Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier

Improving IEEE 802.11ax UORA Performance: Comparison of Reinforcement Learning and Heuristic Approaches

KATARZYNA KOSEK-SZOTT[1], SZYMON SZOTT[1], FALKO DRESSLER[2] (Fellow, IEEE)
[1]AGH University of Science and Technology, Kraków, Poland
[2]Technical University of Berlin, Berlin, Germany

Corresponding author: Katarzyna Kosek-Szott (e-mail: kks@agh.edu.pl).

This research was funded by the National Science Centre, Poland (DEC-2020/39/I/ST7/01457) as well as by the German Research Foundation (DFG DR 639/28-1).

ABSTRACT Machine learning (ML) has gained attention from the network research community because it can help solve difficult problems and potentially lead to groundbreaking achievements. In the Wi-Fi domain, ML is applied to solve challenges such as efficient channel access and fair coexistence with other technologies in unlicensed bands. In this paper, we address the performance of uplink orthogonal frequency division multiple random access (UORA) in IEEE 802.11ax networks. Optimization of UORA is a good case for applying ML because of its inherent complexity and dependence on situation and time-dependent parameters. In particular, we use deep reinforcement learning to tune UORA parameters. Our simulation results show that even though the ML-based solution leads to close to optimal results, its operation is comparable to a much simpler, non-ML heuristic. Therefore, we conclude that ML-based solutions to improve IEEE 802.11 performance need not exceed well-designed heuristics.

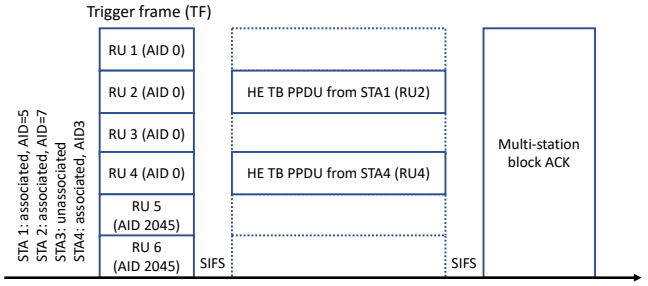
INDEX TERMS deep Q-learning, IEEE 802.11ax, machine learning, OFDMA, reinforcement learning, UORA, uplink orthogonal frequency division multiple random access

I. INTRODUCTION

THE IEEE 802.11ax amendment introduces uplink (UL) multi-user (MU) orthogonal frequency-division multiple access (OFDMA) to improve the efficiency of Wi-Fi networks. OFDMA-based channel access divides radio channel resources into subcarrier groups, called resource units (RUs), which are then allocated to stations. Stations can transmit simultaneously, which improves efficiency compared to single-user transmissions. OFDMA has two modes of operation: scheduled access (SA) [1] and random access (RA) [2]. In the former, all decisions are made centrally at the 802.11ax AP. Meanwhile, in the latter, decisions are distributed and there is room for performance improvement. Therefore, in this paper, we focus on the RA mode.

To provide RA OFDMA, 802.11ax defines uplink OFDMA-based random channel access (UORA) [3], which can be used in dense scenarios, e.g., in Internet of Things (IoT) deployments or industrial wireless sensor networks [4], [5]. SA is inappropriate for such scenarios due to the overhead cost of polling all stations to determine their UL buffer status. With RA, only stations that require UL transmission opportunities compete for RUs using UORA rules. UORA is based on two components: the OFDMA contention window (OCW) and OFDMA random access backoff (OBO). Stations select a random OBO counter from the range (0, OCW) and then decrease it by the number of eligible RUs assigned by the access point (AP) for uplink transmissions. OBO is decremented during each UORA frame exchange (Figure 1). Stations transmit when their OBO reaches 0. The OCW range can be configured by the AP in the UORA parameter set element, distributed through beacon frames. Unfortunately, this basic operation (which we refer to as legacy UORA) is highly inefficient under saturation [2].

Researchers have proposed to modify UORA in various ways (cf. Table 1). The efficiency of UORA can be improved by adaptive grouping [6], [7], spatial clustering [8], sub-



(a) UORA frame exchange

	STA1, initial OBO=3	STA2, initial OBO=6	STA3, initial OBO=4	STA4, initial OBO=2
RU1 (AID 0)	OBO-4=-1 (Set OBO=0, randomly select one RU)	OBO-4=2	-	OBO-4=-2 (Set OBO=0, randomly select one RU)
RU2 (AID 0)			-	
RU3 (AID 0)			-	
RU4 (AID 0)			-	
RU5 (AID 2045)	-	-	OBO-2=2	-
RU5 (AID 2045)	-	-		-
	Select new OBO	Resume OBO in next TF	Resume OBO in next TF	Select new OBO

(b) OBO decrementation after TF reception

FIGURE 1: Example of UORA operation in 802.11ax [2].

channel hopping [9], complementary probability instead of backoff [10], additional carrier sensing [11], retransmission awareness [12], OBO modifications [2], [13], grouping-based channel access [14], and considering adjacent channel interference [15][1]. None of the above research uses ML methods, although the application of reinforcement learning (RL) to improve UORA operation is suggested as future work by Kim et al. [13]. In fact, with the proliferation of the use of ML solutions to improve Wi-Fi performance [17], extending UORA with ML is the logical next step.

Thus inspired, in this paper, we present an RL-based OBO procedure (RL-OBO) to adjust the UORA random access backoff operation to the congestion level of the shared channel. After providing a brief description of legacy UORA (Section II) and an existing non-ML heuristic (Section III), our main contributions are:

- We design an RL-based OBO procedure (RL-OBO) for UORA (Section IV), where, based on the observed probability of unsuccessful RUs, the AP learns the level of network congestion and adjusts the OBO countdown to achieve a higher success rate and, whenever possible, avoid empty RUs. To the best of our knowledge, this has not yet been done.
- We evaluate RL-OBO using a simulation model to confirm the accuracy of the RL-based solution (Section V-D). Unlike most of the literature [6]–[12], [14], which considers only static scenarios, we follow [13] and study dynamic network loads and station churn.
- We compare the operation of RL-OBO with a previous approach in Section V-F. This approach (E-OBO) is an existing non-ML-based heuristic exhibiting good performance (Section V-E). However, E-OBO requires the static definition of certain parameters, which is its main disadvantage (cf. Section III).
- We show that, even though RL-OBO can improve the performance of legacy UORA, its behavior can sometimes be slightly worse than that of E-OBO. In particular, RL-OBO can produce suboptimal results and may lead to throughput unfairness in dynamic environments.

We conclude the paper and outline future work in Section VI. The notation and acronyms used are gathered in Tables 2 and 3, respectively.

[1]Other UORA-related research areas include coexistence of RA and SA modes; alternative MAC protocols (including deterministic channel access); scheduler design; and adaptation to real-time, V2X, and healthcare IoT applications [2] [16].

Algorithm 1 Legacy UORA (802.11ax)

$OCW_{\min} \leftarrow 7$
$OCW_{\max} \leftarrow 31$
if first transmission **then**
  $OCW \leftarrow OCW_{\min}$;
else if retransmission **then**
  $OCW \leftarrow 2 \times OCW + 1$;
  if $OCW \geq OCW_{\max}$ **then**
    $OCW \leftarrow OCW_{\max}$;
  end if
end if
$OBO \leftarrow$ random integer$(0, OCW)$;
Station decrements OBO by n_{RU} and selects a random RU for transmission if $OBO = 0$.

II. LEGACY UORA

UORA is summarized in Algorithm 1 while Figure 1 provides an example of its operation. First, a trigger frame (TF) transmitted by the AP ensures the synchronization of participating stations. Each TF can designate one or more RUs for random access. The AP sets the association identifier (AID) field in the transmitted TF to indicate the RA RUs assigned to associated stations (AID = 0) and unassociated stations (AID = 2045).

After the successful reception of a TF, stations contend to access eligible RA RUs if they have pending data frames destined to the AP. Each contending RA station maintains two variables: OCW (initialized to $OCW_{\min}$) and the OBO counter (initialized with an integer randomly selected from a uniform distribution from 0 to OCW). If the OBO counter is smaller than the number of available RA RUs, a station randomly selects one of the RUs for data transmission. Otherwise, it decrements the OBO counter by the number of eligible RUs and waits for the next TF.

In the event of an unsuccessful transmission, the station retransmits as follows. First, the station updates its OCW counter to $2\times$OCW$+1$ every time $OCW \leq OCW_{\max}$. Once $OCW = OCW_{\max}$, the OCW value remains unchanged

TABLE 1: Literature review. Evaluation types: theoretical (T) and simulation (S).

Key	UORA Enhancement	Result	Evaluation Type	Year
[6]	Adaptive grouping scheme	Improved throughput	T+S	2018
[7]				2019
[8]	Spatial clustering	Improved area throughput		2019
[9]	Extra backoff stage, opportunistic sub-channel hopping	Improved bandwidth utilization and reduced collisions		2019
[10]	Stations transmit without backoff with a complementary probability after unsuccessful transmissions	Improved throughput, reduced packet delay		2019
[11]	Hybrid OFDMA RA with carrier sensing, secondary backoff mechanism	Improved throughput		2020
[12]	Retransmission-aware channel access	Reduced delay	S	2020
[13]	New OBO control scheme	Improved throughput		2021
[14]	Grouping-based channel access	Improved fairness and QoS	T+S	2021
[2]	Enhanced UORA backoff procedure	Improved throughput, fairness, and delay	S	2022

TABLE 2: Notation used

Parameter	Description
α	OBO countdown rate
a_t	Agent action at step t
ϵ	Exploration rate
f_t, f_{p_c}	Fairness of throughput and collision probability
γ	Discount factor
ω	Learning rate
OBO	OFDMA random access backoff
OCW	Current OFDMA contention window value
$OCW_{min(max)}$	OCW minimum (maximum) values
n_{RU}	No. of RUs allocated for unassociated stations
p^e_{RU}	Probability of empty RUs
p^u_{RU}	Probability of unsuccessful RUs
p_c	Collision probability
Q	Q value (quality of a state–action combination)
r_t	Agent reward at step t
r_S, r_U, r_E	Reward for successful, unsuccessful, and empty RUs
s_t	Agent state at step t
t	Time step (cf. Figure 2)
ζ	Measuring interval (in contention rounds)

TABLE 3: List of acronyms

AID	association identifier
AP	access point
BACK	block acknowledgement
CW	contention window
DQL	deep Q-learning
DQN	deep Q network
E-OBO	efficient OBO
GI	guard interval
HE	high efficiency
IoT	Internet of Things
ML	machine learning
MPDU	MAC protocol data unit
MU	multi-user
OBO	OFDMA random access backoff
OCW	OFDMA contention window
OFDMA	orthogonal frequency-division multiple access
PPDU	PLCP protocol data unit
RA	random access
ReLU	Rectified linear unit
RL	reinforcement learning
RL-OBO	RL-based OBO
RU	resource unit
SA	scheduled access
SIFS	short interframe space
TB	trigger-based
TF	trigger frame
UL	uplink
UORA	uplink OFDMA-based random channel access

for subsequent retransmissions. The station then randomly selects a new OBO value in the range of 0 and OCW.

The AP can indicate the OFDMA contention window (OCW) range (i.e., $OCW_{\min}$ and $OCW_{\max}$) in the UORA Parameter Set element, which is a part of management frames (such as beacons and association frames). Alternatively, stations use the default OCW settings, i.e., $OCW_{\min} = 7$ and $OCW_{\max} = 31$.

III. UORA WITH EFFICIENT OBO

Recently, we proposed an UORA improvement called efficient OBO (E-OBO), which exhibits good performance [2]. We briefly explain the operation of E-OBO in this section to compare it later with RL-OBO in Section V-F.

In E-OBO, the AP changes the rate of station OBO countdown based on the RU states observed in previous UORA frame exchanges. We classify the RU states as *successful* (the frame in the RU is acknowledged by the AP), *unsuccessful* (more than one station selected the RU that resulted in a collision), and *empty* (no station selected the RU). By observing these states, the AP can determine whether congestion (many unsuccessful RUs and few empty RUs) or nonsaturation (few unsuccessful RUs and many empty RUs) conditions occur. Then, the AP reacts by increasing or decreasing the rate of OBO countdown with the α parameter, which is later passed to the stations.

E-OBO is formally defined in Algorithm 2. The AP measures the probability of unsuccessful RUs (p^u_{RU}) and empty RUs (p^e_{RU}) within a ζ interval and uses $\alpha \in [0.1, 3]$ to modify OBO:

$$OBO \longleftarrow OBO - \alpha \times n_{\text{RU}}. \quad (1)$$

Algorithm 2 E-OBO procedure

1: $\alpha \leftarrow 1$
2: Measuring interval: ζ contention rounds
3: **if** $p^u_{\text{RU}} \geq 0.33$ and $p^e_{\text{RU}} < 0.33$ **then** $\alpha \leftarrow \max(0.1, \alpha - 0.1)$
4: **else if** $p^u_{\text{RU}} \leq 0.5$ and $p^e_{\text{RU}} \geq 0.5$ **then** $\alpha \leftarrow \min(3, \alpha + 0.2)$
5: **else**
6: $\alpha \leftarrow \alpha$;
7: **end if**
8: AP sends α in the TF to inform stations of the level of contention in the network.
9: Stations decrement their OBO counters by $\alpha \times n_{\text{RU}}$ and select random RUs for transmission if $OBO = 0$.

By default, $\alpha = 1$. Then, if $p^u_{\text{RU}} \geq 0.33$ and $p^e_{\text{RU}} < 0.33$ (i.e., under congestion), the AP decreases α by 0.1. If $p^u_{\text{RU}} \leq 0.5$ and $p^e_{\text{RU}} \geq 0.5$ (i.e., under nonsaturation), the AP increases α by 0.2. Otherwise, α remains unchanged. The selected α is transmitted in TFs that initialize each UORA frame exchange. Then, the stations decrement their OBO counters using (1). The remainder of legacy UORA is left unchanged.

IV. RL-BASED OBO MECHANISM

In this section, we explain how UORA can be improved with an RL-based OBO (RL-OBO) mechanism. In particular, we apply deep Q-learning (DQL) [18] to support IEEE 802.11ax[2] APs in adjusting the α parameter to the congestion level of the shared channel. Similarly to E-OBO, we implement a centralized operation. Therefore, stations do not decide on the α value but obtain this information from the TFs transmitted by the AP before each contention round.

The implemented DQL model consists of three densely connected layers. The first two layers are composed of 32 nodes and they use the rectified linear unit (ReLU) activation functions. The output layer has three nodes (corresponding to the size of the action space) and it uses the linear activation function.

In RL-OBO, the agent is installed at the AP and learns (in the offline training phase) how to update α to reduce collisions under varying congestion levels. After training, the agent can be used to adjust the α value in online operation.

In RL-OBO, at each training step, the agent observes the probability of unsuccessful RUs in state s_t and selects an action based on previous observations. The agent has three possible actions to choose from:

- Action 1 – increase α, i.e., set the α parameter as $\min(3, \alpha + 0.1)$,
- Action 2 – decrease α, i.e., set the α parameter as $\max(0.1, \alpha - 0.1)$,
- Action 3 – leave α unchanged.

[2]DQL has previously been successfully applied to improve IEEE 802.11 performance in various areas: rate selection [19], CW tuning [20]–[22], multi-AP association [23], and RU selection in OFDMA [24].

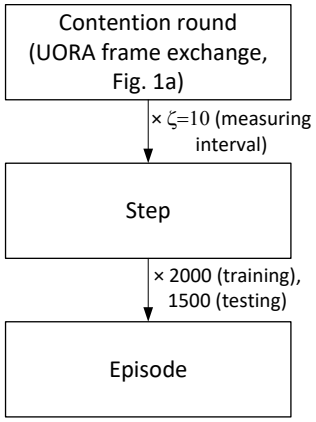


FIGURE 2: UORA simulator elements for training and testing RL-OBO.

After taking each action, the agent receives feedback in the form of a reward r_t and a new state s_{t+1}. Based on the above, the state space is one-dimensional (it stores the probability of unsuccessful transmission) and the action space is three-dimensional (increase α, decrease α, or leave α unchanged).

We notice that a collision is less desirable than an empty RU, since empty RUs may be the result of low congestion. Obviously, a successful transmission is the most desirable outcome. Therefore, the reward is decreased by $r_E = 1.5$ in the case of each empty RU, increased by $r_S = 3$ in the case of each successful RU, and decreased by $r_U = 2$ in the case of each unsuccessful RU. The motivation behind selecting these particular values is given in Appendix A. Additionally, Actions 1 and 2 result in decreasing the reward by 0.1 to promote Action 3 whenever possible (i.e., leave α unchanged if the performance is satisfactory).

The agent calculates the probability of unsuccessful RUs as the fraction of the total number of unsuccessful RUs divided by the sum of the number of successful, unsuccessful, and empty RUs. Additionally, to limit the number of possible states, the agent rounds the results to two decimal places.

At each step (composed of 10 contention rounds, as presented in Figure 2) in the training process, the agent stores s_t, a_t, r_t, s_{t+1} and, after each action taken, updates the Q-value:

$$Q'(s_t, a_t) \longleftarrow Q(s_t, a_t) + \\ + \omega(r_t + \gamma(\max_a Q(s_{t+1}, a)) - Q(s_t, a_t)) \quad (2)$$

where Q is the old and Q' is the new Q-value. Furthermore, we define the mean squared error as the loss function and use the ϵ-greedy strategy to balance exploration and exploitation.

The exploration rate (ϵ) is set to 1 at the beginning of the first episode. Then, with each time step, it anneals linearly from 1 to 0.1 (with a decay of 0.995) to increase the probability of exploitation. Additionally, at each time step, a random number is generated from a uniform distribution over $[0, 1)$. The sampled value is then checked with the current ϵ value. If it is lower, a random action is taken. Otherwise, the learned

Algorithm 3 RL-OBO procedure

Train agent:
(1) For state s_t agent selects action a_t.
(2) AP sends α in TF to inform stations of network contention level.
(3) Stations decrement OBO counters by $\alpha \times n_{\text{RU}}$ and transmit in a randomly selected RU if $OBO = 0$.
(4) For a period of ζ contention rounds, the agent calculates the reward as the sum of the results for each RU i:
if R_i is successful **then** $r_t \leftarrow r_t + r_S$
else if R_i is unsuccessful **then** $r_t \leftarrow r_t - r_U$
else $r_t \leftarrow r_t - r_E$ ▷ R_i is empty
end if
(5) Agent calculates new state $s_{t+1} = p^u_{RU}$, stores s_t, a_t, r_t, s_{t+1}, and selects new action a_{t+1}.
(6) Update the reward after action selection
if $a_{t+1} \in \{1, 2\}$ **then** $r_t \leftarrow r_t - 0.1$
end if
Repeat steps 2-5 until training is finished, i.e., when the reward stabilizes.
Use trained agent:
(1) AP sends α, selected by the trained agent, in the TF to inform stations of the contention level in the network.
(2) Stations decrement their OBO counters by $\alpha \times n_{\text{RU}}$ and transmit in a randomly selected RUs if $OBO = 0$.

TABLE 4: ML parameters

Parameter	Value
Exploration rate ϵ	1
Epsilon minimum	0.01
Epsilon decay	0.995
Discount rate γ	0.95
Batch size	100
Learning rate ω	0.001
Maximum memory length	100,000
Action space size	3
State space size	1
No. of nodes in different layers	32, 32, 3
Activation function	Rectified linear unit
Optimizer	Adam
Loss function	Mean squared error

action is performed. Therefore, initially, the agent starts exploring the environment, and then it steadily increases exploitation.

Table 4 summarizes all the parameters and settings of the proposed machine learning model. The values of the hyperparameters of the model were selected empirically (cf. Appendix A) to provide good performance results.

In summary, the described model allows the AP to map congestion levels (reflected by the number of empty, successful, or unsuccessful RUs) to optimal α settings, which are then announced to the stations in the TFs. The rest of the legacy UORA operation is left unchanged. RL-OBO operation is summarized in Algorithm 3.

V. RESULTS

To evaluate the performance of the two OBO selection schemes, we implement both E-OBO and RL-OBO in a custom 802.11ax UORA simulator. First, we provide details regarding the simulator design. Then, we describe the simulation scenario and define the performance metrics used. Next, we explain the RL-OBO training process and show how the trained agent performs in testing scenarios. Subsequently, we show how E-OBO performs under similar network conditions (such an analysis was not carried out previously [2]). Finally, we compare the ML-based and heuristic solutions.

A. SIMULATOR

Our custom 802.11ax UORA simulator is written in Python, with the RL parts written using Keras. Unfortunately, to the best knowledge of the authors, there are no real devices available with an UORA implementation, which would make an experimental evaluation possible.

The implemented simulator analyzes consecutive 802.11ax UORA frame exchange sequences (Figure 1a) called *contention rounds*. Decisions about future behavior are made after each measurement interval, i.e., ζ contention rounds. In accordance with the ML nomenclature, we refer to these intervals as *steps* and to each simulation run – as *episodes* (Figure 2). The simulator code is available to the research community[3].

B. SIMULATION SCENARIO

We study a scenario with a single AP without outside interference. We assume there are no channel errors, no hidden nodes, and that stations always have data frames to send (a full buffer model). There are two main input parameters that we modify in the analysis: the number of stations n_s and the number of RUs n_{RU}. We refer to the (n_s, n_{RU}) pair as the current *network configuration*. The former parameter denotes the number of stations transmitting to the AP. This number fluctuates over time as stations join and leave the network. We evaluate both fixed changes in the number of stations as well as random ones. In the latter case, the number of stations arriving or departing is randomly chosen from the ranges $[1, 5]$, $[1, 15]$, or $[1, 30]$, which represents small, moderate, and large network dynamicity, respectively. The second network configuration parameter, the number of RUs, is selected from the set $\{4, 8, 16, 32\}$. Since we are interested in measuring upper-bound performance, we evaluate only configurations in which the number of stations is greater than the number of available RUs. Table 5 summarizes the general simulation parameters.

C. PERFORMANCE METRICS

We consider the following performance metrics:

- throughput – measured as the sum of successfully transmitted bytes divided by the simulation time (unless

[3]https://github.com/KatarzynaKosek/RL-UORA

TABLE 5: Simulation parameters

Parameter	Value
PHY/MAC	802.11ax
$OCW_{\min}$	7
$OCW_{\max}$	31
Number of RUs (n_{RU})	4, 8, 16, 32
Data rate per RU (r)	6.67 Mb/s
Slot time	9 µs
SIFS time	16 µs
PHY header duration	40 µs
TF duration	100 µs
MU-BACK duration	68 µs
MPDU length (L)	10 kb
Guard interval (GI)	1.6 µs
OFDM symbol duration	12.8 µs

TABLE 6: RL-OBO training and testing parameters

Parameter	Value
No. of episodes	30 (training), 5 (testing)
No. of steps per episode	2000 (training), 1500 (testing)
Measuring interval ζ	10 contention rounds
Frequency of new station arrivals (i.e., change of the network configuration)	Every 100 steps
No. of network configurations	20 (training), 45 (testing)
r_S	3
r_U	2
r_E	1.5

otherwise indicated, throughput refers to the aggregate network throughput),
- efficiency – measured at the AP as the number of successful RUs divided by the total number of RUs,
- collision probability – measured by each station as the ratio of successfully transmitted data frames and all transmission attempts (we report the average probability across all stations),
- fairness – measured using Jain's fairness index calculated either over the throughput or collision probability of each station.

We do not measure airtime since we evaluate only AP-triggered frame exchanges, i.e., the channel contains only consecutive UORA frame exchanges (Figure 1a). How much data is contained in these exchanges is reflected in the efficiency metric mentioned above. Furthermore, we are interested in determining the upper bound of RL-OBO and E-OBO performance, hence in some cases, the number of stations varies up to a dense network of 90 stations.

D. RL-OBO PERFORMANCE

In this section, we first explain the RL-OBO training process, which is performed in a scenario where the number of stations increases by a fixed number over time. Then, we show how the trained agent performs in dynamic scenarios, where the number of stations increases randomly over time. Table 6 provides the training and testing parameters for RL-OBO.

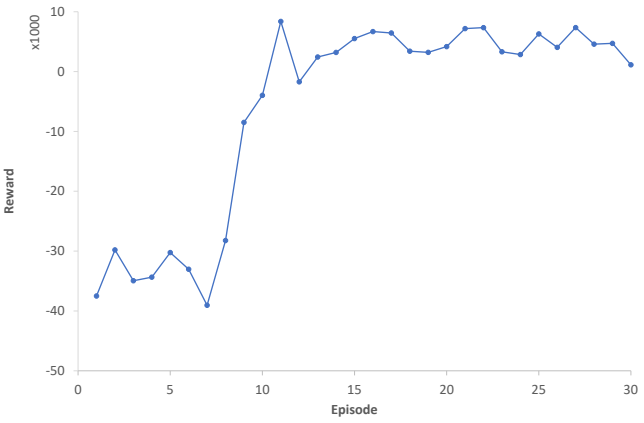


FIGURE 3: Reward for RL-OBO across 30 training episodes.

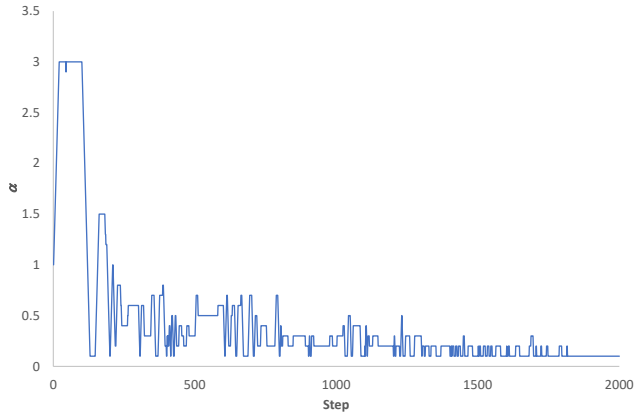


FIGURE 4: Evolution of RL-OBO's α in the final training episode.

1) Training

In the training scenario, a fixed number of new transmitting stations (five) arrive in the network every 100 steps. A single step consists of $\zeta = 10$ contention rounds, which gives the agent time to estimate the congestion level in the network. Additionally, every 500 steps we increase the number of RUs (from 4 to 32) and the number of transmitting stations is then set to $2 \times n_{\mathrm{RU}}$. Therefore, there are 20 (n_s, n_{RU}) configurations in each training episode.

Simulations show that a training duration of 30 episodes is sufficient; the reward stabilizes after about 10 episodes (Figure 3). These results confirm the correct operation of the implemented learning and its fast convergence.

The results for the final (30-th) training episode are shown in Figures 4 and 5. RL-OBO allows UORA to adjust the α parameter values to the number of contending stations and the number of available RUs. This results in a moderately high collision probability (p_c), high network efficiency and throughput, as well as high throughput fairness (f_t) and high collision probability fairness (f_{p_c}).

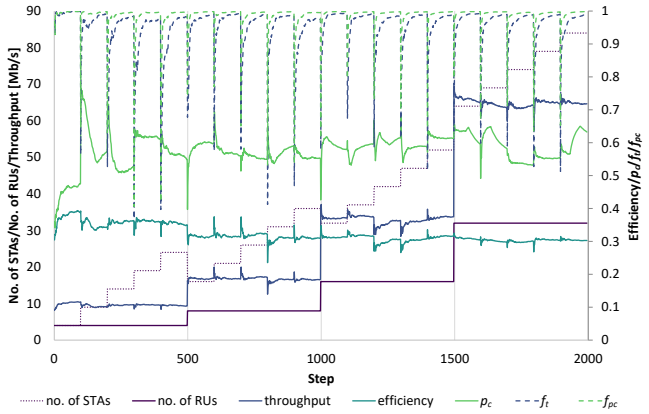


FIGURE 5: RL-OBO operation in the final training episode.

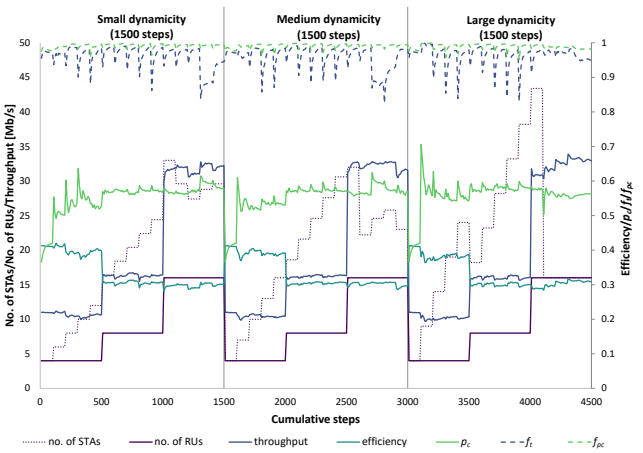


FIGURE 6: RL-OBO performance (the average of five testing episodes) for three dynamic test scenarios (small, moderate, and large network dynamicity). Results are gathered at the end of the measuring interval ζ. The number of stations arriving or departing (since step 1100) is randomly chosen from the ranges $[1, 5]$, $[1, 15]$, and $[1, 30]$, for the three scenarios respectively.

2) Testing

After training in fixed-increase scenarios, we test the operation of RL-OBO in three scenarios with varying network dynamics. We select the number of new station arrivals randomly from the ranges $[1, 5]$, $[1, 15]$, and $[1, 30]$ to reflect small, moderate, and high network dynamicity, respectively. Additionally, starting from step 1100, stations start leaving the network. The number of stations leaving the network is again randomly selected from the ranges $[1, 5]$, $[1, 15]$, and $[1, 30]$, respectively. We also set the minimum number of stations as $n_s = n_{\text{RU}}$, because if the number of stations is less than the number of available RUs, the resources would be underutilized, leading to low observed efficiency. All other aspects are similar to the training phase.

For each of the three network dynamicity scenarios, we perform five independent runs (episodes), each composed of 1500 testing steps in total (i.e., for each episode 15 different network configurations are tested). These settings amount to 45 different network configurations per episode (3 network dynamics $\times$ 15 configurations). The following metrics are measured: network efficiency, throughput, throughput fairness (f_t), and collision probability fairness (f_{p_c})

In Figure 6, we present the average results of the five testing episodes. Each point represents the results obtained for a different configuration. The performance of RL-OBO is highly satisfactory. For all measured metrics, the observations are similar to those for the training scenario.

E. E-OBO PERFORMANCE

To measure E-OBO performance[4], which serves as a non-ML benchmark, we use the parameters in Tables 5 and 7. First, we test E-OBO under constant changes to the number of transmitting stations, i.e., every 100 steps five new stations appear in the network, similarly to the RL-OBO training scenario. E-OBO allows UORA to adjust its operation to the number of competing stations, resulting in high values of the five measured metrics (Figure 7).

Next, we test E-OBO under more varying conditions, with low, moderate, and high network dynamicity. To ensure a fair comparison, we use a configuration similar to the RL-OBO dynamic case: the number of active stations in the network changes every 100 steps, while the analysis of each of the three network dynamics lasts 1500 steps (after which the

TABLE 7: Simulation parameters for E-OBO

Parameter	Value
Measuring interval ζ	10 contention rounds
Frequency of new station arrivals (i.e., change of the network configuration)	Every 100 steps
No. of steps per scenario	2000 (fixed), 1500 (dynamic)
No. of network configurations	20 (fixed), 45 (dynamic)

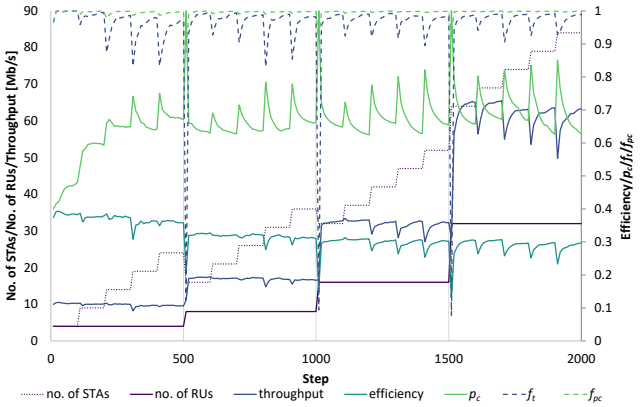


FIGURE 7: E-OBO performance in fixed scenarios.

[4]In [2], E-OBO was shown to outperform OBO-CTRL [13]. However, E-OBO was not previously analyzed for such dynamic scenarios; the results presented in this section are novel and are provided for completeness.

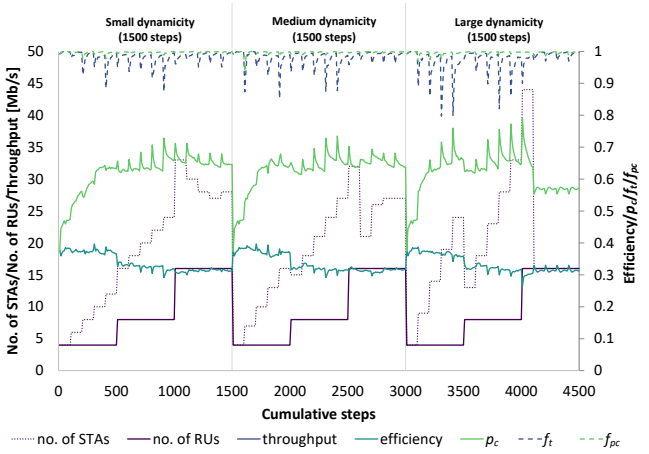


FIGURE 8: E-OBO performance in dynamic scenarios (small, moderate, and large network dynamicity). Results are gathered at the end of the measuring interval ζ.

number of stations and RUs is reset to four). Finally, we change the number of RUs every 500 steps to check the performance under different contention levels[5]. The results are shown in Figure 8. Once again, E-OBO allows UORA to quickly adjust to the number of competing stations and maintain high throughput, efficiency, throughput fairness, and collision probability fairness.

F. RL-OBO VS. E-OBO

We now compare the performance of RL-OBO with E-OBO. Figure 9 shows the final performance metrics for each configuration (i.e., the converged results at the end of a multiple of 100 steps). In general, both mechanisms perform comparably under low, moderate, and high dynamicity. Analyzing the performance in detail reveals the following. Although both methods have high throughput and collision probability fairness, E-OBO has more stable results. For RL-OBO, especially throughput fairness shows temporary decreases when the trend in the number of stations changes from increasing to decreasing. However, fairness remains high (above 0.9). Meanwhile, RL-OBO usually has a slightly lower collision probability. This translates to efficiency – lower (if there are too many empty RUs, i.e., the RL-OBO mechanism is too conservative) or higher (if the prediction of current network conditions is correct). However, the throughput values of both mechanisms are comparable as both adapt to changing network conditions quite well. To better highlight the similarities between the measured metric values, we compare the E-OBO and RL-OBO throughput results in fixed and dynamic scenarios in Figure 10. Clearly, regardless of supply (number of RUs) and demand (number of stations), both methods lead to comparable results.

[5]Congestion decreases with the same number of stations and an increasing number of available RUs.

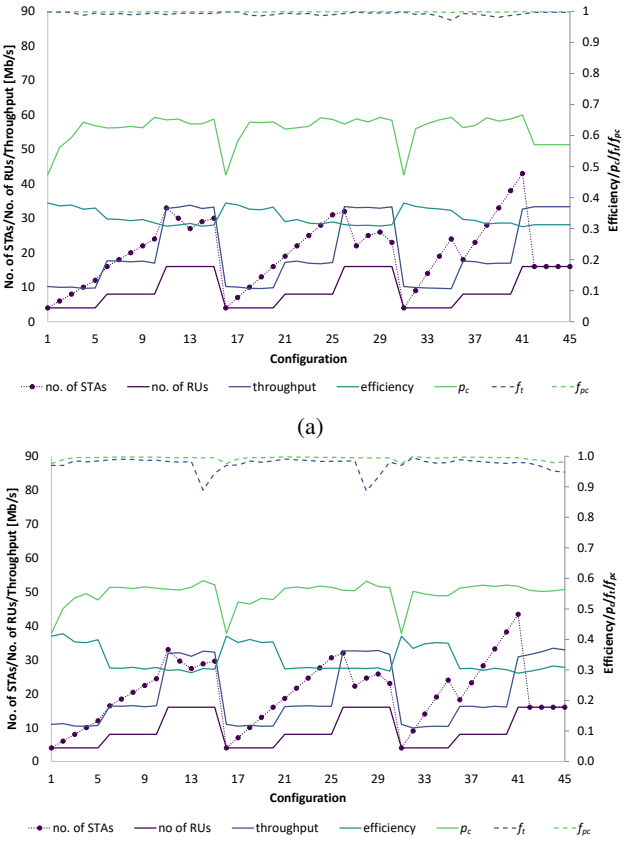


FIGURE 9: Performance for dynamic change of the number of transmitting stations at the end of a multiple of 100 steps: (a) E-OBO, (b) RL-OBO (the average of five testing episodes).

VI. CONCLUSION

In this paper, we addressed the problem of low UORA efficiency in dense 802.11ax deployments with high station contention. We have shown that ML can be used to improve the performance of the OBO mechanism, an important part of UORA. Furthermore, we compared the performance of the new RL-OBO mechanism with the E-OBO heuristic, which does not implement ML. The simulation results confirm that the proposed approach gives satisfactory results in various dynamic settings; however, when compared to E-OBO, RL-OBO does not provide meaningful advantages. Both mechanisms provide similar outcomes (i.e., throughput, channel access fairness, efficiency) and, therefore, the need for RL-OBO training and appropriate configuration of ML-related hyperparameters becomes a disadvantage. The selection of hyperparameters needs to be done carefully (e.g., empirically with a grid search approach), since different values may lead to completely different (and often worse) results. In contrast, E-OBO adjusts the α parameter on the fly using predefined thresholds. In summary, the result of the assessment of whether the ML-based solution (RL-OBO) outperforms a

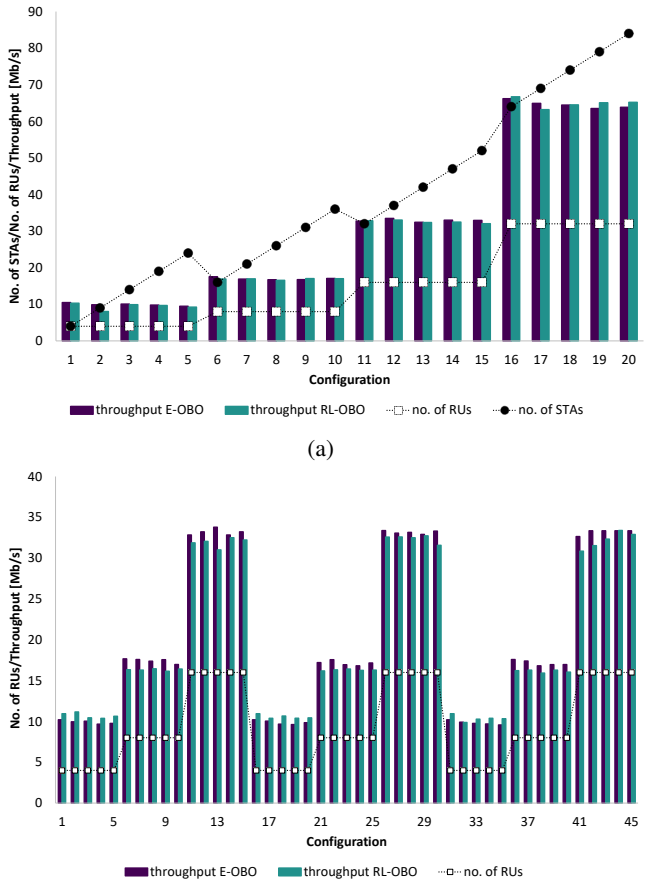


FIGURE 10: Comparison of E-OBO and RL-OBO performance in (a) fixed increase and (b) dynamic increase scenarios.

heuristic (E-OBO) can be interpreted as negative.

We conclude that ML should be used with care to resolve existing network problems. Although ML may provide satisfactory results, its application in a given area should be well thought out. As we have shown, in the case of UORA, the non-ML-based solution is already efficient, and the use of ML does not bring important advantages. At the same time, ML may be useful in distinguishing between collisions and channel errors [25] to improve the efficiency of E-OBO in more complex scenarios, e.g., with time-sensitive services [26]. However, this requires further validation. Furthermore, as future work, we envision the analysis of UORA in a full-protocol stack simulator such as ns-3.

REFERENCES

[1] B. Bellalta and K. Kosek-Szott, "AP-initiated Multi-user Transmissions in IEEE 802.11ax WLANs," *Ad Hoc Networks*, vol. 85, pp. 145–159, 2019.

[2] K. Kosek-Szott and K. Domino, "An Efficient Backoff Procedure for IEEE 802.11 ax Uplink OFDMA-Based Random Access," *IEEE Access*, vol. 10, pp. 8855–8863, 2022.

[3] M. Yang, B. Li, and Z. Yan, "MAC Technology of IEEE 802.11ax: Progress and Tutorial," *Mobile Networks and Applications*, vol. 26, no. 3, pp. 1122–1136, Jun. 2021.

[4] D.-J. Deng, Y.-P. Lin, X. Yang, J. Zhu, Y.-B. Li, J. Luo, and K.-C. Chen, "IEEE 802.11ax: Highly Efficient WLANs for Intelligent Information Infrastructure," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 52–59, 2017.

[5] E. Khorov, A. Kiryanov, A. Lyakhov, and G. Bianchi, "A Tutorial on IEEE 802.11ax High Efficiency WLANs," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 197–216, 2019.

[6] J. Bai, H. Fang, J. Suh, O. Aboul-Magd, E. Au, and X. Wang, "Adaptive Uplink OFDMA Random Access Grouping Scheme for Ultra-dense Networks in IEEE 802.11ax," in *IEEE/CIC International Conference on Communications in China (ICCC 2018)*. IEEE, 2018, pp. 34–39.

[7] ——, "An Adaptive Grouping Scheme in Ultra-dense IEEE 802.11ax Network Using Buffer State Report Based Two-stage Mechanism," *China Communications*, vol. 16, no. 9, pp. 31–44, 2019.

[8] Y. Li, B. Li, M. Yang, and Z. Yan, "A Spatial Clustering Group Division-based OFDMA Access Protocol for the Next Generation WLAN," *Wireless Networks*, vol. 25, no. 8, pp. 5083–5097, 2019.

[9] J. Kim, H. Lee, and S. Bahk, "CRUI: Collision Reduction and Utilization Improvement in OFDMA-Based 802.11ax Networks," in *IEEE Global Communications Conference (GLOBECOM 2019)*. IEEE, 2019.

[10] J. Wang, M. Wu, Q. Chen, Y. Zheng, and Y.-h. Zhu, "Probability Complementary Transmission Scheme for Uplink OFDMA-based Random Access in 802.11ax WLAN," in *IEEE Wireless Communications and Networking Conference (WCNC 2019)*. IEEE, 2019.

[11] L. Lanante, C. Ghosh, and S. Roy, "Hybrid OFDMA Random Access With Resource Unit Sensing for Next-Gen 802.11ax WLANs," *IEEE Transactions on Mobile Computing (TMC)*, vol. 20, no. 12, pp. 3338–3350, Dec. 2021.

[12] Y. Zheng, J. Wang, Q. Chen, and Y. Zhu, "Retransmission Number Aware Channel Access Scheme for IEEE 802.11ax Based WLAN," *Chinese Journal of Electronics*, vol. 29, no. 2, pp. 351–360, 2020.

[13] Y. Kim, L. Kwon, and E.-C. Park, "OFDMA Backoff Control Scheme for Improving Channel Efficiency in the Dynamic Network Environment of IEEE 802.11ax WLANs," *Sensors*, vol. 21, no. 15, p. 5111, 2021.

[14] A. Yang, B. Li, M. Yang, Z. Yan, and Y. Xie, "Utility Optimization of Grouping-based Uplink OFDMA Random Access for the Next Generation WLANs," *Wireless Networks*, vol. 27, no. 1, pp. 809–823, 2021.

[15] L.-H. Shen, K.-H. Liao, and K.-T. Feng, "Queue-Aware Uplink Arbitration-based Contention and Downlink Resource Allocation for Multi-APs for IEEE 802.11ax WLANs," in *IEEE Wireless Communications and Networking Conference (WCNC 2020)*, 2022, pp. 1755–1760.

[16] Y. A. Qadri, A. Nauman, A. Musaddiq, E. Garcia-Villegas, and S. W. Kim, "Preparing wi-fi 7 for healthcare internet-of-things," *Sensors*, vol. 22, no. 16, p. 6209, 2022.

[17] S. Szott, K. Kosek-Szott, P. Gawłowicz, J. Torres Gómez, B. Bellalta, A. Zubow, and F. Dressler, "Wi-Fi Meets ML: A Survey on Improving IEEE 802.11 Performance with Machine Learning," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 3, pp. 1843–1893, Jul. 2022.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," arXiv, cs.LG 1312.5602, Dec. 2013.

[19] S.-C. Chen, C.-Y. Li, and C.-H. Chiu, "An Experience Driven Design for IEEE 802.11ac Rate Adaptation based on Reinforcement Learning," in *40th IEEE International Conference on Computer Communications (INFOCOM 2021)*. Virtual Conference: IEEE, May 2021.

[20] Y. Zhao, J. Hu, K. Yang, and S. Cui, "Deep reinforcement learning aided intelligent access control in energy harvesting based wlan," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 14 078–14 082, 2020.

[21] W. Wydmanski and S. Szott, "Contention Window Optimization in IEEE 802.11ax Networks with Deep Reinforcement Learning," in *IEEE Wireless Communications and Networking Conference (WCNC 2021)*. Nanjing, China: IEEE, Mar. 2021.

[22] A. Kumar, G. Verma, C. Rao, A. Swami, and S. Segarra, "Adaptive Contention Window Design Using Deep Q-Learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2021)*. Toronto, Canada: IEEE, Jun. 2021.

[23] T. H. L. Dinh, M. Kaneko, K. Wakao, K. Kawamura, T. Moriyama, H. Abeysekera, and Y. Takatori, "Deep Reinforcement Learning-based User Association in Sub6GHz/mmWave Integrated Networks," in *IEEE Consumer Communications and Networking Conference (CCNC 2021)*. Las Vegas, NV: IEEE, Jan. 2021.

[24] D. Kotagiri, K. Nihei, and T. Li, "Distributed Convolutional Deep Reinforcement Learning based OFDMA MAC for 802.11ax," in *IEEE Interna-*

tional Conference on Communications (ICC 2021). Virtual Conference: IEEE, Jun. 2021.

[25] F. Frias, H. Alzamly, M. Kretschmer, A. Oliveira, and W. Moreira, "Applying Machine Learning Methods to Classify Wireless Link Errors, their Causes and Solutions," in *31st IEEE Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2020)*, 2020.

[26] S. Li, Y. Lu, and J. Li, "CAD-IDS: A Cooperative Adaptive Distributed Intrusion Detection System with Fog Computing," in *IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2022)*. Hangzhou, China: IEEE, May 2022.

.

APPENDIX A REWARD PARAMETER CALCULATION

In this appendix, we provide the rationale for selecting the constant values used by the proposed RL-OBO procedure when updating the reward. First, recall that in lines 6-8 of Algorithm 3, the agent's reward at step t (r_t) is increased by r_S, decreased by r_U, or decreased by r_E in case of successful, unsuccessful, and empty RUs, respectively.

To find good parameter settings, we simulate different congestion levels with the network configurations listed in Table 8. Then we perform model training (each training consisting of 15 episodes), in which $r_S = 3$ while r_U and r_E are changed linearly from 1 to 3 with a step of 0.5.

The results are presented in Figure 11 in the form of boxplots of the performance metrics achieved in the final training round. In this figure, we also define thresholds for each metric to better visualize the performance of each set of parameters.

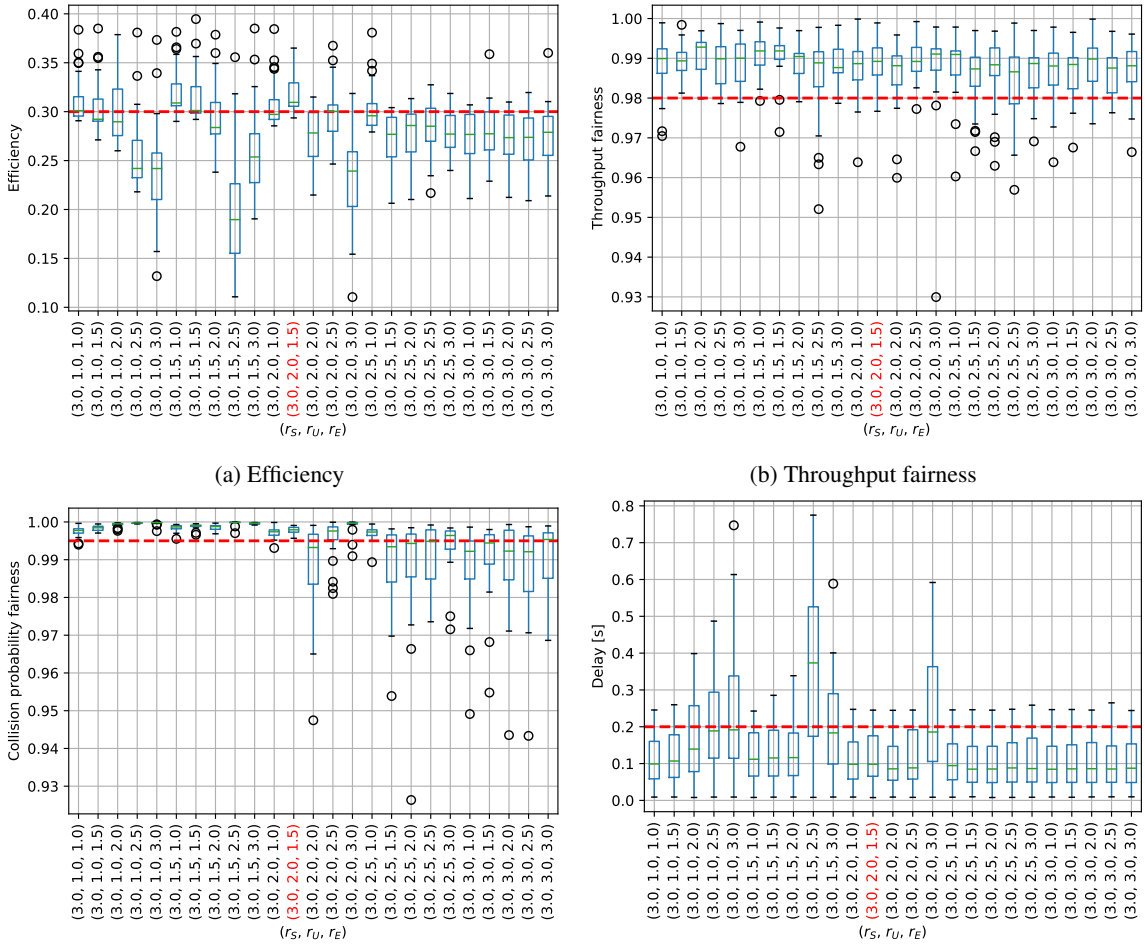
We conclude that the chosen set $(r_S, r_U, r_E) = (3, 2, 1.5)$ exhibits the best performance, i.e., the first (third) quartiles are above (below) the defined thresholds. In addition, $(3, 1.5, 1.5)$ and $(3, 1.5, 1)$ exhibit similar good performance but have slightly larger delay.

TABLE 8: Network configurations used to compare the performance of various reward parameter values.

Conf.	n_S	n_{RU}
1	4	4
2	9	4
4	14	4
6	19	4
3	24	4
5	16	8
7	21	8
8	26	8
10	31	8
9	36	8
11	32	16
12	37	16
13	42	16
14	47	16
15	52	16
16	64	32
17	69	32
18	74	32
19	79	32
20	84	32

KATARZYNA KOSEK-SZOTT received her M.Sc. and Ph.D. degrees in telecommunications (both with honors) from the AGH University of Science and Technology, Krakow, Poland, in 2006 and 2011, respectively. In 2016, she received her habilitation degree. Currently, she is working as an associate professor at the Institute of Telecommunications, AGH University of Science and Technology. Her general research interests are focused on wireless networking. The major topics include quality of service provisioning, novel amendments to the IEEE 802.11 standard, 5G networks, and beyond. She is a reviewer for international journals and conferences. She has been involved in several European projects: DAIDALOS II, CONTENT, CARMEN, FLAVIA, PROACTIVE, RESCUE, as well as grants supported by the Polish Ministry of Science and Higher Education and the National Science Centre. She has co-authored more than 70 research papers.

SZYMON SZOTT received his MSc and PhD degrees in telecommunications (both with honors) from the AGH University of Science and Technology, Krakow, Poland in 2006 and 2011, respectively. Currently, he is working as an associate professor at the Institute of Telecommunications of AGH University. In 2013, he was a visiting researcher at the University of Palermo (Italy) and at Stanford University (USA). His professional interests are related to wireless local area networks (channel access, quality of service, security, and inter-technology coexistence). He is a member of the IEEE 802.11 Working Group. In the past, he has been a member of ETSI's Network Technology Working Group Evolution of Management towards Autonomic Future Internet (AFI), a senior member of IEEE, and on the management board of the Association of Top 500 Innovators. He is a reviewer for international journals and conferences. He has been involved in several European projects (DAIDALOS II, CONTENT, CARMEN, MEDUSA, FLAVIA, PROACTIVE, RESCUE) as well as grants supported by the Ministry of Science and Higher Education and the National Science Centre. He is the author or co-author of over 70 research papers.



(a) Efficiency

(b) Throughput fairness

(c) Collision probability fairness

(d) Delay

FIGURE 11: Training results for various combinations of r_S, r_U, and r_E values. The combination used in the simulation results presented in the paper is highlighted on the X-axis in red. The horizontal dashed lines indicate arbitrary performance thresholds: efficiency should be larger than 0.3, throughput fairness – larger than 0.98, collision probability fairness – larger than 0.995, and delay – lower than 0.2 s.

FALKO DRESSLER is a full professor and Chair for Telecommunication Networks at the School of Electrical Engineering and Computer Science, TU Berlin. He received his M.Sc. and Ph.D. degrees from the Dept. of Computer Science, University of Erlangen in 1998 and 2003, respectively. He has been an associate editor-in-chief for IEEE Transactions on Mobile Computing and Elsevier Computer Communications, as well as an editor for journals such as IEEE/ACM Trans. on Networking, IEEE Trans. on Network Science and Engineering, Elsevier Ad Hoc Networks, and Elsevier Nano Communication Networks. He has chaired conferences such as IEEE INFOCOM, ACM MobiSys, ACM MobiHoc, IEEE VNC, IEEE GLOBECOM. He authored the textbooks Self-Organization in Sensor and Actor Networks published by Wiley & Sons and Vehicular Networking published by Cambridge University Press. He has been an IEEE Distinguished Lecturer as well as an ACM Distinguished Speaker. He is an IEEE Fellow and an ACM Distinguished Member. He is a member of the German National Academy of Science and Engineering (acatech). He has served on the IEEE COMSOC Conference Council and the ACM SIGMOBILE Executive Committee. His research objectives include adaptive wireless networking (sub-6GHz, mmWave, visible light, molecular communication) and wireless-based sensing with applications in ad hoc and sensor networks, the Internet of Things, and Cyber-Physical Systems.