for people to improve on the work of others under the GNU General Public License v3.0. The code used in this work can be found at https://github.com/sergiobarra/MARLforChannelBondingWLANs

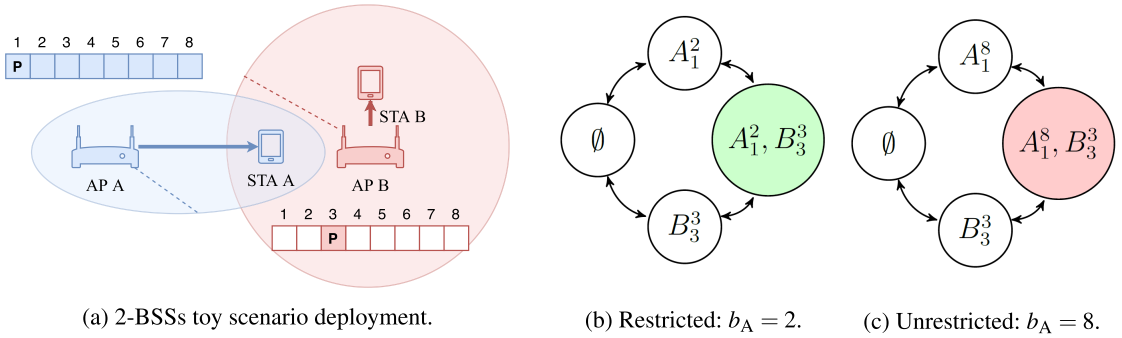


FIGURE 3. The effect of the maximum bandwidth *b*. BSS B (always using single-channel) interferes with STA A, which perceives: (b) negligible interference and (c) critical interference.

demanding traffic loads, not limiting the maximum bandwidth may also raise drawbacks, and such limitation should be adapted to each scenario based on the heterogeneity and chaotic nature of Wi-Fi deployments.

2) ACTION SPACE

Given the relevance of the primary channel and the maximum bandwidth, we consider below two configurable attributes each agent-empowered BSS can modify during the learning process:

- *Primary Channel* $\boldsymbol{p}_w$*:* Primary channel where the backoff procedure is executed, $p \in C_w$, where C_w is the channel allocation of BSS w.
- *Maximum Bandwidth* $\boldsymbol{b}$*:* Maximum bandwidth (in number of basic channels), $b \in \beta = \{1, 2, 4, \dots, |C_w|\}$. Recall that, with DCB, the transmitter can adapt to the sensed spectrum on a per-frame basis. So, the bandwidth limitation just sets an upper bound on the number of basic channels to bond.

The size of the action space $\mathscr{A}_w$ of a particular BSS w is then

$$O(\mathscr{A}_w) = O(C_w) \times O(\beta_w), \tag{1}$$

where $O(\cdot)$ represents the number of elements or cardinality of a set, and every action, or spectrum configuration, $a = (p_w, b_w) \in \mathscr{A}_w$ is a pair of the primary channel $p_w \in C_w$ and maximum bandwidth $b_w \in \beta_w$ attributes. Should we consider a central single agent managing $W' \leq W$ BSS's, the action space increases exponentially to

$$O(\mathscr{A}) = \prod_{w=1}^{W'} O(C_w) \times O(\beta_w). \tag{2}$$

Definition 4.1 (Spectrum Configuration): The spectrum configuration of a BSS w is defined as the pair primary channel p_w and maximum bandwidth b_w.

If we assume that BSS's have the same attribute spaces C and β, $\forall w$, then we clearly observe the exponential growth of the whole action space $O(\mathscr{A}) = \Big(O(C) \times O(\beta)\Big)^{W'}$.

C. STATUSES, OBSERVATIONS AND STATES

Before addressing what is a *state*, which its definition highly depends on the RL framework in use, let us coin a new concept called system *status*.

Definition 4.2 (System Status): The system status $\mathscr{H}_t$ is the minimum *information* we need to describe the entire WLAN in a given time instant t.

The system status $\mathscr{H}_t$ of a static WLAN deployment[3] at time t is given by three key parameters of each of the W' agent-empowered BSS in the WLAN like if we were taking a screenshot of the system: the primary channel $p_{w,t}$, the maximum bandwidth $b_{w,t}$, and the quantized mean load $\ell_{w,t}$. Formally, a status is defined by the set

$$\mathscr{H}_t = \{(p_{1,t}, b_{1,t}, \ell_{1,t}), \dots, (p_{W',t}, b_{W',t}, \ell_{W',t})\}. \tag{3}$$

First, notice that we only consider the attributes of the agent-empowered BSS's since the rest are not allowed to modify their configurations; thus, they do not convey any extra information for the system status. For instance, let us consider a deployment with just two BSS's, A and B. Assume A is agent-empowered, whereas B is not, so $W' = 1$ and $W = 2$. The system information relies just on the configuration of A since B will never change its own. So, the number of system statuses is just the number of possible configurations A can take.

Second, we rely on a quantization $\mathscr{L}$ of the mean traffic load ℓ to discretize its continuous domain. Further, notice that the agent cannot modify the load ℓ by any means since it is application-dependent. Naturally, the performance of a BSS depends not only on its traffic load and spectrum configuration but also on the rest of BSS's configurations and loads.

Definition 4.3 (World): The world (or environment) of a particular BSS w is defined as the combination of spectrum configurations and loads of each of the rest BSS's.

From the point of view of a particular BSS w, the combination of current spectrum configurations and loads of the rest BSS's compose its *world* or environment, which is

[3]Static refers to the fixed locations of all the nodes in the BSS's, including APs and STAs.

determined as

$$\mathcal{I}_{w,t} = \{p_{w',t}, b_{w',t}, \ell_{w',t} | w' \neq w\}. \tag{4}$$

Finally, the amount of possible statuses a certain WLAN deployment may have is given by the possible values each of the status parameters can take,

$$O(\mathcal{H}) = \prod_{w=1}^{W'} O(p_w)O(b_w)O(\ell_w), \tag{5}$$

which raises again exponentially with the number of agents and size of the attributes' spaces. Assuming all the agent-empowered BSS's have the same capabilities,

$$O(\mathcal{H}) = \Big(O(p)O(b)O(\ell)\Big)^{W'}. \tag{6}$$

Generally, the agents cannot know the system status in a real-time fashion. However, they could rely on partial and delayed observations of it to then infer an *state* through some discretization function. Hence, a state is a mapping of the observation a BSS makes of its world into a manageable piece of information about such a world. However, since observations are architecture-dependent – given they may be limited to local sensing capabilities or, on the contrary, be shared via a central controller – the states' definition is also strictly dependent on the RL model under consideration.

D. PROBLEM DEFINITION

In general terms, our goal is to improve the BSS's performance (e.g., in terms of throughput, delay, energy efficiency, etc). The key then is to let the BSS's find (learn) the best actions (p, b), where *best* depends on the problem formulation itself. For instance, we could assess a max-min approach to maximize the throughput satisfaction experienced by the less favored BSS. Another common aim in network management is to boost the fairness of some performance metrics among contending BSS's in the BSS. This paper does not stick to one problem formulation, but we instead evaluate different performance metrics, including individual and aggregated ones.

In any case, we must define the individual performance metrics of each BSS to assess whether such metrics tend to improve (i.e., the learning is working) or not. As a particular use case, we focus on the throughput satisfaction ξ, simply defined as the ratio of throughput $\Gamma_{w,t}$ to generated traffic load $\ell_{w,t}$,

$$\xi_{w,t} = \frac{\Gamma_{w,t}}{\ell_{w,t}}. \tag{7}$$

So, $\xi_{w,t} \in [0, 1], \forall w, t$. The throughput satisfaction is a commonly used parameter to indicate the ratio of packets acknowledged to the number of packets generated by a BSS in a given time window. Thus, a satisfaction value $\xi = 1$ indicates that all the traffic has been successfully received at the destination.

Once the main performance metric has been defined, we may formulate the reward function R, the cornerstone of any RL algorithm,

$$R(\xi) = \xi. \tag{8}$$

In this case, we assume the reward is just throughput-based, but other performance metrics could be included in such formulation. Regardless of the reward function R, we aim at maximizing the cumulative reward *as soon as possible*,

$$G = \sum_{t=1}^{T} r_t, \tag{9}$$

where T is the number of iterations we have to execute the learning, and r_t is the reward at iteration t. Let us emphasize the complexity of the problem by noting that r_t depends on the reward definition R, on the action a_t selected at t, and also on the world at each iteration $\mathcal{I}_{w,t}$.

V. MULTI-ARMED BANDITS

Among model-free RL approaches, we highlight MABs, Temporal Difference (TD) methods like Q-learning, and deep reinforcement learning (DRL) methods like deep Q-learning (DQN).[4] Fig. 4 depicts a schematized representation of each of them in order of increasing complexity.

Different variations of these approaches have been applied to the spectrum management problem in wireless networks. However, as stated in [1], a key benefit of stateless RL formulations like MABs is the absence of states, which greatly simplifies design by relying only on action-reward pairs. Accordingly, we propose to have a MAB instance in each agent-empowered AP to adapt to the environment without the need of states. The reason lies in the fact that meaningful states are normally multidimensional and their effectiveness depends heavily on the application and the type of scenario under consideration. In the end, a state is a piece of information representing the environment that is expected to help the agent by fitting it into its policy. Then, if the state is meaningless, or worst, misleading, it is preferable to not rely on it and go for a stateless approach.

A. THE MAB PROBLEM

The multi-armed bandit problem is a classic RL problem that exemplifies the exploration vs. exploitation dilemma, where no knowledge of the reward probability distribution of each action is assumed. The typical example to introduce the MAB problem is the gambler who faces multiple slot machines, each with different probability distributions over the money he/she can get after pulling the slots. Naturally, the gambler wants to get as much money as possible, thus achieving the highest reward in the mid/long-term. As for what strategy to follow, MAB formulations arise in different variations. For instance, a naive approach would be to pull each machine slot

[4]We discard Monte Carlo methods since they require episodes to eventually terminate in order to work, which contrasts our vision of WLANs in continuous online adaptation in a so-called infinite episode.

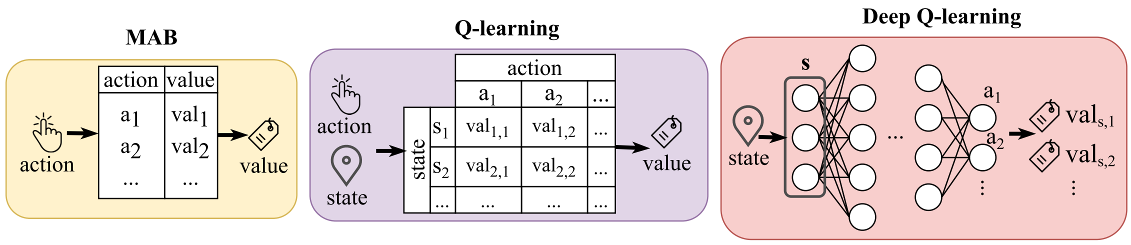


FIGURE 4. Learning storage in MABs, Q-learning, and Deep Q-learning.

a lot of times, so the gambler could guess the "authentic" reward probability of each slot according to the law of large numbers, and then select always the best slot machine. However, such an approach has obvious pitfalls like determining how many times are enough to discover the "true" reward probability or wasting too many trials (or iterations) in a priori lousy machines. Further, in environments where the distribution of such rewards may change due to external factors (as is the case in WLANs), it would not be viable to get accurate estimates of the reward probability distributions. Simply because from one iteration to the next, the underlying reward distribution may be completely changed.

Formally, a MAB can be described as a tuple of the set of actions and the reward function $<\mathscr{A}, R>$, where:

- There are K arms (slot machines in the previous example or spectrum management configurations in this paper) with hidden reward probabilities $\{\theta_1, \dots, \theta_K\}$. $\mathscr{A}$ is the set of actions (or action space), each referring to the interaction with one arm.
- At each time step (or iteration) t, we take an action (or arm) a and observe a reward r_t resulting from performing such action.
- We refer to the *value* of an action a as the expected reward, $Q(a) = \mathbb{E}_{\theta_a}[r|a]$, determined by the reward distribution θ_a.
- Finally, the reward function R defines the reward values obtained at any time step t, $r_t = R(a_t)$. For instance, $R(a_t)$ could be defined in the previous example as the money earned in the last slot pull or as the throughput experienced by a WLAN when using action a_t during the last iteration t.

Notice the difference between $R(a_t)$ and $Q(a_t)$. While $r_t = R(a_t)$ is the *actual* reward observed when applying action a_t in iteration t (e.g., throughput observed when playing a_t at iteration t), $Q(a_t)$ is the *expected* value of playing a_t (e.g., mean future throughput when playing a_t regardless t). In other words, $R(a_t)$ is a particular outcome of the reward distribution $\theta(a_t)$, whereas $Q(a_t)$ is a representation (or statistic) of the arm's value (e.g., mean reward).

The goal is to maximize the cumulative reward G over a finite number of iterations T (9). The simplest action selection rule (or policy π) is to select one of the actions with the highest estimated value, that is, one of the greedy actions, $\pi_t = a_t = \arg\max_{a \in \mathscr{A}} Q_t(a)$. However, greedy selection always exploits current knowledge to maximize the immediate reward, i.e., it does not sample the rest of a priori inferior actions. So, it might be the case that the previously selected action is sub-optimal in the long term. There are more sophisticated selection rule alternatives like ϵ-greedy methods, which force exploration with some probability ϵ to achieve better knowledge on the rewards of the actions in the whole action space. We depict some of the most relevant ones in §V-C.

B. MAB TAXONOMIES

MABs can be categorized according to three main concepts: the type of reward, the use of contexts, and the temporal horizon.

1) REWARD TAXONOMY

According to their reward distribution, there are two main classes of MABs: stochastic and adversarial. The **stochastic** MAB is the most common one, and it is completely determined by the distributions of rewards $\{\theta_1, \dots, \theta_K\}$ of the actions in the action space. In particular, at iteration t, the distribution of the reward observed by a learner that chooses action $a_t \in \mathscr{A}$ is θ_a, regardless of the past rewards and actions. The rewards for each action are independent and identically distributed (i.i.d). That is, every time an action is chosen, the reward is sampled independently from this distribution. Both for simplicity and for guaranteeing theoretical convergence in certain MAB algorithms, the reward function is bounded, $r_t = R(a_t) \in [0, 1], \forall a, t,$ so normalized performance metrics (e.g., throughput satisfaction) are normally used. The main limitation of stochastic MABs is the difficulty of finding real-world problems that rely on suitable distributions. Indeed, rewards can be non-stationary; thus, the reward distribution probability for a given action may change over time. Next, we present adversarial MABs, a paradigmatic example of non-stationary rewards.

In **adversarial** MABs, learning is still possible in the sense that the regret can be kept sub-linear. However, selection rules methods must be completely different from the stochastic ones since adversarial rewards can be arbitrary as if they are chosen by an "omniscient adversary", so the stochastic assumption on the rewards being generated from a fixed distribution does not longer hold. In plain, there are no probability distributions to learn whatsoever. Then, what can the agent do to face such an adversary? The key idea is to introduce noise in the exploration to choose random actions that the adversary could not expect.

So, what class of MAB suits better the joint spectrum management problem? We anticipate that the stochastic one. Even though the different actions' reward distributions are generally complex to estimate, no omniscient adversary tells what the rewards will be at every iteration. Instead, we have different actors (nodes) that behave according to a certain configuration. Such configurations determine what the actors will do, resulting in a stochastic setting. Nonetheless, even though the problem is likely to be stochastic, it is so complex that an adversarial approach may also be appropriate to design a fruitful algorithm. Accordingly, we assess in further sections the convenience of both stochastic and adversarial MABs.

2) CONTEXT TAXONOMY

While the general MAB formulation does not consider states, *contexts* may be used to tune the arm selection based not only on the gathered rewards but also on further observations about the world $\mathscr{I}$. Essentially, we may categorize MABs in context-less and contextual MABs. Context-less is the general formulation where no contexts are assumed.

Conversely, in **contextual** bandits, reward distributions depend on a context, which is fit to the RL algorithm before making a decision. Now, the observed reward r_t in each iteration t depends both on the context Ω_t and the chosen action a_t. For instance, assume that an AP observes that a given basic channel, say basic channel 2, is busy most of the time. The derived context would then represent that channel 2 is active. Thus, it is not the same to apply the action corresponding to pick primary $p = 2$ when the context indicates channel 2 is busy rather than applying the same action when such channel is idle. Henceforth, it seems that contexts may aid the spectrum management problem, especially if we leverage knowledge from previous contexts (e.g., similar contexts stochastically repeated every day). However, contextual MABs also pose the challenge of appropriately defining contexts, meaning that a good representation of the world $\mathscr{I}$ should be captured in each context.

There are different approaches to the contextual bandit problem. The simplest one, while the most versatile, considers a separate MAB per context (as if each context was a state). Then, the goal is to find the optimal action $a^*(\Omega_t) = \pi^*(\Omega_t)$ per context Ω_t. Defining effective contexts is challenging. And what would be an appropriate context definition? Any observed parameter in the MAB's world might be chosen, such as the current action $\Omega_t = a_t$, the current reward $\Omega_t = r_t$ or possible combinations such as a mix of the action selected and the reward $\Omega_t = f(a_t, r_t)$ or A mix including also historical information $\Omega_t = f(\{a_{t-t'}, \dots, a_t\}, \{r_{t-t'}, \dots, r_t\})$ and so on. Unfortunately, we do not find any definition that suits our multi-agent and non-stationary reward problem. The reason lies in the fact that we want to boost the learning speed to raise the user experience in the short/mid-term. Then, relying on a different MAB per action in a contextual fashion would entail large and potentially unfruitful learning periods.

3) TEMPORAL TAXONOMY

The last taxonomy we identify is the temporal one, defined by the time limit imposed to the MAB to converge to a good solution, which determines the *horizon*: finite or infinite.

Finite-Horizon: Refers to when there is a need to perform (explore and exploit) during a given time window (e.g., 5 minutes). So, the number of iterations T available to the agent is finite: the RL algorithm must try to maximize the accumulated reward (9) within that time limit.

Infinite-Horizon: In contrast, poses no time limit to the agent to converge, i.e., $T \rightarrow \infty$. As a matter of fact, in many problems, a finite time horizon cannot be easily specified, so the infinite horizon formulation fits more naturally. More importantly, stationary problems with infinite time horizon lead to optimal stationary strategies, which offer great simplicity.

For the joint spectrum management problem of this paper, we will rely on finite horizons to assess the *myopic* nature of the MABs. We call it myopic in the sense that they try to optimize rewards now without any explicit use of forecast information or any direct representation of decisions in the future. Meaning we want WLANs to improve their performance as soon as possible.

C. MAB EXPLORATION STRATEGIES

We depict the different action selection strategies of choice below.

1) EXPLORATION-FIRST

The simplest MAB algorithm dedicates the first T_{exp} rounds to exploration, and the remaining $T - T_{\text{exp}}$ rounds to exploitation. In this paper, we mandate the full exploration of the action space, i.e., $T_{\text{exp}} = K$ in order to explore all the arms. Notice that in the event of an environment change after the exploration phase, action rewards may drastically change. Exploration-first copes with this issue by simply ranking actions according to their last observed reward, i.e., the estimated action value $\hat{Q}(a)$ of action a is set as the reward r of applying a the last time it was chosen. So, if the best action in iteration $t - 1$ turns out to be the worst in t, since its reward is updated, the second-best action in $t - 1$ is now the best in t and it is picked accordingly in $t + 1$. Consequently, exploration is implicitly carried when action rewards vary over time.

2) ϵ-GREEDY

The ϵ-greedy algorithm takes the best known action with probability $1 - \epsilon$, and explores a random action (previously explored or not) with probability ϵ [32]. The action value Q is estimated, such estimation being represented as $\hat{Q}$, according to the past experience by averaging the rewards associated with the target action a that we have observed so far (up to the current time step t), i.e., $\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{\tau=1}^{t} r_\tau \mathbb{1}[a_\tau = a]$, where $\mathbb{1}$ is a binary indicator function telling whether the action a was used in a given iteration, and $N_t(a)$ is the number of times the action a has been selected so far, i.e., $N_t(a) = \sum_{\tau=1}^{t} \mathbb{1}[a_\tau = a]$. In an exploring iteration, any action is picked with same probability, whereas in an exploitation iteration, the best estimated action is picked, i.e., $a = \arg\max_{a\in\mathscr{A}} \hat{Q}_t(a)$. To avoid inefficient exploration after *enough* iterations, we decrease the parameter *epsilon* in time to reduce the probability of exploring over time [33].

3) UCB1

Random exploration allows trying out options that we have not known much about. However, due to the randomness, we may end up exploring a bad action that we have confirmed in the past. To avoid such inefficient exploration, one approach is to decrease the parameter ϵ in time (as done in ϵ-greedy), and the other is to be *optimistic* about options with high uncertainty and thus to prefer actions for which we have not had a confident value estimation yet. Or in other words, we favor the exploration of actions with a strong potential to have an optimal value. The Upper Confidence Bounds (UCB1) algorithm, introduced first in [34] and further analyzed in [33], measures this potential by an upper confidence bound of the reward value, $\hat{U}_t(a)$, so that the true value is below the bound, $Q(a) \leq \hat{Q}_t(a)+\hat{U}_t(a)$ with high probability. The upper bound $\hat{U}_t(a)$ is a function of $N_t(a)$; a larger number of trials $N_t(a)$ should give a smaller bound $\hat{U}_t(a)$. In UCB1, we always select the greediest action to maximize the upper confidence bound: $a_t = \arg\max_{a\in\mathscr{A}} \hat{Q}_t(a)+\hat{U}_t(a)$. For the cases where no prior knowledge on the distributions is given, the upper confidence bound can be derived as $\hat{U}_t(a) = \sqrt{\frac{2\log t}{N_t(a)}}$. So, by looking at $\hat{U}_t(a)$, for a given value estimation $\hat{Q}_t(a)$, we see that the more times an action is picked – the larger $N(a)$ – the less probable it becomes since the uncertainty about its reward distribution is reduced.

4) THOMPSON SAMPLING

In the previous algorithms, we do not assume any prior on the reward distribution θ_a of action a and therefore we have to rely on pure randomness or a generalized estimation of the upper confidence bound. If we can know the distribution upfront, we would be able to make better bound estimations. For example, if we expect the mean reward of every arm to be Gaussian, we can set the upper bound as 95% confidence interval by setting $\hat{U}_t(a)$ to be twice the standard deviation. Bayesian bandits follow a well-known approach from Bayesian statistics: posit that the unknown quantity is sampled from a known distribution, and optimize in expectation over this distribution [35].

At each time t, we draw a sample from the prior distribution of every action $a \in \mathscr{A}$, $\tilde{Q}(a) \sim \theta_a$. The best action is then selected among the drawn samples: $a_t = \arg\max_{a\in\mathscr{A}} \tilde{Q}(a)$. After the last actual reward $r(a_t)$ of the selected action a_t is observed, we update its reward distribution parameters (e.g., mean and standard deviation for the normal distribution, or α and β parameters for the Beta distribution) accordingly, which is essentially performing Bayesian inference to compute the posterior with the known prior and the likelihood of getting the sampled data. In this work, we assess two widely used distributions for Thompson Sampling: Beta distribution and Gaussian distribution.

5) EXP3

Unfortunately, reward probability distributions are so complex that having optimism is arguably naive, especially when it comes to competitive scenarios like WLANs. To conceive rewards that could operate in any manner, the adversarial model was proposed, where the agent must face an omniscient adversary that secretly chooses a sequence of rewards at the start of the game. An illustrative example made by Auer is to think about a gambler that plays in a rigged casino [36]. In this work, we use the well-known algorithm for adversarial bandit learning, the Exponential-weight algorithm for Exploration and Exploitation (Exp3) [37].

Exp3 works by maintaining a list of weights for each of the actions, using these weights to decide which action to take next at random, and increasing (decreasing) the relevant weights when a reward is good (bad). We further introduce an egalitarianism factor $\gamma \in [0, 1]$ which tunes the desire to pick an action uniformly at random ($\gamma > 0$ to avoid probabilities P too close to 0). That is, if $\gamma = 1$, the weights have no effect on the choices at any step. In other words, the distribution P is a mixture of the uniform distribution and a distribution that assigns to each action a probability mass exponential in the estimated cumulative reward for that action. The uniform distribution encourages exploration, whereas the other probability encourages exploitation. Then, parameter γ controls the exploration.

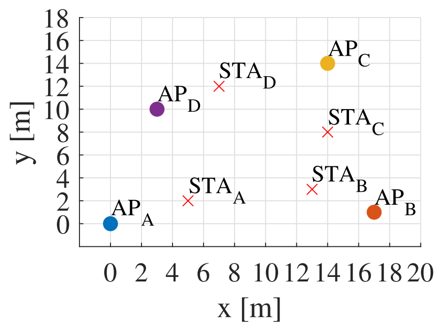
VI. EVALUATION

In this section, we evaluate the performance of the MABs depicted in §V-C. We first benchmark them against the optimal global configuration in a toy self-contained deployment, from which we know the optimal configuration for every traffic load. Then, we generalize the results to denser random deployments.

A. A SELF-CONTAINED DATASET

We study the deployment presented in [1] illustrated in Figure 5a, consisting of $W = 4$ BSS's (with one AP and one STA each) in a system of $C = 4$ basic channels. The traffic load is quantized and can take three values, $\ell_w \in \mathscr{L} = \{20, 50, 150\}$ Mbps $\forall w$.[5] As for the action attributes, the primary channel can take any of the C channels in the system, $p \in \{1, 2, 3, 4\}$, and the maximum bandwidth is allowed to be set to $b \in \{1, 2, 4\}$ fulfilling the IEEE 802.11ac/ax channelization restrictions for 20, 40, and 80 MHz bandwidths.

[5]The 4-BSS's self-contained dataset for spectrum management in BSS's can be found at https://www.upf.edu/web/wnrg/wn-datasets



(a) Location of nodes.

	A	B	C	D
A	-	40	20	80
B	40	-	80	40
C	20	80	-	80
D	80	40	80	-

(b) Interference matrix.

FIGURE 5. Toy deployment of the self-contained dataset.

TABLE 1. Komondor simulation parameters.

Param.	Description	Value
f	Central frequency	5180 MHz
C	No. of 20-MHz channels	8
P_{tx}	Transmission power	14 dBm
PL	Path loss model	TMB [30]
MCS	MCS index	0-11
CCA	CCA threshold	-82 dBm
CE	Capture effect threshold	10 dB
N_o	Background noise level	-95 dBm
L_{d}	Length of a data packet	12000 bit
N_{a}	Max. no. of agg. packets per frame	64
T_{e}	Duration of an empty slot	9 µs
T_{SIFS}	SIFS duration	16 µs
T_{DIFS}	DIFS duration	34 µs
T_{PIFS}	PIFS duration	25 µs
L_{RTS}	Length of an RTS	160 bits
L_{CTS}	Length of a CTS	112 bits
L_{ACK}	Length of an ACK	112 bits
L_{BACK}	Length of a block ACK	$\leq$ 432 bits
CW$_{\text{min}}$	Min. contention window	16
m	No. of backoff stages	5

So, according to (5), the total number of statuses when considering all BSS's to have an agent-empowered AP (i.e., $W' = W = 4$) raises to $O(\mathscr{H}) = (4 \times 3 \times 3)^4 = 1,679,616$. Notice that even a petite deployment like this one leads to a vast number of possible statuses.

The interference matrix elements in Figure 5b indicate the minimum bandwidth in MHz that causes two APs to overlap. So, we note that this deployment is complex in the sense that multiple different one-to-one overlaps appear depending on the distance and the transmission bandwidth in use since the higher the bandwidth, the lower the power per Hz. This leads to exposed and hidden node situations hard to prevent beforehand. For instance, AP$_{\text{A}}$ and AP$_{\text{C}}$ can only overlap when using 20 MHz, whereas AP$_{\text{A}}$ and AP$_{\text{D}}$ do always overlap regardless of the bandwidth because their proximity. Instead, AP$_{\text{A}}$ and AP$_{\text{B}}$ overlap whenever 40 MHz (or 20 MHz) bandwidth is used.

To generate the self-contained dataset, we simulated all the possible statuses $\mathscr{H}$ in Komondor v3.0.1b [10].[6] So, we first deploy the APs and STAs in fixed positions in the map (as in Figure 5a). Then, we define the values of the modifiable attributes and the mean traffic load experienced by each BSS. Likewise, we set the configuration capabilities of each BSS, so we empower each agent with the action space $\mathscr{A}_w = \{\{p_w\}, \{b_w\}\}$. For simplicity, we consider all the BSS's to have the same action space $\mathscr{A}_w = p \times b, \forall w$. Finally, for every BSS's global configuration, we must generate all the input files corresponding to each possible status $\mathscr{H}$ to be later simulated with Komondor. Notice that the input $x = \mathscr{H}$ is mapped to the output y, where y is an array containing multiple performance metrics of each BSS, including throughput and delay. Every status $\mathscr{H}$ is evaluated during $T_{\mathscr{H}} = 5$ seconds. Given that the BSS remains steady during any status, we empirically found that 5 seconds is enough to accurately estimate the stationary performance of a system status. We use Wi-Fi parameters according to 802.11ax [15] as detailed in Table 1.

[6]In this work, we directly tackle the simulation analysis of RL algorithms given the high complexity of dynamic channel bonding WLAN deployments, which are unfeasible to accurately assess through analytical models.

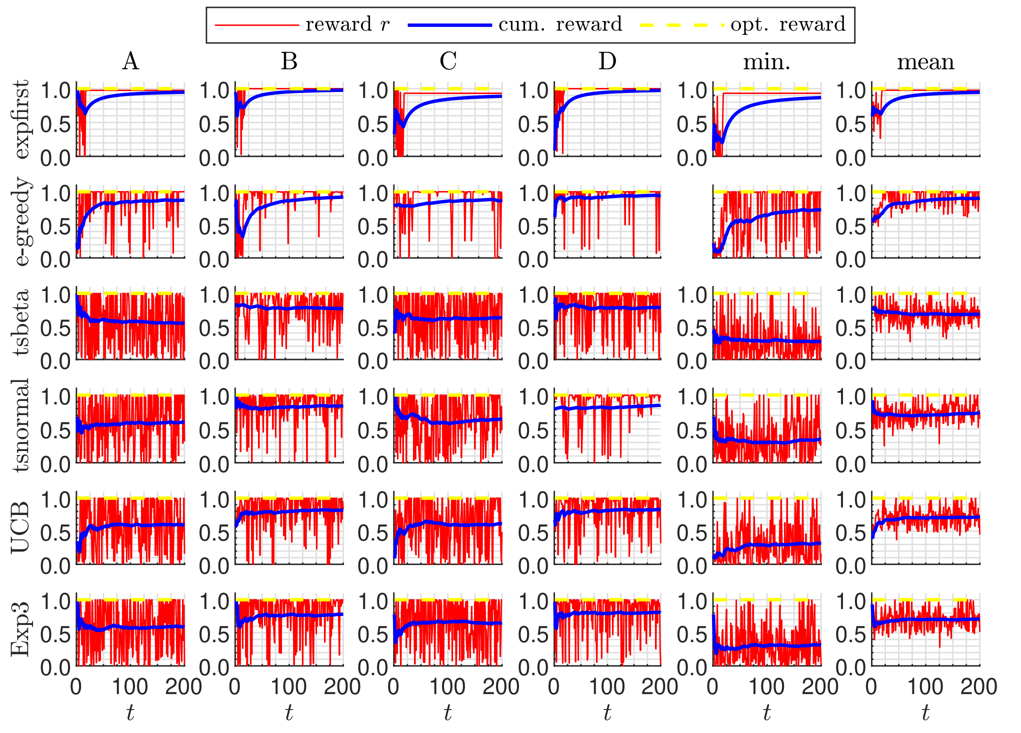
Once we gather the whole dataset of the deployment through Komondor, we use it to benchmark the performance of the MAB algorithms depicted in §V-C under different traffic patterns. For the sake of explanation, assume that the WLAN starts at $t_0 = 0$ with status $\mathscr{H}_0$. At t_1, one agent-empowered BSS picks an action a_1 (e.g., switches the primary channel), leading to status $\mathscr{H}_1$. Notice that, since we already have the dataset generated, it is straightforward (and fast) to check the performance y of all the BSS's at such given status. We essentially use the dataset as a lookup table $x = \mathscr{H} \rightarrow y$, to speed up the computation speed of RL simulations.

B. BENCHMARKING RL ALGORITHMS

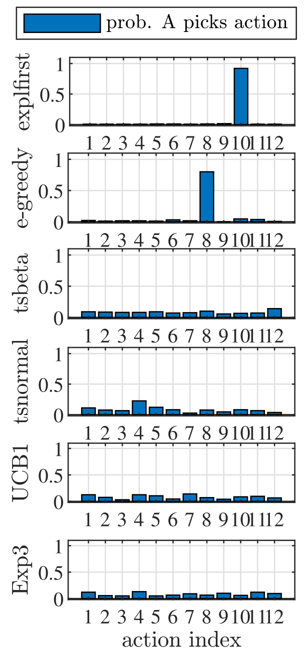
1) LEARNING UNDER CONSTANT TRAFFIC DEMANDS

We first aim at assessing the learning capacity, including the achievable reward and the corresponding convergence speed of the presented MABs in a BSS with constant traffic loads. The purpose is to determine whether MABs converge to *good* rewards under steady load conditions and, if so, how long does the process take.

We assume each BSS has a high average traffic load, $\ell_w = 50$ Mbps, $\forall w$. So, this experiment's long-term dynamism is due exclusively to the MABs progression in the multi-agent



(a) Run chart of the instant, cumulative, and optimal reward of each BSS, minimum and mean of the BSS under fixed traffic loads.



(b) Prob. BSS A to select each action.

FIGURE 6. **Self-contained toy dataset under constant traffic loads.**

setting. The inner traffic load condition does not vary, but the spectrum management configurations (or actions) do. We run a single simulation for each MAB intending to display a realization of how they behave.

Figure 6a shows the throughput satisfaction evolution of the MABs. For each MAB, and each BSS w, we plot the instant reward $r_{w,t}$, the normalized cumulative reward $G_{w,t}/t$, the optimal reward of each separate BSS, and the minimum and maximum optimal of the whole BSS. Notice that some BSS's can only achieve their optimal by forcing the others to underperform, so we refer to the minimum of the BSS ("min." in the Figure) as the most important metric to assess the MAB fairness. We observe that while exploration-first and e-greedy do converge (learn) to higher satisfaction values close to the optimal, the rest of MABs seem not to learn at all. This relates to the fact that learning based on parameter estimation is fruitless in dynamic and chaotic multi-agent deployments like this. So, it seems preferable to act quickly with a low-level knowledge of the actions' rewards. Besides, we find that win-win relations are established as shown by the minimum and mean convergence of the whole BSS. That is, on average, BSS's tend to benefit from others' benefits. As for the learning speed, exploration-first takes approximately a full exploration of the action space with erratic rewards to provide a super-steady performance, whereas e-greedy keeps switching configurations throughout the simulation.

Figure 6b shows the probability of BSS A (picked as an example) to select each possible action. We observe that exploration-first and e-greedy tend to use one single action that behaves *sufficiently* well, so they waste less time in exploring than exploiting. In contrast, the rest of MABs *waste* too many iterations exploring the different actions.

Finding: In small deployments with constant traffic load, exploration-first and e-greedy (so-called *ligthweight* MABs) achieve near-optimal rewards after *enough* exploration, showcasing their ability to learn in multi-agent Wi-Fi deployments by establishing win-win relationships between BSS's. Further, exploration-first is the only one to lead to a super-steady performance at the cost of experiencing vacillating rewards during the early exploration phase.

2) THE USEFULNESS OF MAB PARAMETERS INTO THE SPOTLIGHT: WHY DO SOPHISTICATED MABs NOT WORK?

From the previous experiments, we concluded that UCB1, Thompson Sampling, and Exp3 are not an option for the multi-agent setting of the joint spectrum management problem in BSS's. First, UCB1 cannot benefit from computing an estimation of the upper confidence bound of any action a. In essence, the action selection relies on the sum of such confidence bound and the estimated value $\hat{Q}(a)$. Given the non-stationarity of the multi-agent setting, the estimation $\hat{Q}(a)$ is lousy and may highly vary from one iteration to the other. The algorithm then falls in a loop where most of the

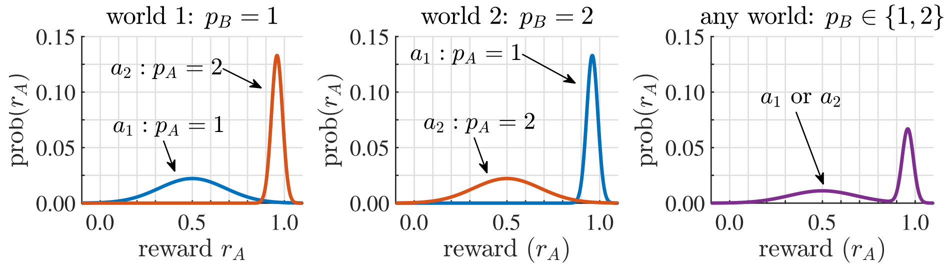


FIGURE 7. **Probability distributions of the reward of each action BSS A can take.**

actions are selected with similar probability, thus wasting lots of iterations with low performance and generating even more dynamism to the BSS.

Second, even though Thompson Sampling is usually known to outperform the rest of the presented MABs in stationary environments [38], its performance is well below exploration-first and e-greedy under the multi-agent setting. Similarly to UCB1, the reason lies in trying to estimate the parameters (e.g., α and β in the Beta distribution) defining the probability distribution of the reward of each action while assuming that it is constant, or at least, slowly varying in time, so it would be worth to learn. However, that is not the case in the multi-agent setting, where from one iteration to the next, the actual probability distribution of any action may change completely. Therefore, the gathered knowledge from past iterations becomes not useful at all to estimate the new distribution.

To illustrate the low utility of estimating the probability distribution of an agent's actions in a multi-agent setting, let us consider an elementary example. Assume two single-channel BSS's overlap, and there are two possible primary channels to pick in the system $C_A = C_B = \{1, 2\}$. We call the BSS's A and B, respectively, and assume they have a constant traffic load. From the point of view of A, its world $\mathscr{I}_A$ is determined just by B's action selection. That is, B selecting primary $p_B = 1$ generates a first world for A where it is better to select $p_A = 2$ to avoid contention and collisions. Likewise, B selecting primary $p_B = 2$ generates a second world for A where it is better to select $p_A = 1$. Figure 7 shows possible probability distributions (modeled as Gaussian variables for simplicity) of the reward of A in terms of throughput satisfaction. As anticipated, it is much better for A to select B's opposite action, i.e., the probability of getting a better reward is much higher. However, since MABs do not count on states, the agent does not have a separate reward distribution of each action per world (or state). In contrast, it generates a state-blind distribution like the one shown in the rightmost plot in Figure 7. And the information contained in there, beyond simple statistics like the mean, is of very little use.[7]

Finally, Exp3 tackles adversarial settings, where the *urgency* of an action is the sum of two terms: an exponential term in the success of the action for exploitation, and a constant term for exploration. Like UCB1 and Thompson Sampling, Exp3 relies on estimating the reward for the exponential term related to exploitation. And, as in the previous algorithms, such estimations are not reliable in the short-term.

Finding: In contrast to the sound performance of exploration-first and e-greedy, the rest of MABs (i.e., Thompson Sampling, UCB1, and EXP3) cannot learn due to their need to over-explore the action space for estimating algorithm parameters resulting in flawed and unfruitful in uncoordinated multi-agent setups.

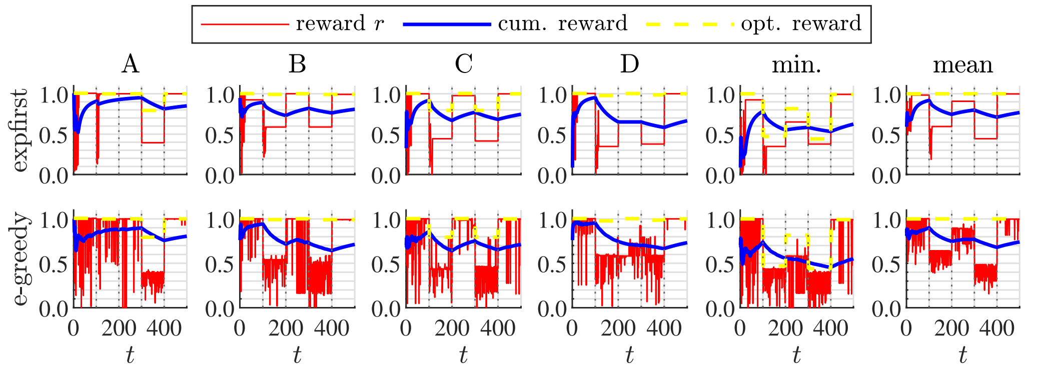
3) ADAPTING TO VARYING TRAFFIC LOADS

In this experiment, we focus on the performance of the MABs under varying traffic loads patterns. We now consider a traffic load pattern where each BSS w changes its mean traffic load to $\ell_w \in \{20, 50, 150\}$ Mbps every 100 iterations uniformly at random. This time, we focus our analysis only on exploration-first and e-greedy because they were the only MABs capable of learning under non-varying traffic loads.

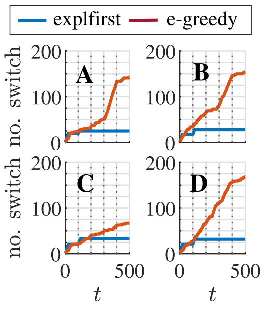
Figure 8a shows the throughput satisfaction evolution as in Figure 6a, this time considering load changes every 100 iterations. We find that both exploration-first and e-greedy can raise their performance after every traffic change, so they learn after every environment alteration. Again, we observe that exploration-first is much more *steady* than e-greedy. In fact, after the initial full exploration phase, exploration-first can quickly adapt to each environment change.

This steadiness is corroborated by looking at the number of action switches shown in Figure 8b. Exploration-first hardly changes an action after the full exploration phase in the early stages, whereas e-greedy keeps changing often, although decreasing the update function of the ϵ parameter. This number of switches is an essential metric to the BSS performance: after any change in the spectrum management configuration, the PHY and MAC overheads required to set up the AP and STAs in the BSS entail delays hindering the overall performance. In this regard, exploration-first seems a much more favorable choice. Indeed, one well-known pitfall of e-greedy is its poor asymptotic behavior because it usually continues to explore long after the optimal solution becomes apparent [39].

[7]Moreover, notice that generating the reward distribution with accuracy for every state-action pair is a costly task itself that requires extensive exploration.



(a) Run chart of the instant, cumulative, and optimal reward of each BSS, minimum, and mean of the BSS in the scenario with varying traffic loads.



(b) Evolution of the no. of switches.

FIGURE 8. **Self-contained toy dataset under varying traffic loads.**

Finding: Both exploration-first and e-greedy perform similarly in terms of cumulative (or long-term) reward, getting close to the optimal minimum in the WLAN after *relatively* few iterations. Nonetheless, exploration-first adapts much more *smoothly* than e-greedy after each load alteration. That is, it requires very few iterations to select its best action once the first exploration phase is completed. This has substantial benefits given that every time a different action is chosen by the AP, the corresponding configuration must be broadcast to the STAs, which require some non-negligible time to apply it, leading to service interruptions and detrimental QoE.

4) CAN CONTEXTS OR STATES AID LEARNING?

Part of the reasoning we conducted on why MABs seemed the most suitable approach was related to the fact of avoiding states. This way, there is no need to seek meaningful state definitions, and the learning can be executed more quickly since only actions and rewards are used to execute the policy.

Nonetheless, in this experiment, we go beyond such qualitative reasoning and provide a quantitative measure on Q-learning performance (relying on states), and contextual MABs (relying on contexts). For the sake of boosting the learning speed, given we consider a relatively short time-horizon, we define the state s and context Ω the same: as merely the binary throughput satisfaction of the BSS,

$$s = \Omega = \begin{cases} \text{true} & \xi > 0.95 \\ \text{false} & \text{otherwise,} \end{cases} \tag{10}$$

where ξ is the throughput satisfaction, which we compare against a 0.95 threshold to provide a safety margin. Notice that state space $\mathscr{S} = \{\text{true}, \text{false}\}$ is small and therefore does not provide much information on the environment. However, other more meaningful state definitions considering parameters like the selected action or spectrum occupancy statistics would exponentially increase the size of $\mathscr{S}$ and reduce the learning speed dramatically. For instance, if the state was defined simply as the action, i.e., $s = a$, the state-action table would be 12×12 given the $|\mathscr{A}| = 12$ actions in the action space. Accordingly, 144 state-action elements would exist. So, it would take *at least* 144 iterations to try each of the pairs once. In contrast, the proposed state space generates a more manageable state-action table of 2×12.[8]

Q-learning is a temporal difference algorithm seeking the best action to take given the current state. It is considered off-policy because the Q-learning function learns from actions outside the current policy, like taking random actions. The learning rate α or step size determines to what extent newly acquired information overrides old information. A factor $\alpha = 0$ makes the agent learn nothing (exclusively exploiting prior knowledge). In contrast, a factor $\alpha = 1$ makes the agent consider only the most recent information (ignoring prior knowledge to explore possibilities). We use a high value of $\alpha = 0.7$ because of the non-stationary deployment. The discount factor γ determines the importance of future rewards. A factor $\gamma = 0$ will make the agent myopic (or short-sighted) by only considering current rewards, while a factor approaching $\gamma \to 1$ will make it strive for a long-term high reward. We use a relatively small $\gamma = 0.3$ to foster exploitation (for rapid convergence) in front of exploration.

Another way in which agents use partial information gathered from the environment is in the form of contexts. In plain, contexts can be formulated as *states for stateless approaches*. That is, a contextual MAB can be instantiated like a particular MAB running separately per context. The learning is separated from one context to the other so that no information is shared between contexts like it is the case for Q-learning and other state-full approaches. In this case, we propose having a separate MAB instance for each of the two contexts defined (true and false).

Figure 9 is an extension of Figure 6a, this time showing the performance of the proposed Q-learning and contextual MAB. While we observe that both tend to learn, i.e., the

[8]From this brief discussion on the size of the state-action space we can conclude that more complex approaches like DRL do not suit the scenarios of this work. DRL relies on huge state-action spaces and normally takes long periods to start learning.

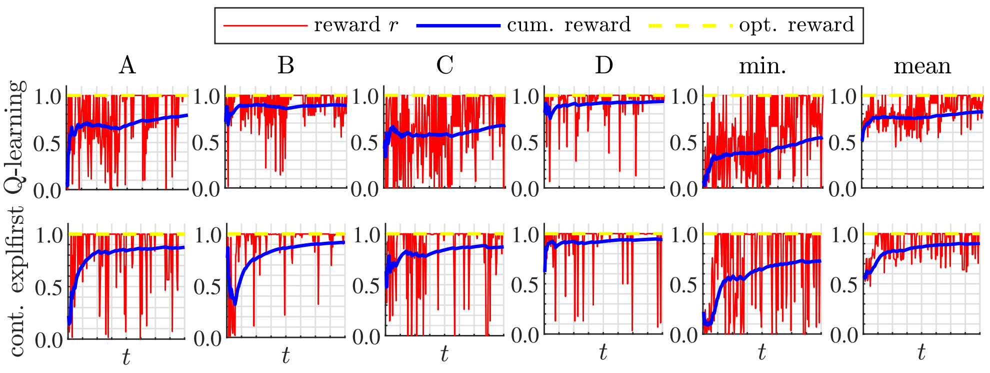


FIGURE 9. Q-learning and contextual MABs under constant traffic loads.

individual reward of all BSS's increases over time, we find unsteadiness with peaks of low (even zero) throughput satisfaction throughout all the iterations. Remarkably, such low throughputs may generate service interruptions causing poor user experience. So, we have seen that, by considering states and contexts, agents also tend to learn in the proposed finite horizon, but at a slower pace than the lightweight MABs.

We envision using states and context for larger time horizons when the BSS's may support periods of poor performance until the WLAN reaches steady dynamics providing even higher rewards than MABs. In such cases, more complex and larger state spaces may aid the task. We leave the study of such approaches as future work since this work focuses on rapid-adaptation mechanisms for zero-prior knowledge scenarios.

Finding: RL algorithms harnessing states or contexts do not boost the learning rate in short-term finite horizons like those studied in this work. Therefore, we corroborate that stateless approaches like MABs are a preferable choice for such a task.

C. GENERALIZATION TO HIGH-DENSITY DEPLOYMENTS

We now evaluate dense multi-agent deployments presenting more challenging dynamics.

1) DENSE MULTI-AGENT SETUPS

We propose a 20 × 20 m^2 map where multiple agent-empowered BSS's (from $W' = 6$ to 16) are spread at random over the area. We simply force a 4 m separation between APs, and STAs being separated up to 10 m from their associated AP. As in the previous experiments, each BSS w is assigned a random initial traffic load $\ell_w \in \{20, 50, 150\}$ Mbps, $\forall w$. As for the channelization, we consider now a standard-compliant maximum bandwidth of 160 MHz, thus having 8 possible 20-MHz channels (e.g., from channels 36 to 64 in the IEEE 802.11ac/ax standards). Accordingly, the action space of a BSS w is $\mathcal{A}_w = \{p_w, b_w\} | p_w \in \{1, 2, \ldots, 8\}, b_w \in \{1, 2, 4, 8\}, \forall w$, thus having 32 (8 × 4) possible configurations. Notice that, in contrast to the self-contained toy dataset, we now lose the

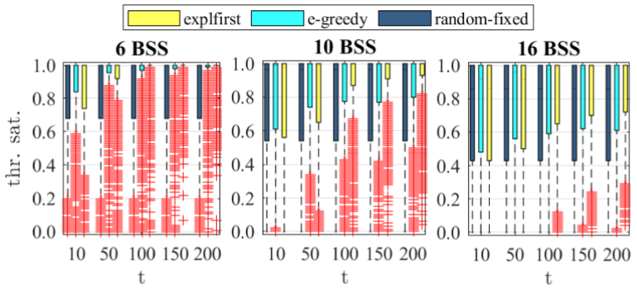


FIGURE 10. Reward distributions at different iterations.

optimal baseline reference since it is unfeasible to simulate all the statuses of the different deployments.

In this first experiment, we aim to generalize the assessment on deployments of increasing BSS density investigating whether the lightweight MABs are still able to learn effectively. We generate 100 WLAN random deployments for three different numbers of BSS's $W' = W \in \{6, 10, 16\}$ and run the same MAB algorithm at every AP in the network. In particular, we simulate exploration-first, e-greedy, and a fixed random configuration (random-fixed), where agents are idle and do never change the initial configuration whatsoever. We run each scenario for two different random seeds to contemplate the randomness of the MAB algorithms. Notice that the random-fixed algorithm can be viewed as an ablation method of the MAB algorithms since a component of the AI system (the agent's action selection) is removed. With such a baseline algorithm, we further motivate the need for learning-based solutions.

Figure 10 shows the distribution of the throughput satisfaction ξ of every BSS for each algorithm and each number of deployed BSS's. Notice that each boxplot inside the subplots is filled with the reward values gathered at a given iteration (10, 50, 100, 150, and 200) in the different simulations, thus covering a vast heterogeneous set of deployments. We observe that, regardless of the node density, exploration-first and e-greedy can learn, thus clearly outperforming the static configurations. In particular, exploration-first outperforms e-greedy as the simulation progresses. However, we also notice that e-greedy is better at earlier stages of the

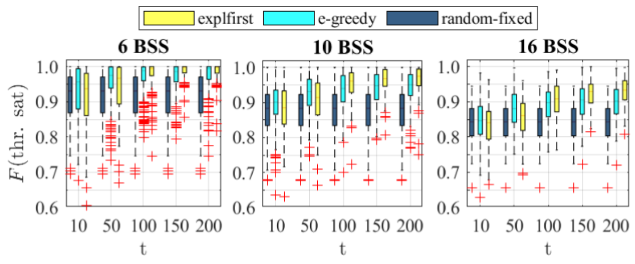


FIGURE 11. Jain's fairness distributions at different iterations.

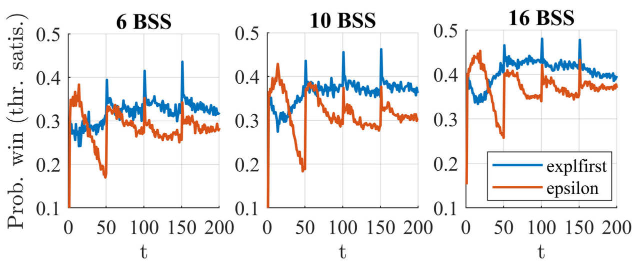


FIGURE 12. Run-chart of the probability of providing the highest instant reward.

simulation. This is due to the mandatory rule of exploration-first of exploring the full action space, no matter whether the agent is satisfied or not. If agents start from zero knowledge on the reward per action, exploration-first will take more time to *converge*. Still, once the full action space is explored, its performance is significantly higher than e-greedy.

Finding: Both exploration-first and e-greedy can learn even in high-density deployments, outperforming by far static configurations. Besides, while exploration-first performs worse than e-greedy at the early stages of the simulation, it reaches much better performance as the simulation progresses.

2) ON THE FAIRNESS

We observed that both exploration-first and e-greedy were able to learn for every assessed BSS density. We investigate now whether the MAB strategy of greedy reward optimization has any impact on the overall system's fairness. We keep studying the 100 WLAN random deployments from the previous experiment.

Figure 11 shows the distribution of the Jain's fairness index,

$$J = \frac{(\sum_{w=1}^{W} \xi_w)^2}{W \sum_{w=1}^{W} \xi_w^2}, \tag{11}$$

for the throughput satisfaction ξ. Like in Figure 10, each boxplot inside the subplots is filled with the reward values gathered at a given iteration in the different simulations.

We observe a similar behavior between the evolution of the reward (Figure 10) and the evolution of the fairness (Figure 11): e-greedy is fairer at the early stages, whereas exploration-first provides significantly higher fairness as the simulation progresses. This is an interesting result showing that multi-agent settings like WLANs can result in fair settings without coordination between agents. The key factor is the mutual interests forming a win-win relationship among agents. In plain, always on average terms, any pair of BSS's would mutually benefit if they do not overlap when trying to satisfy their traffic load demands. Unfortunately, we also find unfair outlier scenarios where some BSS's do not perform well due to different reasons like exposed/hidden nodes. In that kind of situation, it seems necessary to coordinate, thus changing the local greediness of the reward definition towards a collaborative one. In turn, that would require communication overheads and compatibility issues (or conflicts of interests between BSS's).

Finding: The WLAN's fairness tends to increase, thus reaching global *stable* configurations where most (or all) the BSS's perform similarly. In particular, exploration-first tends to outperform e-greedy, leading to fairer setups as simulations progress.

3) ADAPTATION TO VARYING TRAFFIC LOAD

In this experiment, we assess how the lightweight MABs respond to varying traffic loads, potentially affecting each action's reward. We provide a direct comparison between exploration-first and e-greedy by assessing each agent's reward at every iteration in a 1-vs-1 fashion. In particular, we compute the evolution of the probability of each MAB outperforming the other. That is, for every agent, we count the number of deployments per iteration where one MAB provided the highest instant reward $r_{w,t}$.

Figure 12 shows the probability of each MAB outperforming the other in terms of throughput satisfaction. Notice that the curves do not sum 1 in any of the iterations because of the cases where both MABs provided the same reward, most often corresponding to maximum satisfaction, $\xi = 1$. We corroborate that exploration-first is only outperformed by e-greedy in the first full exploration phase. Afterward, it performs better even after the changes in the traffic loads. So, it is preferable for every node density we studied to use exploration-first for mid/long term reward and quicker adaptation to environmental changes.

Finding: After *paying the price* of executing the first full exploration phase, exploration-first outperforms e-greedy both in terms of mid/long-term reward and fast adaptation in front of environment alternations. We conclude that in situations where there is room to spend a *reasonable* time for learning, exploration-first should be the MAB of choice.

4) LIGHTWEIGHTING THE ACTION SPACE

Finally, we now ask if it is possible to increase the learning convergence speed and reach good performance levels sooner. To that aim, we propose lightening the action space of each agent by reducing the possible values the maximum bandwidth attribute may take: from $b \in \{1, 2, 4, 8\}$ to $b \in \{1, 8\}$, i.e., to allow only single-channel ($b = 1$) or to remove any bandwidth restriction ($b = 8$). So, we expect to raise

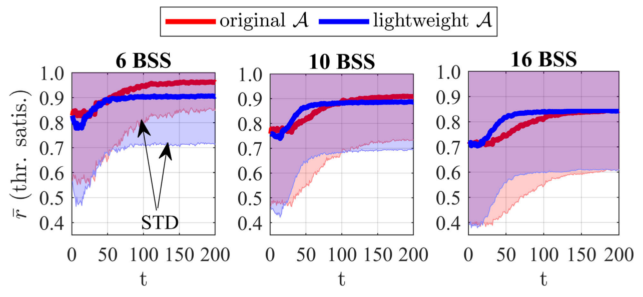


FIGURE 13. Comparison between the original and the lightened action space when using exploration-first. The curves represent the mean value of the rewards, and the shaded areas correspond to the standard deviation (STD), respectively.

the convergence at the cost of renouncing to potentially better configurations.

We run the MABs in the same deployments of the two experiments before, but we now reduce the maximum bandwidth space b to two values: 1 (20 MHz) and 8 (160 MHz). Notice that the resulting action space has 16 actions rather than 32, i.e., we halve the action space going from $|\mathscr{A}_w| = 32$ to $|\mathscr{A}_w| = 16, \forall w$.

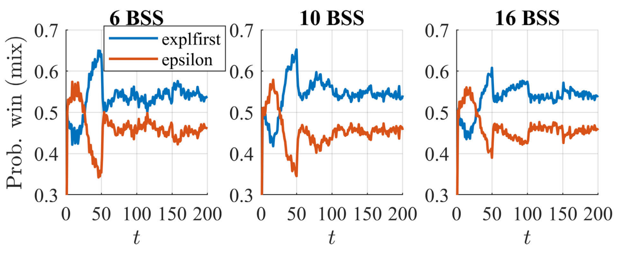
Figure 13 shows the mean and standard distribution of the throughput satisfaction when applying exploration-first with the original and the reduced action space. We observe that both action spaces have a similar convergence rate for scenarios with moderate density (6 BSS's). However, the best performance is only reached with the original action space (i.e., keeping all the configuration options in the max. bandwidth). However, as the node density increases, it turns out that having a reduced action space allows significantly faster convergence while reaching a similar performance than the original action space in the long term.

So, is there any rule of thumb for reducing action spaces? We suggest limiting the domain of the less critical attributes, always bearing in mind the trade-off between fast convergence and maximum reachable reward. However, such a task is not straightforward and requires expertise from the RL designer. In our case, we have proposed reducing the maximum bandwidth's possible values $\{b\}$ and keeping all the values for the primary channel $\{p\}$ since it is well-known that the primary channel is critical in CSMA/CA networks.

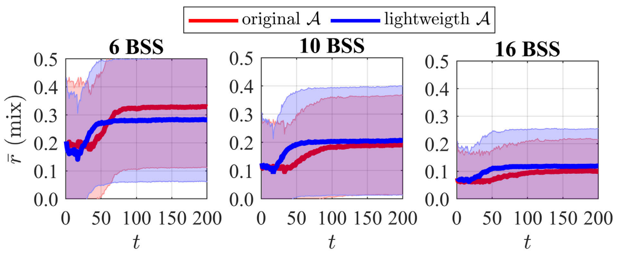
Finding: It is preferable to lighten the action space in challenging (dense and highly loaded) scenarios since it will contribute to increasing the learning speed. The downside is to renounce to a potentially better long-term performance. So, we suggest reducing the less critical attributes from the action space whenever possible.

5) MIXED REWARD: THROUGHPUT AND DELAY

Up to now, we have considered the reward to be exclusively dependent on the throughput satisfaction, i.e., on the throughput and the traffic load (7). In this experiment, we aim to answer whether another reward definition, including the delay, would vary our MAB analysis in any regard.



(a) Run-chart of the probability of providing the highest instant mixed reward (complementary to Figure 12).



(b) Original and lightweight action space when using exploration-first for the mixed reward (complementary to Figure 13).

FIGURE 14. Mixed throughput-delay reward assessment.

Delay is defined as the mean data packet delay, simply computed as the time difference between a data packet being generated and acknowledged. Unfortunately, the delay is not bounded, which is convenient for guaranteeing most RL algorithms' convergence. Thus, we propose a bounded delay ratio defined as

$$\delta_{w,t} = \frac{d^*}{d_{w,t}}, \tag{12}$$

where $d_{w,t}$ is the mean delay during iteration t, and d^* is the known minimum achievable delay experienced when transmitting a single data packet at maximum MCS 11. So, $\delta_{w,t} < 1, \forall w, t$ holds because every BSS will always experience a delay higher than d^* simply due to the MAC interframes (SIFS, DIFS) and control packets exchange (RTS, CTS, ACK).

We repeat the experiments on the action space lightweighting (§VI-C4) and the direct comparison between exploration-first and e-greedy (§VI-C3), this time using a mixed throughput-delay reward definition,

$$R(\xi, \delta) = \xi \cdot \delta. \tag{13}$$

Notice that we propose the latter's product operation given the high correlation between throughput satisfaction and delay. Generally, a high ξ leads to low δ and vice versa. However, for the cases where $\xi = 1$, there may exist configurations providing different delay values. Thus, we anticipate it is worth explicitly considering the delay as well.

Figure 14a and Figure 14b show the probability of each MAB outperforming the other in terms of the mixed reward, and the mean and standard distribution of the throughput satisfaction when applying exploration-first with the original and the reduced action space respectively. That is, such figures are complementary to Figure 12, and Figure 13, respectively. We observe similar patterns in both pairs of figures, only varying the reward scale, which is expected since the mixed reward is by definition lower than the throughput satisfaction. Therefore, we conclude that exploration-first also outperforms e-greedy when introducing other performance parameters like the delay in the reward definition R.

Finding: Exploration-first keeps learning and outperforming e-greedy for the mixed throughput-delay reward. We derive that exploration-first is convenient for the Wi-Fi deployments considered in this thesis regardless of the performance parameters included in the reward definition.

VII. CONCLUSION AND FUTURE WORK

In this paper, we postulate MABs as an efficient ready-to-use formulation for spectrum management in multi-agent channel bonding WLANs. We call into question whether the use of complex RL algorithms actually helps the quest and derive that stateless algorithms, especially in the form of lightweight MABs, are an efficient solution for rapid adaptation avoiding extensive or meaningless states. We anticipate this paper will contribute to the design of novel spectrum management techniques for next-generation WLANs, we and expect that our findings on channel bonding will help overcome the main challenges imposed by spectrum sharing. Besides, we expect that our reasoning on why stateless RL algorithms in the form of lightweight MABs are an efficient solution for rapid deployment will be useful as well for other techniques like scheduling or transmission power control.

As for future work, we envision the use of states for larger time horizons when the focus is not put on rapid-adaptation but on reaching higher potential performance in the long-run. Hereof, we believe that using MABs for generating training datasets to be later fit in more complex ML models is an interesting venue for research. Finally, we leave the design of advanced channel allocation mechanisms that consider the use of domain knowledge-empowered MABs for future work.

ACKNOWLEDGMENT

This work is an extension of the proceedings paper [1] [DOI: 10.1109/WD52248.2021.9508323].

REFERENCES

[1] S. Barrachina-Muñoz, A. Chiumento, and B. Bellalta, "Stateless reinforcement learning for multi-agent systems: The case of spectrum allocation in dynamic channel bonding WLANs," in *Proc. Wireless Days (WD)*, 2021, pp. 1–5, doi: 10.1109/WD52248.2021.9508323.

[2] *IEEE 802.11n Enhancements for Higher Throughput*, IEEE, Manhattan, NY, USA, 2009, pp. 1–682.

[3] *IEEE 802.11ac Very High Throughput*, IEEE, 2013.

[4] *IEEE P802.11ax/D4.0 High Efficiency WLAN*, IEEE, 2019, pp. 1–682.

[5] (2016). *IEEE 802.11be Extremely High Throughput, 802.11-18/1231r6*. [Online]. Available: http://www.ieee802.org/11/Reports/ehtsg_update.htm

[6] S. Barrachina-Munoz, F. Wilhelmi, and B. Bellalta, "Dynamic channel bonding in spatially distributed high-density WLANs," *IEEE Trans. Mobile Comput.*, vol. 19, no. 4, pp. 821–835, Apr. 2020.

[7] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, 4th Quart., 2019.

[8] A. Zappone, M. Di Renzo, and M. Debbah, "Wireless networks design in the era of deep learning: Model-based, AI-based, or both?" *IEEE Trans. Commun.*, vol. 67, no. 10, pp. 7331–7376, Oct. 2019.

[9] S. de Bast, R. Torrea-Duran, A. Chiumento, S. Pollin, and H. Gacanin, "Deep reinforcement learning for dynamic network slicing in IEEE 802.11 networks," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2019, pp. 264–269.

[10] S. Barrachina-Munoz, F. Wilhelmi, I. Selinis, and B. Bellalta, "Komondor: A wireless network simulator for next-generation high-density WLANs," in *Proc. Wireless Days (WD)*, Apr. 2019, pp. 1–8.

[11] M. Park, "IEEE 802.11ac: Dynamic bandwidth channel access," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2011, pp. 1–5.

[12] X. Yue, C.-F. Wong, and S.-H. G. Chan, "CACAO: Distributed client-assisted channel assignment optimization for uncoordinated WLANs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 9, pp. 1433–1440, Sep. 2011.

[13] Y. Xu, J. Wang, Q. Wu, A. Anpalagan, and Y.-D. Yao, "Opportunistic spectrum access in unknown dynamic environment: A game-theoretic stochastic learning solution," *IEEE Trans. Wireless Commun.*, vol. 11, no. 4, pp. 1380–1391, Apr. 2012.

[14] J. Zheng, Y. Cai, N. Lu, Y. Xu, and X. Shen, "Stochastic game-theoretic spectrum access in distributed and dynamic environment," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4807–4820, Oct. 2015.

[15] S. Barrachina-Munoz, F. Wilhelmi, and B. Bellalta, "Online primary channel selection for dynamic channel bonding in high-density WLANs," *IEEE Wireless Commun. Lett.*, vol. 9, no. 2, pp. 258–262, Feb. 2020.

[16] J. Herzen, R. Merz, and P. Thiran, "Distributed spectrum assignment for home WLANs," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 1573–1581.

[17] B. Bellalta, A. Checco, A. Zocca, and J. Barcelo, "On the interactions between multiple overlapping WLANs using channel bonding," *IEEE Trans. Veh. Technol.*, vol. 65, no. 2, pp. 796–812, Feb. 2016.

[18] S. Khairy, M. Han, L. X. Cai, Y. Cheng, and Z. Han, "A renewal theory based analytical model for multi-channel random access in IEEE 802.11ac/ax," *IEEE Trans. Mobile Comput.*, vol. 18, no. 5, pp. 1000–1013, May 2019.

[19] A. Nabil, M. J. Abdel-Rahman, and A. B. MacKenzie, "Adaptive channel bonding in wireless LANs under demand uncertainty," in *Proc. IEEE 28th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Oct. 2017, pp. 1–7.

[20] L. Cigler and B. Faltings, "Decentralized anti-coordination through multi-agent learning," *J. Artif. Intell. Res.*, vol. 47, pp. 441–473, Jul. 2013.

[21] H. Qi, H. Huang, Z. Hu, X. Wen, and Z. Lu, "On-demand channel bonding in heterogeneous WLANs: A multi-agent deep reinforcement learning approach," *Sensors*, vol. 20, no. 10, p. 2789, May 2020.

[22] Y. Chen, J. Zhao, Q. Zhu, and Y. Liu, "Research on unlicensed spectrum access mechanism based on reinforcement learning in LAA/WLAN coexisting network," *Wireless Netw.*, vol. 26, no. 3, pp. 1643–1651, Apr. 2020.

[23] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, "Deep reinforcement learning for dynamic multichannel access in wireless networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 2, pp. 257–265, Jun. 2018.

[24] C. Zhong, Z. Lu, M. C. Gursoy, and S. Velipasalar, "Actor-critic deep reinforcement learning for dynamic multichannel access," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Nov. 2018, pp. 599–603.

[25] O. Naparstek and K. Cohen, "Deep multi-user reinforcement learning for distributed dynamic spectrum access," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 310–323, Jan. 2019.

[26] W. Dai, Y. Gai, and B. Krishnamachari, "Online learning for multi-channel opportunistic access over unknown Markovian channels," in *Proc. 11th Annu. IEEE Int. Conf. Sens., Commun., Netw. (SECON)*, Jun. 2014, pp. 64–71.

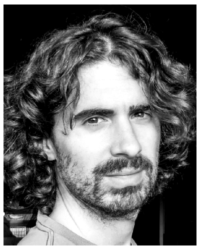
[27] K. Nakashima, S. Kamiya, K. Ohtsu, K. Yamamoto, T. Nishio, and M. Morikura, "Deep reinforcement learning-based channel allocation for wireless LANs with graph convolutional networks," *IEEE Access*, vol. 8, pp. 31823–31834, 2020.

[28] W. Wydmański and S. Szott, "Contention window optimization in IEEE 802.11ax networks with deep reinforcement learning," 2020, *arXiv:2003.01492*. [Online]. Available: http://arxiv.org/abs/2003.01492

[29] R. Karmakar, S. Chattopadhyay, and S. Chakraborty, "SmartLA: Reinforcement learning-based link adaptation for high throughput wireless access networks," *Comput. Commun.*, vol. 110, pp. 1–25, Sep. 2017.

[30] T. Adame, M. Carrascosa, and B. Bellalta, "The TMB path loss model for 5 GHz indoor WiFi scenarios: On the empirical relationship between RSSI, MCS, and spatial streams," in *Proc. Wireless Days (WD)*, Apr. 2019, pp. 1–8.

[31] S. Barrachina-Muñoz, F. Wilhelmi, and B. Bellalta, "To overlap or not to overlap: Enabling channel bonding in high-density WLANs," *Comput. Netw.*, vol. 152, pp. 40–53, Apr. 2019.
[32] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, London, U.K., 1989.
[33] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2, pp. 235–256, 2002.
[34] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Adv. Appl. Math.*, vol. 6, no. 1, pp. 4–22, Mar. 1985.
[35] A. Slivkins, "Introduction to multi-armed bandits," 2019, *arXiv:1904.07272*. [Online]. Available: http://arxiv.org/abs/1904.07272
[36] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "Gambling in a rigged casino: The adversarial multi-armed bandit problem," in *Proc. IEEE 36th Annu. Found. Comput. Sci.*, Oct. 1995, pp. 322–331.
[37] Y. Seldin, C. Szepesvári, P. Auer, and Y. Abbasi-Yadkori, "Evaluation and analysis of the performance of the EXP3 algorithm in stochastic environments," in *Proc. Eur. Workshop Reinforcement Learn.*, 2013, pp. 103–116.
[38] L. Liu, R. Downe, and J. Reid, "Multi-armed bandit strategies for non-stationary reward distributions and delayed feedback processes," 2019, *arXiv:1902.08593*. [Online]. Available: https://arxiv.org/abs/1902.08593
[39] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.



SERGIO BARRACHINA-MUÑOZ received the Ph.D. degree in information and communication technologies from Universitat Pompeu Fabra (UPF), in 2012. He is currently working as a Postdoctoral Researcher with the SMARTECH Department, Centre Tecnològic de Telecomunicacions de Catalunya, Barcelona, Spain. His research interests include machine learning for next-generation Wi-Fi and beyond 5G systems.

ALESSANDRO CHIUMENTO (Member, IEEE) received the Ph.D. degree in electrical engineering from KU Leuven, in 2015. He is currently an Assistant Professor with the Pervasive Systems Group, University of Twente, The Netherlands. He has been working on distributed management solutions for dynamic systems for large and heterogeneous wireless networks. His research interests include reconfigurable networks, cross-layer analysis of wireless systems, the softwarized IoT networks, dynamic quality of service, machine learning for wireless, and wireless for machine learning.

BORIS BELLALTA (Senior Member, IEEE) received the degree in telecommunications engineering from Universitat Politècnica de Catalunya (UPC), in 2002, and the Ph.D. degree in information and communication technologies from Universitat Pompeu Fabra (UPF), in 2007. He is currently an Associate Professor with the Department of Information and Communication Technologies (DTIC), UPF. His research interests include wireless networks, with emphasis on the design and performance evaluation of new architectures and protocols. The results from his research have been published in more than 100 international journal articles and conference papers.