Downlink Scheduler for Delay Guaranteed Services Using Deep Reinforcement Learning

Jiequ Ji, Xiangyu Ren, *Student Member, IEEE*, Lin Cai, *Fellow, IEEE*, and Kun Zhu, *Member, IEEE*

***Abstract*—In this article, we propose a novel scheduling scheme to guarantee per-packet delay in single-hop wireless networks for delay-critical applications. We consider several classes of packets with different delay requirements, where high-class packets yield high utility after successful transmission. Considering the correla-tionship of delays among competing packets, we apply a delay-laxity concept and introduce a new output gain function for scheduling decisions. Particularly, the selection of a packet takes into account not only its output gain but also the delay-laxity of other packets. In this context, we formulate a multi-objective optimization problem aiming to minimize the average queue length while maximizing the average output gain under the constraint of guaranteeing per-packet delay. However, due to the uncertainty in the environment (e.g., time-varying channel conditions and random packet arrivals), it is difficult and often impractical to solve this problem using traditional optimization techniques. We develop a deep reinforcement learning (DRL)-based framework to solve it. Specifically, we decompose the original optimization problem into a set of scalar optimization subproblems and model each of them as a partially observable Markov Decision Process (POMDP). We then resort to a Double Deep Q Network (DDQN)-based algorithm to learn an optimal scheduling policy for each subproblem, which can overcome the large-scale state space and reduce Q-value overestimation. Simulation results show that our proposed DDQN-based algorithm outperforms the conventional Q-learning algorithm in terms of reward and learning speed. In addition, our proposed scheduling scheme can achieve significant reductions in average delay and delay outage drop rate compared to other benchmark schemes.**

***Index Terms*—Resource allocation, packet selection, delay and network utility optimality, deep reinforcement learning.**

I. Introduction

THE increasing popularity of intelligent mobile devices is spurring the development of wireless networks and the emergence of new mobile applications with advanced features such as unmanned driving, extended reality, telemedicine and automatic navigation. Moreover, these applications are time-critical and typically have a stringent delay requirement to guarantee their quality of experience (QoE) which is not fully addressed in existing wireless networks [1]. How to effectively manage premium network resources to ensure QoE for time-critical applications is an open issue.

The resource scheduling problem of wireless networks has been extensively studied in various contexts, where throughput and delay are the main performance index [2], [3], [4]. For the case when the traffic is inside the capacity region, the throughput is equal to the arrival rate and then the throughput maximization problem reduces to a network stability problem. The network can be stabilized by a max-weight policy that schedules links per time to maximize a weighted sum of transmission rates, where the weights are queue backlogs [5]. This is typically shown by the Lyapunov drift theory [6]. For a general case when the traffic is either inside or outside of the capacity region, the max-weight policy can be combined with a flow control policy to jointly maximize the throughput and stabilize the network [7], [8], [9]. In [10] and [11], a delay-based Lyapunov function was proposed, where the delay of the head-of-queue packet is served as a weight of the max-weight decision. In [12], the Lyapunov optimization theory was used to transform the service delay minimization while ensuring long-term accuracy requirements into minimizing a drift-plus-cost.

Although there are few works that use delay-based scheduling to solve joint stability and utility optimization problems, it is not suitable for applications with stringent delay requirements. For many delay-critical applications, packets arriving late are as severe as being dropped. Existing works achieved maximum throughput utility by minimizing the average queue backlog based on the assumption that all packets in the same flow have the same delay requirement and then yield the same utility after successful transmission [13], [14], [15], [16]. However, for many emerging applications where packets of the same flow may have different delay requirements and thus have different utilities after successful transmission. Moreover, suffering excessive delay packets should be dropped early in the network to avoid wasting unnecessary network resources.

To fill the gap, we propose a delay-aware scheduling policy to guarantee the delay of any nondropped packet. Specifically, we consider a single-hop downlink network in which each packet requires transmission over only one link. Moreover, arriving packets have different delay requirements and are divided into several classes. When a packet exceeds its delay budget, it is dropped in advance before being scheduled. To describe the utility of

Manuscript received 23 January 2023; revised 14 April 2023; accepted 4 May 2023. Date of publication 16 May 2023; date of current version 6 March 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62071230, in part by Natural Sciences and Engineering Research Council of Canada, and Compute Canada. Recommended for acceptance by N. Zhang. *(Corresponding authors: Lin Cai; Kun Zhu.)*

Jiequ Ji and Kun Zhu are with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China (e-mail: jiequ@nuaa.edu.cn; zhukun@nuaa.edu.cn).

Xiangyu Ren and Lin Cai are with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC V8W 3P6, Canada (e-mail: jamesrxy@uvic.ca; cai@ece.uvic.ca).

This article has supplementary downloadable material available at https://doi.org/10.1109/TMC.2023.3276697, provided by the authors.

Digital Object Identifier 10.1109/TMC.2023.3276697

1536-1233 © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: UNIVERSITY OF VICTORIA. Downloaded on May 14,2024 at 21:25:37 UTC from IEEE Xplore. Restrictions apply.

different classes of packets, we define an output gain function where packets with high delay requirements yield high output gain. Although scheduling high delay requirement packets can obtain high output gain, it may result in packets in lower classes being dropped unnecessarily due to excessive delay. Therefore, a new delay-laxity concept is introduced where packets with the least laxity are prioritized. In this context, we formulate a multi-objective optimization problem aiming to minimize the average queue length while maximizing the average output gain under the constraints of guaranteeing per-packet delay and achieving fairness among users. Although dynamic programming methods can solve this problem, if the environment changes over time, the solution (i.e., optimal scheduling policy) has to be recalculated which may take a similar amount of time as the initial solution.

Recently, reinforcement learning (RL) has emerged as an effective solution for dealing with environmental uncertainty thanks to its capability to learn an optimal policy by interacting with the environment without any prior knowledge [17], [18], [19]. However, these RL-based algorithms perform poorly during the training process as the parameters to be trained are initialized with random values. Such a setup directly affects the exploration-exploitation dilemma as an inefficient exploration policy will take more suboptimal actions, which thus results in poor performance. To this end, it may not be feasible to directly employ standard RL-based algorithms to solve our problem. We provide a new framework for solving this multi-objective optimization problem through deep reinforcement learning. We decompose it into a set of scalar subproblems using a weighted sum approach and solve each subproblem cooperatively with other subproblems through a neighborhood-based parameter transfer strategy. We then model each subproblem as a partially observable Markov Decision Process (POMDP) and resort to a double deep Q network (DDQN)-based scheduling algorithm to learn. Moreover, our proposed DDQN-based algorithm can address the curse of dimensionality with large state and action spaces and reduce Q-value overestimation.

The main contributions of this paper can be summarized as follows:

- We propose a delay-aware scheduling policy for random arriving packets with different delay requirements, where delay-outage dropping is considered. Moreover, we define an output gain function that combines the delay laxity and priority of different packets to show the penalty for dropping packets and the reward for successful transmission.
- We formulate a multi-objective optimization problem that minimizes the average queue backlog while maximizing the average output gain under the constraints of achieving fairness among users and guaranteeing per-packet delay. A DRL framework is proposed to solve this intractable problem via decomposing it into a set of subproblems. Then we model each subproblem as POMDP and explore a DDQN-based algorithm to solve it.
- Simulation results show that our proposed DDQN-based algorithm converges to an optimal policy at a speed 16 and 20 iterations faster than Q-learning and DQN-based algorithms, respectively. Moreover, our proposed scheduling scheme outperforms other benchmark schemes without admission control and packet selection.

The rest of the paper is organized as follows. In Section II, we discuss existing works on learning-based resource management and traffic control in wireless networks. In Section III, we introduce our system model and formulate a multi-objective optimization problem. To efficiently solve it, we propose a DR-L framework in Section IV. Section V describes the algorithm design in detail. Simulation results are presented in Section VI to evaluate the performance of our scheduling policy and algorithm. Finally, we conclude our work in Section VII.

II. Related Works

There have been many studies on wireless resource management using stochastic optimization. In particular, the theory of Lyapunov drift and optimization has been used to ensure network stability and utility optimization [20]. When the arrival rate is within the capacity region, the throughput optimization problem reduces to the network stability problem and then the Lyapunov drift technique has been employed to stabilize the network by greedily minimizing the Lyapunov drift every time slot. The max-weight or backpressure algorithm was first used for link and server scheduling in [21] and [22] and has since become a promising solution for dealing with stability in various network domains [23], [24], [25]. In a more general case when the arrival rate is either inside or outside of the capacity region, the Lyapunov drift-plus-penalty technique is used to solve joint network stability and utility optimization problems [26], [27], [28]. It is shown that although the max-weight algorithm can achieve throughput optimality, it results in high network delay due to the long queue length [27].

Recently, reinforcement learning has been actively used to tackle network problems such as network traffic and resource control [29], [30], [31], [32]. In [29], a deep Q-learning algorithm based on recurrent neural networks was proposed to learn a scheduling solution for queuing delay optimization by interacting with the environment. In [30], an RL-based scheduling framework was proposed that is capable of selecting different scheduling rules based on the instantaneous scheduler state to minimize packet delays and packet drop rates for applications with strict quality of service (QoS) requirements. In [31], a new learning-based proactive resource sharing policy was proposed for next generation core communication networks. It aims to proactively allocate available forwarding resources on switches to traffic flows to maximize the efficiency of resource utilization with delay satisfaction. The work in [32] proposed a learning-based resource management algorithm that tackles the large queue backlog problem of the max-weight algorithm while achieving throughput optimality. However, most RL works may not be always suitable to address the discrete and high-dimensional action spaces in our formulated multi-objective optimization problem with delay guarantees and utility-optimality.

III. System Model

A. Network Model

We consider the downlink scheduling problem of a single-hop wireless network as shown in Fig. 1, which consists of one base station (BS) and U ground users. Let $\mathcal{U} = \{1, \ldots, U\}$ denote

Authorized licensed use limited to: UNIVERSITY OF VICTORIA. Downloaded on May 14,2024 at 21:25:37 UTC from IEEE Xplore. Restrictions apply.

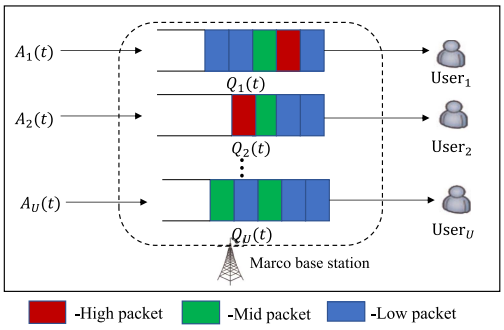

Fig. 1. Illustration of a single-hop downlink scheduling network.

the set of users that are associated with the BS. The single-hop network is assumed to operate in discrete time with normalized time slots $t \in \{0, 1, \ldots\}$. We consider three classes of packets with different delay requirements. We let the application in the end system specify the delay requirement which is carried in each packet header, and the intermediate nodes can make routing and scheduling decisions accordingly to ensure the delay requirement, following the SET architecture [33]. For simplicity, we assume that each packet has the same size and that at most one packet arrives at each downlink at each time slot. The sets of packets with high, medium, and low delay requirements are denoted as $\mathcal{H}$, $\mathcal{M}$ and $\mathcal{L}$, respectively, where $\mathcal{K} \triangleq \mathcal{H} \cup \mathcal{M} \cup \mathcal{L}$. We define $\mathbf{A}(t) \in \{0, 1\}_{U \times K}$ as the packet admission matrix, and its element $A_{u,k}(t) \in \{0, 1\}$ denotes the binary indicator, i.e., $A_{u,k}(t) = 1$ when packet k is admitted to link u at time slot t and otherwise $A_{u,k}(t) = 0$, which is subject to the constraint $\sum_K A_{u,k}(t) \leq 1$ for $u \in \mathcal{U}$. The packet arrival matrix $\mathbf{A}(t)$ is assumed to be independent and identically distributed over time slots. In addition, we assume that the arrival processes for different users and packets in each time slot are independent. Unlike most works that consider the first-in-first-out (FIFO) queue management, our proposed scheduling model selects the optimal packet in all queues for service by jointly considering link conditions and network-utility maximization. We refer this as the packet-selection process, which is discussed in detail in Section III-E. Table I summarizes the notations and definitions used frequently in this paper.

B. Association and Transmission Model

Let $\mathbf{S}(t) = \{S_1(t), \ldots, S_U(t)\}$ be the downlink scheduling vector, and its element $S_u(t) \in \{0, 1\}$ represents the binary variable, i.e., $S_u(t) = 1$ if the BS serves user u at time slot t and otherwise $S_u(t) = 0$. Suppose that the BS serves at most one user device at each time slot, which yields the following constraints

$$S_u(t) \in \{0, 1\}, \ \forall u, t,$$
$$\sum_{u \in \mathcal{U}} S_u(t) = 1, \ \forall t. \tag{1}$$

For simplicity, we assume that at most one packet is transmitted per time slot over the selected downlink. Therefore, we introduce binary variables $\boldsymbol{\eta}(t) \triangleq \{\eta_{u,k}(t), u \in \mathcal{U}, k \in \mathcal{K}\}$ to indicate the transmission states of packets, namely, $\eta_{u,k}(t) = 1$ means that packet k is transmitted to user u at time slot t and otherwise $\eta_{u,k}(t) = 0$. Then the transmission constraints are given by

$$\eta_{u,k}(t) \in \{0, 1\}, \ \forall u, k, t,$$
$$\sum_{k \in \mathcal{K}} \eta_{u,k}(t) \leq 1, \ \forall u, t. \tag{2}$$

Note that the probability of successful packet transmission for each time slot is affected by the channel conditions of the corresponding downlink. Let $\mathbf{C}(t) = \{C_1(t), \ldots, C_U(t)\}$ be the channel vector from the BS to each user, which is assumed to be known by the BS at the beginning of each time slot. Given vectors $\eta_{u,k}(t)$ and $C_u(t)$, the probability of successful packet transmission over downlink u is expressed as

$$Pr(\text{downlink } u \text{ success} | \boldsymbol{\eta}(t), \mathbf{C}(t)) = \Phi_u(\boldsymbol{\eta}(t), \mathbf{C}(t)), \tag{3}$$

where the probability function $\Phi_u(\boldsymbol{\eta}(t), \mathbf{C}(t))$ for $u \in \mathcal{U}$ takes only real numbers between 0 and 1. Moreover, we introduce an auxiliary variable $1_u(t)$ to indicate the successful transmission of downlink u

$$1_u(t) = \begin{cases} 1, & \text{with probability } \Phi_u(\boldsymbol{\eta}(t), \mathbf{C}(t)), \\ 0, & \text{with probability } 1 - \Phi_u(\boldsymbol{\eta}(t), \mathbf{C}(t)). \end{cases} \tag{4}$$

It should be noted that the successful or failed transmission over each downlink can be fully described by a joint success distribution function $\Phi(\boldsymbol{\eta}, \mathbf{C})$ of all 2^U possible successes and failures [34]. However, since the network capacity region is independent among each downlink [34], the maximum utility point is also independent of the interlink success correlations. Thus, it suffices to use only the marginal distribution function

TABLE I
NOTATIONS AND DEFINITIONS

Symbol	Definitions
$\mathcal{K}$,	Sets of packets
$\mathcal{U}$	Sets of users associated the BS
$\mathbf{A}(t)$	Packet admission matrix
$S_u(t)$	Control variable of serving user u at slot t
$\boldsymbol{\eta}(t)$	Transmission status of packets
$\eta_{u,k}(t)$	Variable of transmitting packet k to user u at slot t
$\mathbf{C}(t)$	Channel condition vector
$C_u(t)$	Channel condition of downlink u at slot t
$1_u(t)$	Value of successful transmission of downlink u at slot t
$\mathbf{Q}(t)$	Set of packets backlogged in each queue
$Q_u(t)$	Backlogged packets of queue u at slot t
$x_{u,k}(t)$	Sojourn time of packet k in queue u
Th_H	Maximum tolerable delay of packet $k \in \mathcal{H}$
$D_{u,k}(t)$	Queuing state of packet k in queue u
$L_{u,k}(t)$	Delay budget laxity of packet k in queue u
ω_H	Gain weights of packets with high delay requirements
ρ	Discount factor
$G(t)$	Achievable gain of the system at slot t
$\bar{A}_u$	Average packets admitted to queue u over all slots
λ_i^n	Weight of objective i in the n-th subproblem

Authorized licensed use limited to: UNIVERSITY OF VICTORIA. Downloaded on May 14,2024 at 21:25:37 UTC from IEEE Xplore. Restrictions apply.

$\Phi_u(\boldsymbol{\eta}(t), \mathbf{C}(t))$ for each $u \in \mathcal{U}$. Then the discrete transmission variable $\eta_{u,k}$ in (2) can be rewritten as

$$\hat{\eta}_{u,k}(t) = \eta_{u,k}(t) 1_u(t). \tag{5}$$

We define $\mathbf{Q}(t) = \{Q_1(t), \ldots, Q_U(t)\}$ as the set of packets currently backlogged in network queue for each user. Note that each packet is marked with its integer arrival time slot, which is useful for determining its queuing delay in the system. Let $x_{uk}(t)$ denote the sojourn time of packet k in queue u at time slot t. In addition, the maximum tolerable queuing delay for the three classes of packets is denoted as Th_H, Th_M, and Th_L, respectively. It is clear that packet k will be dropped when its sojourn time in queue u is larger than its maximum tolerable queuing delay. Hence, we introduce discrete binary variables $\{D_{u,k}(t), u \in \mathcal{U}, k \in \mathcal{K}\}$ to indicate the queuing state of each packet, namely, $D_{u,k}(t) = 1$ means that packet k in queue u is dropped at time slot t and otherwise $D_{u,k}(t) = 0$. Then the following queuing constraints need to be satisfied

$$D_{u,k}(t) = \begin{cases} 1, & \text{if } x_{uk}(t) \geq \text{Th}_k, \\ 0, & \text{if } x_{uk}(t) < \text{Th}_k, \end{cases} \quad \forall k \in \mathcal{H} \cup \mathcal{M} \cup \mathcal{L}. \tag{6}$$

To sum up, the packet queuing dynamic of each queue can be expressed as

$$Q_u(t+1) = \max\left\{ Q_u(t) - S_u(t)\hat{\eta}_{u,j} - \sum_{i \neq j, i \in Q_u(t)} D_{u,i}(t), 0 \right\} + A_{u,k}(t), \quad \forall j \in Q_u(t), k \in \mathcal{H} \cup \mathcal{M} \cup \mathcal{L}. \tag{7}$$

C. Scheduling and Gain Model

Given the sojourn time of packet k in queue $u \in \mathcal{U}$, we define the delay laxity [35] as follows:

$$L_{uk}(t) = \text{Th}_k - x_{uk}(t), \forall k \in Q_u(t), \ \forall t, k \in \mathcal{H} \cup \mathcal{M} \cup \mathcal{L}, \tag{8}$$

which measures the remaining queue delay budget. To reduce the packet drop rate per queue, we expect to select the packet with the minimum delay-laxity for transmission. Since the BS only transmits at most one packet per time slot, we can obtain different output gains after successful transmission of different classes of packets. We define the gain weights of packets with high, medium and low delay requirements as ω_H, ω_M and ω_L, respectively, where $\omega_H > \omega_M > \omega_L$. In addition, we consider the discounted potential output gain for all packets backlogged in each queue, which is given as

$$G_u(t) = \rho \sum_{k \in X_u(t)} r_{u,k}(t), \ \forall u \in \mathcal{U}, k \in \mathcal{H} \cup \mathcal{M} \cup \mathcal{L}, \tag{9}$$

where $r_{u,k}(t) = \frac{\omega_k}{L_{uk}(t)}$; $\rho \in [0,1]$ denotes the discount factor; and $X_u(t)$ denotes the set of remaining packets in queue u per time slot after the packet-drop decision is executed. If packet k in queue u is transmitted by the BS in time slot t, the gain received by queue u can be expressed as

$$G_{u,k}(t) = r_{u,k}(t) + \rho \left(\sum_{i \neq k, i \in X_u(t)} r_{u,i}(t) + \sum_{u' \neq u, u' \in U} \sum_{j \in X_{u'}(t)} r_{u',j}(t) \right). \tag{10}$$

Hence, the average achievable gain of all queues at each time slot can be expressed as

$$G(t) = \frac{1}{U} \sum_{u \in \mathcal{U}} \Bigg[\sum_{k \in X_u(t)} \frac{\omega_k}{L_{uk}(t)} (S_u(t)\hat{\eta}_{u,k} + S_u(t)\rho(1 - \hat{\eta}_{u,k}) + \sum_{u=1}^{U} (1 - S_u(t))\rho) \Bigg]. \tag{11}$$

D. Problem Formulation

We define a scheduling problem that considers per-packet delay requirement, network utility, average queuing delay, and admission fairness among users. The gain model introduced in the previous section is defined as the network utility function. Let $F(\bar{\boldsymbol{A}})$ be a concave and non-decreasing function of the U-dimensional vector $\bar{\boldsymbol{A}} = \{\bar{A}_1, \ldots, \bar{A}_U\}$, where $\bar{A}_u$ is used to denote the time-average admission of queue $u \in \mathcal{U}$ (in unit of packets/slot). The following function is useful for addressing network fairness when attributing $\bar{A}_u$ to be non-negative [36]:

$$F(\bar{\boldsymbol{A}}) = \sum_{u=1}^{U} F_u(\bar{A}_u) = \sum_{u=1}^{U} \log(1 + \nu_u \bar{A}_u), \tag{12}$$

where ν_u is a positive constant. This example is useful because each component function $\log(1 + \nu_u \bar{A}_u)$ has a diminishing return property as $\bar{A}_u$ increases and becomes 0 when $\bar{A}_u = 0$. The average queuing delay is proportional to the average queue length. Therefore, we formulate a multi-objective optimization problem aiming to minimize the average queue length while maximizing the average output gain under the constraints of achieving fairness among users and guaranteeing per-packet delay. Mathematically, this problem is written as

$$\text{P1:} \max_{\mathbf{A}, \mathbf{S}, \boldsymbol{\eta}, \mathbf{D}} \left\{ -\frac{1}{T} \sum_{t=0}^{T} \sum_{u \in \mathcal{U}} \mathbb{E}[Q_u(t+1)], \ \frac{1}{T} \sum_{t=0}^{T} G(t) \right\} \tag{13a}$$

$$\text{s.t.} \quad \sum_{u=1}^{U} F_u(\bar{A}_u) \geq \sum_{u=1}^{U} F(\bar{\gamma}_u), \tag{13b}$$

$$\bar{A}_u \leq \frac{1}{T} \sum_{t=0}^{T} \mathbb{E}[S_u(t)\hat{\eta}_{u,k}(t)], \forall u \in \mathcal{U} \tag{13c}$$

$$S_u(t) \in \{0,1\}, \forall u \in \mathcal{U}, \ \sum_{u \in \mathcal{U}} S_u(t) = 1, \tag{13d}$$

$$\eta_{u,k}(t) \in \{0,1\}, \ \sum_{k \in X_u(t)} \eta_{u,k}(t) \leq 1, \forall u, t, \tag{13e}$$

$$D_{u,k}(t) = \begin{cases} 1, & \text{if } x_{uk}(t) \geq \text{Th}_k, \\ 0, & \text{if } x_{uk}(t) < \text{Th}_k, \end{cases} \quad \forall k \in X_u(t). \tag{13f}$$

Authorized licensed use limited to: UNIVERSITY OF VICTORIA. Downloaded on May 14,2024 at 21:25:37 UTC from IEEE Xplore. Restrictions apply.

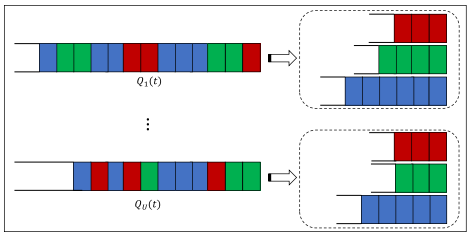

Fig. 2. Implementation of priority-based single-queue packet selection.

where $\bar{A}_u = \frac{1}{T}\sum_{t=0}^{T}\mathbb{E}[A_u(t)]$ denotes the average number of packets admitted to queue u over all time slots and $\bar{\gamma}_u \in \{0,1\}$ denotes the auxiliary variable that is important for achieving fairness among users with random arrival rate [21]. Constraint (13c) guarantees the mean rate stability. In addition, constraint (13d) indicates that the BS only serves at most one user at each time slot while constraint (13e) is used to limit the number of packets transmitted by the BS in each time slot. Constraint (13f) shows the value of $D_{u,k}(t)$ should be either 0 or 1.

E. Reduced-Complexity Scheduling Model

The implementation of the proposed scheduling model may be inefficient in practice, especially for queue maintenance and packet selection. To address this issue, the single-queue model can be replaced by a set of FIFO priority queues as shown in Fig. 2. The number of priority queues is determined by the delay budget of packets with different delay requirements such that packets in the same priority queue follows the sequence of delay laxity L_{uk} from small to large. Instead, the output gain $r_{u,k}$ of packets in each queue follows a descending order from large to small since the weight w_k of the same queue remains the same. Thus, only the first packet of the same queue needs to be considered for scheduling in each packet-selection process. In addition, only the head of queues should be checked for dropping as well. In our example, three priority queues are designed where $w_{u,H} > w_{u,M} > w_{u,L}$. Furthermore, packets enqueued in the same queue have the same delay budget, i.e., $Th^k_{u,1} = Th^k_{u,2} = \ldots = Th^k_{u,Q_{u,k}}$ for $k \in \{H, M, L\}$, where $Q_{u,k}$ is the queue length of priority queue k in downlink u. The potential sojourn time of packet i in priority queue k of downlink u is denoted by $x^k_{u,i}$ for $i = 1, 2, \ldots, Q_{u,k}$, which follows $x^k_{u,1} > x^k_{u,2} > \cdots > x^k_{u,Q_{u,k}}$, while the output gain of all packets in this queue has $r^k_{u,1} > r^k_{u,2} > \cdots > r^k_{u,Q_{u,k}}$. Note that the number of queues with the same weight can be further extended to meet the time granularity requirements at the cost of memory space. However, the problem formulation remains the same as shown in P1.

IV. DRL-Based Throughput and Delay Optimality

The presented problem P1 is difficult to solve mainly pertaining to the following reasons. First, the downlink selection and packet scheduling variables are binary and thus (13d) and (13e) involve integer constraints. Second, in problem P1, the first objective minimizing the average queue length and the second objective maximizing the output gain involve conflicts. Although there are several classical optimization algorithms (e.g., dynamic programming and multi-objective genetic local search) that can be used to solve this problem, they have high computational complexity, especially in large-scale scenarios. In addition, due to the curse of uncertainty (i.e., random arrival of packets), it is difficult and impractical to solve using dynamic programming-based algorithms in practice. To this end, we propose a deep reinforcement learning (DRL)-based method to solve this multi-objective optimization problem, which is used to learn policies by interacting with the environment without any prior knowledge to maximize the cumulative rewards from experiences. Specifically, we decompose problem P1 into a set of scalar optimization subproblems by using the weighted sum approach [37]. We then model each subproblem as a partially observable Markov decision process (POMDP) and explore an efficient double deep Q network (DDQN)-based algorithm to solve it. In particular, we collaboratively optimize the network parameters of all subproblems by applying the neighborhood parameter transfer method [38] and the proposed DDQN-based training algorithm.

A. DRL for Multi-Objective Optimization

In this subsection, we decompose the multi-objective optimization problem into a set of scalar subproblems. There are many scalarizing methods that can be used for decomposition, such as the weighted sum method and the penalty-based boundary intersection method [39]. For simplicity, we use the weighted sum method in which a set of uniformly distributed vectors $\lambda^1, \ldots, \lambda^N$ is given, e.g., $(1,0), (0.99, 0.01), \ldots, (0,1)$ for a bi-objective problem. Note that $\lambda^n = \{\lambda_{n1}, \ldots, \lambda_{nL}\}$, where L is the number of objectives. Thus the original multi-objective problem is transformed into N scalar subproblems and each subproblem is solved collaboratively with other subproblems through the neighborhood-based parameter transfer strategy. Note that solving each scalar subproblem usually results in a Pareto optimal solution and the desired Pareto Front can be obtained when all the scalar optimization subproblems are solved in sequence. In particular, the objective function of the nth subproblem is expressed as

$$\max \lambda^n \times f = \sum_{l=1}^{L} \lambda_{nl} \times f_l. \tag{14}$$

To solve these subproblems by DRL, we model each of them as a neural network. Then the N scalar subproblems are solved collaboratively according to the neighbourhood-based parameter transfer strategy and the proposed DDQN-based algorithm. From (14), two neighbouring subproblems may have very close optimal solutions since their weight vectors are adjacent [40]. In this case, each subproblem can be solved faster by leveraging the knowledge of its neighbouring subproblems. In specific, the parameters of the neural network model of the $(n-1)$th subproblem is described as $[\omega_{\lambda^{n-1}}, b_{\lambda^{n-1}}]$, where $[\omega^*, b^*]$ denotes the optimal parameters of the neural network model and $[\omega, b]$ denotes the parameters that have not been optimized. Once the $(n-1)$th subproblem has been solved, i.e., its network parameters obtained by the proposed algorithm are close to

Authorized licensed use limited to: UNIVERSITY OF VICTORIA. Downloaded on May 14,2024 at 21:25:37 UTC from IEEE Xplore. Restrictions apply.

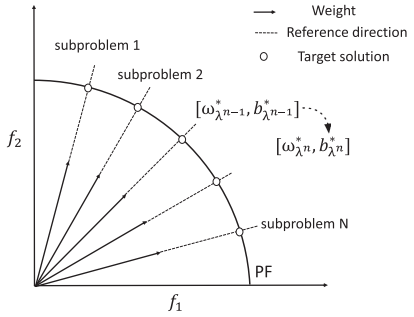

Fig. 3. Illustration of the decomposition and parameter-transfer strategy.

optimal. The best network parameters $[\omega_{\lambda^{n-1}}, b_{\lambda^{n-1}}]$ obtained for the $(n-1)$th subproblem are set as the starting point of the network training for the nth subproblem. Accordingly, the network parameters are transferred sequentially from the previous subproblem to the next subproblem as shown in Fig. 3, which can save a considerable amount of time compared with training all subproblems.

The DRL framework for solving our proposed optimization problem P1 is summarized in Algorithm 1, which consists of the multi-objective decomposition and the neighborhood-based parameter transfer strategy. In this algorithm, each subproblem is modeled as a POMDP and solved by the proposed DDQN algorithm and all the subproblems can be solved sequentially by transferring the network weights. Hence, we can obtain an approximate Pareto solution by solving each subproblem using Algorithm 1. Finally, the desired Pareto Front is obtained when all the subproblems are solved sequentially.

The proposed DRL framework has two main advantages: i) The first is its simplicity and modularity for use, i.e., any of the recently proposed novel DRL-based solvers (including the improved DQN algorithm) can be integrated into the proposed DRL framework to solve multi-objective optimization problems; ii) Once the trained model is available, we can obtain the desired Pareto Front directly by a simple forward propagation of the model.

The DRL framework acts as an outer-loop. The next issue is how to model and solve those decomposed scalar subproblems. In the next subsection, we first formulate each subproblem as a POMDP and then resort to a value iterative-based reinforcement learning algorithm, named Q-learning, to solve it.

B. Preliminaries of POMDP

In our learning framework, we consider the controller on the BS as an agent to learn how to perform data admission control, downlink service scheduling, and packet selection in the system. In addition, since the admission and delivery of each packet may be affected by the current network environment and the actions of the BS, the learning task of the BS-agent can satisfy the Markov property. In this case, we model each scalar optimization subproblem as a POMDP, which is described by the following tuple:

$$\Omega = \{\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}, \tag{15}$$

Algorithm 1: DRL Framework for Multi-Objective Optimization Problems.

Input: The model of the subproblem $M = [\omega, b]$ and weight vectors $\lambda^1, \ldots, \lambda^N$
Process:
Random initialize $[\omega_{\lambda^1}, b_{\lambda^1}]$;
for $n = 1, \ldots, N$ **do**
  if $n == 1$ **then**
    Solve the first subproblem using DDQN algorithm and obtain the network parameters $[\omega^*_{\lambda^1}, b^*_{\lambda^1}]$
  else
    $[\omega^*_{\lambda^n}, b^*_{\lambda^n}] \leftarrow [\omega^*_{\lambda^{n-1}}, b^*_{\lambda^{n-1}}]$
    Solve the nth subproblem using DDQN algorithm and obtain the network parameters $[\omega^*_{\lambda^n}, b^*_{\lambda^n}]$
end for
Output: Pareto Front can be directly computed by $[\omega^*, b^*]$

where $\mathcal{S}$ denotes the set of states describing the environment; $\mathcal{O}$ denotes the observation space; $\mathcal{A}$ denotes the action space; $\mathcal{R}$ denotes the reward function that maps the network state and the joint actions of the agent to rewards; $\mathcal{P}$ denotes the state transition function with $P_{s_t, s_{t+1}}(a_t)$ being the probability that the current state s_t transfers to the next state s_{t+1} when action a_t is performed; and $\gamma \in [0, 1]$ denotes the discount factor. At time slot t, the agent observes a state $s_t \in \mathcal{S}$ and chooses an action $a_t \in \mathcal{A}$ according to a certain policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$, which receives a reward $r_t = r(s_t, a_t)$ and produces a new state s_{t+1} with the transition probability $P(s_{t+1}|s_t, a_t)$.

The detailed definitions of POMDP for the nth subproblem with weight $\lambda_n = (\lambda_{n1}, \lambda_{n2})$ are given below.

1) State and Observation Space: At each time slot t, the BS-agent observes the state information of the environment so as to determine the corresponding policy. The state space at time slot t is denoted by s_t and it contains four elements: the successful transmission probability $1_u(t)$, queue length $Q_u(t)$, packet sojourn time $x_{u,k}(t)$ and delay budget laxity $L_{uk}(t)$, which can be expressed as

$$s_t = \{1_u(t), Q_u(t), x_{u,k}(t), L_{uk}(t)\}, \forall u \in \mathcal{U}, k \in Q_u(t). \tag{16}$$

Thus the state space is expressed as $\mathcal{S} = \{s_t | t = 1, \ldots, T\}$. For the state space, the probability of successful transmission for each user depends on the channel conditions that can only be observed locally and not known by other association pairs. Therefore, according to (4), the observation space at time slot t can be summarized as

$$o_t = \{C_u(t), Q_u(t), x_{u,k}(t), L_{uk}(t)\}, \forall u \in \mathcal{U}, k \in Q_u(t). \tag{17}$$

Accordingly, the observation space of the BS-agent is given by $\mathcal{O} = \{o_t | t = 1, \ldots, T\}$.

2) Action Space: At each slot, the BS-agent determines whether and what types of packets are admitted in each queue. Then the BS chooses which user is served and which packet is transmitted in the queue corresponding to the associated user. In addition, if the sojourn time of the packet exceeds its maximum

Authorized licensed use limited to: UNIVERSITY OF VICTORIA. Downloaded on May 14,2024 at 21:25:37 UTC from IEEE Xplore. Restrictions apply.

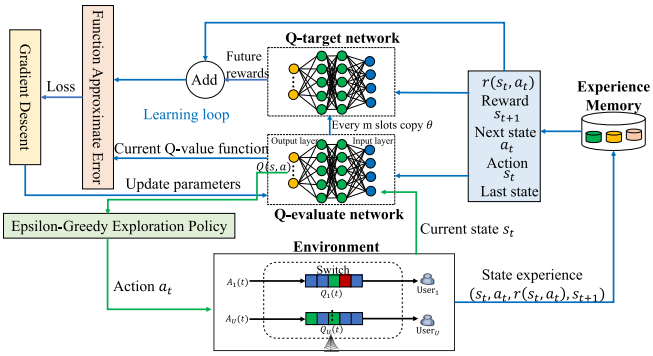

Fig. 4. Illustration of DDQN framework for resource schedule in networks.

tolerable queuing delay, it will be dropped. Thus, the action of the BS-agent at time slot t is expressed as

$$a_t = \{A_{u,k}(t), S_u(t), \eta_{u,k}(t), D_{u,k}(t)\}, \forall u \in \mathcal{U}, k \in \mathcal{K}. \quad (18)$$

Thus the action space is expressed as $\mathcal{A} = \{a_t | t = 1, \ldots, T\}$.

3) Reward Design: In a DRL-based framework, the learning process is driven by the generated reward and the agent makes its policy decision by interacting with environment in terms of maximizing the designed reward. Thus, the reward design is crucial for problems with multiple objectives, while the performance of the system largely depends on the reward. It is obvious that network reward is generally related to the objective function. According to the presented problem P1, our objective has twofold: minimizing the average queue length and maximizing the average output gain. As a result, the immediate network reward is defined as

$$r(s_t, a_t) = \lambda_{n2} \frac{1}{T} \sum_{t=0}^{T} G(t) - \lambda_{n1} \frac{1}{T} \sum_{t=0}^{T} \sum_{u \in \mathcal{U}} \mathbb{E}[Q_u(t+1)]. \quad (19)$$

As shown in Fig. 4, the learning system establishes the relationship between the optimal criterion and the optimal policy by introducing a value function consisting of a state-value function and an action-value function. Specifically, the state-value function $V_\pi(s)$ is defined by the discounted cumulative reward $R_t = \sum_{t=0}^{T} \gamma r_t$ of the agent in state s, which is used to measure the quality of an available state-action pair. Given a policy π, we define the state-value function as

$$V_\pi(s) = \mathbb{E}_\pi[R_t | s_t = s] = \mathbb{E}_\pi \left[\left(\sum_{t=0}^{T} \gamma r_t \right) | s_t = s \right], \quad (20)$$

where $\mathbb{E}_\pi[\cdot]$ denotes the expectation under the policy π. Based on the Bellman equation [41], $V_\pi(s)$ is converted as follows

$$V_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s)(r(s,a) + \gamma \sum_{s'} P_{s,s'}(a) V_\pi(s')), \quad (21)$$

where $\pi(a|s)$ is the action distribution under state s and s' is the state at the next time slot. Similarly, the Q-value function is defined as the expected sum of discounted rewards obtained by performing action a at state s and following policy π in the next state, which is given by

$$Q_\pi(s,a) = r(s,a) + \gamma \sum_{s'} P_{s,s'}(a) \sum_{a' \in \mathcal{A}} \pi(a'|s') Q_\pi(s', a'). \quad (22)$$

There is an optimal state-value function when the optimal policy is used among all of those possible state-value functions. We use $V^*(s) = \max_\pi V_\pi(s)$ to denote the optimal state-value function. Moreover, the optimal state-value function $V^*(s)$ in state s can be estimated by the Q-value function $Q(s,a)$. Thus we have

$$V^*(s) = \max_a Q(s,a). \quad (23)$$

The agent continuously improves its policy with the accumulation of experience to search an optimal policy that yields a maximum value of $Q(s,a)$ for all available states and actions. Overall, the optimal Q-value function is easily obtained when the optimal policy $\pi^*(s) = \max_\pi Q_\pi(s,a)$ that maps the set of states and actions is satisfied. The Bellman optimality equation is used to express the optimal Q-value function, which can be mathematically written as

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s'} P_{s,s'}(a) \max_{a'} Q^*(s', a'), \quad (24)$$

In addition, the optimal Q-value function can be estimated by iteratively updating the following expression at each time slot based on the recursive method.

$$Q_{\text{new}}(s,a) = (1-\alpha) Q(s,a) + \alpha (r(s,a) + \gamma \max_{a'} Q(s', a')), \quad (25)$$

where $\alpha \in (0,1)$ denotes the learning rate.

C. Optimize With Lyapunov Function

We consider two cases where the traffic is inside or outside of the capacity region, i.e., underload or overload. For the first case, the admission rate is equal to the arrival rate and then the network utility is maximized with an optimal scheduling policy that transmits as many packets as possible under the network stability constraint [34]. Moreover, all queues in the network can achieve stability simultaneously by maximizing the weight difference between the output gain and the average queue length, which is demonstrated in Proposition 1. Inspired by these facts, if the optimal objective (13a) is achieved, both constraints (13b) and (13c) are satisfied simultaneously. Thus, the original problem P1 is converted into the following form by introducing a negative Lyapunov drift term $-\nu_1(Q_u(t+1)^2 - Q_u(t)^2)$ into the objective function

$$\text{P2:} \max_{\mathbf{A},\mathbf{S},\boldsymbol{\eta},\mathbf{D}} \left\{ \lambda_{n2} \frac{1}{T} \sum_{t=0}^{T} G(t) + \lambda_{n1} \frac{1}{T} \sum_{t=0}^{T} \sum_{u \in \mathcal{U}} \mathbb{E}[-Q_u(t+1) - \nu_1 (Q_u^2(t+1) - Q_u^2(t))] \right\} \quad (26a)$$

$$\text{s.t.} \quad \nu_1 > 0, \quad (26b)$$

$$S_u(t) \in \{0,1\}, \forall u \in \mathcal{U}, \sum_{u \in \mathcal{U}} S_u(t) = 1, \quad (26c)$$

Authorized licensed use limited to: UNIVERSITY OF VICTORIA. Downloaded on May 14,2024 at 21:25:37 UTC from IEEE Xplore. Restrictions apply.

$$\eta_{u,k}(t) \in \{0,1\}, \sum_{k \in X_u(t)} \eta_{u,k}(t) \leq 1, \forall u, t, \tag{26d}$$

$$D_{u,k}(t) = \begin{cases} 1, & \text{if } x_{uk}(t) \geq \text{Th}_k, \\ 0, & \text{if } x_{uk}(t) < \text{Th}_k, \end{cases} \quad \forall k \in X_u(t). \tag{26e}$$

Although the objective function is changed, we can still obtain an optimal solution that maximizes the average output gain and minimizes the average queue length while making all queues in the network stable. It is clear that the Lyapunov drift term is the increment of queue backlog from time slot t to the next time slot $t+1$. As expected, each queue can be pushed to a low congestion state by minimizing it, which guarantees network stability [21]. Moreover, all queues will not diverge if they are stable, which indicates that the drift term converges to 0 as t tends to infinity and the objective function leaves only the average output gain and the average queue length.

Proposition 1: There exists an optimal policy for problem P2 that makes all queues in the network stable under the inner capacity region while maximizing the average output gain and minimizing the average queue length.

Proof: See Appendix A, available online.

Proposition 1 indicates that the optimal solution of problem P1 is the same as that of problem P2.

Next, we consider a more general case where the traffic is outside of the capacity region. Since all users in our model share limited resources, it is essential to solve the network utility maximization problem in order to fairly allocate network resources while stabilizing the network. For this overload case, the admission rate will no longer be equal to the arrival rate. According to the Lagrangian method [42], the original problem P1 is transformed as follows

$$\text{P3:} \max_{\mathbf{A},\mathbf{S},\boldsymbol{\eta},\mathbf{D}} \left\{ \lambda_{n2} \frac{1}{T} \sum_{t=0}^{T} G(t) - \lambda_{n1} \frac{1}{T} \sum_{t=0}^{T} \sum_{u \in \mathcal{U}} \mathbb{E}[Q_u(t+1)] + \nu_2 \sum_{u=1}^{U} F_u(\bar{A}_u) \right\} \tag{27a}$$

$$\text{s. t} \quad \nu_2 > 0, \tag{27b}$$

$$\bar{A}_u \leq \frac{1}{T} \sum_{t=0}^{T} \mathbb{E}[S_u(t) \hat{\eta}_{u,k}(t)], \forall u \in \mathcal{U}, \tag{27c}$$

$$S_u(t) \in \{0,1\}, \forall u \in \mathcal{U}, \sum_{u \in \mathcal{U}} S_u(t) = 1, \tag{27d}$$

$$\eta_{u,k}(t) \in \{0,1\}, \sum_{k \in X_u(t)} \eta_{u,k}(t) \leq 1, \forall u, t, \tag{27e}$$

$$D_{u,k}(t) = \begin{cases} 1, & \text{if } x_{uk}(t) \geq \text{Th}_k, \\ 0, & \text{if } x_{uk}(t) < \text{Th}_k, \end{cases} \quad \forall k \in X_u(t). \tag{27f}$$

According to the two transformed problems P2 and P3, the reward function (19) is rewritten as

$$r(s_t, a_t) = \lambda_{n2} \frac{1}{T} \sum_{t=0}^{T} G(t) + \lambda_{n1} \frac{1}{T} \sum_{t=0}^{T} \sum_{u \in \mathcal{U}} \mathbb{E}[-Q_u(t+1) - \nu_1 (Q_u^2(t+1) - Q_u^2(t))] \tag{28}$$

for the arrival rate lied inside the capacity region and

$$r(s_t, a_t) = \lambda_{n2} \frac{1}{T} \sum_{t=0}^{T} G(t) - \lambda_{n1} \frac{1}{T} \sum_{t=0}^{T} \sum_{u \in \mathcal{U}} \mathbb{E}[Q_u(t+1)] + \nu_2 \sum_{u=1}^{U} F_u(\bar{A}_u) \tag{29}$$

for the arrival rate lied outside the capacity region.

V. Algorithm Design

Although Q-learning has emerged as a prospective learning algorithm based on value, it relies heavily on a set of records in the sample when searching the optimal policy. Therefore, it is susceptible to the training variance and even the convergence of the Q-value function. The work [43] showed that Q-learning algorithms are very flexible for low-dimensional reinforcement learning problems. However, in our considered time-varying model, the complex and large state-action space can easily trap the table of Q-value functions in the curse of dimensionality, which will waste a lot of time and resources and even exceed the memory size for maintaining this table.

In order to address the above problems, many scholars have focused on the deep Q-network (DQN) algorithm based on the combination of neural networks and Q-learning, which mainly takes the state and action as the input of the neural network to approximate the Q-value function instead of maintaining the Q-table [44]. To be specific, the Q neural network receives an input state s and outputs the estimated Q-value functions for all optional actions, i.e., $Q(s,a;\boldsymbol{\theta}) \approx Q(s,a)$, $a \in A$, where $\boldsymbol{\theta}$ represents the vector of the weights of a Q neural network. In this algorithm, the agent aims to continuously learn an optimal policy for optimizing the Q-value function by minimizing the loss function $\text{Loss}(\boldsymbol{\theta})$, which is expressed as

$$\text{Loss}(\boldsymbol{\theta}) = \mathbb{E}[(y(t) - Q(s,a;\boldsymbol{\theta}))^2], \tag{30}$$

where $y(t)$ denotes the target Q-value and can be mathematically written as

$$y(t) = r(s,a) + \gamma \max_{a'} Q(s', a'; \boldsymbol{\theta}). \tag{31}$$

According to (30), the Q neural network is trained by iteratively updating its weights $\boldsymbol{\theta}$ to optimize the approximation of the Q-value function. The parameter update equation of the Q neural network can be written as

$$\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \mathbb{E}[(y(t) - Q(s,a;\boldsymbol{\theta})) \nabla Q(s,a;\boldsymbol{\theta})]. \tag{32}$$

It is worth noting that the DQN-based framework has an experience replay pool and is usually used to store the experiences collected by the agent denoted by (s_t, a_t, r_t, s_{t+1}), from which a group of experiences can be randomly selected and used to train the neural network. Moreover, this framework employs a dual network approach to further improve the learning stability and algorithm efficiency, which means that there is also a target network with weights $\boldsymbol{\theta}'$ in the training process. Accordingly, the target Q-value $y(t)$ in (31) can be described in a new way as

Authorized licensed use limited to: UNIVERSITY OF VICTORIA. Downloaded on May 14,2024 at 21:25:37 UTC from IEEE Xplore. Restrictions apply.

follows

$$y(t) = r(s,a) + \gamma \max_{a'} Q(s', a'; \boldsymbol{\theta}'). \quad (33)$$

It is easy to see from the Q-value function that DQN uses a single max mathematical estimator to select and evaluate an action. However, this manner tends to make agents sometimes confused about the selection and evaluation of actions, which leads to estimated Q-values that may be higher than the true values. Note that these overestimation problems are caused by the positive bias of the max operator used in DQN to update its Q-value function.

A. DDQN-Based Solution

To address this overestimation problem, we explore a double deep Q network (DDQN)-based method, which decouples the action selection and Q-value calculation into two separated max function estimators to avoid overestimation [45]. Evidently, the maximum Q-value function is computed by the state of the next time slot and all possible actions in the current neural network, DDQN first finds the optimal action under that Q-value function and then uses this action to predict the target Q-value from the target network. Due to the mutual constraints between the dual estimators, they can eliminate the maximum deviation.

Similar to DQN, DDQN also has two neural networks namely online network $Q(s,a;\boldsymbol{\theta})$ and target network $(Q(s,a;\boldsymbol{\theta}')$. The weight vector of the target network $\boldsymbol{\theta}'$ is copied from the online network in several previous iterations. Inspired by this, the target Q-value for DDQN is replaced as follows

$$y(t) = r(s,a) + \gamma Q(s', \arg\max_{a'} Q(s', a'; \boldsymbol{\theta}); \boldsymbol{\theta}'). \quad (34)$$

Note that the selection and evaluation of an action follows an **arg max** operator, which is defined by a set of weights $\boldsymbol{\theta}$. This means that we estimate the value of the greedy policy based on the current Q-value defined by $\boldsymbol{\theta}$. Then another set of weights $\boldsymbol{\theta}'$ is used to evaluate the value of this policy. The agent selects an action based on the ε-greedy policy to balance its exploitation and exploration, where ε is a diminishing value for exploration. Particularly, the agent selects an action that can maximize the Q-value with probability $(1-\varepsilon)$, while randomly choosing other actions with probability ε, which can be described as

$$a(t) = \begin{cases} \text{random action}, & \text{probability } \varepsilon, \\ \arg\max_{a'} Q(s', a'; \boldsymbol{\theta}), & \text{probability } 1-\varepsilon. \end{cases} \quad (35)$$

The loss function defined in (30) is written as

$$\text{Loss}(\boldsymbol{\theta}) = \mathbb{E}[(r(s,a) + \gamma Q(s', \arg\max_{a'} Q(s', a'; \boldsymbol{\theta}); \boldsymbol{\theta}') - Q(s,a;\boldsymbol{\theta}))^2]. \quad (36)$$

In order to reduce the loss value, the gradient descent method is used to update the weights of the target network. The update of $\boldsymbol{\theta}$ can be expressed as

$$\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \mathbb{E}[(r(s,a) + \gamma Q(s', \arg\max_{a'} Q(s', a'; \boldsymbol{\theta}); \boldsymbol{\theta}') - Q(s,a;\boldsymbol{\theta}))\nabla Q(s,a;\boldsymbol{\theta})]. \quad (37)$$

Algorithm 2: DDQN-Based Training Algorithm.

Input: Action set $\mathcal{A}$; number of episodes E; learning rate α; network update period F; discount factor γ; minibatch size B; random selection probability ε.
Output: Optimal policy π^* and maximum reward R
Initialize: the relay memory M with the capacity of C
Initialize: the online Q-network $Q(s,a;\boldsymbol{\theta})$ with weight $\boldsymbol{\theta}$
Initialize: the target Q-network $Q(s,a;\boldsymbol{\theta}')$ with $\boldsymbol{\theta}' = \boldsymbol{\theta}$
for each episode **do**
  Initialize the environment and obtain an initial state s_1
  for each iteration of an episode **do**
    Observe o_t and take an action a_t from $\mathcal{A}$ with a random probability $\hbar$ based on the ε-greedy policy
    if $\hbar < \varepsilon$ **then**
      Randomly choose an action a_t;
    else
      Choose an action $a_t = \arg\max_{a' \in \mathcal{A}} Q(s_t, a'; \boldsymbol{\theta})$;
    end if
    Execute action a_t and receive a reward $r(s,a)$
    Get the next state s_{t+1} for the current action and state
    Update the input to Q_{t+1} based on the observed state
    Store the current experience tuple $\{s_t, a_t, o_t, r_t, s_{t+1}, o_{t+1}\}$ in the experience relay memory
    Randomly choose a minibatch of experiences $\{s_t, a_t, o_t, r_t, s_{t+1}, o_{t+1}\}$ from the experience relay memory
    Calculate the target Q-value by the learning step with optimizer
    if an episode terminates at iteration $t+1$ **then**
      $y(t) = r(s,a)$;
    else
      $y(t) = r(s,a) + \gamma Q(s', a'; \boldsymbol{\theta}); \boldsymbol{\theta}')$;
      $a' = \arg\max_{a' \in \mathcal{A}} Q(s_t, a'; \boldsymbol{\theta})$;
    end if
    Update the weight of the online network by minimizing the loss function as
    $\text{Loss}(\boldsymbol{\theta}) = \mathbb{E}[(y(t) - Q(s,a;\boldsymbol{\theta}))^2]$
    Perform a gradient descent step on $\text{Loss}(\boldsymbol{\theta})$ relative to the weight of the online network $\boldsymbol{\theta}$
    Update the weight of the target network based on the weight of the online network $\boldsymbol{\theta}$
    Rest the target network $Q' \leftarrow Q$ after F iterations
  end for
end for

B. Implementation of DDQN-Based Algorithm

According to the above analysis, we summarize the detailed procedure of our proposed DDQN-based algorithm for solving problems P2 and P3 as Algorithm 2. Specifically, the first step is to input the various parameters of Algorithm 2. The training purpose of Algorithm 2 is to output the optimal policy and the maximum reward.

Authorized licensed use limited to: UNIVERSITY OF VICTORIA. Downloaded on May 14,2024 at 21:25:37 UTC from IEEE Xplore. Restrictions apply.

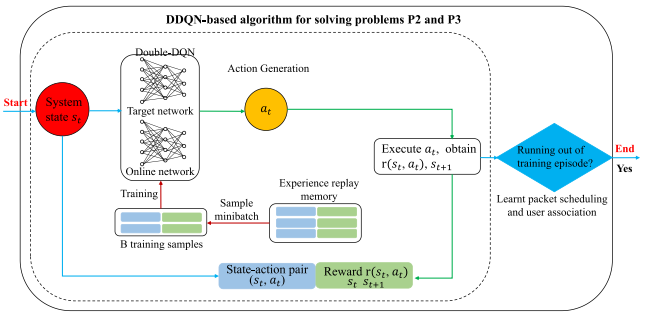

Fig. 5. Training process of DDQN-based algorithm for solving P1 and P2.

Before training, we initialize the experience replay memory and the weights of the online network and the target network. In particular, the weight of the target network is initialized to be the same as that of the online network, i.e., $\boldsymbol{\theta}' = \boldsymbol{\theta}$. As shown in Fig. 5, we develop two neural networks to illustrate the correlation between each state-action pair (s, a) and its value function $Q(s, a)$. Hence, it is crucial to preprocess the admission control and output selection and user association of the downlink network for a sufficiently long time with a random policy. On this basis, the estimation of $Q(s, a)$ can be obtained by means of some state transition information, which is then stored in the experience replay memory. Moreover, we train the neural network with the input state-action pair (s, a) and the outcome $Q(s, a)$. After that, the action selection and Q-value can be achieved. More specifically, in each decision episode, the network environment is initialized and the agent observes the initial state space $\{C_u(t), Q_u(t), x_{u,k}(t), L_{uk}(t)\}$ from the environment simulator. Then, we adopt the ε-greedy policy with the greedy factor $\varepsilon \in [0, 1]$ to select the execution action a. For a random probability $\hbar < \varepsilon$, the online network randomly selects an action from the action space $\mathcal{A}$. Otherwise, the action with the largest Q-value function derived from the online network with the input of the state-action pair (s, a) is selected. After selecting an action a, the environment simulator provides the corresponding reward $r(s, a)$ to the agent and the state is transferred from s_t to the next state s_{t+1}.

To improve training stability, we use the experience replay memory to store the experience tuple $\{s_t, o_t, a_t, r_t, s_{t+1}, o_{t+1}\}$ and randomly select a minibatch of B experiences to train the weights of the online network and the target network at each iteration. During the learning process, the online network and the target network compute the optimal values $Q(s', a'; \boldsymbol{\theta})$ and $Q(s', a'; \boldsymbol{\theta}')$, respectively. Given the current reward $r(s, a)$ and the discount factor γ, the target network yields a target value $y(t) = \begin{cases} r(s,a), \\ r(s,a) + \gamma Q(s', \arg\max_{a'} Q(s', a'; \boldsymbol{\theta}); \boldsymbol{\theta}'), \end{cases}$ which is then compared with the estimated value of the online network to obtain the loss value $\text{Loss}(\boldsymbol{\theta})$. Moreover, the weight of the online network is updated by using a gradient descent step of $[\partial \text{Loss}(\boldsymbol{\theta})/\partial \boldsymbol{\theta}]$. The weight of the target network does not need to be updated iteratively and it can copy $\boldsymbol{\theta}$ from the online network after a certain number of iterations. Note that the value of the loss function gradually decreases by constantly updating the weight of the online network. When the loss value reaches the global minimum, our proposed algorithm outputs the corresponding optimal policy. Fig. 5 describes the detailed procedure of the proposed DDQN-based algorithm for finding the optimal packet scheduling policy for the BS-agent.

VI. Simulation Results

In this section, we present simulation results to demonstrate the effectiveness of our proposed DDQN-based algorithm. We first introduce our simulation setup and network architecture. We then compare the algorithm in this paper with several other algorithms and analyze the simulation results under different scheduling policies.

A. Simulation Setup

We consider a single-hop downlink system in which $U = 4$ ground users are associated with a single BS. In this system, the packet scheduling matrix is selected every time slot within the queue corresponding to each user $u \in \mathcal{U}$, so that at most one packet is served per input and per output at each time slot.[1] The arrival process for each queue follows a Bernoulli distribution, which is independently and identically distributed over time slots with the arrival rate r_{uk}. A total of $K = 2000$ packets consisting of various delay requirements, i.e., $H = 400$, $M = 600$ and $L = 1000$, arrives at the BS over a period of time. The maximum tolerable queuing delays of the three classes of packets are set as $\text{Th}_H = 10$, $\text{Th}_M = 20$ and $\text{Th}_L = 30$, respectively, with the unit of time slots. In addition, the output gain weights of the three classes of packets are set as $\omega_H = 0.5$, $\omega_M = 0.3$ and $\omega_L = 0.2$, respectively. The discount factor of the potential output gain is set as $\rho = 0.3$.

We conduct the experimental simulations using a server with an NVIDIA GTX 2080 Ti GPU. The software platform of the experiment is Python 3.6 with PyTorch [46]. Our developed DDQN-based algorithm consists of the online-network and target-network, each of which has one input layer, two hidden layers and one output layer. Moreover, each hidden layer is assumed to have the same number of neurons and is defined as $e = 64$. We use the rectified linear unit (ReLU) function $f_{\text{ReLU}}(x) = \max\{0, 1\}$ to describe the activation function in each hidden layer. The Adam optimizer is used to update the weights of the online network and the target network. Besides, the target network is updated by the online network every 100 training iterations. The learning rates of both neural networks are set as $\alpha = 0.0001$ to ensure that the training process does not miss all possible local solutions. The discount factor is set as $\gamma = 0.999$. During the training process, the ε-greedy policy is used where the value of ε is set as 0.9 at the beginning and then gradually decreases to 0.1. The training process of our proposed DDQN-based algorithm has $E = 2000$ episodes. The capacity

[1]The proposed solution can be easily extended to the system to schedule packets for multiple orthogonal channels

Authorized licensed use limited to: UNIVERSITY OF VICTORIA. Downloaded on May 14,2024 at 21:25:37 UTC from IEEE Xplore. Restrictions apply.

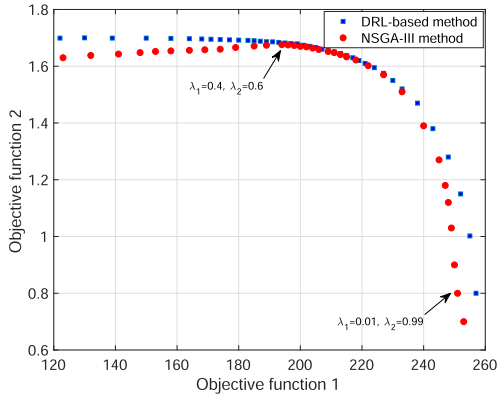

Fig. 6. Pareto-Front achieved by the proposed DRL-based framework.

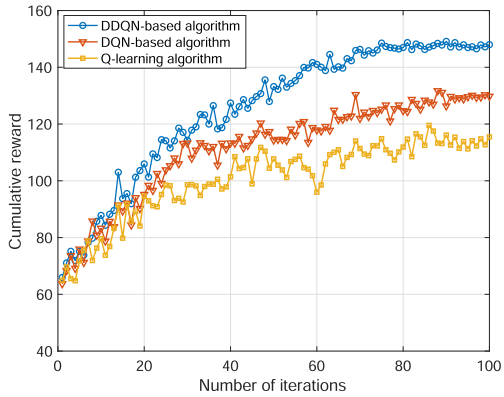

Fig. 7. Convergence of different algorithms for the overload case.

of the experience replay memory is set as $C = 2000$ and the size of each minibatch sampled from this experience replay memory is set as $B = 64$.

B. Result Analysis

Based on the proposed DRL framework, the multi-objective problem of scheduling and association can be solved by using the neighborhood-based parameter transfer strategy. In Fig. 6, we illustrate the Pareto front obtained by the proposed DRL-based method and the NSGA-III method [47]. It can be seen from Fig. 6 that the performance of the DRL-based method is close to that of the NSGA-III method when the weight values λ_1 and λ_2 are close each other (e.g., $\lambda_1 = 0.4$ and $\lambda_2 = 0.6$). When the difference between λ_1 and λ_2 is significant (e.g., $\lambda_1 = 0.01$ and $\lambda_2 = 0.99$), the DRL-based method tends to spare most of its effort in training only one of the objective functions. The obtained policy will then perform relatively poor on the other objective function. Whereas the DRL-based method shows its advantage over the NSGA-III method in terms of computing time since it only needs a very simple forward progression after the training process is completed. This advantage becomes more prominent when the complexity of the multi-objective problem increases, e.g., the objective contains multiple integer decision variables.

To demonstrate the convergence of the proposed algorithm and the other two algorithms, Fig. 7 plots the learning curves of these three algorithms in the case where the arrival rate is outside of the capacity region. With the increase of the number of iterations, the cumulative rewards obtained by the three algorithms have an obvious tendency to increase and converge. It is clear that the proposed DDQN-based algorithm always outperforms the other two algorithms in terms of cumulative reward. The reason is that the proposed algorithm can prevent the overestimation of the action-value function by decoupling the Q-target. Besides, the convergence speed of the proposed DDQN-based algorithm and the DQN-based algorithm is substantially higher than that of the Q-learning algorithm. This is because both the proposed DDQN-based algorithm and the DQN-based algorithm use the neural network $Q(s, a; \boldsymbol{\theta})$ to approximate the target Q-value $Q^*(s, a)$, which can greatly reduce the time of searching the maximum value.

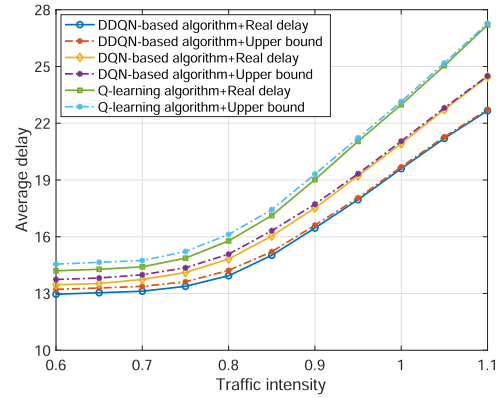

Fig. 8. Delay versus traffic intensity for the inner capacity case.

1) Delay Performance: Fig. 8 shows the delay performance of different algorithms versus traffic intensity ϱ[2] when the arrival rate is within the capacity region. The solid and dash lines indicate the average delay calculated based on the observed delay (real delay) and the maximum tolerable delay per packet (upper bound), respectively. The average delay achieved by our proposed algorithm and the other two algorithms increases with the increase of traffic intensity ϱ and eventually diverges when $\varrho > 1$, i.e., outside the capacity region. In addition, we observe that our proposed DDQN-based algorithm achieves 9% and 20% delay reduction compared with the DQN-based algorithm and the Q-learning algorithm, respectively.

Fig. 9 shows the delay performance versus traffic intensity ϱ for the general case. As can be seen from Fig. 9, for the three algorithms, the average delays increases with ϱ and eventually becomes saturated when ϱ is sufficiently large. In addition, our proposed DDQN-based algorithm shows up to 6% and 13% delay reduction compared to the DQN-based algorithm and the Q-learning algorithm, respectively. From Figs. 8 and 9, we observe that our proposed DDQN-based algorithm outperforms the other two algorithms in terms of average delay, which demonstrates the effectiveness of our proposed algorithm. It is

[2]The traffic intensity ϱ is defined as the ratio of the arrival rate to the capacity region boundary [32] to measure the average resource occupancy.

Authorized licensed use limited to: UNIVERSITY OF VICTORIA. Downloaded on May 14,2024 at 21:25:37 UTC from IEEE Xplore. Restrictions apply.

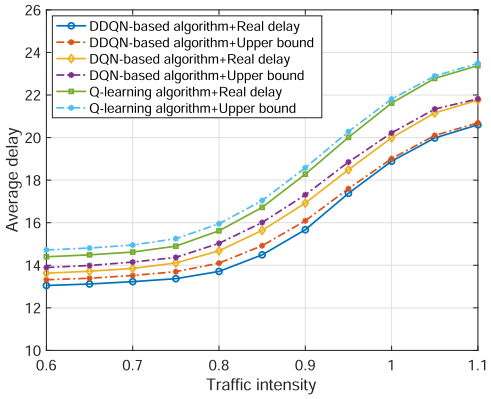

Fig. 9. Delay versus traffic intensity for the overload case.

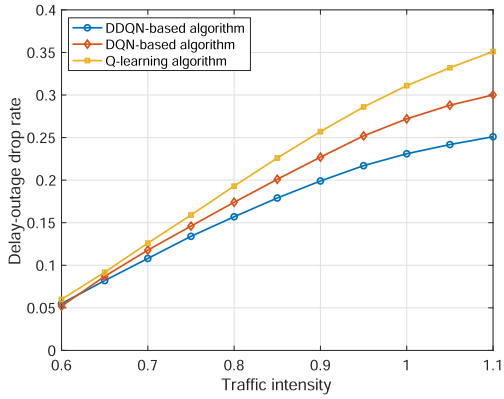

Fig. 11. Drop rate versus traffic intensity for the overload case.

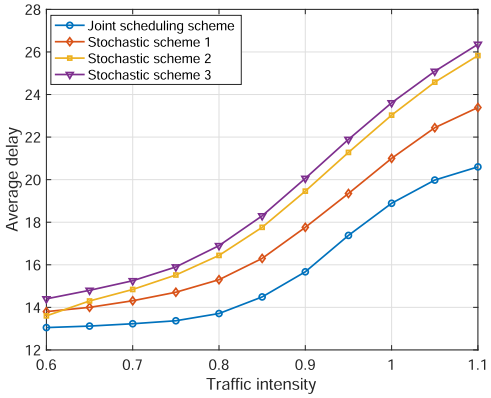

Fig. 10. Delay versus traffic intensity for different design schemes.

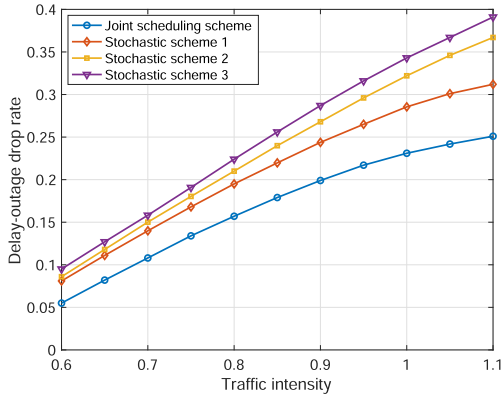

Fig. 12. Drop rate versus traffic intensity for different design schemes.

obvious that the real average delay gradually approaches the upper bound as ϱ becomes large in both cases. The reason is that as ϱ grows, the queue length of each user increases leading to longer queue delay, and thus packets reach their maximum tolerable queuing time. On the other hand, it shows that our proposed model guarantees per-packet delay.

In order to demonstrate the superiority of the proposed joint-scheduling model, we consider the following three schemes: (i) scheme 1 jointly optimizes user association and packet selection but without admission control [32]; (ii) scheme 2 jointly optimizes admission control and user association but with FIFO queue management [34]; and (iii) scheme 3 adopts random packet admission and user association as well as FIFO queue management. Note that all schemes are solved by the proposed DDQN-based algorithm.

Fig. 10 depicts the average delay achieved by the various schemes versus traffic intensity ϱ. Despite the average delay of all the four schemes increases as the traffic intensity ϱ increases, the proposed joint scheduling scheme always achieves the lowest delay. In addition, we note that only our proposed scheme and scheme 1 shows the trend of convergence when ϱ exceeds the capacity region, while the others diverge. The reason is that the packet-selection process becomes more crucial for better control of packets with different delay requirements when ϱ is large. The comparisons between the proposed joint scheduling scheme and scheme 1, and scheme 2 and scheme 3 also show the importance of admission control. In addition, we conclude that the packet-selection process plays an essential role in reducing average delay by comparing our proposed scheme and scheme 2. Overall, our scheme achieves a delay reduction up to 30%.

2) Delay Outage Drop Performance: Since the proposed model guarantees per-packet delay by dropping delay outage packets, minimizing the drop rate is one of the key metrics in evaluating the performance. Fig. 11 plots the delay outage drop rate achieved by the three algorithms versus traffic intensity ϱ in the case where the arrival rate is outside of the capacity region. It is obvious that the drop rate increases for the three algorithms as ϱ becomes large. The reason is that high traffic intensity leads to longer queue lengths where packets with high delay requirements are more likely to be dropped. On the other hand, the proposed DDQN-based algorithm achieves a much slower increase in terms of drop rate, i.e., a better scheduling policy is achieved. In spite of the increase, all algorithms show the trend of convergence when $\varrho > 1$ thanks to the adoption of the packet-selection process which avoids unnecessary delay outage drops within each queue.

Next, we compare the drop rate performance of our proposed scheduling model with the three benchmark schemes as shown in Fig. 12. The proposed joint scheduling achieves a significantly lower drop rate up to 60% than the other three schemes. In

Authorized licensed use limited to: UNIVERSITY OF VICTORIA. Downloaded on May 14,2024 at 21:25:37 UTC from IEEE Xplore. Restrictions apply.

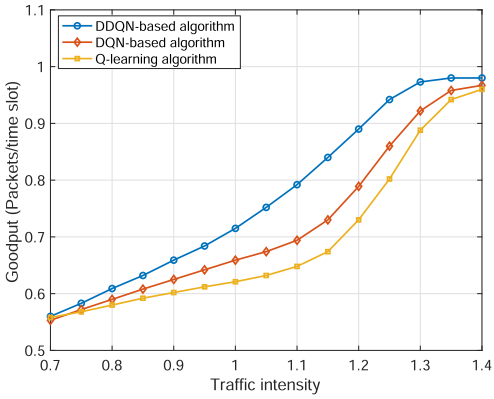

Fig. 13. Goodput versus traffic intensity for the overload case.

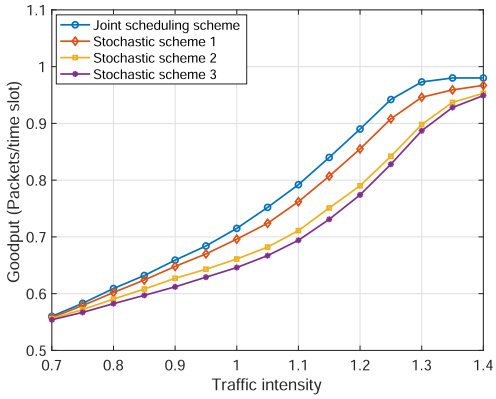

Fig. 15. Goodput versus traffic intensity for different design schemes.

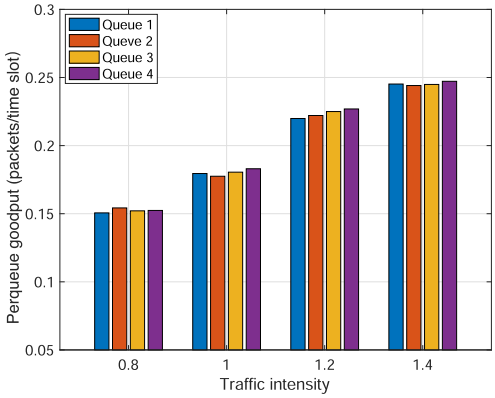

Fig. 14. Goodput of each queue versus traffic intensity.

addition, we observe that the growth trend for our proposed scheme and scheme 1 in Fig. 12 is similar to that in Fig. 10. This is because the packet-selection process allows as many packets as possible to be transmitted before reaching their maximum tolerable queuing time. The increasing gap between the proposed joint design scheme and schemes 2 and 3 also indicates the significance of the packet-selection process in our design.

3) Throughput Performance: As the results presented in the previous theoretical analysis, our proposed algorithm can achieve utility maximization while guaranteeing per-packet delay. Here, we consider goodput, i.e., the number of successfully received packets over the total measured time period. Fig. 13 plots the goodput performance of different algorithms versus traffic intensity ϱ. For the three algorithms, the goodput increases with ϱ and eventually converge to 1 when ϱ is sufficiently large. This is because the BS serves at most one user per time slot, i.e., at most one packet is transmitted per time slot. In other words, since the link capacity is 1 packet/time-slot, the goodput has an upper bound of 1 packet/time-slot. The proposed DDQN-based algorithm is observed to substantially outperforms the Q-learning algorithm and the DQN-based algorithm in terms of goodput, which further demonstrates the effectiveness of the proposed algorithm.

In order to demonstrate fairness among users, Fig. 14 plots the goodput per queue versus traffic intensity ϱ for the outside capacity region case. It can be seen that the goodput of all the queues increases as ϱ becomes large. For different values of ϱ, the total goodput of the four queues is equal to the goodput corresponding to the blue curve in Fig. 13. It can also be seen that the values of the four queues are very close to each other, which confirms that our formulated model can achieve fairness among users.

In Fig. 15, the goodput achieved by the four schemes versus traffic intensity ρ is plotted. As expected, the goodput increases for the four schemes as ρ becomes large and eventually tends to 1 when ρ is sufficiently large. For different values of ρ, our proposed scheme always achieves the highest goodput, while scheme 3 has the lowest goodput. In addition, we observe that the curve of scheme 1 is closer to the curve of our proposed scheme than that of scheme 2. These results further indicate that packet transfer control is more effective than admission control in improving goodput.

VII. Conclusion

In this article, we investigated the scheduling problem to guarantee per-packet delay in a single-hop wireless network for delay-critical applications. All arriving packets have different delay requirements, which were divided into three categories. In addition, packets with high delay requirements are prioritized and yield high utility after successful transmission. To differentiate the utility of different packets, we introduced a novel output gain function. In addition, a scheduling policy based on the delay laxity and output gain was proposed which transmits one packet with the minimal delay laxity and the maximal output gain every time slot. Note that any packet that stays in the queue longer than its delay budget needs to be dropped. In this context, we formulate a multi-objective optimization problem that minimizes the average queue length while maximizing the average output gain under the constraint of guaranteeing per-packet delay. To cope with the uncertainties in the environment (e.g., random packet arrivals and dynamic channel conditions), we transformed our problem into a POMDP and proposed a Q-learning algorithm to solve it. To avoid the curse of dimensionality of Q-learning in large-scale networks and alleviate Q-value overestimation, we proposed a DDQN-based algorithm to derive the joint policy of

Authorized licensed use limited to: UNIVERSITY OF VICTORIA. Downloaded on May 14,2024 at 21:25:37 UTC from IEEE Xplore. Restrictions apply.

link scheduling and packet selection. Simulation results show a better performance of our proposed DDQN-based algorithm than that of the Q-learning and DQN algorithms. However, many other issues in practice, e.g., wireless link dynamics, were not considered in the current model. In future work, more effort will be addressed on incorporating environmental dynamics to improve the scalability of our model.

REFERENCES

[1] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE Netw.*, vol. 34, no. 3, pp. 134–142, May/Jun. 2020.

[2] X. Lan, Y. Chen, and L. Cai, "Throughput-optimal H-QMW scheduling for hybrid wireless networks with persistent and dynamic flows," *IEEE Trans. Wireless Commun.*, vol. 19, no. 2, pp. 1182–1195, Feb. 2020.

[3] Y. Chen, X. Wang, and L. Cai, "On achieving fair and throughput-optimal scheduling for TCP flows in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 12, pp. 7996–8008, Dec. 2016.

[4] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing for time-varying wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 89–103, Jan. 2005.

[5] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Found. Trends Netw.*, vol. 1, no. 1, pp. 1–149, Oct. 2006.

[6] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control," *IEEE Trans. Netw.*, vol. 15, no. 6, pp. 1333–1344, Dec. 2007.

[7] J. Lee, R. Mazumdar, and N. Shroff, "Opportunistic power scheduling for dynamic multi-server wireless systems," *IEEE Trans. Wireless Commun.*, vol. 5, no. 6, pp. 1506–1515, Jun. 2006.

[8] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali, "Routing without routes: The backpressure collection protocol," in *Proc. IEEE/ACM 9th Int. Conf. Inf. Process. Sensor Netw.*, 2010, pp. 279–290.

[9] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, and R. Vijayakumar, "Scheduling in a queuing system with asynchronously varying service rates," *Probab. Eng. Inf. Sci.*, vol. 18, no. 2, pp. 191–217, Apr. 2004.

[10] M. J. Neely, "Super-fast delay tradeoffs for utility optimal fair scheduling in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1489–1501, Aug. 2006.

[11] L. Hai, Q. Gao, J. Wang, H. Zhuang, and P. Wang, "Delay-optimal backpressure routing algorithm for multihop wireless networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 2617–2630, Mar. 2018.

[12] W. Wu, P. Yang, W. Zhang, C. Zhou, and X. Shen, "Accuracy-guaranteed collaborative DNN inference in industrial IoT via deep reinforcement learning," *IEEE Trans. Ind. Inform.*, vol. 17, no. 7, pp. 4988–4998, Jul. 2021.

[13] Y. Cui, R. Wang, H. Huang, and S. Zhang, "A survey on delay-aware resource control for wireless systems large deviation theory, stochastic lyapunov drift, and distributed stochastic learning," *IEEE Trans. Inf. Theory*, vol. 58, no. 3, pp. 1677–1701, Mar. 2012.

[14] G. R. Ghosal, D. Ghosal, A. Sim, A. V. Thakur, and K. Wu, "A deep deterministic policy gradient based network scheduler for deadline-driven data transfers," in *Proc. IFIP Netw. Conf.*, 2020, pp. 253–261.

[15] S. Chilukuri, G. Piao, D. Lugones, and D. Pesch, "Deadline-aware TDMA scheduling for multihop networks using reinforcement learning," in *Proc. IFIP Netw. Conf.*, 2021, pp. 1–9.

[16] L. Qu, C. Assi, and K. Shaban, "Delay-aware scheduling and resource optimization with network function virtualization," *IEEE Trans. Commun.*, vol. 64, no. 9, pp. 3746–3758, Sep. 2016.

[17] F. Tang, Y. Zhou, and N. Kato, "Deep reinforcement learning for dynamic uplink or downlink resource allocation in high mobility 5G HetNet," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 12, pp. 2773–2782, Dec. 2020.

[18] L. Yang, Y. E. Sagduyu, J. Zhang, and J. H. Li, "Deadline-aware scheduling with adaptive network coding for real-time traffic," *IEEE/ACM Trans. Netw.*, vol. 23, no. 5, pp. 1430–1443, Oct. 2015.

[19] M. K. Sharma, T. P. Hui, E. Kurniawan, and S. Sumei, "Packet drop probability-optimal cross-layer scheduling: Dealing with curse of sparsity using prioritized experience replay," in *Proc. IEEE Int. Conf. Commun.*, 2021, pp. 1–6.

[20] Z. Jiao, C. Li, and H. T. Mouftah, "Backpressure-based routing and scheduling protocols for wireless multihop networks: A survey," *IEEE Wireless Commun.*, vol. 23, no. 1, pp. 102–110, Feb. 2016.

[21] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Autom. Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.

[22] G. D. Celik, L. B. Le, and E. Modiano, "Dynamic server allocation over time-varying channels with switchover delay," *IEEE Trans. Inf. Theory*, vol. 58, no. 9, pp. 5856–5877, Sep. 2012.

[23] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1260–1267, Aug. 1999.

[24] M. Marsan, E. Leonardi, and F. Neri, "On the stability of local scheduling policies in networks of packet switches with input queues," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 4, pp. 642–655, May 2003.

[25] E. Leonardi, M. Mellia, and F. Neri, "Bounds on average delays and queue size averages and variances in input-queued cell-based switches," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2001, pp. 1095–1103.

[26] M. McConley, B. Appleby, M. Dahleh, and E. Feron, "A computationally efficient Lyapunov-based scheduling procedure for control of nonlinear systems with stability guarantees," *IEEE Trans. Autom. Control*, vol. 45, no. 1, pp. 33–49, Jan. 2000.

[27] M. J. Neely, E. Modiano, and C.-P. Li, "Fairness and optimal stochastic control for heterogeneous networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 2, pp. 396–409, Apr. 2008.

[28] A. Ferragut and F. Paganini, "Network resource allocation for users with multiple connections: Fairness and stability," *IEEE/ACM Trans. Netw.*, vol. 22, no. 2, pp. 349–362, Apr. 2014.

[29] T. Zhang, S. Shen, S. Mao, and G.-K. Chang, "Delay-aware cellular traffic scheduling with deep reinforcement learning," in *Proc. IEEE Glob. Commun. Conf.*, 2020, pp. 1–6.

[30] I. S. Comsa et al., "Towards 5G: A reinforcement learning-based scheduling solution for data traffic management," *IEEE Trans. Netw. Serv. Manag.*, vol. 15, no. 4, pp. 1661–1675, Dec. 2018.

[31] J. Chen, P. Yang, Q. Ye, and X. Li, "Learning-based proactive resource allocation for delay-sensitive packet transmission," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 2, pp. 675–688, Jun. 2021.

[32] J. Bae, J. Lee, and S. Chong, "Learning to schedule network resources throughput and delay optimally using Q-learning," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 750–763, Apr. 2021.

[33] L. Cai, J. Pan, W. Yang, X. Ren, and X. Shen, "Self-evolving and transformative (SET) protocol architecture for 6G," *IEEE Wireless Commun.*, early access, Aug. 22, 2022, doi: 10.1109/MWC.003.2200022.

[34] M. J. Neely, "Delay-based network utility maximization," *IEEE/ACM Trans. Netw.*, vol. 21, no. 1, pp. 41–54, Feb. 2013.

[35] S. Oh, A. Khil, and S. Yang, "A modified least-laxity first scheduling algorithm for reducing context switches on multiprocessor systems," *J. Comput. Syst. Theory*, vol. 30, no. 12, pp. 68–77, Feb. 2003.

[36] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synth. Lectures Commun. Netw.*, vol. 3, no. 1, pp. 1–211, Jan. 2010.

[37] M. Kaisa, *Nonlinear Multiobjective Optimization*, vol. 12. Norwell, MA, USA: Kluwer, 1999.

[38] X. Ma, Y. Yu, and X. Li, "A survey of weight vector adjustment methods for decomposition-based multiobjective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 24, no. 4, pp. 634–649, Aug. 2020.

[39] R. Wang, Z. Zhou, H. Ishibuchi, T. Liao, and T. Zhang, "Localized weighted sum method for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 3–18, Feb. 2018.

[40] L. Ke, Q. Zhang, and R. Battiti, "MOEA/D-ACO: A multiobjective evolutionary algorithm using decomposition and antcolony," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1845–1859, Dec. 2013.

[41] V. Mnih, K. Kavukcuoglu, D. Silver, and J. Veness, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[42] E. Altman, *Constrained Markov Decision Processes*. Norwell, MA, USA: Kluwer, 1999.

[43] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[44] J. Huang, Y. Yang, and G. He, "Deep reinforcement learning-based dynamic spectrum access for D2D communication underlay cellular networks," *IEEE Commun. Lett.*, vol. 25, no. 8, pp. 2614–2618, Aug. 2021.

Authorized licensed use limited to: UNIVERSITY OF VICTORIA. Downloaded on May 14,2024 at 21:25:37 UTC from IEEE Xplore. Restrictions apply.

[45] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.
[46] A. Paszke, S. Gross, F. Massa, and A. Lerer, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 8026–8037.
[47] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, Aug. 2014.

Jiequ Ji received the PhD degree from the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2021. From October 2018 to January 2020, she was a research assistant with Nanyang Technological University, Singapore, with Prof. Dusit Niyato. She was a research fellow with the Department of Electrical and Computer Engineering, University of Victoria, Canada, from 2021 to 2022. She is currently a post-doctoral research fellow with the Pillar of Information Systems Technology and Design, Singapore University of Technology and Design, Singapore. Her research interests include UAV-enabled wireless communications, wireless content caching, resource allocation in 5G and beyond, mobile edge computing, and physical layer security.

Xiangyu Ren (Student Member, IEEE) received the BSc degree from the Department of Automation Engineering, University of Electronic Science and Technology of China, Chengdu, China, in 2019. He is currently working toward the PhD degree in electrical engineering with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada. He is the recipient of the Graduate Student Fellowship Award from the University of Victoria, in 2021. His research interests include deterministic networks, software-defined network, vehicular networks, machine learning, and optimization with applications in networking.

Lin Cai (Fellow, IEEE) received the MASc and PhD degrees (awarded Outstanding Achievement in Graduate Studies) in electrical and computer engineering from the University of Waterloo, Waterloo, Canada, in 2002 and 2005, respectively. Since 2005, she has been with the Department of Electrical and Computer Engineering, University of Victoria, and she is currently a professor. She is an NSERC E.W.R. Steacie Memorial fellow, an Engineering Institute of Canada (EIC) fellow. In 2020, she was elected as a member of the Royal Society of Canada's College of New Scholars, Artists and Scientists, and a 2020 "Star in Computer Networking and Communications" by N2Women. Her research interests span several areas in communications and networking, with a focus on network protocol and architecture design supporting emerging multimedia traffic and the Internet of Things. She was a recipient of the NSERC Discovery Accelerator Supplement (DAS) Grants, in 2010 and 2015, respectively. She has co-founded and chaired the IEEE Victoria Section Vehicular Technology and Communications Joint Societies Chapter. She has been elected to serve the IEEE Vehicular Technology Society Board of Governors, 2019–2024.

Kun Zhu (Member, IEEE) received the PhD degree from the School of Computer Engineering, Nanyang Technological University, Singapore, in 2012. He was a research fellow with the Wireless Communications Networks and Services Research Group, University of Manitoba, Canada, from 2012 to 2015. He is currently a professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. He is also a Jiangsu specially appointed professor. His research interests include resource allocation in 5G, wireless virtualization, and self-organizing networks. He has published more than fifty technical papers and has served as TPC for several conferences. He won several research awards including IEEE WCNC 2019 Best paper awards, ACM China rising star chapter award.

Authorized licensed use limited to: UNIVERSITY OF VICTORIA. Downloaded on May 14,2024 at 21:25:37 UTC from IEEE Xplore. Restrictions apply.