Distributed DRL-based Resource Allocation for Multicast D2D Communications

Pei-Yu Gong[†], Chih-Hang Wang[♯], Jang-Ping Sheu[†], and De-Nian Yang[♯]
[†]Institute of Communications Engineering, National Tsing-Hua University, Taiwan
[♯]Institute of Information Science, Academia Sinica, Taiwan
E-mail: s106064704@m106.nthu.edu.tw, superwch7805@iis.sinica.edu.tw,
sheujp@cs.nthu.edu.tw, dnyang@iis.sinica.edu.tw

GLOBECOM 2021 - 2021 IEEE Global Communications Conference | 978-1-7281-8104-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/GLOBECOM46510.2021.9685485

***Abstract*—Device-to-device (D2D) communication is one of the promising solutions to improve spectrum efficiency and alleviate the mobile traffic explosion. However, interference mitigation and resource allocation in the underlying cellular network is a challenging task. In this paper, we propose a distributed deep reinforcement learning (DRL) based scheme to solve the interference mitigation and resource allocation problem. According to the channel status, each cellular user (CU) and D2D transmitter (D2D TX) will determine the appropriate reused channel and transmit power to maximize the system throughput. We propose a distributed DRL scheme and integrate two hotbooting algorithms into the scheme to improve the system throughput at the early stage of training. Simulation results show that the proposed distributed DRL with hotbooting outperforms the baselines regarding running time, message overhead, and throughput.**

I. Introduction

Device-to-Device (D2D) communications have been exploited to allow nearby devices to communicate directly in order to improve the spectrum efficiency in cellular networks [1]. However, channel reuse causes the mutual interference between D2D and cellular transmission. When multiple D2D users reuse the channels of cellular users (CUs), *cross-tier interference* will occur between CUs and D2D users, and *co-tier interference* will occur between D2D users [1], which severely degrades transmission rates. To enhance system throughput, multicast D2D allows a transmitter to deliver data to multiple receivers simultaneously on a single channel, whereas they choose different transmission rates for each link to alleviate interference [2]–[4]. Meshgi et al. [2], who maximized the total throughput of CUs and D2D groups under QoS requirements, formulated multicast D2D communications with channel reuse as a mixed-integer nonlinear programming problem [3]. Wu et al. [4] jointly allocated radio and power resources to reduce interference and increase total throughput. However, the above methods are not designed to optimize the resources effectively for dynamic D2D networks with mobile users.

To adapt to dynamic D2D networks, Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) [5] have been leveraged to allocate resources, where an *agent* (e.g., base station) interacts with the networks and observes the interaction results to maximize the reward function. Specifically, differing from traditional algorithms [2]–[4] that reprocess the whole algorithm to find the optimal solution, DRL adaptively makes decisions to optimize the reward based on the observed environmental information. Tan et al. [6] proposed a distributed DRL algorithm to maximize the weighted sum rate for D2D communications. Ye et al. [7] dynamically adjusted the power and channel allocation for vehicle-to-vehicle communications by distributed DRL. Zhang et. al [8] optimized energy-efficiency for hybrid unicast and multicast traffic in 5G by a DRL-based framework. Zhang et. al [9] put forward a DRL-based semi-decentralized algorithm to jointly select transmission mode and allocate resources for vehicle-to-everything communications. However, the above approaches cannot ensure the enhanced performance for dynamic multicast D2D before the AI model is well-trained, which leads to poor performance when users move. In contrast, we design a distributed DRL-based multicast scheme with the designed multicast hotbooting algorithms (detatiled later) to improve the system performance at the early stage of training.

Distributed learning enables devices to collaboratively learn a shared prediction model, while keeping all the training data in the devices to reduce data transmission overhead and maintain privacy [9], [10]. This paper explores the throughput optimization problem for distributed underlying multicast D2D, called D2D Channel Assignment and Power Allocation Problem (DCAPAP). We model DCAPAP as a Markov Decision Process (MDP), where each CU and D2D TX acts as a DRL agent and takes its action (resource allocation) based on its local observation. Then, we present a distributed DRL-based algorithm to assign the channel and allocate power for each D2D transmitter (D2D TXs) and CU. To accelerate the training process and enhance the system throughput, we propose two distributed multicast hotbooting algorithms, 1) Decentralized Interference-based Multicast Resource Allocation (D-IMRA) algorithm and 2) Decentralized Energy-efficient Multicast Resource Adjustment (D-EMRA), by exploiting the personalized user experience of each device. Different from the ϵ-greedy [5] algorithm taking actions randomly to solve MDP, D-IMRA spatially clusters CUs and D2D TXs to avoid mutual interference and examines their channel gains to configure an initial channel and power allocation. D-EMRA then adjusts the channel and power of CUs and D2D TXs according to their energy efficiency, defined as the ratio of UE's data rate to its

978-1-7281-8104-2/21/$31.00 ©2021 IEEE

power consumption, to elevate system performance, and this improvement experience is collected to accelerate the DRL training process. Simulation results show that the proposed distributed DRL with hotbooting outperforms the baselines regarding running time, message overhead, and throughput.

II. System Model and Problem Formulation

We consider a cellular network with a single BS and underlying multicast D2D communications. The network includes K cellular users (CUs) denoted by $\mathcal{K} = \{1, \ldots, K\}$ and N D2D multicast groups denoted by $\mathcal{N} = \{1, \ldots, N\}$. Each CU can directly communicate with the BS [11] and is assigned a unique channel [2]–[4], whereas each D2D user communicates with each other via a direct wireless link. In a D2D multicast group, a D2D transmitter (D2D TX) multicasts messages to the receivers (D2D RXs), and each D2D RX belongs to only one multicast group [2]–[4]. Let g_i denote the RX set belonging to the i-th multicast group, and $|g_i|$ represents the number of RXs in the group. Following [2], we assume there are K uplink channels, and each channel is occupied by a CU predetermined by the BS. Therefore, the uplink channel and CU have a one-to-one correspondence, and index k indicates both of them. For channel reuse, each D2D TX can reuse an uplink channel of a CU, and a channel can be reused by multiple D2D TXs, i.e., $\sum_{k\in\mathcal{K}} y_{i,k} \leq 1, \forall i \in \mathcal{N}$, where the binary variable $y_{i,k} \in \{0, 1\}$ indicates if D2D TX i reuses channel k, and $y_{i,k} = 1$ if D2D TX i reuses channel k; otherwise, $y_{i,k} = 0$.

Let $p_{k,k}^{BS}$ and $p_{i,k}^{D2D}$ be the decision variables representing the transmit power of the k-th CU and the i-th D2D TX, respectively. Let G_k^{BS} denote the channel gain from CU k to the BS and $G_{i,k}^{D2C}$ be the channel gain from D2D TX of group i to the BS at channel k. The instantaneous signal-to-interference-plus-noise ratio (SINR) of the received signal at the BS from CU k is expressed as $SINR_k^{BS} = \frac{G_k^{BS} p_{k,k}^{BS}}{\sum_{i\in\mathcal{N}} y_{i,k} G_{i,k}^{D2C} p_{i,k}^{D2D} + \sigma^2}, \forall k \in \mathcal{K}$, where σ^2 is the background noise. The data rate (bit/sec/Hz) of CU k is $R_k^{BS} = \log_2(1 + SINR_k^{BS}), \forall k \in \mathcal{K}$. Similarly, the SINR of D2D RX m in the i-th D2D group at channel k is $SINR_{i,m,k}^{D2D} = \frac{G_{i,m,k}^{D2D} p_{i,k}^{D2D}}{G_{i,m,k}^{C2D} p_{k,k}^{BS} + \sum_{i'\in\mathcal{N}, i'\neq i} y_{i',k} G_{i',m,k}^{D2D} p_{i',k}^{D2D} + \sigma^2}, \forall i \in \mathcal{N}, k \in \mathcal{K}, m \in g_i$, where $G_{i,m,k}^{D2D}$ is the channel gain to the receiver m from D2D TX in group i at channel k, $G_{i,m,k}^{C2D}$ is the channel gain from CU k to the receiver m in group i, and $G_{i',m,k}^{D2D}$ is the channel gain from D2D TX of group i' to the receiver m of group i over channel k. Since the data rate of multicast is bounded by the D2D RX with the worst channel quality [12], the data rate (bit/sec/Hz) of D2D TX i is $R_i^{D2D} = |g_i| \sum_{k\in\mathcal{K}} y_{i,k} \log_2(1 + SINR_{i,k}^*), \forall i \in \mathcal{N}$, where $SINR_{i,k}^* = \min_{m\in g_i} SINR_{i,m,k}^{D2D}$ is the SINR of D2D TX i.

To ensure reliable cellular and D2D multicast transmission, let γ^{BS} and γ^{D2D} respectively be their minimum SINR requirements, which can be specified by the service provider [13]. For CU k and D2D TX i, the SINR constraints are $SINR_k^{BS} \geq \gamma^{BS}$ and $SINR_{i,k}^* \geq y_{i,k}\gamma^{D2D}$, respectively. The power constraints for CUs and D2D TXs are $p_{min}^{BS} \leq p_{k,k}^{BS} \leq p_{max}^{BS}$ and $p_{min}^{D2D} \leq p_{i,k}^{D2D} \leq p_{max}^{D2D}$, respectively. By discretizing the transmit power range into $(\frac{p_{max} - p_{min}}{\Delta O} + 1)$ levels [14], we denote $\mathcal{P}^{BS}$ and $\mathcal{P}^{D2D}$ as the transmit power level set of CUs and D2D TXs, respectively. Equipped with the above models, DCAPAP can be formulated as follows.

Definition 1. *Given a set of cellular users $\mathcal{K} = \{1, \ldots, K\}$, a set of D2D multicast groups $\mathcal{N} = \{1, \ldots, N\}$, and their corresponding channel states, SINR requirements γ^{BS} and γ^{D2D}, and the power constraints $p_{min}^{BS}, p_{max}^{BS}, p_{min}^{D2D}$, and p_{max}^{D2D}, our goal is to maximize the long-term system throughput,*

$$\max_{\boldsymbol{Y},\boldsymbol{P}} \sum_{t=0}^{T} R_{sys}^t(\boldsymbol{Y}, \boldsymbol{P}) = \max_{\boldsymbol{Y},\boldsymbol{P}} \sum_{t=0}^{T} (\sum_{k\in\mathcal{K}} R_k^{BS,t} + \sum_{i\in\mathcal{N}} R_i^{D2D,t}), \tag{1}$$

where $\boldsymbol{Y} = [y_{i,k}, i \in \mathcal{N}, k \in \mathcal{K}]$ is a channel assignment matrix and binary variable $y_{i,k} = 1$ if D2D TX i reuses channel k; otherwise, $y_{i,k} = 0$. $\boldsymbol{P} = [p_{k,k}^{BS}, p_{i,k}^{D2D}, i \in \mathcal{N}, k \in \mathcal{K}]$ is the set of transmit power of CUs and D2D TXs, and R_{sys}^t is the total system throughput of CUs and D2D TXs in time slot t. $R_k^{BS,t}$ and $R_i^{D2D,t}$ are the data rates of CU k and D2D TX i in time slot t, respectively.

III. Proposed Algorithm

Most conventional algorithms are designed for static networks [2], [3], and they are not optimized for dynamic networks since they require a lot of time to execute the whole process when the network changes. To adapt to dynamic D2D multicast, we first follow [6]–[8] to model DCAPAP as an MDP and propose a distributed DRL-based algorithm CAPA to maximize system throughput. Then, we propose the hotbooting algorithms D-IMRA and D-EMRA to accelerate the training speed and improve the performance of CAPA at the early stage.

A. MDP

In the following, we first define the MDP state space, action space, and reward function of CUs and D2D TXs.

1) State-space $\mathbb{S}$: For each CU k, it observes the state $s_{CU,k}^t$ in time slot t containing 1) the current (i.e., time slot $t-1$) transmit power $\mathbf{p}_k^{BS,t-1} = \{p_{k,1}^{BS,t-1}, ..., p_{k,K}^{BS,t-1}\}$ of CU k at each channel, 2) the interference $\mathbf{I}_k^t = \{I_{k,1}^t, ..., I_{k,K}^t\}$ (observed by CU k) on each channel from D2D TXs to the BS in time slot t, 3) the received interference $\mathbf{I}_k^{BS,t} = \{I_1^{BS,t}, ..., I_K^{BS,t}\}$ of the BS on each channel in time slot t, 4) the channel gain $G_k^{BS,t}$ from CU k to the BS in time slot t, and 5) the instantaneous SINR , $SINR_k^{BS,t}$ of CU k in time slot t. Therefore, the state is given as $s_{CU,k}^t = \{\mathbf{p}_k^{BS,t-1}, \mathbf{I}_k^t, \mathbf{I}_k^{BS,t}, G_k^{BS,t}, SINR_k^{BS,t}\}$.

Different from CU, the observed state of D2D TX further includes the interference and channel gain of the D2D RX with the worst channel quality in the group and the number of D2D RXs satisfying the SINR constraint since the multicast data rate is bounded by the D2D RX with the worst channel quality.

Specifically, for each D2D TX i, the observed state in time slot t contains 1) the current transmit power $\mathbf{p}_i^{D2D,t-1}$ of D2D TX i at each channel, 2) the interference $\mathbf{I}_i^{*,t} = \{I_{i,1}^{*,t}, ..., I_{i,K}^{*,t}\}$ of the D2D RX with the worst channel quality in D2D group i on each channel in time slot t, 3) the interference $\mathbf{I}_i^t = \{I_{i,1}^t, ..., I_{i,K}^t\}$ on each channel observed by D2D TX i from other D2D TXs in time slot t, 4) the received interference $\mathbf{I}_i^{BS,t}$ of the BS on each channel in time slot t, 5) the channel gain $G_i^{D2C,t}$ from D2D TX i to the BS in time slot t, 6) the channel gain $G_i^{*,t}$ from D2D TX to the D2D RX with the worst channel quality in D2D group i in time slot t, 7) the worst instantaneous SINR $SINR_i^{*,t}$ of D2D group i in time slot t, and 8) the number of D2D RXs d_i^t satisfying the SINR constraint in D2D group i in time slot t. Therefore, the state is given as $s_{TX,i}^t = \{\mathbf{p}_i^{D2D,t-1}, \mathbf{I}_i^{*,t}, \mathbf{I}_i^t, \mathbf{I}_i^{BS,t}, G_i^{D2C,t}, G_i^{*,t}, SINR_i^{*,t}, d_i^t\}$.

2) Action-space $\mathbb{A}$*:* We denote by $a_{CU}^t = \{p_t\} \in \mathbb{A}_{CU}$ the power allocation of CU, where $p_t \in \mathcal{P}^{BS}$ is the transmit power level, and $\mathbb{A}_{CU}$ is the action space of CU. Let ΔO be the transmit power offset, and the size of $\mathbb{A}_{CU}$ is $(\frac{p_{max}^{BS} - p_{min}^{BS}}{\Delta O} + 1)$. To reuse the CUs' channel, the action of D2D TXs further includes channel assignment. Let $a_{TX}^t = \{h_t, p_t\} \in \mathbb{A}_{TX}$ be the action of a D2D TX, where $h_t \in \{\mathcal{K} \cup \{0\}\}$ is the index of reused channel and $h_t = 0$ is for the case that a D2D TX does not choose any reused channel. $p_t \in \mathcal{P}^{D2D}$ is the transmit power level, and $\mathbb{A}_{TX}$ is the action space of D2D TX. The size of $\mathbb{A}_{TX}$ is $K(\frac{p_{max}^{D2D} - p_{min}^{D2D}}{\Delta O} + 1) + 1$, because there are K reused channels that can be allocated to D2D TXs and $(\frac{p_{max}^{D2D} - p_{min}^{D2D}}{\Delta O} + 1)$ levels for power allocation, where the plus of 1 is for $h_t = 0$.

3) Reward function: The reward function r^t includes the system throughput and the QoS requirement conditions of CUs and D2D TXs to reflect the objective of DCAPAP.

$$\begin{aligned} r^t =& \omega_1 R_{sys}^t - \omega_2 \sum_{i \in \mathcal{K}} J(R_{min}^{BS} - R_k^{BS,t}) \\ & - \omega_3 \sum_{i \in \mathcal{N}} J(R_{min}^{D2D} - R_i^{D2D,t}), \end{aligned} \tag{2}$$

where R_{min}^{BS} and R_{min}^{D2D} are the minimum data rates of CUs and D2D TXs, respectively and $J(x)$ is a piecewise function that is used to calculate penalties. $J(x) = x$, if $x > 0$; otherwise, $J(x) = 0$. ω_1, ω_2, and ω_3 are tuning knobs to adjust the importance of different factors.

B. Preliminaries on DRL

Deep Q-Networks, one of the DRL techniques, exploits a neural network denoted by $Q(s_t, a'; \theta)$ to estimate Q-value, where s_t is the state in time t, a' is the action, and θ is the network parameter (weights). In DRL, a long-term optimization problem is modeled as an MDP, typically includes a state space $\mathbb{S}$, an action space $\mathbb{A}$, and a reward function. In each time t, a DRL agent observes state s_t and then chooses the action a_t leading to the best Q-value.

$$a_t = \arg\max_{a' \in \mathbb{A}} Q(s_t, a'; \theta). \tag{3}$$

After taking an action a_t, the agent gets an immediate reward r_t, enters state s_{t+1}, and updates the Q-value as $Q(s_t, a_t; \theta) + lr\,[r_t + \beta \max_{a' \in \mathbb{A}} Q(s_{t+1}, a'; \theta) - Q(s_t, a_t; \theta)]$, where $lr \in [0, 1]$ is the learning rate, $\beta \in [0, 1]$ is the discount factor to determine the importance of future rewards, and $r_t + \beta \max_{a'} Q(s_{t+1}, a'; \theta)$ is the temporal difference (TD) target estimated by TD prediction [5]–[9]. The agent maintains a replay memory to store the seen experiences (also called transitions) (s_t, a_t, r_t, s_{t+1}) into the replay memory D. In each time slot, the agent randomly samples a mini-batch from D, and a gradient descent backpropagation algorithm based on the loss function $L(\theta)$ is executed to update parameter θ.

$$L(\theta) = \sum_{(s,a) \in D} (y - Q(s, a; \theta))^2, \tag{4}$$

where $y = r + \beta \max_{a' \in \mathbb{A}} Q(s', a'; \theta^-)$ is the target Q-value and r is the corresponding reward. $Q(s, a; \theta)$ is the output of the current Q-network, which is used to evaluate the Q-value of the current state action pair, and $Q(s, a; \theta^-)$ is the output of the target network, which is updated every F steps with parameter θ^-. An ϵ-greedy [5] is usually used in DRL to improve learning efficiency, where ϵ is a probability annealed during training. The agent takes the best action according to (3) with probability $1 - \epsilon$; otherwise, the agent chooses a random action with probability ϵ. However, this approach may select a worse action, leading to poor system throughput.

C. Distributed DRL Scheme

We first propose a distributed DRL-based multicast framework, named Channel Assignment and Power Allocation (CAPA), and the proposed hotbooting algorithms will be detailed later. Since the multicast transmit power is dominated by the worst channel gain in a multicast group, it is crucial to evaluate the interference between multiple users. Different from previous DRL [6] ignoring the influence between the decisions of different users, the BS in CAPA periodically clusters CUs and D2D TXs according to their locations by the K-means method to avoid mutual interference [15]. The BS designates the closest CU or D2D TX in each cluster as the cluster head, which is responsible for channel and power allocation to alleviate the computation loads of the BS. Then, it sequentially sends the information of corresponding cluster members and the candidate reused channels to each cluster head according to the descending order of the number of candidate reused channels, because the cluster with more candidate reused channels can allocate channels more flexibly. When each cluster head receives the notification, it performs D-IMRA (detailed later) to allocate the channel and power for its cluster members and then sends the result to the BS and its members. When the BS receives the result, it informs the next cluster head. After all cluster heads perform D-IMRA, the BS updates the throughput threshold R_{TH} (detailed later). Differing from the ϵ-greedy [5] taking an action randomly, CAPA ensures the enhanced throughput in each training iteration.

For the model training in Fig. 1, the BS maintains two shared prediction models for CU and D2D TX and initializes the weights θ_{CU} and θ_{TX} of the Q-networks. Afterward, it multicasts the weights to CUs and D2D TXs. Each CU and

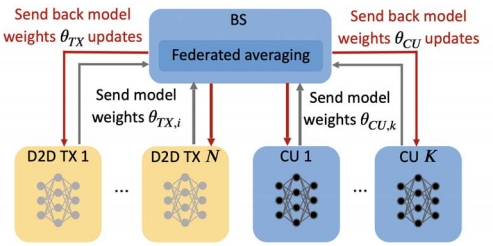

Fig. 1: Distributed learning architecture.

D2D TX then trains its own DQN model by the received weights and its local data. If the total throughput obtained from the learning is lower than the throughput threshold R_{TH}, each cluster head sequentially performs D-EMRA (detailed later), according to the descending order of the number of its candidate reused channels. In other words, the agent switches the strategy from ϵ-greedy to D-EMRA when the system throughput is lower than R_{TH} during training to ensure enhanced performance. Each CU and D2D TX then sends its weights of local models to the BS. Afterward, the BS performs federated averaging [9] for updating the weights of the global model and multicasts the updated weights. A common federated averaging method is minibatch-based stochastic gradient descent [9]. The BS updates the global model of CUs by $\theta_{CU}^{t+1} \leftarrow \sum_{k\in\mathcal{K}} \frac{B_k}{B}\theta_{CU,k}^{t}$, where θ_{CU}^{t+1} and $\theta_{CU,k}^{t}$ are the weights of the global model of CU and local model of CU k, respectively. B_k and B are the local minibatch size of CU k and the total minibatch size of all CUs. Similarly, the BS update the global model of D2D TXs by $\theta_{TX}^{t+1} \leftarrow \sum_{i\in\mathcal{N}} \frac{B_i}{B}\theta_{TX,i}^{t}$, where θ_{TX}^{t+1} and $\theta_{TX,i}^{t}$ are the weights of the global model of D2D TXs and local model of D2D TX i, respectively. B_i and B are the local minibatch size of D2D TX i and the total minibatch size of all D2D TXs.

D. Hotbooting

To avoid the performance loss in the early training stage and adapt to network dynamics, we design a new multicast hotbooting technique [16] to initialize the Q-network parameters through the historical experience of similar scenarios. Specifically, we propose D-IMRA and D-EMRA to allocate the channel and power of CUs and D2D TXs at the early stage of DQN training. Then, the experiences obtained by D-EMRA are used for DQN model training.

1) D-IMRA: From the SINR formula of CU, a higher channel gain G_i^{D2C} represents that the D2D TX i is closer to the BS and more easily interferes with the uplink communications of the corresponding CU. Hence, D-IMRA allocates the channels and transmit power to D2D TXs in the cluster in ascending order of G_i^{D2C} since a larger G_i^{D2C} tends to cause greater interference to the uplink communications. For each D2D TX i, D-IMRA examines the next feasible channel until the channel allocation for D2D TX i is successful or all feasible channels have been examined. When $G_{i,k}^{C2D,*}$ and $G_{i',i}^{D2D,*}$ are higher, CU k and D2D TX i' tend to interfere with the multicast communication of D2D TX i, where $G_{i,k}^{C2D,*}$ and $G_{i',i}^{D2D,*}$ are the maximum channel gain from CU k and D2D TX i' to D2D TX i, respectively. Different from the ϵ-greedy [5] ignoring the interference from nearby users, D-IMRA considers the interference caused by the corresponding CU of the reused channel and other D2D TXs in the cluster, which is estimated from the transmit power and channel gain of them. Let $\mathcal{K}'$ be the feasible channel set that only considers the CUs which meet the SINR constraint and belong to the candidate reused channel set. For each cluster C_c, D-IMRA evaluates the criteria $k^* = \arg\min_{k\in\mathcal{K}'}(p_k^{BS}G_{i,k}^{C2D,*} + \sum_{i'\in\mathcal{N}\cap\mathcal{C}_c, i'\neq i} p_{i,k}^{D2D}G_{i',i}^{D2D,*}y_{i',k})$ for finding the channel that receives the minimum interference from CU k and all other D2D TXs in the cluster using the same channel, where $\mathcal{C}_c$ is the c-th cluster set. After finding the channel k^* for D2D TX i, D-IMRA allocates proper transmit power to D2D TX i by iterating through all potential power in $\mathcal{P}^{D2D}$. Then, it calculates the total data rate and SINR of CU k^* and D2D TXs in the cluster to ensure the minimum SINR requirements. Then, D-IMRA updates the channel assignment matrix and transmit power set and removes the allocated channel from the feasible channel set for the next iteration. D-IMRA stops after finding a channel for D2D TX i, or if no feasible channels can be assigned to D2D TX i. After all D2D TXs in the cluster are allocated in order, D-IMRA will terminate. Finally, we obtain the channel assignment for D2D TXs and power allocation for CUs and D2D TXs and the system throughput, and the BS updates the throughput threshold R_{TH} to the current throughput for the DQN switching strategy from ϵ-greedy to D-EMRA.

2) D-EMRA: A larger transmit power is beneficial to the user's data rate but causes greater interference to other communications. Different from the distributed learning with ϵ-greedy [5] taking an action randomly without global knowledge, D-EMRA iteratively adjusts the allocation of the CU or D2D TX which has the worst energy efficiency in the cluster for improving the overall system throughput. D-EMRA first finds the UE with the worst energy efficiency in the cluster by calculating the ratio of the user's data rate to its power consumption. For each CU, D-EMRA iterates through all potential power in $\mathcal{P}^{BS}$ to look for the optimal power of the CU. It calculates the total data rate of CU j^* and D2D TXs in the cluster based on the potential power of CU j^* and examines their SINR and total data rate to ensure the QoS requirements. For each D2D TX, D-EMRA iterates all possible reused channels in $\mathcal{K}'$ and power combinations of D2D TX j^* to find the optimal channel and power combination. If the optimal allocation is found, the results are then put into the replay memory of DQN for providing effective experiences to help model training.

IV. Simulations

A. Simulation Setup

We consider a single cell network with a coverage area of 1000×1000 m^2 [2] with the CUs and D2D groups being deployed over the whole area. The D2D TXs are deployed

over the whole cell randomly and each D2D TX has $3 \sim 8$ nearby D2D RXs to form a D2D group. The number of D2D groups is set to 14 by default. The distribution of D2D RXs in the D2D group follows the clustered distribution model in [17]. The transmission radius of D2D TX and the SINR requirement are set to 50 m [2], [12] and 5 dB [18], respectively. The network bandwidth is set to 20MHz and is equally divided into 20 channels. Following [19], the path loss models of cellular link and D2D link are $128.1+37.6log(d)$ and $148+40log(d)$, respectively, where d is the distance in kilometers and the noise spectral density is set to -174 dBm/Hz. The minimum and maximum transmit power of devices (i.e., p_{min} and p_{max}) are 10 and 20 dBm. We set the transmit power offset ΔO to 1 dB and the transmit power range is discretized into $(\frac{p_{max}-p_{min}}{\Delta O}+1)$ levels [14], [20]. Each D2D RX moves within the cluster, randomly at a speed of $1\ m/s \sim 2.5\ m/s$.

We implement DQN with Python 3 and Pytorch on a server with Intel i7-7700K 4.2-GHz CPU and 62-GB memory. Following [7], [21], our DQN model is based on a four-layer fully connected neural network with two hidden layers. We set 300 neurons for each hidden layer. To initialize the weights of each layer, we use the normal distribution with mean 0 and variance 0.1 in our DQN model. ReLu is used as an activation function [6], and the Adam optimizer with the learning rate $lr = 10^{-4}$ is used for training [9]. The discount factor $\beta = 0.5$, the batch size is 64, the frequency for updating the target network $F = 100$ and the size of replay memory $D = 10^4$ [22]. For distributed learning, the training model weights on different devices are averaged every 1000 time slots. Following [5], we adopt the ϵ-greedy strategy to balance exploration and exploitation for optimizing rewards. We set ϵ to 1 at the beginning and then subtract it by $(\frac{1-0.1}{1\times10^4})$ in each time slot until it reaches 0.1. We compare CAPA with the state-of-the-art distributed learning algorithm WSR-MADQN [6], and optimization algorithms HRA-M2O [3] and HRA-O2O [2]. Each simulation result is averaged over 10^5 samples.

B. Simulation Results

Fig. 2(a) compares the total data rate of D2D TXs, and it shows that CAPA significantly outperforms the baselines as the number of D2D TXs grows. In CAPA, each CU and D2D TX observes its channel information and the interference of neighbor D2D TXs during the training process to select the reused channel and allocate power. When more channels need to be reused, D2D TX can select channels with less interference and allocates power based on the observations to increase the total data rate of D2D TXs. In contrast, the performance of HRA-O2O remains steady since it does not allow multiple D2D TXs to reuse a single channel. In Fig. 2(b), we set the number of D2D TXs to 14 and vary the number of CUs from 10 to 20. Compared with Fig. 2(a), although CAPA results in similar system throughput to HRA-M2O, it generates much more D2D communication throughput since each cluster head in CAPA distributionally examines the channel gains of D2D TXs by evaluating received interference from neighbors to optimize local D2D throughput.

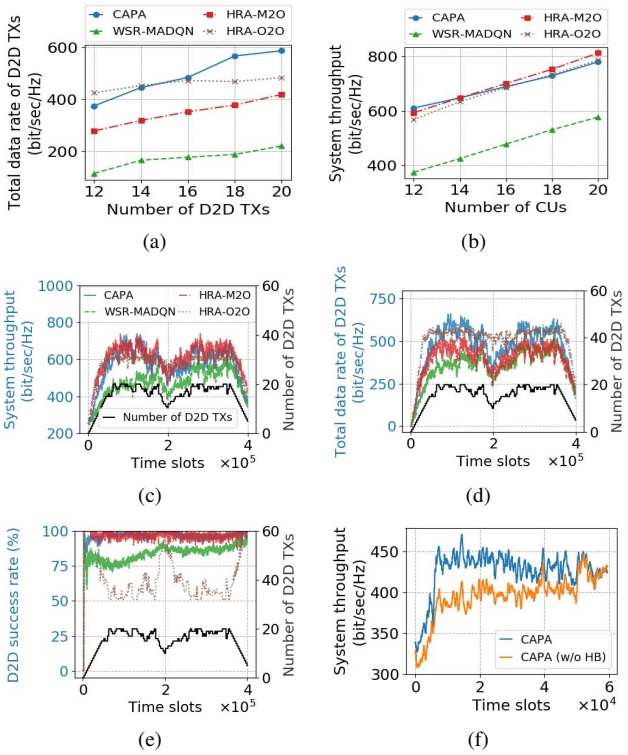

Fig. 2: Performance evaluation of different algorithms.

Figs. 2(c) - 2(e) compare the performance of different algorithms in dynamic networks (i.e., D2D TXs join and leave the network). The D2D success rate is the ratio of the number of D2D RXs whose QoS requirements are satisfied to the total number of D2D RXs. The left y-axis compares the system performance of different algorithms, and the right y-axis shows the number of D2D TXs (black curve) in the simulation time slots. In the beginning, there is no D2D TX, and we deploy a D2D TX every 3,000 time slots (about 5 minutes). The number of D2D TXs gradually rises until 45,000 time slots, and we remove a D2D TX every 3,000 time slots after 355,000 time slots. The number of CUs is fixed at 10. CAPA adapts to dynamic networks and achieves higher D2D and total throughput in Figs. 2(c) and 2(d) since the CUs and D2D TXs periodically exchange channel states of nearby devices to avoid interference during channel and power allocation. Moreover, the BS collects the local weights of the training model and performs federated averaging to ensure the global system performance (i.e., total throughput). In contrast, WSR-MADQN leads to worse throughput since it makes decisions based on only local observations and the interference may be severe to degrade the total throughput. CAPA also achieves better D2D throughput than WSR-MADQN and HRA-M2O. In Fig. 2(e), CAPA achieves nearly 100% D2D success rate at any moment, which is identical to the centralized optimization algorithm HRA-M2O. This is because CAPA avoids mutual interference when clustering CUs and D2D TXs, and it results in a higher D2D success rate even when the network changes.

TABLE I: Running time over $100,000$ time slots (Unit: min)

Algorithm	Number of D2D TXs				
	12	14	16	18	20
CAPA	42.98	76.08	93.19	95.51	101.12
CAPA (w/o HB)	0.68	0.66	0.53	0.62	0.60
WSR-MADQN	0.40	0.38	0.36	0.34	0.32
HRA-M2O	208.97	280.11	355.14	477.80	581.42
HRA-O2O	84.39	109.39	119.76	129.11	138.20

TABLE II: Message overhead (#messages, Unit: million)

Algorithm	Number of D2D TXs				
	12	14	16	18	20
CAPA	2.46	2.69	2.90	3.09	3.29
WSR-MADQN	13.2	18.2	24	30.6	38

However, HRA-M2O has a lower D2D throughput in Fig. 2(d) since it may lower the transmit power of D2D TXs to elevate the throughput of CUs for maximizing the total throughput.

Fig. 2(f) compares the convergence time of CAPA and CAPA (w/o HB), where CAPA (w/o HB) represents CAPA without hotbooting and the numbers of CUs and D2D TXs are set to 10. The multicast hotbooting accelerates the training speed more than 80% to achieve the converged system throughput of more than 420 bits/sec/Hz since it collects transitions by the proposed D-IMRA and D-EMRA, instead of acting randomly at the beginning. Table I evaluates the total running time over 100,000 time slots under different number of D2D TXs. CAPA (w/o HB) and WSR-MADQN have a shorter running time since they directly make decisions according to the learning result. Although the running time is similar, WSR-MADQN results in much worse performance as shown in Figs. 2(b) - 2(f). In contrast, HRA-M2O and HRA-O2O require much more time to search for the solution repeatedly when the network is dynamic. Together with Fig. 2, it can be seen that CAPA can achieve better performance with less running time. We also evaluate the total message overhead of the distributed learning algorithms in Table II. It shows that WSR-MADQN generates much more overhead since it exchanges information with every neighbor device. When the network size becomes larger with more D2D TXs, the message overhead of CAPA increases slightly since more devices join the training process with more training weights returned to the BS for the global model update. In general, CAPA can reduce more than 80% overhead compared to WSR-MADQN.

V. Conclusion

In this paper, we study the resource allocation problem in an underlaying D2D multicast network and formulate DCAPAP. In the proposed distributed DRL scheme, each CU and D2D TX allocates a reuse channel and the transmission power based on its local and exchange channel information to maximize the system throughput. To enhance the performance at the early stage of training, we proposed D-IBRA and D-EERA to ensure enhanced performance and accelerate the training speed. Simulation results show that the running time of CAPA is much shorter than that of the optimization algorithms, and it achieves better system throughput with less message overhead.

Acknowledgment

This work was supported by Qualcomm Technologies, Inc. under Grant SOW NAT-435533.

References

[1] F. Jameel, Z. Hamid, F. Jabeen, S. Zeadally, and M. A. Javed, "A survey of device-to-device communications: Research issues and challenges," *IEEE Commun. Surv. Tutorials*, vol. 20, no. 3, pp. 2133–2168, 2018.

[2] H. Meshgi, D. Zhao, and R. Zheng, "Joint channel and power allocation in underlay multicast device-to-device communications," in *IEEE ICC*, 2015.

[3] ——, "Optimal resource allocation in multicast device-to-device communications underlaying LTE networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 9, pp. 8357–8371, 2017.

[4] X. Wu, Y. Chen, X. Yuan, and M. E. Mkiramweni, "Joint resource allocation and power control for cellular and device-to-device multicast based on cognitive radio," *IET Communications*, vol. 8, no. 16, pp. 2805–2813, 2014.

[5] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[6] J. Tan, L. Zhang, and Y.-C. Liang, "Deep reinforcement learning for channel selection and power control in D2D networks," in *IEEE GLOBECOM*, 2019.

[7] H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep reinforcement learning based resource allocation for V2V communications," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, 2019.

[8] X. Zhang, P. Yu, L. Feng, F. Zhou, and W. Li, "A drl-based resource allocation framework for multimedia multicast in 5g cellular networks," in *IEEE BMSB*, 2019.

[9] X. Zhang, M. Peng, S. Yan, and Y. Sun, "Deep-reinforcement-learning-based mode selection and resource allocation for cellular v2x communications," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6380–6391, 2020.

[10] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.

[11] K. Doppler, M. Rinne, C. Wijting, C. B. Ribeiro, and K. Hugl, "Device-to-device communication as an underlay to LTE-advanced networks," *IEEE Commun. Mag.*, vol. 47, no. 12, pp. 42–49, 2009.

[12] M. Hmila, M. Fernández-Veiga, M. Rodríguez-Pérez, and S. Herrería-Alonso, "Energy efficient power and channel allocation in underlay device to multi device communications," *IEEE Trans. Commun.*, vol. 67, no. 8, pp. 5817–5832, 2019.

[13] F. A. Onat *et al.*, "Threshold selection for snr-based selective digital relaying in cooperative wireless networks," *IEEE Trans. Wireless Commun.*, vol. 7, no. 11, pp. 4226–4237, 2008.

[14] K. K. Nguyen *et al.*, "Non-cooperative energy efficient power allocation game in d2d communication: A multi-agent deep reinforcement learning approach," *IEEE Access*, vol. 7, pp. 100 480–100 490, 2019.

[15] H. Ren, F. Jiang, and H. Wang, "Resource allocation based on clustering algorithm for hybrid device-to-device networks," in *IEEE WCSP*, 2017.

[16] L. Xiao, Y. Li, C. Dai, H. Dai, and H. V. Poor, "Reinforcement learning-based noma power allocation in the presence of smart jamming," *IEEE Trans. Veh. Technol.*, vol. 67, no. 4, pp. 3377–3389, 2017.

[17] B. Kaufman and B. Aazhang, "Cellular networks with an overlaid device to device network," in *IEEE ACSSC*, 2008.

[18] M. Jung, K. Hwang, and S. Choi, "Joint mode selection and power allocation scheme for power-efficient device-to-device (d2d) communication," in *IEEE VTC Spring*, 2012.

[19] 3GPP, "Selection procedures for the choice of radio transmission technologies of the umts," 3GPP, Technical report (TR) 30.03U, 1998.

[20] B. Gu, X. Zhang, Z. Lin, and M. Alazab, "Deep multiagent reinforcement-learning-based resource allocation for internet of controllable things," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3066–3074, 2020.

[21] J. Tang *et al.*, "A reinforcement learning approach for d2d-assisted cache-enabled hetnets," in *IEEE GLOBECOM*, 2019.

[22] D. Van Le and C.-K. Tham, "A deep reinforcement learning based offloading scheme in ad-hoc mobile clouds," in *IEEE INFOCOM WKSHPS*, 2018.