


Article

On-Demand Channel Bonding in Heterogeneous WLANs: A Multi-Agent Deep Reinforcement Learning Approach

Hang Qi ^{1,2,3} , Hao Huang ^{1,2,3}, Zhiqun Hu ⁴, Xiangming Wen ^{1,2,3} and Zhaoming Lu ^{1,2,3,*}

¹ School of Information and Communication Engineering, Beijing University of Posts and Telecommunication, Beijing 100876, China; qh.sirius@gmail.com (H.Q.); huanghao@bupt.edu.cn (H.H.); xiangmw@bupt.edu.cn (X.W.)

² Beijing Key Laboratory of Network System Architecture and Convergence, Beijing University of Posts and Telecommunications, Beijing 100876, China

³ Beijing Laboratory of Advanced Information Networks, Beijing University of Posts and Telecommunications, Beijing 100876, China

⁴ School of Computing and Information Engineering, Hubei University, Wuhan 430062, China; zhiqunhu520@163.com

* Correspondence: lzy_0372@163.com; Tel.: +86-189-0138-1605

Received: 11 March 2020; Accepted: 12 May 2020; Published: 14 May 2020



Abstract: In order to meet the ever-increasing traffic demand of Wireless Local Area Networks (WLANs), channel bonding is introduced in IEEE 802.11 standards. Although channel bonding effectively increases the transmission rate, the wider channel reduces the number of non-overlapping channels and is more susceptible to interference. Meanwhile, the traffic load differs from one access point (AP) to another and changes significantly depending on the time of day. Therefore, the primary channel and channel bonding bandwidth should be carefully selected to meet traffic demand and guarantee the performance gain. In this paper, we proposed an On-Demand Channel Bonding (O-DCB) algorithm based on Deep Reinforcement Learning (DRL) for heterogeneous WLANs to reduce transmission delay, where the APs have different channel bonding capabilities. In this problem, the state space is continuous and the action space is discrete. However, the size of action space increases exponentially with the number of APs by using single-agent DRL, which severely affects the learning rate. To accelerate learning, Multi-Agent Deep Deterministic Policy Gradient (MADDPG) is used to train O-DCB. Real traffic traces collected from a campus WLAN are used to train and test O-DCB. Simulation results reveal that the proposed algorithm has good convergence and lower delay than other algorithms.

Keywords: WLAN; 802.11; channel bonding; multi-agent deep reinforcement learning

1. Introduction

Nowadays, IEEE 802.11 based Wireless Local Area Networks (WLANs) have been widely deployed around the world due to their low-cost and convenience. More and more users take WLAN as their first choice to access the Internet, which results in a fast growth of traffic. In order to meet the ever-increasing traffic, channel bonding is introduced in IEEE 802.11 standards. Channel bonding was first introduced in the IEEE 802.11n standard [1] where 802.11n nodes can transmit data packets in 40 MHz channel by bonding two contiguous non-overlapping 20 MHz basic channels. Then, the 802.11ac standard [2] further extends this capability by allowing the use of 80 MHz and 160 MHz channels by bonding four and eight basic channels, respectively. 802.11ax [3], which is the next generation WLAN standard, is expected to continue to develop channel bonding, such as supporting

wider channels and non-contiguous channel aggregation (CA) [4]. At present, the heterogeneous WLANs where nodes have different channel bonding capability are very common.

802.11 standards introduce two channel bonding schemes, static channel bonding (SCB) and dynamic channel bonding (DCB). The former is mandatory and the latter is optional. The access procedure of SCB and DCB is shown in Figure 1. At the beginning, a node determines its channel bonding parameter (P, B) , where P is the index of primary channel and B denotes the number of bonded channels. A 20 MHz basic channel is selected as primary channel, and other basic channels are secondary channels. In the example of Figure 1, the channel bonding parameter is $(1, 4)$. The legacy Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) is performed on the primary channel. Specifically, after a Distributed Coordination Function (DCF) Inter-Frame Spacing (DIFS) period, the node samples a backoff time from contention window (CW) and waits for the backoff time to decrease to zero. Before the backoff time reaches zero, the node senses secondary channels for one Point Coordination Function (PCF) Inter-Frame Spacing (PIFS) period. If all the primary and secondary channels are idle, the node will start a transmission. The above procedure is same for SCB and DCB. The difference between SCB and DCB lies in the bonding choice when not all secondary channels are idle in PIFS. If the node uses SCB, it will give up transmission, reselect the backoff time and proceed previous procedure until all secondary channels are both idle. However, if the node uses DCB, it will bond idle secondary channels which are contiguous to the primary channel as many as possible, and start a transmission. Of course, the bonded channel should be consistent with the channelization regulated by 802.11 standards, as shown in Figure 2.

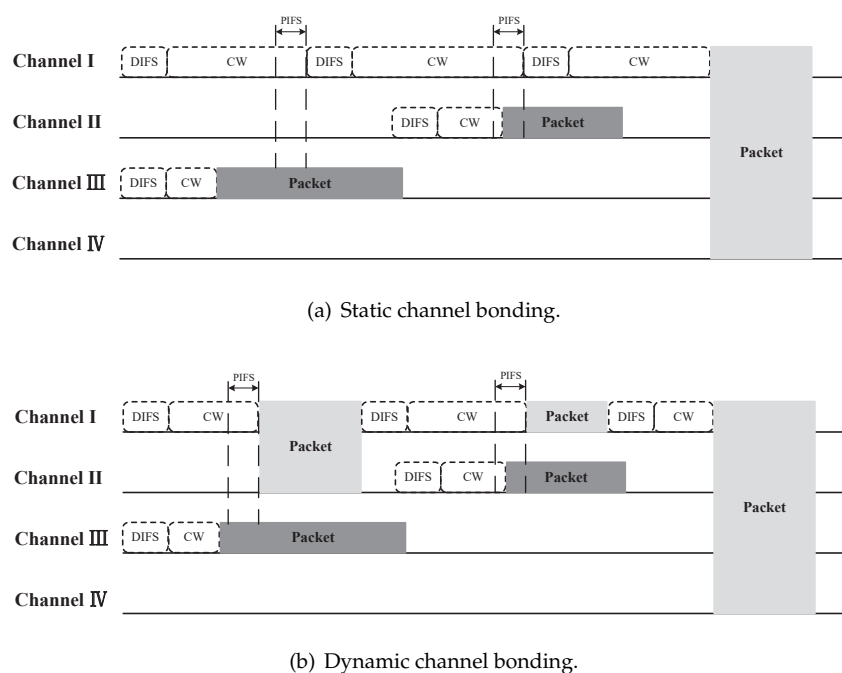


Figure 1. SCB and DCB in 802.11 standards.

The pros and cons of channel bonding are both prominent [5–9]. On the one hand, larger channel bandwidth effectively increases transmission rate, which can achieve a lower delay. On the other hand, a wider channel will cover a bigger frequency scope, and this wider channel tends to increase the channel overlapping probability, which may cause severe performance degradation due to packet collision and overlength backoff time. The experiments and theoretical analyses in [6,10] show that the channel bonding parameter (P, B) has an important effect on the performance of channel bonding and should be selected carefully to guarantee the performance gains.

In fact, the traffic load differs from one WLAN access point (AP) to another. Furthermore, the traffic load changes significantly depending on the time of day [11–13]. For example, the APs

located in a conference room have heavy traffic load only at meeting time and light load at other times, and the APs in offices usually have high load at working time. This raises a question of how to dynamically select channel bonding parameters for each AP to meet different and varying traffic demands and guarantee the performance gains.

According to the above analyses, in this paper, we proposed an On-Demand Channel Bonding (O-DCB) algorithm based on Deep Reinforcement Learning (DRL) [14] to select channel bonding parameter for heterogeneous WLANs, where the APs have different channel bonding capability. Selecting DRL is based on the following two reasons. The first reason is that DRL can learn the best policy by real-time interaction with the environment and only very minimal prior knowledge is needed. DRL embraces the learning ability of Reinforcement Learning (RL) and the generalization capability and approximation capacity of the Deep Neural Network (DNN). By online learning, it can effectively exploit the traffic pattern and adapt to the varying environment. Up to now, DRL and DNN have been applied in wireless communication to solve various problems [15], e.g., handoff management [16], base station sleeping [17] and performance prediction [18]. The second reason is that the periodicity and mobility of user behavior make the traffic loads exhibit some temporal and spatial correlations. According to the research findings in WLANs [12,17,19], the traffic load fluctuation shows repetitive pattern in weekday or week. DRL can catch this pattern and utilize it effectively.

To the best of our knowledge, this is the first work designing an on-demand channel bonding algorithm based on DRL. Simulation results reveal that the proposed algorithm has better performance than other channel bonding algorithms in terms of transmission delay. The main contributions of this paper can be summarized as follows:

- The on-demand channel bonding algorithm is designed in heterogeneous WLANs to decrease transmission delay, where APs have different channel bonding capability.
- The feasibility of DRL in channel bonding is explored in this paper. Real traffic traces collected from a campus WLAN [17] are used to train and test O-DCB, and simulation results show that O-DCB has lower delay than other channel bonding algorithms.
- In this problem, the state space is continuous and the action space is discrete. However, the size of action space increases exponentially with the number of APs by using single-agent DRL, which severely affects the learning rate. To accelerate learning, Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [20] is used to train O-DCB.

The remainder of this paper is organized as follows. Section 2 presents related works, and Section 3 presents system model and problem definition. The preliminaries of RL is introduced in Section 4. Section 5 gives a detailed description of O-DCB. In Section 6, the convergence and performance of the proposed algorithm are verified by numerical results. Finally, Section 7 concludes the paper.

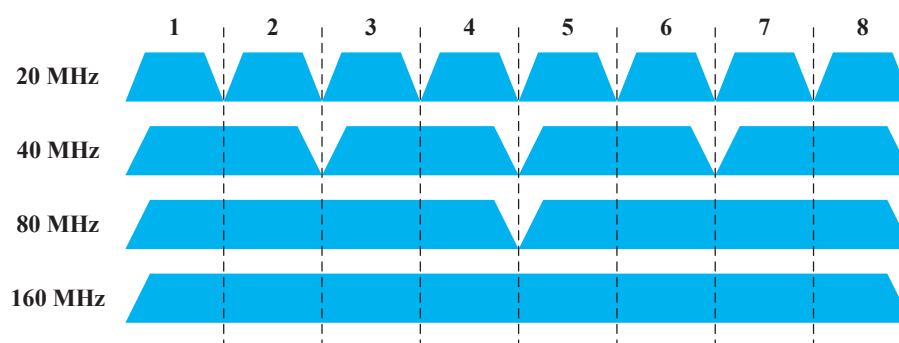


Figure 2. The channelization in 802.11 standards.

2. Related Works

2.1. Channel Bonding

The performance of channel bonding is studied in [5–9] via plenty of experiments and simulations, and the conclusions can be summarized as follows: (1) channel bonding can effectively increase throughput but is more susceptible to interference; (2) DCB usually has a better performance than SCB due to its flexibility, especially in node-intensive scenarios; (3) the primary channel and channel bonding bandwidth have important effects on the performance of channel bonding, and should be selected carefully to guarantee the performance gains.

On the other hand, a lot of works are devoted to analyzing the performance of channel bonding in theory. Bellalta et al. study the performance analysis of channel bonding, including SCB and DCB, in [10,21] by the use of Continuous Time Markov Network (CTMN), where the packet collisions are omitted and the traffic is saturated. Further research of channel bonding in high density WLANs, where nodes are not required to be within the carrier sense range of each other, are presented in [22]. Non-saturated traffic load scenarios are considered in [23–25] presents an analytical framework of Opportunistic Channel Bonding (OCB) where multiple channels shared by both legacy nodes and 802.11ac nodes with channel bonding capability. In [4], a renewal theory based analytical model is developed to study the performance of the DCB in 802.11ac, and non-contiguous CA in 802.11ax, with coexisting legacy single channel nodes.

Meanwhile, there are a number of papers which aim to improve channel bonding. In [26], the authors propose a MAC protection mechanism which effectively combats the hidden node problem on secondary channels. Stelter designs Channel Width Selection Scheme (CWSS) [27] which adjusts channel bandwidth according to data frame size, so as to improve bandwidth utilization. In [28], a channel allocation algorithm to achieve maximal throughput in DCB WLANs is proposed. Specifically, the throughput maximization is modeled as an integer nonlinear programming (INLP) problem and an optimal channel allocation algorithm based on the Branch-and-Bound Method (BBM) is used to solve the INLP. In [29], a reinforcement learning based channel bonding adaptation algorithm termed PoBA is developed to solve the hidden channel problem in 802.11ac networks. Quality of Service (QoS) requirements are considered in [30] and an enhanced dynamic channel bonding combining with the Transmission Opportunity (TXOP) [31] is developed. A primary channel selection scheme for 802.11ac nodes coexisting with legacy nodes to maximize network throughput is proposed and verified in [32]. Recently, a dynamic channel bonding algorithm [13] is proposed for 802.11ac WLANs, which considers the traffic demands of each WLAN. However, the authors do not consider the heterogeneity of nodes.

2.2. Spectrum Assignment

Other works concerning channel bonding in WLANs focus on spectrum assignment. Spectrum assignment is usually modeled as a graph coloring problem (an NP-complete problem) or non-convex optimization problem, and is solved by heuristic algorithm [33–36].

The works which likewise consider traffic demand and use spectrum assignment to solve include [11,37–39]. In [38], an on-demand channel bandwidth adaptation algorithm, SampleWidth, is proposed, which is designed for a single isolated link. Ref. [11] develops a load-aware adaptive channel bandwidth architecture for WLANs to improve spectrum utilization and per-load fairness. Further research of [11] termed FLUID is presented in [37]. However, these works assume that the demand of each AP is fixed, and the spectrum division is not standard-compatible. The scenario with uncertain demand is studied in [39]. The authors propose an adaptive channel bonding algorithm where center frequency and channel bandwidth are jointly allocated. However, the probability distribution of each AP demand is needed to know in advance, which is difficult to implement in practice. In the same way, above works both do not consider the heterogeneity of nodes.

3. System Model and Problem Definition

3.1. System Model

In this paper, we consider a wireless network consisting of N WLANs, $\mathcal{V} = \{1, 2, \dots, N\}$, and a centralized controller (The proposed algorithm is especially suitable for enterprise and campus WLANs). Downlink transmission is considered in this paper. The type of WLANs include 802.11n, 802.11ac and 802.11ax. The centralized controller can control APs to adjust their channel bonding parameter and collect the traffic load information (e.g., load size, traffic arrival rate) from each AP. Without loss of generality, the time is discretized into isometric time slots $\{t_1, t_2, \dots, t_j, \dots, t_{end}\}$ with the slot duration of δ . In each time slot, the channel bonding parameter of each WLAN remains unchanged. Because the number of users in each WLAN is unrelated to the research in this paper, for simplicity, we assume that WLAN i consists of an AP i and an user i .

We assume that the distance between AP and its user is short and the packet transmission failure is only caused by packet collision. As shown in Figure 3, a conflict graph $G = (\mathcal{V}, \mathcal{E})$ which is an undirected graph is used to represent the interference relationship between WLANs, where \mathcal{E} is the set of edges. If there is an edge $e_{i,i'} \in \mathcal{E}$ and the channels of WLAN i and WLAN i' are overlapping, there may be packet collision between them. $\bar{\mathcal{V}}_i = \{v \mid e_{v,i} \in \mathcal{E}, v \in \mathcal{V}\}$ denotes all the WLANs adjacent to WLAN i in G . For example, in Figure 3, $\bar{\mathcal{V}}_1 = \{2\}$ and $\bar{\mathcal{V}}_2 = \{1, 3, 4, 5\}$.

There are K basic channels with the size of 20 MHz where the total channel bandwidth is $20K$ MHz, and the channelization which is regulated by 802.11 standards is shown in Figure 2. The set of optional channel bonding parameters for AP i , \mathcal{C}_i , is determined by the AP type and K . For example, when $K = 4$, there are 8 channel bonding parameters for 802.11n APs, i.e., $\mathcal{C}_i = \{(1, 1), (2, 1), (3, 1), (4, 1), (1, 2), (2, 2), (3, 2), (4, 2)\}$, and there are 12 channel bonding parameters for 802.11ac/ax APs, i.e., $\mathcal{C}_i = \{(1, 1), (2, 1), (3, 1), (4, 1), (1, 2), (2, 2), (3, 2), (4, 2), (1, 4), (2, 4), (3, 4), (4, 4)\}$. It is assumed that all other parameters affecting the transmission rate (e.g., transmit power, modulation and coding scheme, the number of spatial streams and guard interval) are fixed except for the channel bandwidth, which is denoted as R_B .

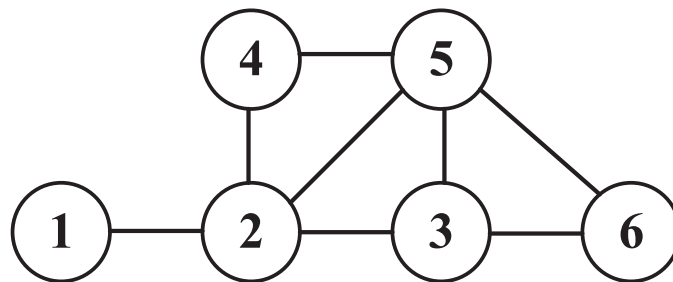


Figure 3. Conflict graph G .

3.2. Problem Definition

Consider a wireless network contains multiple WLANs with different channel bonding capability. There may be interference between WLANs and each AP has varying traffic demand. Our objective is to find the optimal channel bonding parameters for all APs with the consideration of channel bonding capability, varying traffic demand and potential interference relationships, in order to decrease the average transmission delay of the total network. The delay is defined as the time between the packet arriving at the AP buffer and the packet being received correctly.

As is mentioned above, this problem is a complicated sequential decision problem and traditional optimization algorithms are unsuitable. Fortunately, the research findings in [12,17,19] show that the traffic load fluctuation in WLANs shows repetitive pattern in weekday or week, which is caused by the periodicity and mobility of user behavior. On the other hand, DRL can catch the pattern and find

the optimal channel bonding parameters by interacting with the environment. As a result, we design O-DCB algorithm based on DRL. Before presenting O-DCB in detail, the preliminaries on RL will be briefly introduced in the next section.

4. Preliminaries on RL

This section gives a brief description of RL. For a comprehensive presentation, please refer to [14,40,41].

RL is learning how to map state to action, so as to maximize a numerical reward. The learner, i.e., the agent, is not told which actions to take, but instead must discover which actions yield the most reward by trying them, as shown in Figure 4. Markov Decision Processes (MDPs) are a mathematically idealized form of the RL problem [14]. An MDP is defined by a 4-tuple $(\mathcal{S}, \mathcal{A}, P, r)$, where \mathcal{S} is a finite set of states, \mathcal{A} is a finite set of actions, P is a transition probability from state s to state s' after the action a is executed, and r is the immediate reward received after transitioning from state s to state s' due to action a . The goal of an MDP is to find a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ to maximize the accumulated reward. The expected accumulated reward starting in state s and following policy π , termed state-value function, is defined as

$$v_{\pi}(s) = \mathbb{E}_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid S_t = s \right], \quad (1)$$

where $\gamma \in [0, 1]$ is the discount factor. Therefore, the optimal policy π^* can be defined as

$$\pi^* = \underset{\pi}{\operatorname{argmax}} v_{\pi}(s). \quad (2)$$

Similarly, the expected accumulated reward of taking action a in state s under a policy π is defined as

$$Q(s, a) = \mathbb{E}_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid S_t = s, A_t = a \right], \quad (3)$$

which is called action-value function or Q -function. The optimal action-value function $Q^*(s, a) = \max_{\pi} Q_{\pi}(s, a)$ can be used to choose the optimal actions. With Q^* , for any state s , the agent can simply find any action that maximizes $Q^*(s, a)$. That is,

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} Q^*(s, a). \quad (4)$$

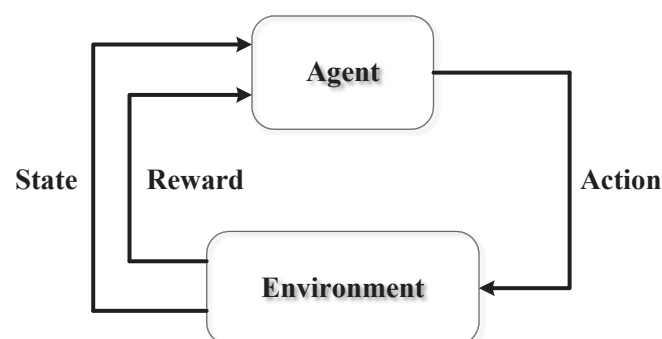


Figure 4. The basic component and form of reinforcement learning.

Based on Equation (4), the most effective and widely used method to find π^* without the need of the environment model is Q -learning [42]. In Q -learning, the action-value function is obtained by iterative processes, and the update rule is defined by

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right], \quad (5)$$

where α denotes learning rate and usually gradually decreases as the iteration. The core idea in Q -learning is to find the Temporal-Difference (TD) error between predicted value and current value, i.e., $r + \gamma \max_{a'} Q(s', a') - Q(s, a)$. Watkins et al. proved that Q -learning can converge to the optimal action-value function Q^* with probability one [42].

Q -learning can efficiently obtain an optimal policy when the state space and action space are small. However, it is not applicable to the complex problems which have continuous and high dimensional state spaces. In order to overcome this shortcoming, the function approximator should be introduced to process large state spaces. Deep Q -network (DQN), which uses a DNN to represent the action-value function in Q -learning, can learn the optimal policy in continuous and high dimensional state spaces. In DQN, the action-value function can be denoted as $Q(s, a | \theta)$, where θ are the weights of the DNN. θ are updated to reduce the mean-squared TD error $L(\theta)$ by the use of stochastic gradient descent, and $L(\theta)$ can be denoted as

$$L(\theta) = \mathbb{E} \left[\left(r + \gamma \max_{a'} Q(s', a' | \theta^-) - Q(s, a | \theta) \right)^2 \right], \quad (6)$$

where θ^- are the DNN weights in previous iteration.

Because RL is unstable or even to diverge when a nonlinear function approximator such as a neural network is used [43], experience replay and target Q -network are used to guarantee the convergence. In experience replay, a replay memory is used to store the agent's experiences (s, a, r, s') . If the replay memory is full, the oldest experiences will be discarded. The DNN is updated by sampling a minibatch randomly from the replay memory. As a result, the correlations between experiences are broken, and the variance of learning is reduced. In target Q -network, a copy of the DNN is created and updated periodically. It is used to generate the target values as shown in Equation (6), which further improving the stability of learning.

Further, to deal with the complex tasks which have both continuous, high dimensional state spaces and action spaces, Deep Deterministic Policy Gradient (DDPG) [41] is introduced. DDPG is a model-free off-policy actor-critic DRL algorithm. In particular, the DDPG algorithm maintains a parameterized actor neural network $\mu(s | \theta^\mu)$ which specifies the current policy by deterministically mapping states to a specific action. The parameterized critic neural network $Q(s, a | \theta^Q)$ is used to approximate the action-value function. The actor network is updated by applying the chain rule to the expected return from the start distribution J with respect to the actor parameters as follows:

$$\begin{aligned} \nabla_{\theta^\mu} J &\approx \mathbb{E} \left[\nabla_{\theta^\mu} Q(s, \mu(s | \theta^\mu) | \theta^Q) \right] \\ &= \mathbb{E} \left[\nabla_a Q(s, \mu(s) | \theta^Q) \nabla_{\theta^\mu} \mu(s | \theta^\mu) \right]. \end{aligned} \quad (7)$$

Similarly, experience replay and target network are used in DDPG to guarantee the convergence. Not the same as DQN, the weights of target networks are updated by slowly tracking the original networks: $\theta' \leftarrow \tau \theta + (1 - \tau) \theta'$ with the tracking parameter $\tau \ll 1$.

5. On-Demand Channel Bonding Algorithm

In this section, we design an on-demand channel bonding algorithm based on DRL. However, single-agent DRL is not suited to this problem due to the vast action space. In particular, in single-agent DRL, the agent should select channel bonding parameters for all APs, therefore the action is a N -dimensional vector where the i -th element is the channel bonding parameter of AP i . As a result, the size of action space equals $\prod_{i=1}^N |C_i|$ and increases exponentially with N , where $|\cdot|$ denotes the cardinality of a set. The vast action space severely affects the learning rate and even makes the algorithm unachievable in practice. On the contrary, multi-agent DRL has good scalability. In multi-agent DRL,

the size of action space is $|\mathcal{C}_i|$ for agent i . As such, we use a multi-agent DRL algorithm, MADDPG [20], to accelerate learning.

MADDPG is an extension of DDPG in multi-agent settings. It adopts the framework of centralized training with decentralized execution where the critic network is augmented with extra information about the policies of other agents, while the actor network only has access to local information. The extra information is only used to ease training and not used during the test. In MADDPG, each AP corresponds to an agent, and the basic three elements (state, action and reward) for agent i are designed as follows.

5.1. State

The state—more accurately, the observation—of agent i in time slot t_j is defined as

$$s_{i,j} = \{l_{i,j}, \lambda_{i,j-1}\}, \quad (8)$$

where $l_{i,j}$ denotes the load size of AP i at the beginning of t_j , and $\lambda_{i,j-1}$ is the average load arrival rate in the last time slot.

5.2. Action

Obviously, the action $a_{i,j}$ is

$$a_{i,j} = (P, B), \quad (9)$$

where $(P, B) \in \mathcal{C}_i$. \mathcal{C}_i is determined by the type of AP i and the number of basic channels, and may be different for different APs.

5.3. Reward

Because our objective is to reduce delay of the total network, the reward is designed as

$$r_{i,j} = \eta \cdot \bar{d}_j + \epsilon \cdot \bar{l}_{j+1}, \quad (10)$$

where \bar{d}_j is the average transmission delay of the total network in t_j . \bar{l}_{j+1} is the average load size of all APs at the end (start) of t_j (t_{j+1}), and $\bar{l}_{j+1} = \frac{\sum_i l_{i,j+1}}{N}$. η and ϵ are both weighting factor and less than zero.

5.4. O-DCB

The pseudo-code of O-DCB is presented in Algorithm 1. First of all, the actor network μ and the critic network Q are initialized for each agent, and the corresponding target networks are built. A replay buffer with a fixed size S_{RB} is also built. It should be noted that the replay buffer is shared by all agents but their actor-critic networks are personal.

At the start of each episode, the state of each agent is initialized to $\{0, 0\}$ because there are not any packet in each AP. At the start of time slot t_j , agents obtain the actions by their current actor networks μ_i , as shown in line 4. To support discrete actions in this problem, the Gumbel-Softmax estimator [44] is used. The output of actor network is the probability mass function (PMF) of all actions, and the action is determined by sampling from this PMF. Then, the APs change their channel bonding parameters according to the actions and the new parameters remain unchanged until the end of t_j . Next, the rewards and new states are obtained, and the global state s_j , global action a_j , global reward r_j and new global state s_{j+1} in this time slot are stored in the replay buffer. Next, for each agent i , it samples a minibatch with a size of L from the replay buffer. The agent calculates *loss* using the target networks of all agents, the samples in the replay buffer and its critic network, as shown in lines 11–12. The critic network parameters are updated by minimizing *loss* (line 12) and the actor network parameters are updated by policy gradient (line 14). Finally, after all agents updated their

actor network and critic network, agents update their target networks by slowly tracking the original networks (line 15).

Algorithm 1 O-DCB

Initialization:

Randomly initialize actor and critic neural network $\mu_i(s | \theta_i^\mu)$ and $Q_i(s, a | \theta_i^Q)$ for each agent.

Initialize corresponding target network Q_i' and μ_i' with weights $\theta_i^{Q'} \leftarrow \theta_i^Q, \theta_i^{\mu'} \leftarrow \theta_i^\mu$.

Initialize replay buffer RB .

Algorithm:

- 1: **for** episode in $\{1, 2, 3, \dots\}$ **do**
- 2: Receive initial state $s_{i,1} \leftarrow \{0, 0\}$.
- 3: **for** t_j in $\{t_1, t_2, \dots, t_{end}\}$ **do**
- 4: for each agent i , select $a_{i,j} = \text{Sample}[\mu_i(s_{i,j})]$.
- 5: All agents execute actions.
- 6: Calculate reward $r_{i,j}$ according to Equation (10).
- 7: Get new state $s_{i,j+1}$.
- 8: Store (s_j, a_j, r_j, s_{j+1}) in RB , where $s_j = (s_{1,j}, \dots, s_{N,j})$, $a_j = (a_{1,j}, \dots, a_{N,j})$, $r_j = (r_{1,j}, \dots, r_{N,j})$
and $s_{j+1} = (s_{1,j+1}, \dots, s_{N,j+1})$.
- 9: **for** agent i in $\{1, 2, \dots, N\}$ **do**
- 10: Sample a random minibatch of L samples (s_l, a_l, r_l, s_{l+1}) from RB .
- 11: Set $y_l = r_{i,l} + \gamma Q_i'(s_{l+1}, a_{1,l+1}, \dots, a_{N,l+1}) |_{a_{k,l+1} = \mu_k'(s_{k,l+1})}$
- 12: Update critic by minimizing the loss:

$$loss = \frac{1}{L} \sum_l (y_l - Q_i(s_l, a_l))^2.$$

- 13: Update actor using the sampled policy gradient:

$$\nabla_{\theta_i^\mu} J \approx \frac{1}{L} \sum_l \nabla_{\theta_i^\mu} \mu(s_{i,l}) \nabla_{a_i} Q_i(s_l, \mu_1(s_{1,l}), \dots, \mu_N(s_{N,l})).$$

- 14: **end for**
- 15: Update the target networks for each agent i :

$$\theta_i^{Q'} \leftarrow \tau \theta_i^Q + (1 - \tau) \theta_i^{Q'},$$

$$\theta_i^{\mu'} \leftarrow \tau \theta_i^\mu + (1 - \tau) \theta_i^{\mu'}.$$

- 16: **end for**
 - 17: **end for**
-

5.5. Implementation

O-DCB can be implemented in the centralized controller due to limited computation ability and storage space in APs. At the end of each time slot, the centralized controller collects essential information (load size, traffic arrival rate and so on) from each AP. Then, MADDPG is executed. The channel bonding parameters are obtained by the actor neural networks. Finally, new channel bonding parameters are distributed to each AP and APs use the new parameters in the next time slot. Because of the temporal and spatial correlations of traffic load, the learning process will gradually

stabilize. Besides, in order to avoid the poor performance in learning, initial learning can be executed in the background by the use of recorded traffic traces.

6. Simulation and Performance Evaluation

In this section, we present simulation to evaluate the performance of O-DCB. The simulation is performed with TensorFlow 1.13 [45] and Python 3.6. We consider a wireless network where there are 6 APs, and n , ac , ax denote the number of 802.11n APs, 802.11ac APs and 802.11ax APs, respectively. The interference relationship is shown in Figure 3. The number of basic channels K is equal to 4. We use real traffic traces dataset provided in [17], which are captured from a campus WLAN from September 2014 to January 2015 (The dataset is publicly available at <https://github.com/zaxliu/deepnap>). Each record in the dataset contains the following information: session ID, user ID, session start time and end time (UTC time), the total number of HTTP requests and bytes requested, etc. Similar with [17], we make the following assumption to mitigate the imperfection in the dataset: we assume a constant packet size S_p and uniform packet arrival in each session to translate coarse session-level summary to fine-grained packet arrival process. The traffic load changes of all APs per 6-minutes within 4 days are shown in Figure 5, where the data in each AP are processed with min-max normalization. We only consider traffic load from 8 a.m. to 8 p.m. (UTC+8) in everyday, because there is very little load at night. It is not difficult to see that the repeatability of traffic load fluctuation in a 7-day cycle, which is due to the periodicity of users activities in campus. We can also see that the traffic load is bursty in the daytime, which is a great challenge. In the simulation, the traffic traces of the first three days are used as training set, while the traffic traces in the last day are selected as test set. Feed-forward fully connected neural networks are used as function approximator in MADDPG. The main simulation parameters are listed in Tables 1–3.

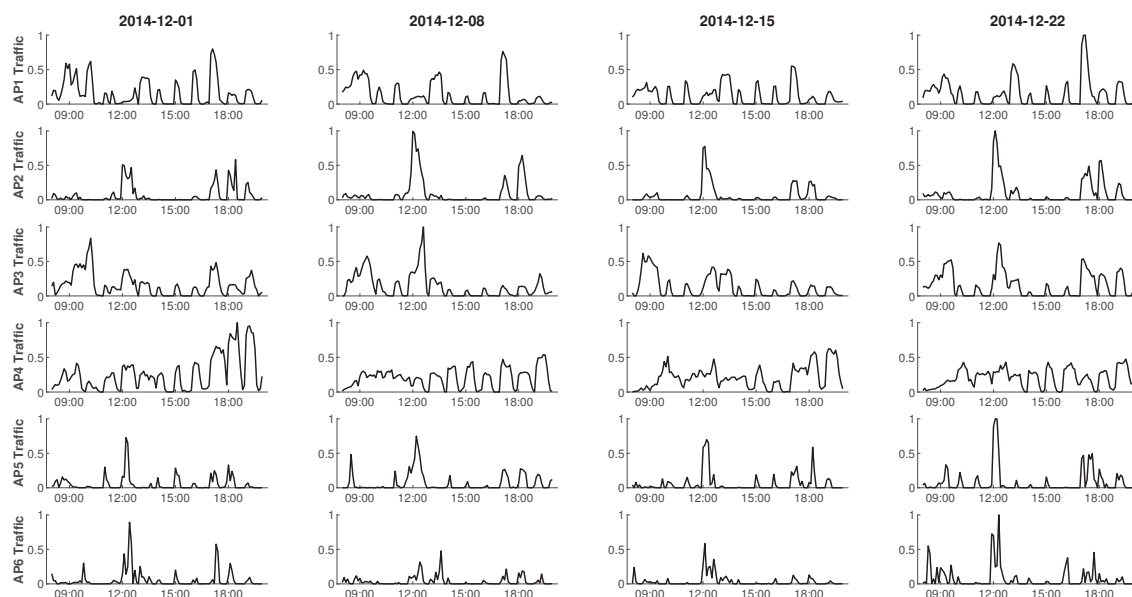


Figure 5. Traffic load changes of AP1–AP6 from 8 a.m. to 8 p.m. in 4 days.

In particular, independent DQN (IDQN), random selection and fixed configuration are used as benchmark.

- **IDQN.** The simplest approach of learning in multi-agent settings is to use independently learning agents, where each agent independently maximizes its individual reward and treats other agents as part of the environment. This is attempted with Q -learning in [46], which is called independent Q -learning (IQL). As Q -learning can be extended as DQN, IQL can be also upgraded to IDQN easily. In particular, IDQN uses the same state, action and reward with O-DCB. Besides,

the hyperparameters of IDQN are also the same with O-DCB, such as the learning rate, the size of replay buffer and minibatch, the structure of DNN, etc.

- **Random selection.** Random selection is very straightforward. In time slot t_j , for each AP i , the channel bonding parameter is randomly selected from its action space \mathcal{C}_i .
- **Fixed configuration.** Fixed Configuration (FC) is always used in real-life WLANs where the channel parameter is fixed for each AP and the widest channel is used.

Table 1. Simulation Parameters.

Notation	Definition	Value
N	The number of WLANs	6
K	The number of channels	4
S_p	Packet size	3 kB
CW_{min}	Min. contention window	16
CW_{max}	Max. contention window	1024
R_B	The transmission rate	13, 27, 42.5, 58.5 Mbps
δ	The length of time slot	6 min
η	The weighting factor	-0.25
ϵ	The weighting factor	-1
α	Learning rate	0.001
τ	The tracking parameter	0.01
S_{RB}	The size of replay buffer	4096
L	The size of minibatch	32
γ	The discount factor	0.95

Table 2. Parameters of actor neural network for agent i .

Name	Number	Size	Activation Function
Input Layer	1	2	NA
Hidden Layer	2	256, 256	ReLU, ReLU
Output Layer	1	8 (802.11n) 12 (802.11ac/ax)	NA

Table 3. Parameters of critic neural network for agent i .

Name	Number	Size	Activation Function
Input Layer	1	$14N - 4n$	NA
Hidden Layer	2	256, 256	ReLU, ReLU
Output Layer	1	1	NA

6.1. The Coexistence of 802.11n and 802.11ac

In this part, the coexistence scenario of 802.11n APs and 802.11ac APs is considered, and all APs use DCB. Apparently, 802.11ac APs have larger action space and better channel bonding capability than 802.11n APs. Concretely, we consider three different scenarios as follows: (1) six 802.11n APs; (2) the first three APs are 802.11ac and the rest APs are 802.11n; (3) six 802.11ac APs. In first two scenarios, for each AP, the channel bonding parameters of FC are $[(1, 2), (3, 2), (1, 2), (1, 2), (3, 2), (3, 2)]$ and $[(4, 4), (1, 4), (1, 4), (4, 2), (1, 2), (4, 2)]$ respectively. In the last scenario, the channel bonding parameters of FC are that all APs bond four basic channels. During the training of O-DCB, for each scenario, the absolute value of Total Reward Per Episode (TRPE) is shown in Figure 6. We can see that the training is fluctuant in the second scenario and is relatively stable in other scenarios. During the test, each agent in O-DCB only uses its trained actor network and its current state to generate actions without any extra information. Meanwhile, argmax is used to determine actions from the PMF other than sampling, i.e., $a_i = \text{argmax } \mu_i(s_i)$.

The average delay of total network over different algorithms and 802.11ac AP number is shown in Figure 7. It is easy to see that random selection has the worst performance. IDQN has better performance than random selection but is not as good as FC and O-DCB. This is because IDQN agents are independently updating their policies as learning progresses, which results in non-stationary of the environment from the view of any one agent. Although FC has good performance, there is much room for improvement. After training, O-DCB has shorter delay than FC. For example, in the first scenario, the channel bonding parameters of all APs used most often by O-DCB is [(4, 2), (2, 2), (4, 1), (3, 2), (3, 2), (2, 2)], which proves that using the widest channel is not always optimal. We can also see that the average delay of different algorithms decreases with the increase of 802.11ac APs except IDQN. This is because 802.11ac APs bring better channel bonding capability. However, due to the independent learning and ignoring other agents in IDQN, IDQN is very unstable. For example, when the number of 802.11ax APs is 3, the average delay of IDQN is 162 ms, which is much shorter than other scenarios. As a result, IDQN is unsuitable for the design of on-demand channel bonding algorithm.

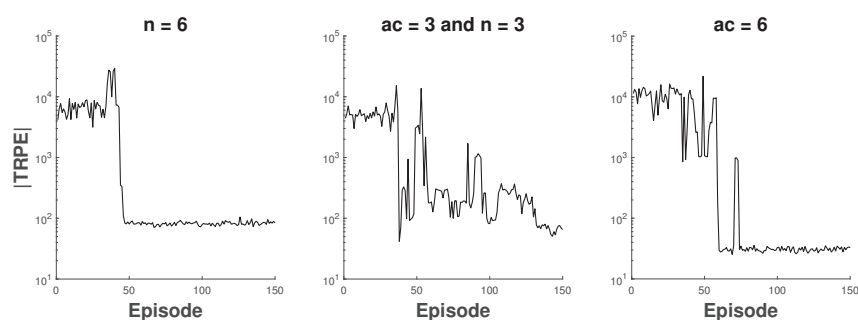


Figure 6. The total reward of each episode during training.

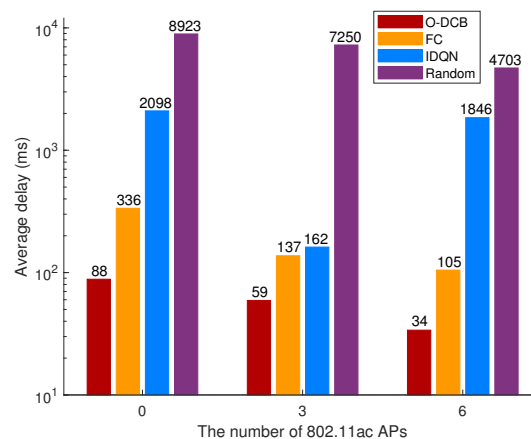


Figure 7. Comparisons of average delay over different algorithms and 802.11ac AP number.

6.2. The Coexistence of 802.11ac and 802.11ax

Besides SCB and DCB, 802.11ax can use non-contiguous CA [4]. When not all secondary channels are idle in PIFS, CA can use all idle secondary channels without any contiguity restriction. For example, 802.11ax APs can bond three arbitrary basic channels, which is unachievable in 802.11n/ac. In this part, the coexistence scenario of 802.11ac APs and 802.11ax APs is considered, where 802.11ac APs use DCB and 802.11ax APs use CA. Concretely, we consider three different scenarios as follows: (1) the first two APs are 802.11ax and the rest APs are 802.11ac; (2) the first four APs are 802.11ax and the rest APs are 802.11ac; (3) six 802.11ax APs. In all above scenarios, the channel bonding parameters of FC are that all APs bond four basic channels. The absolute value of TRPE in above scenarios during training is shown in Figure 8.

The average delay over different algorithms and 802.11ax AP number is shown in Figure 9. Same as the previous subsection, random selection has the worst performance and O-DCB has the best performance. Similarly, IDQN only has better performance than random selection, which is due to the isolated learning of each agent. The delay of FC is unchanged in three scenarios and is same with the scenario consisting of six 802.11ac APs. This is because CA and DCB will equivalent if all APs bond total channels. Due to the more flexible channel bonding of 802.11ax APs, the average delay of O-DCB is similar or shorter compared with the scenario consisting of six 802.11ac APs, and decreases with the increase of the number of 802.11ax APs.

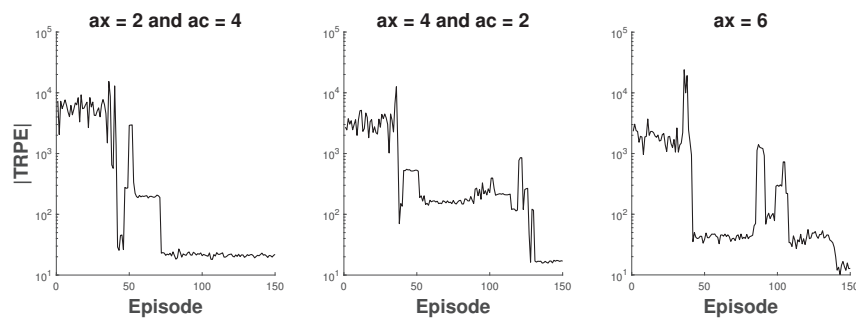


Figure 8. The total reward of each episode during training.

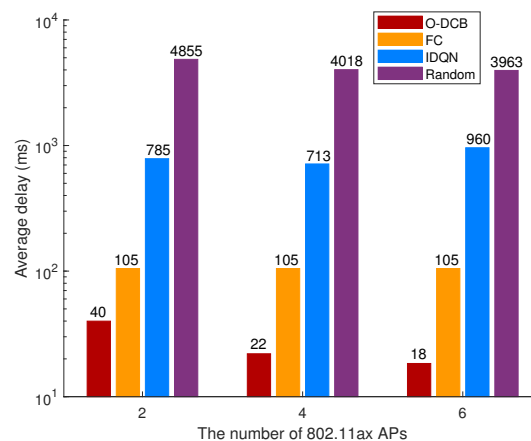


Figure 9. Comparisons of average delay over different algorithms and 802.11ax AP number.

7. Conclusions

In this paper, an on-demand channel bonding algorithm based on MADDPG for heterogeneous WLANs was proposed, where the APs had different channel bonding capability. In O-DCB, the state, action and reward were carefully designed under the consideration of reducing average transmission delay. Real traffic traces collected from a campus WLAN were used to train and test O-DCB. Simulation results showed that O-DCB had good convergence and lower delay than other algorithms.

Author Contributions: Conceptualization, H.Q., Z.H. and X.W.; methodology, H.Q., H.H.; software, H.Q., H.H.; validation, H.Q., H.H.; formal analysis, H.Q., Z.H.; investigation, H.Q., Z.H.; resources, X.W.; writing—original draft preparation, H.Q.; writing—review and editing, H.Q.; visualization, H.Q.; supervision, Z.L.; project administration, X.W. and Z.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by National Natural Science Foundation of China (Grant No.61671073).

Acknowledgments: Thank Jingchu. Liu et al. for their public dataset.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gast, M. *802.11 n: A Survival Guide*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2012.
2. Gast, M. *802.11 ac: A Survival Guide: Wi-Fi at Gigabit and Beyond*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2013.
3. Khorov, E.; Kiryanov, A.; Lyakhov, A.; Bianchi, G. A tutorial on IEEE 802.11 ax high efficiency WLANs. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 197–216. [[CrossRef](#)]
4. Khairy, S.; Han, M.; Cai, L.X.; Cheng, Y.; Han, Z. A renewal theory based analytical model for multi-channel random access in IEEE 802.11 ac/ax. *IEEE Trans. Mobile Comput.* **2018**, *18*, 1000–1013. [[CrossRef](#)]
5. Deek, L.; Garcia-Villegas, E.; Belding, E.; Lee, S.J.; Almeroth, K. The impact of channel bonding on 802.11 n network management. In Proceedings of the Seventh Conference on Emerging Networking EXperiments and Technologies, Tokyo, Japan, 6–9 December 2011; ACM: New York, NY, USA, 2011; pp. 1–12.
6. Park, M. IEEE 802.11 ac: Dynamic bandwidth channel access. In Proceedings of the 2011 IEEE International Conference on Communications (ICC), Kyoto, Japan, 5–9 June 2011; pp. 1–5.
7. Arslan, M.Y.; Pelechrinis, K.; Broustis, I.; Krishnamurthy, S.V.; Addepalli, S.; Papagiannaki, K. Auto-configuration of 802.11 n WLANs. In Proceedings of the 6th International Conference, Braga, Portugal, 1–3 September 2010; ACM: New York, NY, USA, 2010; pp. 1–12.
8. Bellalta, B.; Faridi, A.; Barcelo, J.; Checco, A.; Chatzimisios, P. Channel bonding in short-range WLANs. In Proceedings of the European Wireless 2014 20th European Wireless Conference, Barcelona, Spain, 14–16 May 2014; pp. 1–7.
9. Daldoul, Y.; Meddour, D.E.; Ksentini, A. IEEE 802.11 ac: Effect of channel bonding on spectrum utilization in dense environments. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–6.
10. Faridi, A.; Bellalta, B.; Checco, A. Analysis of dynamic channel bonding in dense networks of WLANs. *IEEE Trans. Mobile Comput.* **2016**, *16*, 2118–2131. [[CrossRef](#)]
11. Moscibroda, T.; Chandra, R.; Wu, Y.; Sengupta, S.; Bahl, P.; Yuan, Y. Load-aware spectrum distribution in wireless LANs. In Proceedings of the 2008 IEEE International Conference on Network Protocols, Orlando, FL, USA, 19–22 October 2008; pp. 137–146.
12. Hernández-Campos, F.; Karaliopoulos, M.; Papadopoulou, M.; Shen, H. Spatio-temporal modeling of traffic workload in a campus WLAN. In Proceedings of the 2nd Annual International Workshop on Wireless Internet, Boston, MA, USA, 2–5 August 2006; ACM: New York, NY, USA, 2006.
13. Lee, S.S.; Kim, T.; Lee, S.; Kim, K.; Kim, Y.H.; Golmie, N. Dynamic Channel Bonding Algorithm for Densely Deployed 802.11 ac Networks. *IEEE Trans. Commun.* **2019**, *67*, 8517–8531. [[CrossRef](#)]
14. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
15. Luong, N.C.; Hoang, D.T.; Gong, S.; Niyato, D.; Wang, P.; Liang, Y.C.; Kim, D.I. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 3133–3174. [[CrossRef](#)]
16. Han, Z.; Lei, T.; Lu, Z.; Wen, X.; Zheng, W.; Guo, L. Artificial intelligence-based handoff management for dense WLANs: A deep reinforcement learning approach. *IEEE Access* **2019**, *7*, 31688–31701. [[CrossRef](#)]
17. Liu, J.; Krishnamachari, B.; Zhou, S.; Niu, Z. DeepNap: Data-driven base station sleeping operations through deep reinforcement learning. *IEEE Internet Things J.* **2018**, *5*, 4273–4282. [[CrossRef](#)]
18. Xu, L.; Wang, J.; Wang, H.; Gulliver, T.A.; Le, K.N. BP neural network-based ABEP performance prediction for mobile Internet of Things communication systems. *Neural Comput. Appl.* **2019**. [[CrossRef](#)]
19. Redondi, A.E.; Cesana, M.; Weibel, D.M.; Fitzgerald, E. Understanding the WiFi usage of university students. In Proceedings of the 2016 International Wireless Communications and Mobile Computing Conference (IWCMC), Paphos, Cyprus, 5–9 September 2016; pp. 44–49.
20. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, O.P.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 6379–6390.
21. Bellalta, B.; Checco, A.; Zocca, A.; Barcelo, J. On the interactions between multiple overlapping WLANs using channel bonding. *IEEE Trans. Veh. Technol.* **2015**, *65*, 796–812. [[CrossRef](#)]
22. Barrachina-Muñoz, S.; Wilhelmi, F.; Bellalta, B. Dynamic channel bonding in spatially distributed high-density WLANs. *IEEE Trans. Mobile Comput.* **2019**, *4*, 821–835. [[CrossRef](#)]

23. Kim, M.S.; Ropitault, T.; Lee, S.; Golmie, N. A throughput study for channel bonding in IEEE 802.11 ac networks. *IEEE Commun. Lett.* **2017**, *21*, 2682–2685. [[CrossRef](#)]
24. Barrachina-Muñoz, S.; Wilhelmi, F.; Bellalta, B. To overlap or not to overlap: Enabling channel bonding in high-density WLANs. *Comput. Netw.* **2019**, *152*, 40–53. [[CrossRef](#)]
25. Han, M.; Khairy, S.; Cai, L.X.; Cheng, Y. Performance analysis of opportunistic channel bonding in multi-channel WLANs. In Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016; pp. 1–6.
26. Gong, M.X.; Hart, B.; Xia, L.; Want, R. Channel bounding and MAC protection mechanisms for 802.11 ac. In Proceedings of the 2011 IEEE Global Telecommunications Conference-GLOBECOM, Houston, TX, USA, 5–9 December 2011; pp. 1–5.
27. Stelter, A. Channel width selection scheme for better utilisation of WLAN bandwidth. *Electron. Lett.* **2014**, *50*, 407–409. [[CrossRef](#)]
28. Kai, C.; Liang, Y.; Huang, T.; Chen, X. To bond or not to bond: An optimal channel allocation algorithm for flexible dynamic channel bonding in WLANs. In Proceedings of the 2017 IEEE 86th Vehicular Technology Conference (VTC-Fall), Toronto, ON, Canada, 24–27 September 2017; pp. 1–6.
29. Jang, S.; Shin, K.G.; Bahk, S. Post-CCA and reinforcement learning based bandwidth adaptation in 802.11 ac networks. *IEEE Trans. Mobile Comput.* **2017**, *17*, 419–432. [[CrossRef](#)]
30. Mammeri, S.; Yazid, M.; Bouallouche-Medjkoune, L.; Mazouz, A. Performance study and enhancement of multichannel access methods in the future generation VHT WLAN. *Future Gener. Comput. Syst.* **2018**, *79*, 543–557. [[CrossRef](#)]
31. Hu, Z.; Wen, X.; Li, Z.; Lu, Z.; Jing, W. Modeling the TXOP sharing mechanism of IEEE 802.11ac enhanced distributed channel access in non-saturated conditions. *IEEE Commun. Lett.* **2015**, *19*, 1576–1579. [[CrossRef](#)]
32. Khairy, S.; Han, M.; Cai, L.X.; Cheng, Y.; Han, Z. Enabling efficient multi-channel bonding for IEEE 802.11 ac WLANs. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–6.
33. Ko, B.J.; Rubenstein, D. Distributed self-stabilizing placement of replicated resources in emerging networks. *IEEE/ACM Trans. Netw.* **2005**, *13*, 476–487.
34. Mishra, A.; Banerjee, S.; Arbaugh, W. Weighted coloring based channel assignment for WLANs. *Mob. Comput. Commun. Rev.* **2005**, *9*, 19–31. [[CrossRef](#)]
35. Gong, D.; Zhao, M.; Yang, Y. Distributed channel assignment algorithms for 802.11 n WLANs with heterogeneous clients. *J. Parallel Distrib. Comput.* **2014**, *74*, 2365–2379. [[CrossRef](#)]
36. Wang, X.; Derakhshani, M.; Le-Ngoc, T. Self-organizing channel assignment for high density 802.11 WLANs. In Proceedings of the 2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC), Washington, DC, USA, 2–5 September 2014; pp. 1637–1641.
37. Rayanchu, S.; Shrivastava, V.; Banerjee, S.; Chandra, R. FLUID: improving throughputs in enterprise wireless lans through flexible channelization. *IEEE Trans. Mobile Comput.* **2012**, *11*, 1455–1469. [[CrossRef](#)]
38. Chandra, R.; Mahajan, R.; Moscibroda, T.; Raghavendra, R.; Bahl, P. A case for adapting channel width in wireless networks. *Comput. Commun. Rev.* **2008**, *38*, 135–146. [[CrossRef](#)]
39. Nabil, A.; Abdel-Rahman, M.J.; MacKenzie, A.B. Adaptive channel bonding in wireless LANs under demand uncertainty. In Proceedings of the 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), Montreal, QC, Canada, 8–13 October 2017; pp. 1–7.
40. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [[CrossRef](#)] [[PubMed](#)]
41. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
42. Watkins, C.J.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [[CrossRef](#)]
43. Tsitsiklis, J.N.; Van Roy, B. An analysis of temporal-difference learning with function approximation. *IEEE Trans. Autom. Control* **1997**, *42*, 674–690. [[CrossRef](#)]
44. Jang, E.; Gu, S.; Poole, B. Categorical Reparametrization with Gumble-Softmax. In Proceedings of the International Conference on Learning Representations (ICLR 2017), Toulon, France, 24–26 April 2017.
45. Google. TensorFlow. Available online: <https://www.tensorflow.org/> (accessed on 1 January 2020).

46. Tan, M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth International Conference on Machine Learning*, Amherst, NY, USA, 27–29 June 1993; ACM: New York, NY, USA, 1993; pp. 330–337.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).