Beyond Max-weight Scheduling: A Reinforcement Learning-based Approach

Jeongmin Bae
Department of Electrical Engineering
KAIST
Email: jmbae93@kaist.ac.kr

Joohyun Lee
Division of Electrical Engineering
Hanyang University
Email: joohyunlee@hanyang.ac.kr

Song Chong
Department of Electrical Engineering
KAIST
Email: songchong@kaist.edu

***Abstract*—As network architecture becomes complex and the user requirement gets diverse, the role of efficient network resource management becomes more important. However, existing network scheduling algorithms such as the max-weight algorithm suffer from poor delay performance. In this paper, we present a reinforcement learning-based network scheduling algorithm that achieves both optimal throughput and low delay. To this end, we first formulate the network optimization problem as an MDP problem. Then we introduce a new state-action value function called W-function and develop a reinforcement learning algorithm called W-learning that guarantees little performance loss during a learning process. Finally, via simulation, we verify that our algorithm shows delay reduction of up to 40.8% compared to the max-weight algorithm over various scenarios.**

I. Introduction

Recently, wireless network has been supporting unprecedented wireless traffic due to the dramatic growth of mobile devices such as smartphones, the development of various applications and implementation of internet or things. Thus, wireless network architecture comes to be more intricate and user's requirement becomes various and complex. Hence, the role of network resource management has come to be crucial.

The resource scheduling problem of communication networks has been extensively studied in various contexts and formulations, where most of them consider throughput and delay as main objectives. The max-weight algorithm is one of the most widely used scheduling algorithms achieving throughput optimality by stabilizing the network when the arrival rate is inside the network capacity region [1], [2]. These studies employed the Lyapunov technique that greedily minimizes the Lyapunov drift at every time, given the instantaneous network state without any prior knowledge on its distribution[1]. For a more general case, where the arrival rate may be either inside or outside of the capacity region, the Lyapunov optimization technique for optimizing functions of time average called min-drift-plus-penalty [3] can be used to solve the network utility maximization problem. It aims to allocate the restricted resources to users fairly by maximizing the sum of network utilities under a network stability constraint[2].

[1]Lyapunov drift is defined as the difference in the Lyapunov function, which represents a network congestion level, from one time slot to the next.

[2]The utility of a user is typically defined as an increasing function of the user's throughput.

Recently, learning algorithms have been actively applied to the resource management problem of wireless networks [4]. Chen et al. [5] addressed the resource allocation problem in virtual reality wireless networks by formulating the problem as a noncooperative game and solving it through a learning algorithm based on the framework of echo state networks(ESN). Challita et al. [6] proposed an interference-aware path planning scheme for a network of cellular-connected unmanned aerial vehicles(UAVs) leveraging deep reinforcement learning based on ESN. Xu et al. [7] proposed a deep reinforcement learning based framework for power-efficient resource allocation in cloud RANs, and [8] suggested an experience-driven approach using deep reinforcement learning for enabling model-free control framework for traffic engineering(TE). He et al. [9] addressed the problem of finding an optimal user selection policy in cache-enabled opportunistic interference alignment(IA) wireless networks with the time-varying channel.

However, a wide range of user QoS requirements such as high throughput, low latency, and low energy consumption makes the existing resource management algorithm based on constrained optimization framework unavailing. Indeed, although the max-weight algorithm achieves the throughput optimality, it is known to suffer from high network delay because of the long queue length. Although a learning-based approach has been actively applied recently to tackle the issue, general learning-based algorithms suffer from poor performance during the learning process, since the parameters to be learned are usually initialized with random values before learning starts. Therefore, it is still not viable to apply learning-based algorithms in practice, whose performance is poor during the learning period.

In this paper, we propose a novel learning-based resource management algorithm for the downlink scheduling scenario which achieves optimal throughput while addressing the large queue backlog problem of the existing algorithms by using statistics of environments. We first transform the network optimization problem into a Markov Decision Process (MDP) problem so that dynamic programming can be applied to. With the transformed MDP problem, however, it is impossible to solve it directly through dynamic programming in the real world, due to the curse of dimensionality and uncertainty (i.e., a large action-state space and unknown model of the environment). Hence, reinforcement learning is applied to

learn the model by interacting with the environment without any prior knowledge.

To address the problem of poor performance during the learning period, we design a learning algorithm that adopts existing resource management algorithms such as max-weight algorithm as a baseline policy during the learning period and then gradually converges to an optimal policy. For this purpose, we introduce a new action-value function called W-function, which is a variant of Q-function (i.e., action-value function) and enables us to initialize the policy to a baseline policy. Then, we propose W-learning, which is similar to Q-learning, but uses W-function instead of Q-function. The contribution of this paper is summarized as follows.

- We transform the network optimization problem into an MDP problem so that dynamic programming can be applied to. We show that the designed MDP problem achieves throughput optimality while minimizing the average delay.
- We design a novel reinforcement algorithm called W-learning to solve the transformed MDP problem, which guarantees the minimal performance loss during the learning period by introducing a new action-value function, called W-function. We show that the W-learning algorithm converges to an optimal policy of the MDP problem.
- Via simulation, we show that W-learning can reduce delay up to 40% compared to the max-weight scheduling algorithm while achieving the throughput optimality. Furthermore, we show that W-learning can guarantee the performance as good as the max-weight scheduling algorithm during the learning period.

The rest of the paper is organized as follows. In Section II, we present our system model and notations. In Section III, we formulate the network optimization problem into an MDP problem. In Section IV, we introduce our learning algorithm to solve the problem formulated in Section III. In Section V, we give a theoretical analysis of our algorithm and in Section VI, we show the performance evaluation of our algorithm with simulations. Finally, in Section VII, we conclude our work.

II. System Model

We consider a general downlink scheduling scenario with a single base station and multiple users as depicted in Figure 1. We assume that the system operates in discrete time $t \in \{0, 1, \cdots\}$. Let $\mathcal{N} = \{0, 1, \cdots, N\}$ denote the set of N users that are associated to the base station. We denote $c(t) = \{c_1(t), \cdots, c_N(t)\}$ as the channel states, where $c_n(t)$ represents the amount of data (bits) that can be transmitted over channel n at time slot t when user n is scheduled, and $\theta(t) = \{\theta_1(t), \cdots, \theta_N(t)\}$ as the control variables for user scheduling at time slot t. We assume that the channel state of each user is independent and identically distributed (i.i.d.) over time and takes one of $C = \{C_1, \cdots, C_m\}$ at every $t \in \{0, 1, \cdots\}$ and that the base station can serve one user at every $t \in \{0, 1, \cdots\}$, i.e., $\theta_n(t) \in \{0, 1\}, \forall n \in \mathcal{N}$ and $\sum_{n \in N} \theta_n(t) = 1$. Let $i(t) = \{i_1(t), \cdots, i_N(t)\}$ be

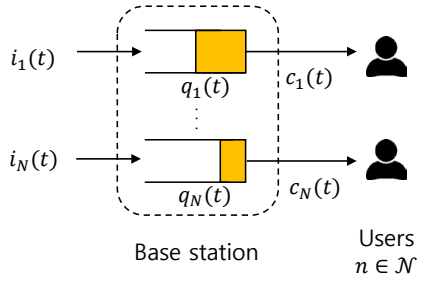

Figure 1: System model

the amount of data (bits) arriving to each user at the end of each time slot t. We assume that the arrival process is also i.i.d. over time with the average value $\lambda_n = \mathbb{E}[i_n(t)]$ and the maximum value of the arrival processes at each t is $I_{\max}$ i.e., $i_n(t) \leq I_{\max}, \forall n \in \mathcal{N}$. We denote the admitted data that is injected to the queue for each user n at t as $x_n(t) \in [0, i_n(t)]$ and long-term time average of $x_n(t)$ as $x_n = \lim_{t \to \infty} \frac{1}{T} \sum_{\tau=0}^{T-1} \mathbb{E}[x_n(\tau)], \forall n \in \mathcal{N}$. At each time slot t, the base station selects a user to be served with control variable $\theta_n(t), \forall n \in \mathcal{N}$. Let $q(t) = \{q_1(t), \cdots, q_N(t)\}$ be the backlog of network queue for each user. The queueing dynamics of $q(t)$ can be written as below.

$$q_n(t+1) = \max\left(q_n(t) - c_n(t)\theta_n(t), 0\right) + x_n(t), \forall n \in \mathcal{N}. \tag{1}$$

In Section III, we will introduce the virtual queue, $z(t) = \{z_1(t), \cdots, z_N(t)\}$ with admitted data $y(t) = \{y_1(t), \cdots, y_N(t)\}$, where $z_n(t)$ and $y_n(t)$ is defined as the backlog of the virtual queue for user n and admitted data arriving to the queue at time slot t. We also denote the utility function of user n, which is non-decreasing and concave, as $U_n(x_n)$.

Under the aforementioned assumptions, we can model our system as an MDP problem with the finite set of states, $\mathcal{S}$ and the finite set of possible actions, $\mathcal{A}$. We represent the state of the system at time t as $s_t = (q(t), z(t), c(t), i(t)), s_t \in \mathcal{S}$ and the action as $a_t = (\theta(t), x(t), y(t)), a_t \in \mathcal{A}$. At each time step t, the agent observes the current state s_t from the environment, decides an action a_t based on a deterministic policy $\pi : \mathcal{S} \to \mathcal{A}$, and receives reward $r_t = r(s_t, a_t)$ where $r : (\mathcal{S}, \mathcal{A}) \to \mathbb{R}$. In the rest of this paper, we sometimes omit the time index t and use $s = s_t$ and $a = a_t$ when we consider an arbitrary time slot, for brevity. Given a policy π, we also define the expected sum of discounted rewards of taking action a at state s and following π in next states as

$$Q_\pi(s, a) = r(s, a) + \alpha \sum_{s'} P^a_{ss'} Q_\pi(s', \pi(s')) \tag{2}$$

where $P^a_{ss'}$ denotes the transition probability from state s to the successive state s' when action a is taken, and $\alpha \in [0, 1)$ denotes the discount factor. The Bellman equation of the Q-function of the optimal policy π^*, $Q^*(s, a)$ would be written

as

$$Q^*(s,a) = r(s,a) + \alpha \sum_{s'} P^a_{ss'} \max_{a'} Q^*(s',a'). \quad (3)$$

III. Problem formulation

In this section, we define an optimization problem for stabilizing a network, which is equivalent to stabilizing all queues in the network, while jointly optimizing throughput and average delay. Then we transform it into an MDP problem to apply dynamic programming. For a general scenario, where the arrival rate may be either inside or outside of the capacity region, the goal of maximizing throughput becomes achieving optimal fairness among users measured by a utility function of throughput, $U_n(x_n)$, which is non-decreasing and concave [10]. Here, the average delay is proportional to the average queue length [11]. Hence, we formulate an optimization problem with the objective of minimizing average queue length with constraints of achieving optimal utility and mean rate stability [3], as below:

$$\textbf{P0:} \max_{\pi} \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{n\in\mathcal{N}} -\mathbb{E}_\pi \left[q_n(t+1)\right] \quad (4)$$

$$\text{s.t.} \quad \sum_{n\in\mathcal{N}} U_n(x_n) = U^{opt} \quad (5)$$

$$x_n \le \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[c_n(t)\theta_n(t)], \forall n \in \mathcal{N} \quad (6)$$

$$\theta_n(t) \in \{0,1\}, \sum_{n\in\mathcal{N}} \theta_n(t) = 1, \forall n \in \mathcal{N}, \quad (7)$$

where U^{opt} denotes an optimal utility, an optimal solution of the utility maximization problem [12]. Constraint (6) represents the mean rate stability constraint, which is equivalent to

$$\lim_{t\to\infty} \frac{\mathbb{E}[q_n(t)]}{t} = 0,$$

from [3]. We first consider a case where the arrival rate is within the capacity region. We will consider a more general case that the traffic arrival rate may be outside the capacity region later in this section. Since the arrival rate is within the capacity region, we do not need to consider a flow control and the only thing that has to be decided at every time step is $a_t = \theta(t)$[3]. For this inner capacity region scenario, the throughput is simply equal to the arrival rate for work-conserving policies that transmit as much data as possible under network stability condition and the throughput maximization problem simply reduces to the network stability problem [12]. However, by minimizing average queue length, we can simultaneously stabilize all queues in the network, which is shown in Proposition III.1. Therefore, the constraints (5) and (6) are satisfied simultaneously if (4) is achieved. Therefore, we define an MDP problem **P1** for the inner capacity region of which an optimal solution is also an optimal solution of **P0**. In addition to the average queue length term, we added the negative Lyapunov drift term $-\nu_1(q_n(t+1)^2 - q_n(t)^2))$ to the objective for a technical reason which will be described in Section IV in detail. Although the objective is transformed, we can still get an optimal solution which stabilizes all queues in network while minimizing the average queue length. We will show this in the following proposition. Intuitively, since the Lyapunov drift term represents the increment of queue backlog from one slot t to the next slot $t+1$, minimizing it pushes queues toward a low congestion state, which maintains network stability [3]. Furthermore, if all queues are stable, they do not diverge, which means that the drift term converges to 0 as t goes to infinity, leaving the average queue length term only.

$$\textbf{P1:} \max_{\pi} \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{n\in N} \mathbb{E}_\pi[-q_n(t+1) - \nu_1(q_n(t+1)^2 - q_n(t)^2)] \quad (8)$$

$$\text{s.t.} \quad \nu_1 > 0 \quad (9)$$

$$(7).$$

Proposition III.1. *An optimal policy of* **P1**

1) *makes all queues in the network mean rate stable for the inner capacity region.*
2) *minimizes the average queue length.*

Proof. Please refer to the technical report [13]. □

By Proposition III.1, we show that by solving **P1**, we can achieve throughput optimality as well as minimal average queue length. Therefore, an optimal solution of **P1** is also an optimal solution of **P0**.

The MDP problem of which the objective is defined by an expected average reward can be solved using average reinforcement learning such as R-learning [14]. However, its performance is often sensitive to the initial parameter and it may not converge to an optimal policy [15]. Therefore, we transform **P1** to a discounted dynamic programming problem with discount factor α. It is shown that by taking α close to 1, an optimal policy of discounted problem can be approximated to an optimal policy of average problem [16]–[20]. The transformed discounted dynamic programming problem can be written as below.

$$\textbf{P1}': \max_{\pi} \lim_{T\to\infty} \sum_{t=0}^{T-1} \alpha^t \mathbb{E}_\pi[-q_n(t+1) - \nu_1(q_n(t+1)^2 - q_n(t)^2)]$$

$$\text{s.t.} \quad (7), (9).$$

We now consider the more general case when traffic may lie in both inside and outside of the capacity region. Since multiple users share limited resources, we have to solve the network utility maximization problem to fairly allocate network resources while stabilizing the network. For outer capacity region, throughput is no longer equal to the arrival traffic. Instead, we denote the throughput of user n as $x_n(t)$

[3]We abuse this definition for simplicity.

and let $x_n(t) \in [0, i_n(t)]$. We first transform **P0** into the following form using the Lagrangian approach [21].

$$\textbf{P0}': \max_{\pi} \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\sum_{n\in\mathcal{N}} -\mathbb{E}_\pi\left[q_n(t+1)\right] + \nu \sum_{n\in\mathcal{N}} U_n(x_n) \tag{10}$$

$$\text{s.t.} \quad \nu > 0 \tag{11}$$
$$(6), (7).$$

However, the above **P0′** does not comply with the structure of dynamic programming, since the objective includes nonlinear function of average of variable $x_n(t)$ for infinite time. Therefore, the **P0′** is transformed into **P2** with an auxiliary variable as below [12]. We can show that an optimal solution of **P2** is also an optimal solution of **P0′**, which is similar to the proof shown in [10].

$$\textbf{P2:} \max_{a(t)} \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\sum_{n\in\mathcal{N}} \left(-\mathbb{E}_\pi\left[q_n(t+1)\right] + \nu\mathbb{E}_\pi\left[U_n(y_n(t))\right]\right) \tag{12}$$

$$\text{s.t.} \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}_\pi\left[y_n(t)\right] \le x_n \tag{13}$$
$$(6), (7), (11).$$

Proposition III.2. *An optimal solution of* **P2** *is an optimal solution of* **P0′**.

Proof. Please refer to the technical report [13]. □

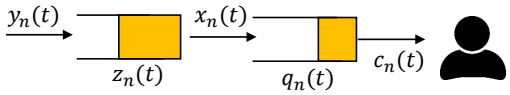

Figure 2: auxiliary variable and virtual queue

To solve **P2**, we introduce virtual queues $z(t) = (z_n(t))_{n\in\mathcal{N}}$ as [3] for auxiliary variable $y_n(t)$ as Figure 2 to ensure that (13) is satisfied [3]. We assume that $z_n(0) = 0, \forall n \in \mathcal{N}$ and $y(t)$ is finite for all t. Then the queuing dynamics of $z_n(t)$ is written as below.

$$z_n(t+1) = \max[z_n(t) - x_n(t), 0] + y_n(t), \forall n \in \mathcal{N}. \tag{14}$$

The virtual queue $z_n(t)$ can be viewed as a queue with data admitted $y_n(t)$ and served $x_n(t)$ and by making $z(t)$ mean rate stable, (13) is satisfied [3]. With **P2**, we formulate the original problem **P0′** as below. Again, the mean rate stability of $q(t)$ and $z(t)$ can be achieved by counting negative Lyapunov drift term to the objective. This is shown in Proposition III.3. Note that the mean rate stability of $q(t)$ still holds without the drift term, which is added for the similar reason as **P1**, since the average length of $q(t)$ is included in the objective.

$$\textbf{P3:} \max_{\pi} \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}_\pi[\sum_{n\in\mathcal{N}} -q_n(t+1) + \nu_2 U_n(y_n(t)) - \nu_3(q_n(t+1)^2 - q_n(t)^2) - \nu_4(z_n(t+1)^2 - z_n(t)^2)] \tag{15}$$

$$\text{s.t.} \quad \nu_2, \nu_3, \nu_4 > 0 \tag{16}$$
$$(7).$$

Proposition III.3. *An optimal policy of* **P3**
1) *makes all queues in network mean rate stable.*
2) *is Pareto-optimal with respect to the average queue length and utility.*

Proof. Please refer to the technical report [13]. □

By Proposition III.3, we can show that an optimal solution of **P3** is also an optimal solution of **P0′**. Note that by increasing weight ν, we can find a policy with the minimum delay among policies with utility differs from the optimal utility by the same amount of value as minimum drift-plus-penalty algorithm [3]. Similar to the inner capacity case, we can transform the **P3** defined by average reward into dynamic programming problem with discounted reward as below.

$$\textbf{P3}': \max_{\pi} \lim_{T\to\infty} \sum_{t=0}^{T-1} \alpha^{t-1}\mathbb{E}_\pi[\sum_{n\in\mathcal{N}} -q_n(t+1) + \nu_2 U_n(y_n(t)) - \nu_3(q_n(t+1)^2 - q_n(t)^2) - \nu_4(z_n(t+1)^2 - z_n(t)^2)]$$
$$\text{s.t.} \quad (7), (16).$$

IV. Algorithm Design

A. W-function

In this section, we introduce a novel reinforcement learning algorithm to solve the MDP problem presented in Section III based on a new action-value function called W-function. Before that, we define the reward function $r(s_t, a_t)$ as

$$r(s_t, a_t) = \sum_{n\in N} -q_n(t+1) - \nu_1(q_n(t+1)^2 - q_n(t)^2) \tag{17}$$

for **P2** and

$$r(s_t, a_t) = \sum_{n\in\mathcal{N}} -q_n(t+1) + \nu_2 \log\left(y_n(t)\right) - \nu_3(q_n(t+1)^2 - q_n(t)^2) - \nu_4(z_n(t+1)^2 - z_n(t)^2) \tag{18}$$

for **P3**.

One of long-standing challenges of learning-based algorithms is that they generally present poor performance during learning period. This is because at the beginning of learning, the action-value function is initialized with random values. Hence, before observing the reward and following state by taking a certain action at certain state, the action-value function for that state and action remains at random values. Our goal is to solve the problem of poor performance during learning

process by utilizing a baseline algorithm (e.g., max-weight algorithm) as a base policy and changing to a learning-based policy after a sufficient learning period. To this end, we present a new action-value function that is a variant of Q-function, which can be written in (2). Our motivation is that the immediate reward from a given state and action is a value that can be calculated immediately, which does not have to be learned. Instead, we define a new action-value function $W_\pi(s,a)$ which is an expected sum of discounted future rewards except for an immediate reward following policy π given that the current state is s and action a is taken.

Definition IV.1. *Given a policy π, the W-function $W : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is defined as*

$$W_\pi(s,a) = \sum_{s'} P^a_{ss'} Q_\pi(s', \pi(s')) \tag{19}$$

From the definition, we can derive the Bellman equation for $W_\pi(s,a)$ as

$$W_\pi(s,a) = \sum_{s'} P^a_{ss'} [r(s', \pi(s')) + \alpha W_\pi(s', \pi(s'))] \tag{20}$$

Furthermore, we can represent the optimal W-function $W^*(s,a)$ as

$$\begin{aligned} W^*(s,a) &= \sum_{s'} P^a_{ss'} \max_{a'} Q^*(s',a') \\ &= \sum_{s'} P^a_{ss'} \max_{a'} [r(s',a') + \alpha W^*(s',a')]. \end{aligned}$$

Intuitively, although learning with the Q-function and W-function may converge to the same point, we can expect that during the learning process, learning with W-function surpass the performance of the Q-function, since the W-function excludes the immediate reward, minimizing the uncertainty to be learned. In Section IV-B, we will introduce a learning algorithm using the W-function called W-learning, which is a variant of the Q-learning and in Section V we will show that under the W-learning, we can use other baseline algorithms as a base policy during the learning process. That is, for the unvisited states, which have never been experienced, the control action is determined based on the baseline algorithm.

B. W-learning

With the new action-value function defined in the previous section, we now propose a learning algorithm called W-learning to find the optimal W-function, which is similar to the Q-learning algorithm. Then we propose a practical version of W-learning called W-learning-approx where W-function is approximated with parameter ω, i.e., $W(s,a) \approx W(s,a;\omega)$. Algorithm 1 elaborates on the W-learning algorithm. While Q-learning initializes the Q-function with random value, W-function is initialized with 0, to utilize a baseline algorithm as an initial policy, which will be addressed in detail in Section V. The rest of the algorithm is similar to the Q-learning algorithm. Note that the W-function is updated using the temporal difference (TD) method.

Algorithm 1: W-learning

```
Initialize W(s,a) = 0, ∀s ∈ S, a ∈ A.
Initialize s
while for each step of episode do
    Given state s
    With probability ε choose random action a
    otherwise Select a = max_a[r(s,a) + αW(s,a)]
    Execute action a and observe next state s'
    W(s,a) ← W(s,a) + β[max_a' [r(s',a') + αW(s,a')] − W(s,a)]
    s ← s'
end
```

$$W(s,a) \leftarrow W(s,a) + \beta[\max_{a'} [r(s',a') + \alpha W(s,a')] - W(s,a)] \tag{21}$$

However, since the state and action space in the real world is huge, it is almost impossible to represent the actual $W(s,a)$ value with a tabular method. Therefore, the action-value function is often estimated with a function approximator. which may be either linear or nonlinear such as deep neural network [22]. Therefore, we propose an approximated version of W-learning similar to deep-Q-learning [22] to implement it in practice. W-learning algorithm learns W-function instead of Q-function. W-function is approximated with parameter ω. The full algorithm is presented in Algorithm 2.

Algorithm 2: W-learning-approx

```
Initialize target network update period K
Initialize replay memory D to capacity |D|
Initialize W(s,a;φ) = 0, W(s,a;ω) = 0, ∀s ∈ S, a ∈ A
for episode i = 1, ··· , I do
    Given state s_i
    With probability ε choose random action a_i
    otherwise select a_i = max_a[r(s_i,a) + αW(s_i,a;φ)]
    Execute action a_i and observe next state s_{i+1}
    Store transition (s_i, a_i, s_{i+1}) in D
    if more than |M| samples are collected then
        Sample random mini-batch M of transition
          (s,a,s') from D
        Set for (s,a,s') ∈ M
        calculate target value
        y_(s,a,s') = max_a[r(s',a) + αW(s',a;φ)]
        Perform gradient descent on
          Σ_(s,a,s',y)∈M (y − W(s,a;ω))^2.
        if mod (i,K) = 0 then
            φ = ω.
        end
    end
end
```

Note that we assume a general function approximator parametrized by ω which may be either linear or non-linear

function. For each time slot, a control action is taken according to the Q-function (line 6). As [23], the technique called experience replay is used to efficiently utilize the collected samples and eliminate correlations between samples. After taking an action, the agent observes the next state s_{i+1} and store samples (s_i, a_{i+1}, s_{i+1}) to the replay buffer (line 8). Note that unlike [23], W-learning does not have to store reward, since the immediate reward can be calculated with a given state. After more than $|M|$ samples are collected, W-function is updated in a way similar to Q-learning with a mini batch with size $|M|$ (lines 10-14). We also used the fixed target technique as [22] that hold the target W-function parameterized with ϕ and updates the target every K steps (lines 15-17).

V. Theoretical Analysis

In this section, we first show that the W-learning algorithm converges to an optimal solution of **P1′** and **P3′**. After that, we elaborate on the property of W-learning that guarantees the performance as good as baseline algorithms (minimum-drift algorithm for the inner capacity region and the minimum-drift-plus-penalty algorithm for the outer capacity region) during the learning period introduced in Section IV.

To show the convergence of W-learning, we follow the steps of [24] showing the convergence property of Q-learning, starting from defining an operator for value iteration as below:

Definition V.1. *An operator* Ψ:$\mathbb{R}^{|\mathcal{S}\times\mathcal{A}|} \mapsto \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$ *is defined as*

$$\Psi W(s,a) = \sum_{s'} P^a_{ss'} \max_{a'} [r(s',a') + \alpha W(s',a')] \qquad (22)$$

With the new operator Ψ, the update rule (21) can be written as

$$W(s,a) \leftarrow W(s,a) + \beta[\Psi W(s,a) - W(s,a)].$$

Similar to [25], the convergence property of W-learning can be shown using the contraction mapping property of the value iteration operator Ψ, which is given by Theorem V.2.

Theorem V.2. *W-learning algorithm converges to an optimal solution of* **P1′** *and* **P3′**.

Proof. Please refer to the technical report [13]. □

Now we consider the control policy of W-learning for the unvisited states, which determines the performance during the learning period.

Proposition V.3. *Under W-learning, for the unvisited states* $s \in \mathcal{S}$*, we have*

$$\pi(s) = \max_a r(s,a)$$

Proof. Please refer to the technical report [13]. □

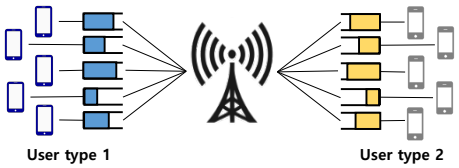

Figure 3: Simulation setup : Cellular downlink scenario

Therefore, with reward (17) and (18), if we let $\nu_1, \nu_2, \nu_3, \nu_4 \gg 1$, we have

$$\pi(s) \approx \max_a \sum_{n\in\mathcal{N}} -\nu_1(q_n(t+1)^2 - q_n(t)^2),$$

which follows the minimum-drift algorithm(max-weight algorithm) [3] and

$$\pi(s) \approx \max_a \sum_{n\in\mathcal{N}} [-\nu_2 U_n(y_n(t) - \nu_3(q_n(t+1)^2 - q_n(t)^2) - \nu_4(z_n(t+1)^2 - z_n(t)^2)],$$

which follows the minimum-drift-plus-penalty [3] for the unvisited states $s \in \mathcal{S}$, respectively. Note that Q-learning takes a random action for the unvisited states since $Q(s,a)$ is initialized with a random value. In order to utilize a baseline policy as an initial policy, one requires the value of Q-function for a policy (e.g., the max-weight algorithm), which is much more complex and takes a much longer time than our W-learning algorithm.

VI. Simulation

In this section, we present simulation results of our W-learning algorithm. We first describe our simulation setting and show the detailed performance of the simulation.

A. Simulation Setup

We consider a cellular downlink scenario as Figure 3 with 10 users $n \in \{1, 2, \cdots, 10\}$ associated to a single base station. We assume that there are two types of users, each with five users with homogeneous channel distribution and arrival rate. For each type, the channel state is i.i.d. and follows the following Bernoulli-type distributions, respectively:

$$c^1(t) = \begin{cases} 0, & \text{with probability } 0.8 \\ 1, & \text{with probability } 0.2 \end{cases} \qquad (23)$$

for user type 1, and

$$c^2(t) = \begin{cases} 1, & \text{with probability } 0.4 \\ 2, & \text{with probability } 0.6 \end{cases} \qquad (24)$$

for user type 2.

We assume that the size of arrival data follows Weibull distribution since the data arrival occurs in batches in practice. Figure 4 shows the data arrival of We used $k = 0.4$ as [26], and adjusted λ according to the mean arrival rate of each user type. At each time slot t, the base station observes current state and chooses one user to serve. For W-learning-approx algorithm, W-function is estimated with a linear function approximator, which is linear combination of hand-crafted features.

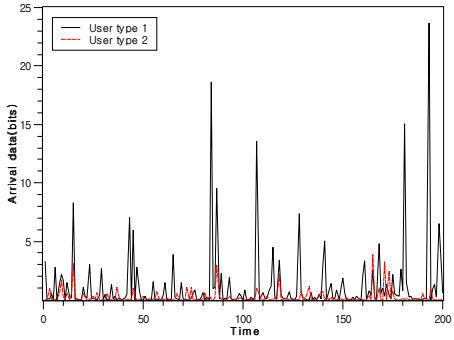

Figure 4: Arrival rate

B. Performance Analysis

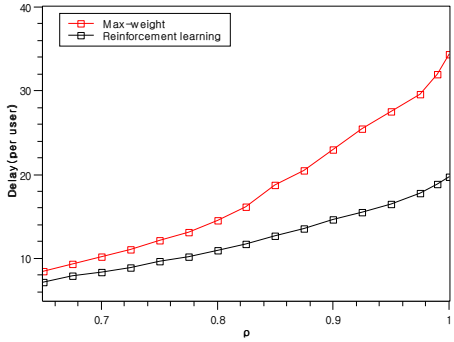

Figure 5: Delay performance the inner capacity region

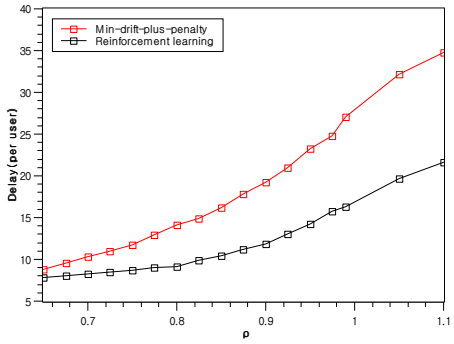

Figure 6: Delay performance for general case

In this section, we present the simulation scenario then analyze the detailed performance of the simulations. First, we evaluate the delay performance of W-learning algorithm by changing ρ, which represents a measure of distance between the arrival rate and capacity region boundary [3]. Figure 5 compares the delay performance of the W-learning algorithm(denoted as Reinforcement learning) with reward (17) and max-weight algorithm for the inner capacity region. In a similar way, Figure 6 compares the delay performance of the W-learning algorithm(denoted as Reinforcement learning) with reward (18) and min-drift-plus-penalty [3] algorithm for both the inner and outer capacity region. We adjusted the value

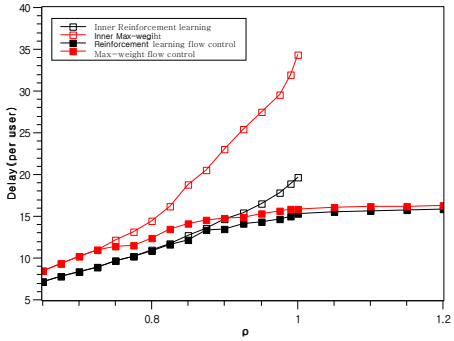

Figure 7: Delay performance for outer capacity region with given flow controller

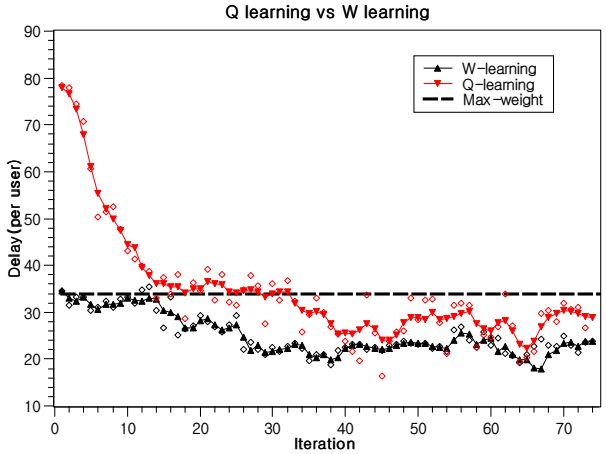

Figure 8: Delay performance of W-learning and Q-learning

of ν_2 in (18) so that the sum of utility for both algorithms are equivalent. As shown in Figure 5, the delay of both algorithms do not diverge to infinity when $\rho = 1$, which means that both achieve maximal throughput since they guarantee network stability. However, the delay of max-weight algorithm starts to increase abruptly as the arrival rate approaches capacity region boundary, while our algorithm shows up to 40.8% delay reduction compared to the max-weight algorithm. Figure 6 also shows up to 37.9% delay reduction compared to the min-drift-plus-penalty algorithm.

Figure 7 also shows the delay performance of max-weight and W-learning algorithm measured with arrival rate at both inner and outer capacity region. However, in this case, we assume that we use the W-learning or max-weight algorithm for scheduling and a existing flow controller such that limits the arrival based on current queue length [27]. The flow controller is adopted to handle arrival data outside of the capacity region. However, since the actual capacity region is unknown, the flow controller can only observe the current queue backlog. As a result, it can be activated when the queue backlog becomes large due to the burst arrival traffic. As it is shown in Figure 7, the flow controller starts to operate for ρ smaller than 1 for both max-weight and W-learning algorithm,

but much smaller ρ for the max-weight algorithm. In the long term, both algorithms can serve all arrival traffic inside the capacity region without the flow controller. Therefore, the activation of the flow controller for ρ smaller than 1 results in throughput loss, and the loss would be much larger for the max-weight algorithm.

Finally, we compared the delay performance during the learning period for Q-learning and W-learning. As Figure 8 shows, the delay performance of W-learning does not exceed that of the max-weight algorithm. However, since Q-function is initialized with a random parameter for Q-learning, the delay performance during the learning period is even worse than the max-weight algorithm.

VII. Conclusion

In this paper, we reformulated the network optimization problem into an MDP problem to solve it through dynamic programming. Thereby, the problem of high average delay of the existing network resource management algorithms can be addressed by exploiting the future information. To solve the curse of dimensionality and uncertainty of dynamic programming, we presented a new action-value function and a reinforcement learning algorithm that ensures the minimal performance loss during the learning process. Finally, via simulation we showed that our algorithm showed delay reduction after learning compared to max-weight algorithm while maintaining performance as good as max-weight algorithm during the learning process.

VIII. Acknowledgment

This work was supported by the ICT R&D program of MSICT/IITP. [2017-0-00045, Hyper-connected Intelligent Infrastructure Technology Development], the Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921), and Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT)(No.2016-0-00563, Research on Adaptive Machine Learning Technology Development for Intelligent Autonomous Digital Companion). Also this research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(No. 2018R1D1A1B07045181).

References

[1] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE transactions on automatic control*, vol. 37, no. 12, pp. 1936–1948, 1992.

[2] ——, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Transactions on Information Theory*, vol. 39, no. 2, pp. 466–478, 1993.

[3] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[4] C. Jiang, H. Zhang, Y. Ren, Z. Han, K.-C. Chen, and L. Hanzo, "Machine learning paradigms for next-generation wireless networks," *IEEE Wireless Communications*, vol. 24, no. 2, pp. 98–105, 2017.

[5] M. Chen, W. Saad, and C. Yin, "Virtual reality over wireless networks: Quality-of-service model and learning-based resource management," *IEEE Transactions on Communications*, vol. 66, no. 11, pp. 5621–5635, 2018.

[6] U. Challita, W. Saad, and C. Bettstetter, "Cellular-connected uavs over 5g: Deep reinforcement learning for interference management," *arXiv preprint arXiv:1801.05500*, 2018.

[7] Z. Xu, Y. Wang, J. Tang, J. Wang, and M. C. Gursoy, "A deep reinforcement learning based framework for power-efficient resource allocation in cloud rans," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–6.

[8] Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C. H. Liu, and D. Yang, "Experience-driven networking: A deep reinforcement learning based approach," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 1871–1879.

[9] Y. He, Z. Zhang, F. R. Yu, N. Zhao, H. Yin, V. C. Leung, and Y. Zhang, "Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 11, pp. 10 433–10 445, 2017.

[10] L. Georgiadis, M. J. Neely, L. Tassiulas *et al.*, "Resource allocation and cross-layer control in wireless networks," *Foundations and Trends® in Networking*, vol. 1, no. 1, pp. 1–144, 2006.

[11] J. D. Little, "A proof for the queuing formula: L= λ w," *Operations research*, vol. 9, no. 3, pp. 383–387, 1961.

[12] M. J. Neely, "Delay-based network utility maximization," *IEEE/ACM Transactions on Networking (TON)*, vol. 21, no. 1, pp. 41–54, 2013.

[13] J. Bae, J. Lee, and S. Chong. Beyond max-weight scheduling: A reinforcement learning approach. [Online]. Available: http://netsys.kaist.ac.kr/publication/papers/beyondmw.pdf

[14] S. P. Singh, "Reinforcement learning algorithms for average-payoff markovian decision processes," in *AAAI*, vol. 94, 1994, pp. 700–705.

[15] S. Yang, Y. Gao, B. An, H. Wang, and X. Chen, "Efficient average reward reinforcement learning using constant shifting values." in *AAAI*, 2016, pp. 2258–2264.

[16] E. A. Feinberg, P. O. Kasyanov, and N. V. Zadoianchuk, "Average cost markov decision processes with weakly continuous transition probabilities," *Mathematics of Operations Research*, vol. 37, no. 4, pp. 591–607, 2012.

[17] D. Blackwell, "Discrete dynamic programming," *The Annals of Mathematical Statistics*, pp. 719–726, 1962.

[18] M. Schäl, "Average optimality in dynamic programming with general state space," *Mathematics of Operations Research*, vol. 18, no. 1, pp. 163–172, 1993.

[19] L. I. Sennott, *Stochastic dynamic programming and the control of queueing systems*. John Wiley & Sons, 2009, vol. 504.

[20] O. Hernández-Lerma and J. B. Lasserre, *Discrete-time Markov control processes: basic optimality criteria*. Springer Science & Business Media, 2012, vol. 30.

[21] E. Altman, *Constrained Markov decision processes*. CRC Press, 1999, vol. 7.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[24] T. Jaakkola, M. I. Jordan, and S. P. Singh, "Convergence of stochastic iterative dynamic programming algorithms," in *Advances in neural information processing systems*, 1994, pp. 703–710.

[25] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[26] J. Lee, K. Lee, Y. Kim, and S. Chong, "Carriermix: How much can user-side carrier mixing help?" *IEEE Transactions on Mobile Computing*, vol. 16, no. 1, pp. 16–29, 2017.

[27] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control," *IEEE/ACM Transactions on Networking (TON)*, vol. 15, no. 6, pp. 1333–1344, 2007.