IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 20, 2020, accepted June 25, 2020, date of publication June 30, 2020, date of current version July 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3005885

Deep Q-Learning Based Optimization of VLC Systems With Dynamic Time-Division Multiplexing

UMAIR F. SIDDIQI[1], (Member, IEEE), SADIQ M. SAIT[1,2], (Senior Member, IEEE), AND MURAT UYSAL[3], (Fellow, IEEE)

[1]Center for Communications and IT Research, Research Institute, King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia
[2]Department of Computer Engineering, King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia
[3]Department of Electrical and Electronics Engineering, Ozyegin University, 34794 Istanbul, Turkey

Corresponding author: Sadiq M. Sait (sadiq@kfupm.edu.sa)

This work was supported by the King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia.

ABSTRACT The traditional method to solve nondeterministic-polynomial-time (NP)-hard optimization problems is to apply meta-heuristic algorithms. In contrast, Deep Q Learning (DQL) uses memory of experience and deep neural network (DNN) to choose steps and progress towards solving the problem. The dynamic time-division multiple access (DTDMA) scheme is a viable transmission method in visible light communication (VLC) systems. In DTDMA systems, the time-slots of the users are adjusted to maximize the spectral efficiency (SE) of the system. The users in a VLC network have different channel gains because of their physical locations, and the use of variable time-slots can improve the system performance. In this work, we propose a Markov decision process (MDP) model of the DTDMA-based VLC system. The MDP model integrates into deep Q learning (DQL) and provides information to it according to the behavior of the VLC system and the objective to maximize the SE. When we use the proposed MDP model in deep Q learning with experienced replay algorithm, we provide the light emitting diode (LED)-based transmitter an autonomy to solve the problem so it can adjust the time-slots of users using the data collected by device in the past. The proposed model includes definitions of the state, actions, and rewards based on the specific characteristics of the problem. Simulations show that the performance of the proposed DQL method can produce results that are competitive to the well-known metaheuristic algorithms, such as Simulated Annealing and Tabu search algorithms.

INDEX TERMS Deep Q learning, deep reinforcement learning, dynamic time division multiple access, visible light communications, optimization, non-deterministic algorithms.

I. INTRODUCTION

Reinforcement learning (RL), a branch of artificial Intelligence (AI), deals with the development of self-learning intelligent agents that can learn to solve specific tasks [1] by taking a correct sequence of actions without any pre-trained policy. In an reinforcement learning (RL) system, an agent applies actions on the environment. The environment changes its state and returns a reward value in response of each action applied to it. The reward is a numerical feedback that tells the agent about the quality of its last action. The goal of the agent is to learn a policy that contains for each state of the environment, the action which can maximize the cumulative future reward (i.e., reward in long run). The conventional RL uses only tables/memory to store the history, and becomes infeasible when the number of states and actions becomes large. Deep learning (DL) [2] provides a handy solution by approximating the cumulative future rewards of the state action pairs using a deep neural networks (DNN) of many layers.

The algorithms of RL that use the experience to predict the best actions when the model of the environment is unknown are termed as Temporal difference (TD) learning [3]. Q-learning [4] is a very popular model-free temporal difference (TD)-based method that uses a table to store the Q-values of all possible state-action pairs. The Q-value

The associate editor coordinating the review of this manuscript and approving it for publication was Mostafa M. Fouda.

$Q(s, a)$ of a pair of state (s) and action (a) is equal to the expected cumulative future reward of the agent if it chooses the action a in state s, and the states until the end of episode follow a given policy [3]. Deep Q-learning (DQL) is a type of Q-learning that uses deep learning (DL) to approximate the Q-values [4].

Deep Q-networks (DQN) gained popularity when they were trained to learn to play Atari2600 games. They played better than humans in some cases [5]. Some recent applications of Q-learning include: (i) autonomous control of a power network; (ii) autonomous determination of congestion-free paths in global routing [6]; (iii) dynamic path planning [7]; and, (iv) path planning in grid graphs [8].

In visible light communications (VLC) systems, the light emitting diodes (LED)s serve the dual purpose of providing illumination and serving as data-access points. The use of VLC systems is rapidly growing, and the use of reinforcement learning techniques can significantly improve the performance of many vital components such as LEDs and encoders/decoders. Recently, Lee *et al.* applied DL to design a transceiver for VLC systems [9] that use the on-off keying (OOK) method to transmit messages. In a OOK system, the message is first encoded into a binary vector and then transmitted through an LED. The number of ones in the binary vector control the intensity level. DL helped in finding optimal binary encoding for messages to meet the requirements on the intensity level and signal quality.

An LED can serve multiple users by using multiple access schemes such as orthogonal frequency division multiple access (OFDMA) or time-division multiple access (TDMA). OFDMA is the most popular multiple access technique in VLC systems, but it has a peak-to-average power ratio (PAPR) problem that limits its usefulness [10]. Recently, Abdelhady *et al.* proposed dynamic time-division multiple access (DTDMA) which has excellent resource utilization feature and is efficient in satisfying users' requirements [11]. In DTDMA, the duration and power level of the signal in any time slot are variable. They showed that DTDMA is very useful when a single LED needs to send data to several users [11].

In this work, we introduce a deep Q-learning (DQL)-based method for the LED transmitter that exploit the variability of duration of the time-slots of the DTDMA to maximize the overall spectral efficiency (SE) of the system. The main contributions of the proposed work are as follows.

1) The conventional approach of solving optimization problems is to apply (meta)-heuristic algorithms that provide step-wise instructions. The DQL method however, uses the data of the previous runs (or trials) to autonomously choose the steps that can eventually lead to the desired solution. To the best of our knowledge, this is the first work that applies the DQL method to optimize the duration of the time-slots in DTDMA-based VLC system.
2) We propose Markov decision process (MDP) model of the DTDMA-based VLC system, which can integrate into DQL and enable the LED to optimize the duration of the time-slots and improve the SE of the system.
3) The proposed MDP model contains the definitions of states, rewards, and actions. The state consists of the variables that represent the unique characteristics of the VLC system controlling the SE of the system. The set of actions presents the agent with many possible ways to alter the duration of the time-slots of the DTDMA system. The reward function helps the agent to improve the SE of the system and find a globally optimal solution.
4) We performed simulations and demonstrate that the solution quality of the DQL approach with the proposed MDP model is competitive to two well-known meta-heuristic algorithms: simulated annealing (SA) and tabu search (TS). Both these algorithms have already been applied to solve different optimization problems in the realm of VLC systems [12], [13].

The organization of this article is as follows. The second section shows some of the most relevant previous work. In Sections III and IV, we discuss the VLC system model and a brief introduction of the DQL method. Section V contains the details of the proposed model. In section VI, we discuss the simulation results and their analysis, and finally, the article finishes with a conclusion and future work.

II. RELATED WORK

In the recent past, many researchers have applied RL to solve several engineering problems. In this section, we present a brief discussion on the main components (i.e., state representation, actions, and reward functions) of the MDP models used in those works. The design of MDP is critical for the efficient application of RL.

Liang *et al.* applied the double dueling deep Q networks (3DQN) to improve the waiting times of vehicles at a four-way intersection controlled by a traffic light [14]. They mapped the intersection onto a rectangular grid where each grid cell either occupies a vehicle or remains empty. They also mapped the speed of the vehicles to another rectangular grid with the same orientation as the previous one. These two grids are collectively represented as matrices and indicate the state of the system. The action set consists of an increment or decrement of 5s in the green-light timings of any of the two directions at the intersection. The reward of an action is defined as the difference between the total waiting time of vehicles before and after applying that action.

Liao *et al.* applied the DQL to solve the 3D global routing problem [6]. Their proposed approach first decomposes the multiple pin nets into two pin subnets and then routes the subnets on a weighted 3D mesh. The weight of edges denote the available capacity of those edges. A single DQN network iteratively routes all two-pin subnets. The state is a 12 dimension vector in which the first three indicate the x,y,z coordinates of the current location of the agent, and the next three coordinates indicate the distance of the target from the current location along the x, y, and z directions. The next

six coordinates indicate the capacity values of edges that connect to the current location of the agent from any direction. The action space is the six directions ($+x$, $-x$, $+y$, $-y$, $+z$, $-z$) in the 3D mesh in which the agent can move. The reward function is also simple, and the agent receives a reward of $+100$ upon reaching a state, which is the target pin of the currently chosen subnet and a reward of -1 otherwise.

Mocanu *et al.* solved the multi-objective power optimization problem using Deep Q-networks (DQN) [15]. They optimize power by controlling three critical devices: air conditioner (AC), dishwasher, and electric vehicle (EV). The state vector consists of 11 elements and contains information such as time step, baseload, photovoltaic (PV) resource, AC state, EV state, and dishwasher state. The number of states could be huge because the attributes have continuous values. The DQN employs a reward vector of three components in which each component corresponds to an objective of the optimization problem. The Q-value also has three components because the reward value have three components. The action set consists of all possibilities of changing the state of the three devices. The action gets a positive reward if it changes the state toward the goal, and a negative reward otherwise. The DQN has eight outputs that correspond to the possible on-off combinations of the three devices (AC, EV, and dishwasher).

Demiral *et al.* addressed the problem when multiple independent control systems need to communicate over a shared communication resource [16]. They applied DQL to schedule the multiple control systems to use a shared limited communication resource. The goal of the DQL method is to minimize the loss due to delays in communications. The main features of their MDP model are as follows: (i) The state consists of all error values of all control systems at a given time; (ii) The action space consists of the allocation of a subset of control systems to use the communication resource at a given time; and, (iii) The reward function is equal to the negative summation of the error values of the control systems.

Wu *et al.* suggested that the unmanned aerial vehicle (UAV) problem of finding a target is equivalent to a snake game in which a snake needs to find a target in a 2D plane [17]. They applied DQL to find and enable the snake to find the target autonomously. The screen-shot of the game screen at any particular time step serves as the state. To keep the size of the state small, they down-sampled the original image into 80×80. The set of actions consists of the snake's movement in the four directions ($+$x, $-$x, $+$y, $-$y). The value of the reward lies between -100 to $+100$. The reward value is $+100$ when the snake reaches the target. Otherwise, it is chosen based on the proximity of the position of the snake from the target.

The MDP models mentioned above have two limitations for their application to solving the optimization problems with large search space: (i) The reward functions either use constants or parameters whose values should be determined through trial-and-error; and (ii) They do not implement the hill-climbing feature of the optimization methods that enable the search to skip local optimal solutions and find the globally optimal ones. The reward function in our proposed work is free from parameters and is equal to the ratios of the objective function values or the degree of violation of constraints in the current and next states. The reward function in our proposed model also ensures that the search does not get trapped into local optima. The MDP models are generally problem-specific. The MDP models' problem-specific nature has a disadvantage that a change in the target problem requires a redefinition of the MDP model.

III. VISIBLE LIGHT COMMUNICATIONS SYSTEM

For the convenience of readers, we have described the symbols and notations repeatedly used in this article in Table 1.

A. OVERVIEW

In this work, we consider a VLC system that has a single LED transmitter and multiple users. The LED employs the DTDMA method [11] to serve numerous users. In the standard TDMA, each user consumes the entire bandwidth within its fixed time-slot. In contrast, the DTDMA has adjustable time-slots which give the system flexibility to change the duration of the time-slots to improve the system performance. As an example, Fig. 1 illustrates a VLC system in which the LED transmits data to up-to four users (u_0,u_1,u_2,u_3), and the channel between the users and LED are indicated by h_i ($i = 0$ to 3) for users u_0 to u_3, respectively. The VLC networks usually contain mobile users that can change their positions. A change in the position causes a change in the channel condition between the LED and user. Fig. 2 shows the structure of the signal in both TDMA and DTDMA. In both these types, the signal is divided into time slots that are assigned to different users. In TDMA, the duration of the time slots is fixed, whereas, in DTDMA, the duration of time slots is variable, as shown in Fig. 2(b). DTDMA has better resource utilization and data-rates as compared to the conventional TDMA [11]. In DTDMA, the intensity of

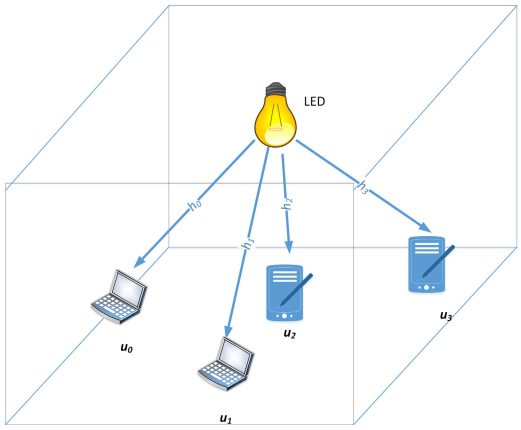


FIGURE 1. Illustration of a VLC system in which LED uses the TDMA method to transmit data to multiple users.

TABLE 1. Description of the frequently used symbols and notations.

Symbol	Description
	VLC network
U	A set that contains all users
u_i	A user that belongs to U
K	Number of users in the VLC network
η_{eo}	Electrical-to-optical conversion efficiency
η_{oe}	Optical-to-electrical conversion efficiency
h_i	Gain of the channel that exists between the LED and the user $u_i \in U$
ϕ_a	Semiangle at half-power of the LED
ψ_a	field-of-view (FOV) of the PD
ψ_i	Angle between the incident light and normal to the user's PD which is placed horizontally facing upwards
σ_n	Variance of the additive white Gaussian noise (AWGN)
γ_i	The channel-to-noise ratio of the user u_i
s_L	Intensity of current at the LED transmitter
s_i	Current intensity received at the PD of user u_i
I	Expected value of s_L
d_i	Distance of the user u_i from the LED
τ_i	Duration of the time-slot allocated to user u_i
$\tau_{\min}$	Minimum duration of the time-slot of any user
$Z_i(\tau_i)$	The data-rate of the user u_i when the duration of its time-slot is τ_i
Z_{th}	Minimum required data-rate for any user
B_v	Bandwidth of the LED
	DQL algorithm
$\mathcal{S}$	State space
$\mathcal{A}$	Action space
$\mathcal{P}$	State transition function
$\mathcal{R}$	Reward function.
γ	Discount factor
s_t	State of the MDP at time step t
a_t	Action chosen by the agent at time step t
r_t	Immediate reward at time step t
$Q(s,a)$	Q-Value function, where $s \in \mathcal{S}$, and $a \in \mathcal{A}$
$Q(s,a;\theta)$	Approximation of Q-value function using the Q-network with parameters θ
$Q(s,a;\theta")$	Approximation of Q-value function using the target network with parameters θ"
α	Learning rate of the gradient descent algorithm
D_M	Replay memory of size m_r transactions
M	Number of episodes
T	Maximum possible steps in any episode
C	Number of steps between successive update of θ" to θ
ϵ_0	Initial value of the parameter ϵ in the ϵ- algorithm
ϵ_δ	A unit decrement in the ϵ value
$\epsilon_{\min}$	Final value of the parameter ϵ
B	Batch size
	Proposed Model
Δ_I	Termination criterion of an episode
δ	Unit change in the duration of time-slot
D	# of hidden layers in the DNN
W	# of neurons in the any hidden layer of DNN

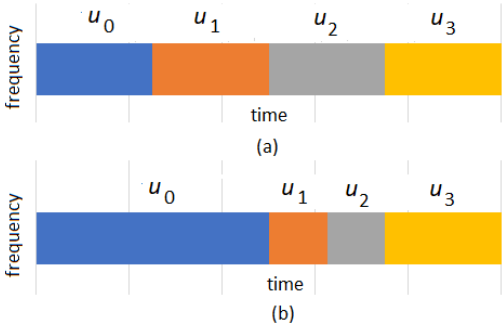


FIGURE 2. Structure of the frame of (a) conventional TDMA and (b) dynamic TDMA.

current at the LED can also be adjusted to improve the system performance further.

B. SIGNAL MODEL

Consider a system that has one LED and up-to K users ($U = \{u_o, u_1, \ldots, u_{K-1}\}$). The LED serves the users by employing the DTDMA technique. The transmitter's circuit encodes the data streams of the users by varying the excitation current of the LED while making sure to encode the data stream of each user within its time slot. Fig. 3 shows a simple schematic diagram in which the LED transmits a signal to user u_i. The descriptions of notations used in the figure are as follows: (i) s_L is the current intensity used to transmit a symbol from the LED; (ii) η_{eo} indicates the electrical-to-optical conversion efficiency of the LED; (iii) h_i indicates the transfer function or gain of the channel that exists between u_i and the LED; (iv) η_{oe} is the optical-to-electrical energy conversion efficiency of the photo-diode of u_i; and, (v) s_i is

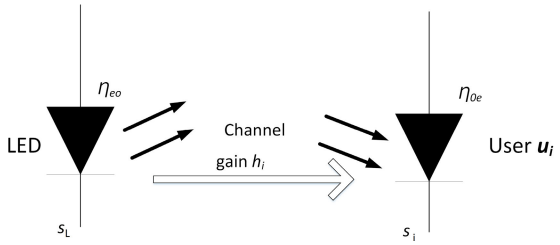


FIGURE 3. A simplified signal model of the VLC system.

the current intensity received at the user u_i, and its value is equal to $s_i = \eta_{oe}\eta_{eo}h_i s_L$. The user also receives noise from the environment which is denoted by n_i and is equal to additive white Gaussian noise (AWGN). The value of channel gain (h_i) can be determined as follows:

$$h_i = \frac{(m+1)A_p}{2\pi d_i^2}\cos^{m+1}(\psi_i)\text{rect}(\frac{\psi_i}{\psi_a}) \tag{1}$$

where m is the order of Lambertian emission and is equal to $\frac{-1}{\log_2(\cos(\phi_a))}$, ϕ_a is the semi-angle at half-power of the LED. d_i is the distance of u_i from the LED, ψ_i is the angle between the incident light and normal to the user's photodiode (it is assumed that the photodiode is placed horizontally facing upwards), and ψ_a is the field of view of users' photodiode. Fig. 4 illustrates the angles that are used in the computation of the channel gain using (1). The function rect(x) is given by:

$$\text{rect}(x) = \begin{cases} 1 & \text{if } |x| \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

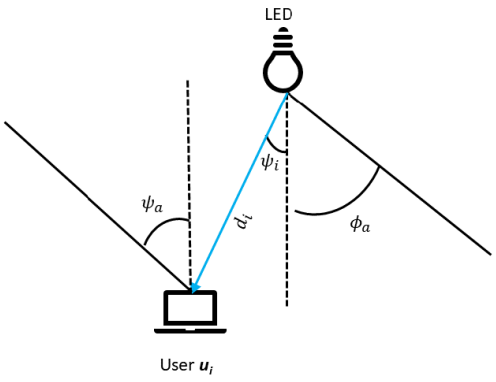


FIGURE 4. The LOS channel of the user u_i from the LED.

The system contains only one LED, therefore, it is free from interference, and the signal to noise ratio (SNR) values depends on the signal strength and noise, and SNR of the user u_i is denoted by γ_i and computed as follows [11]:

$$\gamma_i = \frac{e\eta_{eo}^2\eta_{oe}^2 h_i^2 I^2}{2\pi\sigma_n^2} \tag{3}$$

where $I = \mathbb{E}(s_L)$, i.e., is the average current intensity. The goal of the VLC system is to maximize the overall SE of the system, which is given as follows:

$$\eta_{\text{SE}} = \frac{1}{2}\sum_{i=0}^{K-1}\tau_i \log_2(1+\gamma_i) \tag{4}$$

The objective of optimization is to maximize η_{SE} and satisfy the following constraints:

$$\sum_{i=0}^{K-1}\tau_i = 1 \tag{5}$$

$$\tau_i \geq \tau_{\min}, \quad \forall i \tag{6}$$

$$Z_i(\tau_i) \geq Z_{\text{th}}, \quad \forall i \tag{7}$$

The first constraint indicates that the total duration of the time-slots of all users should be equal to one (Please note that 1 corresponds to 100% utilization). The duration of any time-slot (τ_i) can be changed in discrete steps of size $\pm\delta$, (where, $\delta \in \mathbb{R}^+$). The second constraint ensures that the duration of the time-slot of any user should not be less than a minimum value. The constraint in (7) indicates that the data-rate of the users should be greater than a given threshold. In constraint (7), the term $Z_i(\tau_i)$ denotes the data-rate of the user u_i when the duration of its time-slot is τ_i, and the value of Z_i should be greater than a given minimum value Z_{th}. A constraint on the minimum data-rate ensures that none of the the users faces outage. The value of $Z_i(\tau_i)$ can be computed as follows:

$$Z_i(\tau_i) = \frac{1}{2}B_v\tau_i log_2(1+\gamma_i) \tag{8}$$

We can denote the objective function as follows:

$$\mathcal{F}(h_0, h_1, \ldots, h_{K-1}, \tau_0, \tau_1, \ldots, \tau_{K-1}) \tag{9}$$

The objective function can be computed in two steps: (i) In the first step, we compute the SNR values of all users ($\gamma_0, \gamma_1, \ldots, \gamma_{K-1}$) using (3); and (ii) Compute the overall SE of the system (η_{SE}) using (4).

In this work, we assume that the unit or minimum change in the percentage of the duration of time-slots is equal to δ. The value of $\tau_{\min}$ is minimal, and we can approximate it to zero. The total number of possible solutions in the search space is given by $\binom{\frac{1}{\delta}+K-1}{K-1}$. When we assume a unit change equal to 1% i.e., $\delta = 0.01$, and the number of users (K) is equal to 10, then the search space contains a total of 4.26e12 possible solutions which is a very large number.

The class non-deterministic-polynomial-time (NP) contains problems that cannot be solved in polynomial-time, but a given solution can be verified in polynomial time. A problem is NP-hard if all problems in NP are reducible to it. The NP-hard problems are at-least as hard as every problem in NP. Therefore, an algorithm for solving the NP-hard problem can be transformed in polynomial-time to solve any NP problem as well [18]. The problem considered in our work is a hard non-convex optimization problem [11] and, therefore, belongs to the NP-class [19]. An exhaustive search is infeasible because of the large search space size. We should apply

heuristics to intelligently traverse the search space and find near-optimal solutions. A non-convex optimization problem could have multiple local minima. Therefore, the proposed work also incorporates hill-climbing to reach to the globally optimal solutions. The DQL method is model-free and can solve any problem without any knowledge of the structure of the problem using the data collected with the trial-and-error experiments.

IV. DEEP Q-LEARNING (DQL) ALGORITHM

In this section, we first briefly describe some concepts in RL and then the DQL algorithm.

A. REINFORCEMENT LEARNING (RL)

RL algorithms are an efficient tool to solve the decision problems of choosing a sequence of actions. The agent interacts with the environment to learn an optimal policy. The decision problem of choosing a sequence of actions to solve a task can be mathematically expressed using the MDP. The MDP is defined using a 5-tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}$, where, (i) $\mathcal{S}$ contains a set of finite states of the environment as observed by the agent; (ii) $\mathcal{A}$ contains the set of possible actions available for the agent to apply to the environment; (iii) $\mathcal{P}$ are the state transition probabilities with which the environment can change its state. The expression $P[s_t, s_{t+1}, a_t]$ denotes the probability of state transition from s_t to s_{t+1} through application of the action a_t; (iv) $\mathcal{R}$ denotes the rewards that the agent receives from the environment upon applying any action, and it is a continuous value bounded in an interval $[0, R_{\max}]$. The expression $R(s_t, a_t, s_{t+1})$ denotes the reward when the agent moves from s_t to s_{t+1} through application of the action a_t; and, finally, (v) γ denotes the discount factor and it is the weight of the rewards of the future states in the computation of the cumulative reward.

The goal of the agent in RL is to learn a policy that optimize a V-value function. A policy is denoted by $\pi(s, a) : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, where $\pi(s, a)$ is the probability of applying action a in state s. The set Π contains all possible policies. The V-value function is denoted by $V^{\pi}(s) : \mathcal{S} \rightarrow \mathbb{R}$ and can be determined using the following equation.

$$V^{\pi}(s) = E[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, \pi] \tag{10}$$

In (10), s_t, and s_{t+1} denote the states at time t and $t+1$, respectively. The variable r_t is equal to the expected reward of the agent and is given by, $r_t = \mathop{\mathbb{E}}_{a \sim \pi(s_t, .)} R(s_t, a, s_{t+1})$. We use expected value because the behavior of the environment is probabilistic. The optimal V-value function can be defined as follows:

$$V^*(s) = \max_{\pi \in \Pi} V^{\pi}(s) \tag{11}$$

Another important function in RL is the Q-value function $Q(s, a) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, i.e., it maps the pairs of state and actions to real numbers. It returns the future cumulative reward when the agent chooses the action a on state s. Mathematically,

$$\begin{aligned} Q(s, a) &= E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots | s_t = s, a_t = a, , \pi] \\ &= E[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, a_t = a, \pi] \end{aligned} \tag{12}$$

We represent the above equation using a recursive relationship with the help of Bellman optimality equation as follows:

$$\begin{aligned} Q^{\pi}(s_t, a_t) = \sum_{s_{t+1} \in S} P(s_t, a_t, s_{t+1})(R(s_t, a_t, s_{t+1}) \\ + \gamma Q^{\pi}(s_{t+1}, a_{t+1} = \pi(s_{t+1}))) \end{aligned} \tag{13}$$

The optimal Q-valued function ($Q^*(s, a)$) can be expressed using the following equation.

$$Q^*(s, a) = \max_{\pi \in \Pi} Q^{\pi}(s, a) \tag{14}$$

The above equation can be solved using dynamic programming when the number of states and actions are small. However, dynamic programming becomes infeasible for a large number of states and/or actions. Therefore, we need to search for the near optimal solutions.

B. DEEP Q-LEARNING (DQL)

In this sub-section, we briefly describe the DQL algorithm. The DQN is a DNN or a neural network whose function is to approximate the Q-value function. The DQL algorithm employs two DQNs, which are named as the Q-network and target-network. The weights of the Q-network and target-network are denoted by θ and θ'', respectively. The DQL algorithm also contains a replay memory (D_M), which is also known as experience replay. It also employs a gradient descent algorithm for training the weights of the Q-network, and an ϵ-greedy algorithm for choosing actions. The gradient descent is an optimization algorithm that minimizes a cost function by moving in the direction of steepest descent, and the size of each step is controlled by the parameter learning rate (α).

Algorithm 1 shows the DQL procedure. An agent can follow it to solve up-to M episodes of a given task. An episode refers to completely solving an independent instance of the problem. In our work, the problem is to determine the optimal duration of the time-slots in a DTDMA VLC system. Therefore, an episode finds the optimal duration of the time-slots for a particular configuration of users. The input of the DQL algorithm includes the following: (i) m_r is the size of the replay memory D_M; (ii) γ is the discount factor; (iii) α is the learning rate of the gradient-descent algorithm; (iv) (ϵ_0, ϵ_{min}, and ϵ_{δ}) are the parameters of the ϵ-greedy algorithm; (v) C denotes the number of iterations after which we update θ'' to θ; and finally, (vi) B which denotes the batch size and is a critical parameter in the training of DQN.

Lines 2-3 in Algorithm 1 perform the initialization. Both DQNs (Q-network and target-network) are identical in structure, and we initialize the weights to the Q-network to random

Algorithm 1: The DQL Algorithm

m_r: Size of the replay memory; γ: discount factor; α: learning rate; M: Number of episodes; T: Maximum iterations in an episode; $\{\epsilon_0, \epsilon_{\min}\epsilon_\delta\}$: Probability values; C: Number of steps between successive update of θ" to θ.; Initialize the replay memory D_M.;
Initialize the weights of Q-network (i.e. θ) with random values and that of target network (i.e., θ") with θ.;
set $\epsilon = \epsilon_0$;
for *i= 1,M* **do**
  t=0;
  while *episode does not terminate* **do**
    With a probability ϵ select a random action a_t;
    otherwise select $a_t = \arg\max_a Q(s_t, a_t; \theta)$. ;
    Apply action a_t to the environment and observe the immediate reward r_t and next state s_{t+1}. ;
    Store the transition (s_t, a_t, r_t, s_{t+1}) in D_M. ;
    Sample random mini-batch of B number of transitions from D_M. ;
    Set $y_t =$
    $\begin{cases} r_t \text{ if } s_{t+1} \text{ is the terminal state of the episode.} \\ r_t + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a'; \theta'') \text{ otherwise} \end{cases}$
    Perform a gradient descent step on $(y_t - Q(s_t, a_t; \theta))^2$ with respect to the parameters θ ;
    $\epsilon = max(\epsilon - \epsilon_\delta, \epsilon_{\min})$;
    Every C steps reset $\theta'' = \theta$;
    $t = t+1$;
  end
end

values and the weights of the target-network equal to the weights of the Q-network. We also initialize ϵ, that indicates the probability of the random selection of the action to its initial value (ϵ_0). The replay memory is initially empty and stores the transactions as the execution proceeds. The outer-most **for** loop executes for the number of times equal to the number of test cases or the number of episodes. The MDP also contains a function that can terminate the episode upon the satisfaction of some criterion. The steps inside an episode proceed as follows: (i) The agent sends the current state s_t to the Q-network which returns the Q-values for each action in the action-space; (ii) The agent chooses an action following the ϵ-greedy algorithm in which the agent chooses an action a_t that could be a random action with probability ϵ, or the action that has the maximum Q-value among all actions in the action-space with a probability 1-ϵ; (iii) The agent applies the action to the environment, due to which the environment changes its state from s_t to s_{t+1}, and returns a reward r_t, where $r_t = R(s_t, a_t, s_{t+1})$; (iv) The agent stores the complete transaction of 4-tuple (s_t, a_t, r_t, s_{t+1}) into the replay memory D_M; (v) The agent also performs the training of the DQN by applying the following steps: (a) Retrieves a batch of B random observations from D_M; (b) Obtains a target value y_t using the target-network which is equal to r_t if the episode terminates in the next iteration, and if the next iteration is not the last iteration, then y_t is equal to $r_t + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a'; \theta)$", i.e., the sum of the immediate rewards and the maximum Q-value of any action of state s_{t+1} from the target-network; and, (c) The last step in training is to apply the gradient descent algorithm and update the parameters θ (i.e., the weights of the Q-network).

Two important characteristics of the above-mentioned DQL algorithm are as follows: (i) The use of replay memory (D_M); and, (ii) The use of a separate target-network for the computation of target values. It has been found that the consecutive transactions are correlated with each other, whereas, for stable training, the data should be uncorrelated. The replay memory breaks this correlation by sampling a batch of random transactions. We update the weights of the Q-network in every iteration. If we also use it to compute the target value y_t, then the target value changes in every iteration. Therefore, we employ a target-network whose weights remain unchanged for up-to C iterations. The readers can refer to [5] for a more information on the DQL algorithm.

V. PROPOSED MODELS

In this section, we first discuss the design of the MDP model of the environment and then show the architecture of the DQN used to approximate the Q-values.

A. MDP MODEL

The MDP model represents the environment that provides the feedback necessary for the DQN to learn and adjust the duration of time-slots in a DTDMA-based VLC system. The three major components of the MDP model are: (i) states, (ii) actions, and, (iii) rewards. In the following, we discuss them in detail.

1) STATES

We denote the state as a combination of the channel gains of the users and their current time-slots values. The following expression denotes the state representation.

$$s_t = \{h_0, h_1, \ldots, h_{N-1}, \tau_0, \tau_1, \ldots, \tau_{N-1}\} \tag{15}$$

where s_t denotes the state at time t, and all attributes are real numbers. The values of channel gains (h_i) can be obtained using Equation (1). The values of duration of time-slots (τ_i) lie between (0,1), and indicate the percentage duration of the time-slot allocated to the user u_i.

2) ACTIONS

The duration of time-slots of the users is expressed as the percentages of the total frame duration. To increase the duration of the time-slot of any one user we should reduce the duration of the time-slot of some other user by an equal amount because the sum of the percentages of all users should always remain equal to one. An action comprises the following

two steps. The first step is to choose two users, and the second step is to increase the duration of the time-slot of the first user by an amount equal to δ, and decrease the duration of the time-slot of the second user by an amount equal to δ. The value of δ is kept very small such as 0.1, 0.01. The action space contains a total of $1 + 2 \times \binom{K}{2}$ actions without any duplication. Any arbitrarily action a^k (where, $k = 0$ to $2 \times \binom{K}{2}$) can be denoted using the following mathematical expression:

$$a^k = \{\tau_i + \delta\sigma(\delta), \tau_j - \delta\sigma(\delta)\}, \text{ where,}$$
$$\sigma(\delta) = \begin{cases} 0 & \text{if } k = 0 \\ 1 & \text{otherwise} \end{cases}$$
$$i \neq j, \text{ , and } i, j \in \{0, ..K-1\} \quad (16)$$

The first action is denoted by a^0 and it does not change the duration of the time-slots. The remaining actions contain all possible combinations to increase and decrease the duration of the time-slots of any two users at a time. Fig. 5 illustrates an example that has three users ($\{u_0, u_1, u_2\}$), and the initial duration of the time-slots is $\{\tau_0, \tau_1, \tau_2\}$. The agent has up-to seven actions, it can increase/decrease the duration of the time-slots of any two users at a time, and it can also leave the time-slots unchanged.

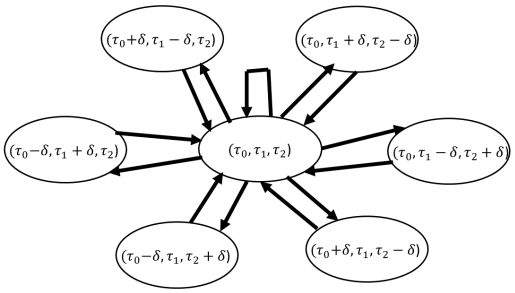


FIGURE 5. Illustration of the action space.

3) TRANSITION TO A NEW STATE

The state vector has two terms: channel gain, and, duration of time-slots. In the previous subsection, we mentioned that the application of actions changes the duration of time-slots. However, the state also changes if any user changes its position, and this causes a change in the value of the channel gain. In this subsection, we specify the two exceptional conditions in which the state of the environment does not change.

$$s_{t+1} = \begin{cases} s_t & \text{if } a_t = a^0 \\ s_t & \text{if } \exists \tau_i \in a_t \text{ s.t. } \tau_i - \delta < \tau_{min} \\ = s \in S & \text{otherwise} \end{cases} \quad (17)$$

The first case in the above equation shows that the application of action a^0 on any state, does not change the duration of time-slots, and hence the state remains unchanged (i.e., $s_{t+1} = s_t$). The second case is for any action $a_t = a^k$, where $(k > 0)$ applied on the state s_t, but the action a_t has a problem that it reduces the duration of at least one time-slot $\tau_i \in s_t$ whose existing value is $\tau_i < \tau_{\min} + \delta$, i.e., any decrease in the τ_i value is the violation of the constraint (6). In both these cases, the environment returns a reward but does not change its state.

4) REWARDS

Reward is the feedback on the last action taken by the agent. The reward function is usually closely tied to the goal or objective function. In our work, the goal is to maximize the SE of the system. The reward function is therefore expressed using the following equation.

$$R(s_t, a_t, s_{t+1}) = \begin{cases} -1 & \text{if } a_t = a^0 \\ -1 & \text{if } \exists \tau_i \in a_t \\ & \text{s.t. } \tau_i - \delta < T_{min} \\ -\sum_{i=0}^{K-1}(1 - \min(\frac{Z_i}{Z_{\text{th}}}, 1)) & \text{if } \exists \tau_i \in s_{t+1}, \\ & \text{s.t. } Z(\tau_i) < Z_{\text{th}} \\ \frac{\mathcal{F}(s_{t+1})}{\mathcal{F}(s_t)} & \text{if } \mathcal{F}(s_{t+1}) > \mathcal{F}(s_t) \\ -\frac{\mathcal{F}(s_t)}{\mathcal{F}(s_{t+1})} & \text{if } \mathcal{F}(s_{t+1}) < \mathcal{F}(s_t) \\ 0, & \text{otherwise} \end{cases} \quad (18)$$

In the above equation, the first two cases do not change the state of the environment, as discussed in the previous subsection. In the first two cases, we assigned a negative reward to prevent the agent from choosing actions that do not change the state of the environment. In the optimization process, it is critical that the search continues and should not freeze at any local minima. The third case occurs when the data-rates of one or more users in the state s_{t+1} is lesser than the minimum required value (Z_{th}). In the third case, the reward expression returns a negative value of magnitude equal to the summation of the ratio $\frac{Z_i}{Z_{\text{th}}}$ of those users whose data-rate is less than the minimum required value. In the fourth case of (18), we assign a positive reward because it refers to the condition when the new state is better than the previous one. The fifth case refers to the condition when the SE value of the new state is worst than the previous state, and we assign it a negative reward. The function $\mathcal{F}$ is described in (4) and (9) and determine the average SE value of the state.

5) TERMINATION CONDITION OF THE EPISODES

The initial state in the proposed model can be any random state that meets the constraint on the minimum duration of time-slots (i.e., constraint (6)). The episodic tasks come to an end after a finite number of iterations. The termination criterion in the episodes that solve a combinatorial optimization problem can be the attainment of minimum solution quality. We assume that the termination criterion of the episode is Δ_I iterations after the episode reaches a given target SE value denoted by Δ_Q. We can set the value of Δ_Q using the

user requirements and/or information of the other existing methods.

B. ARCHITECTURE OF THE DQN

As mentioned earlier, the DQL architecture employs two DQNs that are essentially DNNs. Fully connected neural networks are mostly a general type of DNN and have no requirements about the type of input data. The mathematical equation of the output of a neuron in the fully connected neural network is as follows.

$$y = \sigma\left(\sum_{i=0}^{L-1} w_i x_i\right) \tag{19}$$

In the above equation, the number of inputs is equal to L, and the weights are denoted by w_i and inputs by x_i. σ denotes the activation function. We also observed through simulations that the DQN of fully neural networks produce good results. Fig. 6 shows the architecture of the DQN that has $D+2$ layers, the first and the last layers are the input and output layers and have a number of nodes (or neurons) equal to the number of attributes in the state, i.e., $2K$, and number of possible actions, respectively. The DNN has up-to D hidden layers, and the number of neurons in any hidden layer is equal to W. W and D are also often referred to as the depth and width of the DNN. The activation function of all layers is the Rectified Linear Unit (ReLu). The output of ReLu is given by $y = max(0, x)$, where x is the input. The suitable values for the parameters will be determined through simulations.

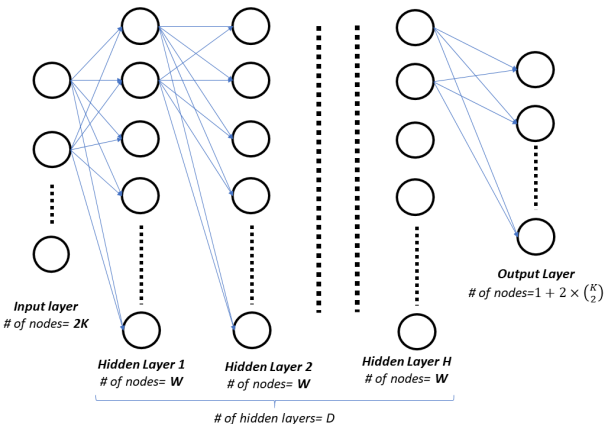


FIGURE 6. Architecture of the DQN which is used to approximate the Q-value.

C. COMPUTATIONAL COMPLEXITY

In each iteration, in addition to the computation of the objective function, the DQL-based optimization methods update the weights of the DNN. In this subsection, we discuss the computational complexity of the process that updates the weights of the DNN in each iteration. The process comprises several forward propagations and one backpropagation, and the computations mostly are matrix multiplications. We can assume that the number of neurons in any layer is equal to W (the number of neurons in the input and output layers cannot be more than W, and the hidden layers contain W neurons). Thus the complexity of the feed-forward propagation step is given as $O((D+2)W^3)$ or $O(DW^3)$, where D is the number of hidden layers and W^3 is the complexity of a matrix multiplication operation. The two main steps of the backpropagation are: (i) back propagate the error of the neurons from the output to the input layer; and, (ii) computation of new weights using the error values of the neurons. The back propagation of error has a complexity of $O(DW^3)$, and the updating of the weights has a complexity of $O(DW^2)$. The complexity of the backpropagation step is equal to $O(DW^3)$. In each iteration, we have forward propagation steps equal to twice the batch size and one step of backpropagation. Hence, the complexity of the process to update the weights in each iteration is equal to $O(DW^3 + 2BDW^3)$, which can be reduced to $O(W^3)$, considering that D and B are constants and have small values.

VI. SIMULATIONS

We implemented the VLC system and the DQN model using Python and Pytorch. Table 2 lists the values of the parameters of the VLC system. Abdelhady *et al.* [11] proposed a range of values of their DTDMA-based VLC system and the values mentioned in Table 2 are within the suggested range. We generated a total a 6000 test problems, in which the room size is equal to 10m×10m×5m, and the locations of users in the room is random. The number of users in each of the 2000 test cases are equal to 6, 8, and 10, respectively, and the LED is located in the center of the room. Each test case is solved by an episode of the proposed model.

TABLE 2. Parameters values of the VLC system.

parameter	value
N_o	10e-22 W/Hz
A_p	1 cm^2
ϕ_a	60
ψ_a	85
η_{oe}	0.6 A/W
η_{eo}	1
I	3.5 A
$\tau_{\min}$	1e-8 sec
B_v	20 MHz
Z_{th}	5e4 bps

We also implemented two well-known optimization metaheuristic algorithms: (i) SA algorithm; and, (ii) TS algorithm [20] to benchmark the performance of the proposed model. Both of these metaheuristic algorithms have successfully solved different optimization problems in the VLC systems [10], [12], [13]. The results of the TS algorithm are significantly better than the SA algorithm and are used as target values, (Δ_Q), to be reached by the proposed model. The DTDMA is a recent development, and there are no specific heuristic algorithms for it. Table 3 shows the parameters values of both the algorithms. The neighbor function in both SA and TS algorithms is to randomly choose two users and increase and decrease the duration of their time-slots by amounts equal to δ (where $\delta = 0.01$). The aspiration criterion

TABLE 3. Parameters values of the SA and TS algorithm.

Parameter		
Symbol	Description	Value
SA algorithm		
α	Cooling rate	0.98
β	Rate of increase in the value of M_s	1
M_s	Number of iterations in the Metropolis function	4
T_0	Initial temperature	1e12
Δ_S	Termination criterion (iterations)	50,000
TS algorithm		
T_s	Size of the Tabu list	7
N_m	Number of neighbor solutions	5
Δ_T	Termination criterion (iterations)	50,000

in TS algorithm is to allow the moves from the Tabu-list if they improve the SE value of the current solution. The SA and TS algorithms were executed for 50,000 iterations on each test case, which is a very large value and allows the search to converge to its best value.

The values of hyperparameters are critical in any DQL model. We set the values of the hyperparameters of the proposed DQN model using either the available guidelines or determine the most suitable values by experimenting with several alternatives. Table 4 lists the values of the hyperparameters used in this work. An important component of the proposed model is a DNN for which we need to decide the number of layers (D) and the number of neurons in each layer (W). The role of the DNN is to approximate the Q-function, and D equal to 2 is considered sufficient to approximate functions. The number of neurons in any layer should lie between the number of neurons in the input and output layers. In our case, the input layer has $2K$ (where K is the number of users) neurons, and the output layer has $\binom{1+2K}{2}$ neurons. Based, on these guidelines, we selected $D = 2$, and W values between $2K$ and $2\times\binom{K}{2}+1$ [21]. Another important parameter is ϵ, and we should define its starting value, final value, and a unit decrement in its value. The proposed DQL model adopts the ϵ-greedy method in choosing actions. In the ϵ-greedy method, the agent selects a random action with a probability equal to ϵ and selects the action that has the maximum Q-value as returned by the DQN with probability equal to $1-\epsilon$. The initial value of ϵ is usually set to 1, and its final value to a small non-zero value that enables the DQL to keep on exploring new states, actions, and rewards. In our problem, the size of the search space is huge. Therefore, continuous exploration benefits the search process to avoid getting trapped in local optima. It was empirically determined that when the minimum value of ϵ is equal to 0.25, then there is a good balance between exploration and exploitation.

TABLE 4. Hyperparameters values of the DQL model.

Hyperparameter	
parameter	Value
ϵ_0	1
$\epsilon_{\min}$	0.25
ϵ_δ	1e-5
γ	0.99
C	500
B	8
m_r	200,000
Δ_I	200
δ	0.01
D	3
W	64

Fig. 7 shows the results of the proposed DQN model when the number of users (K) is equal to 6, 8, and 10. The graph shows the SE values obtained in each episodic task solved by the proposed model. Each episodic task solves the problem of optimizing the duration of time-slots for a particular position of users. The graph conveys the following information about the SE values, when $K = 6$, 8, and 10. Here, the SE values lie in the ranges of 3.576–4.492, 2.892–4.436, and 3.431–4.4.465, respectively. This observation shows that the system can handle users from 6–10 without any significant effect on the performance.

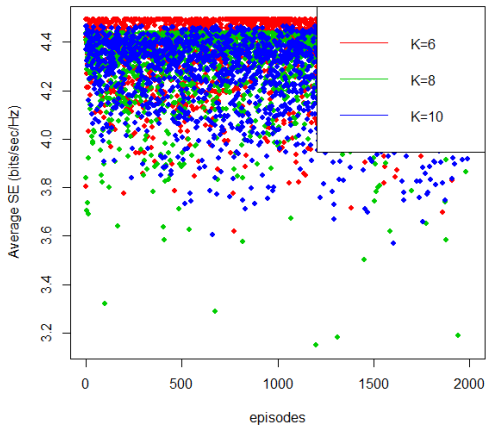


FIGURE 7. SE values of the solutions returned by the proposed DQN method for different values of K.

Fig. 8 shows the number of steps or iterations in each episode. The medians of the number of iterations per episodes are equal to 325, 569, and 707, for $K = 6$, 8, and 10, respectively. The graph also shows that the third quartile (Q3) of the number of steps is equal to 456, 902, and 1352, for

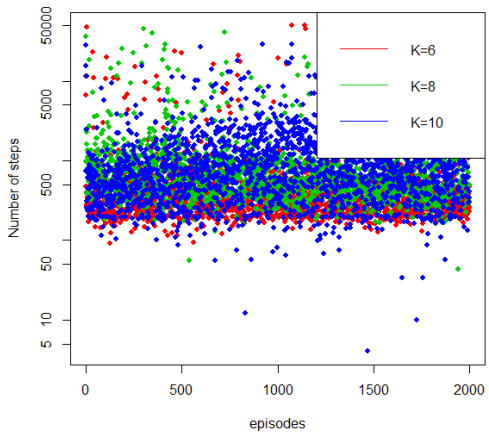


FIGURE 8. Number of steps or iterations in each episodes for different number of users (K).

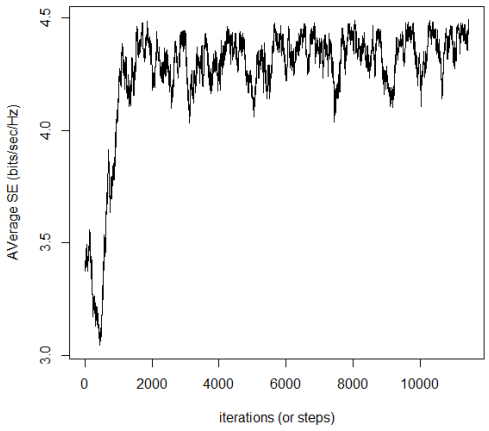


FIGURE 9. The curve showing the change in the value of SE within an episode.

$K = 6$, 8, and 10, respectively. The curve in Fig. 9 shows the change in the average SE value of the solution within an episode. It illustrates the effect on the SE value of the solution in response to the actions chosen by the agent. The curve contains many down-hills and up-hills before it can reach the maximum value in the 11460^{th} iteration. This curve also shows that the proposed model, similar to hill climbing, enables the agent to avoid trapping into local maxima and continue to search for the global maxima.

In the remaining part of this section we use box-plots to show the results, therefore, it is worthwhile to briefly discuss their key features. The key features of box-plots are: (i) The lines in the middle of the rectangles (or box) show the median of a data series; (ii) The lower and upper edges of the rectangle represent the Q1 and Q3 percentile of a data-series; (iii) The whiskers are small horizontal lines. The values of which terminate the vertical lines originating from the rectangles, and denote the minimum and maximum values of a data-series; and, (iv) The values denoted by points below the whiskers are known an outliers and denote the unexpected values.

Now, we start discussion on the comparison of the proposed model with that of the TS and SA algorithms. The box-plots in Fig. 10 show the SE values of solutions returned by the proposed model, TS, and SA algorithms. The plots show that based on the average SE values of solutions, the proposed model outperforms both TS and SA algorithms. In Fig. 11, we used the box-plots to show the difference between the SE values of the solutions of the proposed model with the solutions of TS and SA algorithms. In Fig. 11, the positive values indicate that the SE value of the solution of the proposed model is better than the solution of the SA or TS algorithms by that amount. The plots show that medians of the difference between the SE values of the proposed model

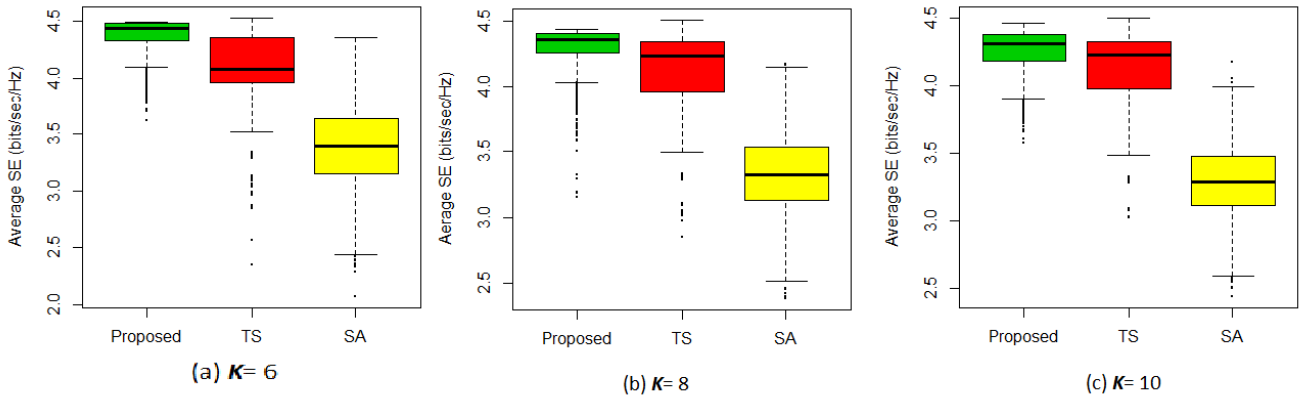


FIGURE 10. SE values of the proposed model and that of the TS and SA algorithms for different values of the number of users (K).

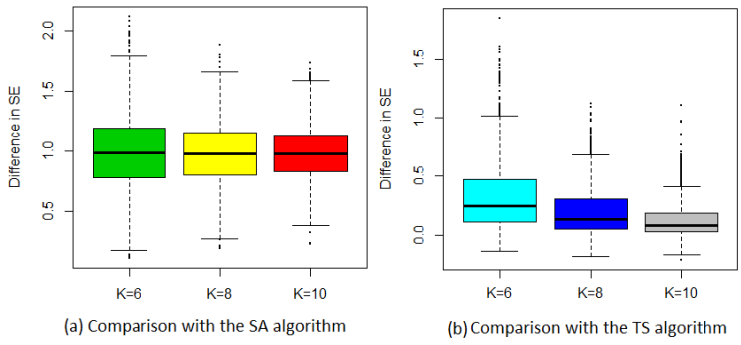


FIGURE 11. Difference between SE values of the proposed model and that of the TS and SA algorithms for different values of the number of users (K).

and that of the SA algorithm are equal to 0.986, 0.975, and 0.977, when $K = 6$, 8, and 10, respectively. The medians of the difference between the SE values of the proposed model and that of the TS algorithm are equal to 0.246, 0.132, and 0.080 for $K = 6$, 8, and 10, respectively. It is important to mention again that the TS and SA algorithms executed for up-to 50,000 iterations. In contrast, the proposed model only executed an average of 1112 iterations, and a maximum of 49,000 iterations in one episode. Therefore, the performance of the proposed model is better than SA and TS algorithms in terms of its ability to converge to good quality solutions.

Finally, we also used the paired Wilcoxon test [22], [23] to compare the results of the proposed model with that of SA and TS algorithms. The results indicated that the average SE values of the proposed model are significantly better than that those of TS and SA algorithms.

In short, the simulations in this section show that the proposed model is efficient in applying the DQL to solve the problem to optimize the duration of time-slots. The results of the DQL using the proposed model are competitive to two well-known metaheuristic algorithms.

VII. CONCLUSION AND FUTURE WORK

The DTDMA-based VLC system offers a high data-rate and does not suffer from the high PAPR problem. In a VLC system, the users are spread in a room and experience different channel strengths. We can maximize the SE of the system by adjusting the duration of the time-slots of the users. In this work, we proposed a model of the MDP that captures the functionality of the DTDMA based VLC system and enables the DQL algorithm to get trained and optimize the duration of the time-slots. The definition of the MDP includes innovative and problem-specific descriptions of the state, actions, and rewards. We considered episodic tasks whose goal is to adjust the duration of the time-slots of users; and an episode terminates when SE of the system reaches a given target value. The DNN used in our work has two-layers, is fully connected, with ReLu deployed as an activation function. Simulations showed that the proposed MDP model is efficient, and can integrate into the DQL algorithm to optimize the duration of time-slots and find globally optimal solutions. Simulations also showed that the performance of the proposed model is competitive to two metaheuristic algorithms: SA and TS. The current DTDMA technique is for the single LED-based multi-users VLC networks. An extension of the DTDMA technique to the multi-LEDs and multi-users VLC networks introduces interference in the network. In a multi-LED network, an LED cannot act in isolation and need to collaborate with other LEDs to achieve overall maximum efficiency. Extension of DTDMA to multi-LEDs and development of DQL based method of optimization is an important direction of future.

REFERENCES

[1] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Found. Trends Mach. Learn.*, vol. 11, nos. 3–4, pp. 219–354, 2018, doi: 10.1561/2200000071.

[2] A. Shawahna, S. M. Sait, and A. El-Maleh, "FPGA-based accelerators of deep learning networks for learning and classification: A review," *IEEE Access*, vol. 7, pp. 7823–7859, 2019.

[3] 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.

[4] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "Q-learning algorithms: A comprehensive classification and applications," *IEEE Access*, vol. 7, pp. 133653–133667, 2019.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

[6] H. Liao, W. Zhang, X. Dong, B. Poczos, K. Shimada, and L. Burak Kara, "A deep reinforcement learning approach for global routing," *J. Mech. Des.*, vol. 142, no. 6, pp. 061701-1–061701-12, Jun. 2020, doi: 10.1115/1.4045044.

[7] X. Lei, Z. Zhang, and P. Dong, "Dynamic path planning of unknown environment based on deep reinforcement learning," *J. Robot.*, vol. 2018, pp. 5781591-1–5781591-10, Sep. 2018, doi: 10.1155/2018/5781591.

[8] A. I. Panov, K. S. Yakovlev, and R. Suvorov, "Grid path planning with deep reinforcement learning: Preliminary results," *Procedia Comput. Sci.*, vol. 123, pp. 347–353, Jan. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050918300553

[9] H. Lee, T. Q. S. Quek, and S. H. Lee, "A deep learning approach to universal binary visible light communication transceiver," *IEEE Trans. Wireless Commun.*, vol. 19, no. 2, pp. 956–969, Feb. 2020.

[10] U. F. Siddiqi, S. M. Sait, M. S. Demir, and M. Uysal, "Resource allocation for visible light communication systems using simulated annealing based on a problem-specific neighbor function," *IEEE Access*, vol. 7, pp. 64077–64091, 2019.

[11] A. M. Abdelhady, O. Amin, A. Chaaban, B. Shihada, and M.-S. Alouini, "Downlink resource allocation for dynamic TDMA-based VLC systems," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 108–120, Jan. 2019.

[12] Y. Chen, S. Li, and H. Liu, "Dynamic frequency reuse based on improved tabu search in multi-user visible light communication networks," *IEEE Access*, vol. 7, pp. 35173–35183, 2019.

[13] U. F. Siddiqi, O. Narmanlioglu, M. Uysal, and S. M. Sait, "Joint bit and power loading for adaptive MIMO OFDM VLC systems," *Trans. Emerg. Telecommun. Technol.*, p. e3850, Jan. 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3850

[14] X. Liang, X. Du, G. Wang, and Z. Han, "A deep reinforcement learning network for traffic light cycle control," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1243–1253, Feb. 2019.

[15] E. Mocanu, D. C. Mocanu, P. H. Nguyen, A. Liotta, M. E. Webber, M. Gibescu, and J. G. Slootweg, "On-line building energy optimization using deep reinforcement learning," *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 3698–3708, Jul. 2019.

[16] B. Demirel, A. Ramaswamy, D. E. Quevedo, and H. Karl, "DeepCAS: A deep reinforcement learning algorithm for control-aware scheduling," *IEEE Control Syst. Lett.*, vol. 2, no. 4, pp. 737–742, Oct. 2018.

[17] C. Wu, B. Ju, Y. Wu, X. Lin, N. Xiong, G. Xu, H. Li, and X. Liang, "UAV autonomous target search based on deep reinforcement learning in complex disaster scene," *IEEE Access*, vol. 7, pp. 117227–117245, 2019.

[18] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*. Basel, Switzerland: Birkhäuser, 2013.

[19] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA, USA: Freeman, 1979.

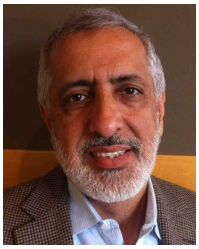
[20] S. M. Sait and H. Youssef, *Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems*, 1st ed. Los Alamitos, CA, USA: IEEE Computer Society Press, 1999.

[21] J. Heaton, *Introduction to Neural Networks for Java*, 2nd ed. Chesterfield, MO, USA: Heaton Research, 2008.

[22] U. F. Siddiqi, Y. Shiraishi, and S. M. Sait, "Memory-efficient genetic algorithm for path optimization in embedded systems," *IPSJ Online Trans.*, vol. 6, no. 6, pp. 28–36, 2013.

[23] *R: A Language and Environment for Statistical Computing*, R Core Team, R Found. Stat. Comput., Vienna, Austria, 2013. [Online]. Available: http://www.R-project.org/

UMAIR F. SIDDIQI (Member, IEEE) was born in Karachi, Pakistan. He received the B.E. degree in electrical engineering from the NED University of Engineering and Technology, Karachi, Pakistan, in 2002, the M.Sc. degree in computer engineering from the King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia, in 2007, and the D.Eng. degree from Gunma University, Japan, in 2013. He is currently a Research Engineer with the Center of Communications and Information Technology Research of Research Institute, KFUPM. He has authored over 30 research papers in international journals and conferences. He also has several U.S. patents. His areas of research interests include reinforcement learning, optimization, soft computing, evolutionary algorithms, metaheuristics, and electronic design automation.



SADIQ M. SAIT (Senior Member, IEEE) was born in Bengaluru. He received the bachelor's degree in electronics engineering from Bangalore University, in 1981, and the master's and Ph.D. degrees in electrical engineering from the King Fahd University of Petroleum & Minerals (KFUPM), in 1983 and 1987, respectively. He is currently a Professor of computer engineering and the Director of the Center for Communications and IT Research, Research Institute, KFUPM. He has authored over 200 research papers, contributed chapters to technical books, granted several US patents, and lectured in over 25 countries. He is also the Principle Author of two books. He received the Best Electronic Engineer Award from the Indian Institute of Electrical Engineers, Bengaluru, in 1981.

MURAT UYSAL (Fellow, IEEE) received the B.Sc. and M.Sc. degrees in electronics and communication engineering from Istanbul Technical University, Istanbul, Turkey, in 1995 and 1998, respectively, and the Ph.D. degree in electrical engineering from Texas A&M University, College Station, TX, USA, in 2001. He is currently a Full Professor and the Chair of the Department of Electrical and Electronics Engineering, Ozyegin University, Istanbul. He is also the Founding Director of the Center of Excellence in Optical Wireless Communication Technologies (OKATEM). Prior to joining Ozyegin University, he was a Tenured Associate Professor with the University of Waterloo, Waterloo, ON, Canada, where he still holds an Adjunct Faculty Position. He has authored some 300 journal and conference papers on his research topics and received more than 11,000 citations. His research interests are in the broad areas of communication theory and signal processing with a particular emphasis on the physical layer aspects of wireless communication systems in radio, acoustic, and optical frequency bands. He was a recipient of the Marsland Faculty Fellowship, in 2004, the NSERC Discovery Accelerator Award, in 2008, the University of Waterloo Engineering Research Excellence Award, in 2010, the Turkish Academy of Sciences Distinguished Young Scientist Award, in 2011, the Ozyegin University Best Researcher Award, in 2014, the National Instruments Engineering Impact Award, in 2017, the Elginkan Foundation Technology Award, in 2018, and the IEEE Communications Society Best Survey Paper Award in 2019, among others. He was involved in the organization of several IEEE conferences in various capacities. He was the TPC Chair of major IEEE conferences, including the Wireless Communications and Networking Conference 2014, the International Symposium on Personal, Indoor and Mobile Radio Communications 2019, and the Vehicular Technology Conference–Fall 2019. In addition, he was the Chair of the Communication Theory Symposium of the IEEE International Conference on Communications 2007, the Chair of the Communications and Networking Symposium of the IEEE Canadian Conference on Electrical and Computer Engineering 2008, the Chair of the Communication and Information Theory Symposium of International Wireless Communications and Mobile Computing Conference 2011, and the General Chair of the IEEE International Workshop in Optical Wireless Communications 2015. He is the Chair of the IEEE Turkey Section. He is currently an Editorial Board Member of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS. In the past, he was an Editor of the IEEE TRANSACTIONS ON COMMUNICATIONS, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE COMMUNICATIONS LETTERS, *Wiley Wireless Communications and Mobile Computing* (WCMC), *Wiley Transactions on Emerging Telecommunications Technologies* (ETT), as well as the Guest Editor of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (2009 and 2015) and Physical Communication (Elsevier) (2018).