A Reliable Reinforcement Learning for Resource Allocation in Uplink NOMA-URLLC Networks

Waleed Ahsan, *Student Member, IEEE,* Wenqiang Yi, *Member, IEEE,* Yuanwei Liu, *Senior Member, IEEE,* and Arumugam Nallanathan, *Fellow, IEEE*

***Abstract*—In this paper, we propose a deep state-action-reward-state-action (SARSA) λ learning approach for optimising the uplink resource allocation in non-orthogonal multiple access (NOMA) aided ultra-reliable low-latency communication (URLLC). To reduce the mean decoding error probability in time-varying network environments, this work designs a reliable learning algorithm for providing a long-term resource allocation, where the reward feedback is based on the instantaneous network performance. With the aid of the proposed algorithm, this paper addresses three main challenges of the reliable resource sharing in NOMA-URLLC networks: 1) user clustering; 2) Instantaneous feedback system; and 3) Optimal resource allocation. All of these designs interact with the considered communication environment. Lastly, we compare the performance of the proposed algorithm with conventional Q-learning and SARSA Q-learning algorithms. The simulation outcomes show that: 1) Compared with the traditional Q learning algorithms, the proposed solution is able to converges within 200 episodes for providing as low as 10^{-2} long-term mean error; 2) NOMA assisted URLLC outperforms traditional OMA systems in terms of decoding error probabilities; and 3) The proposed feedback system is efficient for the long-term learning process.**

***Index Terms*—Deep SARSA$-\lambda$ learning, non-orthogonal multiple access, power allocation, ultra-reliable low-latency communication, user clustering**

I. Introduction

As a key design feature for 5G and beyond 5G (B5G) networks, ultra-reliable low-latency communication (URLLC) attracts increased attention recently [2]. Thanks to the empirical advantages of URLLC that focuses on the transmission with finite blocklength (FBL) to support emerging applications, such as intelligent transportation systems [3], factory automation (industry 4.0), healthcare (remote surgery), virtual reality, and augmented reality. Such type of services requires guaranteed reliability, latency, and data rates for wireless communications [4] [5]. Note that these emerging real-time applications of the B5G network demands long-term optimizations for not only URLLC but also high data rates [2]. Different from orthogonal multiple access (OMA) schemes, NOMA is able to reduce the transmission delay for each user by providing additional access in the power domain, especially for extremely-high-frequency communications with low-rank channels [6]. Therefore, NOMA offers the remedy for new rigorous requirements of the URLLC in two different ways [7]. Firstly, NOMA enhances the spectral efficiency for URLLC. Secondly, due to simultaneously serving multiple users in the same resource block, NOMA is able to further reduce the latency for URLLC, especially for the scenario with massive users. Finally, to find an optimal balance between latency, reliability, and data rates, an intelligent-learning-based mechanism can be applied to dynamically operate the resource allocation in NOMA-URLLC networks.

W. Ahsan, W. Yi, Y. Liu, and A. Nallanathan are with Queen Mary University of London, London, UK (email:{w.ahsan, w.yi, yuanwei.liu, a.nallanathan}@qmul.ac.uk).

Part of this work was submitted in IEEE Global Communications Conference (GLOBECOM), December, Spain, 2021 [1].

A. Related Works

1) NOMA-URLLC: Due to the simple design of OMA systems, most of the recent works concentrate on exploring URLLC with OMA instead of NOMA communications [8]–[10]. As URLLC uses FBL transmissions, the well-known Shannon capacity is no longer the accurate approximation of data rates [11]. To characterise the FBL transmissions, the authors in [12] and [13] designed a fundamental simulation framework. However, due to the design complexity of NOMA, evolving OMA-URLLC to NOMA-URLLC faces numerous challenges [14]. In [15], the authors designed downlink multiple-input multiple-output (MIMO)-NOMA framework for the URLLC networks. The design focused on a two-user case, which can be extended to multi-user scenarios to enhance the connectivity. Similarly, the work [16] compared packet sizes, required signal to noise ratio (SNR), and delay constraints for the NOMA-URLLC systems for the two-user case. The simulation outcomes showed that the required SNR is inversely proportional to the quality of service exponent and error probabilities. The authors in [17] designed a static multi-user NOMA-URLLC framework based on hybrid automatic repeat request re-transmissions. In this framework, fixed user connectivity was simulated for different network settings to show the performance of NOMA-URLLC. The results section showed that the performance of NOMA-URLLC is better than OMA-URLLC schemes in all settings. The authors in [18] provided a detailed numerical framework for NOMA-URLLC using a dynamic programming approach. Based on error probabilities and the maximum transmit power, the simulation outcomes proved that NOMA-URLLC systems perform efficiently in all discussed cases. Finally, in [19], the authors presented the advantages of NOMA based URLLC transmission use cases over OMA-URLLC systems. Notably, existing works do not address data rates, reliability, and latency constraints properly for a long-term basis.

2) AI-NOMA: In comparison to the model-free methods, the traditional model-based approaches are not suitable for

simultaneously controlling multiple network parameters. Due to frequent fluctuations in wireless systems, a significant amount of model-free frameworks are being investigated [20]. For example, the authors in [21] proposed communication deep neural network (CDNN) to optimise the sum utility of the MIMO-NOMA system. The simulation results showed that CDNN outperforms the conventional methods by providing efficient system utility. In this model, the stochastic gradient descent (SGD) optimization was used to minimise the training loss with fully connected layer of linearly rectified (ReLU) activations. Similarly, inspired by the grant-free NOMA, the authors in [22] designed a DNN framework to address low latency access for tactile internet of things (IoT) system. The performance outcomes showed that the proposed model provides better results than multi-user shared access (MUSA) technique. A meta deep learning based framework is also proposed in [23] to handle dynamic network parameters for the supervised learning scheme. In which a general DNN model is trained on pre-generated data-set to learn the communication network settings. In simulation outcomes meta learning based model free system was able to learn the system efficiently than traditional learning technique. In a recently proposed framework [24], the authors also used DNN to design the deep-NOMA system. According to their simulation outcomes, in most of the cases the deepNOMA framework performs better than other schemes like conventional OMA and model-based NOMA systems. Different from others, the authors in [25] exploited DNN for the NOMA based caching scheme for the dynamic power control. The final results were verified using Monte Carlo simulations that showed the learning-based algorithm performs better than traditional model-based schemes. Interestingly, all the existing DNN based models require huge data-sets for training that is difficult to be acquired for practical NOMA systems. Therefore, apart from other advantages, the similar models suffer from the same limit, where the training is only based on data-sets instead of the actual environment. This limit may result in an unreliable decision since the real-time feedback is not effectively considered. Hence, practical NOMA systems, especially for URLLC, require a reliable model-free design that can learn from actual wireless network environments.

B. Motivations

For emerging NOMA-URLLC applications, long-term reliable optimization of resource allocation to improve the latency, reliability, and data rates is challenging but important [26]–[28]. Because in long-term optimization process the learning strategy is developed to predict future resource allocations. Therefore, the long-term strategy is based on the real experience of the agent (BS) that is gained by interactions with the physical environment to ensure the stability and reliability. As discussed in the related works, the existing solutions for URLLC are mainly based on conventional OMA or model-based NOMA (short-term) conventional schemes. A new solution with prompt real-time responses is urgently needed to enhance the reliability in learning based resource allocation process. Recently, human-level control accuracy over the continuous control is achieved in the game of GO by the use of reliable learning algorithms, such as (n)-step reinforcement learning (RL) algorithms and google deepmind [29]–[31]. Therefore, such type of techniques are suitable to handle dynamic networks. Compared with other RL techniques, e.g., traditional Q-learning [32], multi-arm bandits [33], and DNN's, the (n)-step RL is a promising direction to learn long-term reliable policies. Since, (n)-step RL performs the future actions/decisions depending on the knowledge from multiple ((n)-step) interactions for the purpose of enhanced efficiency. In other words, as the process of RL algorithms is based on the rewards and punishments for this reason with the help of ((n)-step) optimizations the agent finds the long-term effective allocation mechanism to avoid the punishments. Consequently, the learned policy is suitable for long-term in dynamically changing wireless network scenarios. So that most recent models in wireless networks are based on multi-step learning [34], [35].

Motivated from the aforementioned discussions, for time-varying network environment, we consider to design a model-free ((n)-step) learning framework to address long-term resource allocation problem for URLLC with NOMA transmissions. Moreover, a multi-user NOMA scheme is considered to further enhance the performance.

C. Contributions and Organization

In the result of the above motivations, we design a reliable framework to learn NOMA-URLLC uplink communication environments in terms of the long-term performance. With a reliable (n)-step learning framework, we first formulate the considered problem as the reward maximization issue by jointly optimising the resource allocation and error probabilities. Then, we propose the Deep SARSA$-\lambda$ algorithm for the long-term optimization with rewards. Lastly, to enhance the reliability, we utilise the sparse neuron activation that are based on the ReLU and backward trace table. The detailed summary of our contributions are listed as follows:

- We design a 2D model-free clustering framework to serve NOMA-URLLC users dynamically based on available network resources. Considering this framework, we devise a mean error minimization problem to handle URLLC constraints. These constraints consider long-term variables in the proposed NOMA-URLLC systems, such as the packet size of users, the number of users and channel gains. To portray fully online behaviour, these parameters are alterable at each time step.
- We propose a reliable deep RL (DRL) algorithm, as deep SARSA$-\lambda$ with dynamic λ and reliability tracing, to address the long-term reliability problem. The deep neural network is used for forward tracing to increase short-term reliability. Long-term reliability is handled by backward tracing via dynamic λ.
- We design a reward function $D_r(t)$, that is used to train the BS for optimal resource allocation under URLLC constraints. Additionally, to avoid simple (non-practical) case with a static reward function that is predefined by the designer. The proposed reward function $D_r(t)$ is based

on the instantaneous system output (system utility) at each timestep (t_s), rather than conventional user defined values. Furthermore, to increase the exploration safety and reliability we design backwards trace table that assists the DRL agent to avoid unreliable state visits.

- We demonstrate that: 1) Based on time varying properties, reliable resource allocation can be performed simultaneously in time varying systems; 2) According to the proposed framework, the reward function provides optimal trade-off for the long-term error (Fig. 4) and the sum rate (Fig. 3) by best convergence; 3) For deep SARSA$-\lambda$, the long-term mean error probability is proportional to the network density and packet size; 4) For DNN prediction accuracy, the mean squared error (MSE) is also proportional to the network density and packet size; and 5) NOMA is more suitable for URLLC due to providing the efficient spectral utility with low decoding error probabilities as compared to OMA systems.

II. Network Model and Problem Formulation

A. Network Settings

This paper investigates NOMA-aided uplink URLLC with short data blocks as shown in Fig. 1. One BS is located at the center of the considered cell. The BS communicates with N_u users via N_s orthogonal sub-channels. Both the BS and users are assumed to be equipped with a single antenna. The set of data block sizes $m_k \in \Upsilon_m$ ($k \in [1, N_u]$) for all users $\Upsilon_m = \{m_1, m_2, \cdots, m_{N_u}\}$ are assumed to be the same, which consists of D bits. The transmission of these blocks is subject to a latency constraint, i.e., for each sub-channel the transmission of one data block should be finished within M seconds per unit bandwidth. The sub-channels are indexed by $\Upsilon_s = \{s_1, ..., s_{N_s}\}$. Regarding users, the set for users severed by the BS through a sub-channel (i.e., a NOMA cluster) $s_j \in \Upsilon_s$ ($j \in [1, N_s]$) is defined as $\Upsilon_u^j = \{u_1, ..., u_{N_u^j}\}$, where N_u^j is the number of the users connected to the BS via the sub-channel s_j and $\sum_{j=1}^{N_s} N_u^j = N_u$. The defined notations in this system model are listed in the TABLE I.

B. Signal Model

In NOMA, two or more users are able to access the same resource block (time/frequency). Therefore, each sub-channel has $N_u^j \geq 2$ [36]. To simplify the analysis, we assume the BS contains perfect CSI of all users. Based on such CSI, BS is capable to simultaneously optimise the sub-channel allocation for active users in long-term communications. For an arbitrary user u_k, we define its active index variable at time t as $c_k^j(t) = 1$ when the user u_k is served by the NOMA cluster s_j. Similarly, $c_k^j(t) = 0$ means the users are silent in the sub-channel s_j. The set of clustering parameters is defined as $\mathbf{C}_t$ and $c_k^j(t) \in \mathbf{C}_t, \forall j, k$. The size of $\mathbf{C}_t = N_u \times N_s$ in this model.

In a NOMA cluster s_j, when the BS first receives the transmitted messages from the connecting users from Υ_u^j, the signal of each user is decoded by applying SIC [6]. Additionally, we assume that the decoding order in this model is the reverse of the channel gain order [37]. In a time slot t, the instantaneous signal-to-interference-plus-noise ratio (SINR) for the intra-cluster user $u_k^j \in \Upsilon_u^j$ is given by

$$\gamma_k^j(t) = \frac{c_k^j(t) p_k^j(t) g_k^j(t)}{\sum_{k'=1}^{k-1} c_{k'}^j(t) p_{k'}^j(t) g_{k'}^j(t) + \sigma^2(t)}, \quad (1)$$

where the SINR for the last user is defined as

$$\gamma_1^j(t) = \frac{c_1^j(t) p_1^j(t) g_1^j(t)}{\sigma^2(t)}, \quad (2)$$

where $p_k^j(t)$ is the transmit power of the user u_k and the set of transmit power is given by $\mathbf{P}_t$ ($p_k^j(t) \in \mathbf{P}_t, \forall j, k$) and the size of $\mathbf{P_t}$ obeys that $|\mathbf{P_t}| = N_u$. The $\sigma^2(t)$ is the additive white Gaussian noise (AWGN) noise factor.

C. Achievable Data-rate with Finite Block Length

In uplink finite block-length NOMA transmission, the decoding of user u_k is based on the SIC process of its previous user u_{k+1}. For all of the users in the finite block-length transmissions the decoding error probabilities are denoted by the set $\varphi = \{\varepsilon_1^j(t), \varepsilon_2^j(t), ..., \varepsilon_{N_u^j}^j(t)\}$. Additionally, if the data rate of successfully completing the SIC process is R_k^{th}, according to [19], the decoding rate (in bps/Hz) of user u_k^j obeys

$$R_k^j(t) \approx \log_2(1 + \gamma_k^j(t)) - \sqrt{V/M} \frac{Q^{-1}(\varepsilon_k^j(t))}{\ln 2}, \quad (3)$$

where

$$V = 1 - (1 + \gamma_k^j(t))^{-2}, \quad (4)$$

If $R_k^j(t) < R_k^{th}$ the target rate threshold R_k^{th}, the decoding of all the users $u_k, ..., u_1$ fails, namely $R_k^j(t) = ... = R_1^j(t) \equiv 0$. $Q^{-1}(.)$ is the inverse function of the following equation:

$$Q(\zeta) = \frac{1}{\sqrt{2\pi}} \int_{\zeta}^{\infty} e^{\frac{t^2}{2}} dt, \quad (5)$$

Based on the results in [13], the expression for the decoding error probability and the channel dispersion with the Rayleigh fading can be expressed as

$$\varepsilon_k^j(t) = Q(\Psi(\gamma_k^j(t), M, D)), \quad (6)$$

where

$$\Psi(\gamma_k^j(t), M, D) = \ln 2 \sqrt{\frac{M}{V}} (\log_2(1 + \gamma_k^j(t)) - \frac{D}{M}), \quad (7)$$

and $(\frac{D}{M})$ is the required decoding rate for satisfying the latency constraint.

D. Problem Formulations

In this section, we construct uplink-NOMA-URLLC based optimization problem. This work minimizes the decoding errors, since reducing decoding errors is capable for both enhancing the reliability and decreasing the latency. The

TABLE I
TABLE OF NOTATIONS

Symbol	Definition
N_s, s_j	Number of sub-channels (NOMA clusters), symbol of sub-channels (NOMA clusters)
N_u, u_k	Number of users, symbol of users
Υ_s	Set of sub-channels (NOMA clusters)
Υ_u^j,u_k^j	Set of users connected to BS via sub-channel s_j, user k in the set Υ_u^j
Υ_m, m_k	Set of block sizes with Block size m for the user k
$c_k^j(t)$	Clustering variable for user u_k connecting to BS via sub-channel s_j at time t
$p_k^j(t)$	Transmit power for user u_k^j at time t
$g_k^j(t)$	Channel gain for user u_k^j at time t
$\sigma^2(t)$	Additive white Gaussian noise at time t
D	Total number of the transmission bits for each user
M	Maximum block size
V	channel dispersion
$\Psi(\gamma_k^j(t), M, D)$	Q value Input calculation function
$\varepsilon_k^j(t)$	Decoding error probability
φ	Set of error probabilities
$\gamma_k^j(t)$	Instantaneous SINR for user u_k^j at time t
$R_k^j(t)$	Instantaneous data rate for user u_k^j at time t
R_k^{th}	Rate requirement for the SIC process of user u_k^j
$D_r(t)$	reward function
T	Duration of the considered long-term communication
$\tau(s,a)$	Reliability tracing table
$Q(s,a)$	Q table
$\varrho(\vartheta)$	activation function with input values (ϑ)
$\mathbb{S}$	State space
$\mathbb{A}$	Action space
$\mathbb{P}$	Transition probabilities
π	q learning policy
κ	Experience memory
$Q(.)$	Error function
λ	Learning step
$\mathbf{C}_t$, $\mathbf{P}_t$	Matrix for clustering parameters, matrix for transmit power
θ	DQN error/loss
$P_{s(t)\to s(t+1)}$	state transition probability

function to minimize the average decoding errors for long-term communications with a period T is as follows:

$$\min_{\mathbf{C_t},\mathbf{P_t}} \quad \mathbb{E}\left[\sum_{t_s=1}^{T}\sum_{j=1}^{N_s}\sum_{k=1}^{N_u^j}\varepsilon_k^j(t_s)\right] \tag{8a}$$

$$\text{s.t}: \quad p_1^j g_1^j(t_s) \leq, ..., \leq p_{N_u^j}^j g_{N_u^j}^j(t_s),\ \forall j, t_s, \tag{8b}$$

$$\sum_{k=1}^{N_u^j} c_k^j(t_s) p_k^j(t_s) \leq P_s, \forall j, t_s, \tag{8c}$$

$$R_k^j(t_s) > R_k^{th},\ \forall k, t_s, \tag{8d}$$

$$2 \leq \sum_{k=1}^{N_u^j} c_k^j(t_s) \leq N_u,\ \forall j, t_s, \tag{8e}$$

$$\sum_{j=1}^{N_s} c^j(t_s) \in \{1, 0\},\ \forall j, t_s, \tag{8f}$$

where (8b) shows the channel ordering for the network user based on the perfect CSI. (8c) is to impose the power constraint of each sub-channel. (8d) ensures all clustered NOMA users can be successfully decoded for maximizing the connectivity by ensuring that the rate $R_k^j(t_s)$ of each user is more than the target data-rate $R_k^{th} = 0$. (8e) limits the number of clustered users for the entire system and each sub-channel. The constraint (8f) indicates that each user belongs to only one cluster. All of these constraints in our coding are checked in the form of sequential steps, if the value is out of range from (8b-8e) then that try is discarded to check it again. The joint optimization problem (8a) is a non-convex and NP-hard problem. Therefore, all the sub-problems (i.e., resource allocation and simultaneous user connectivity) in (8a) are also non-convex and NP-hard, especially when considering (8c) and (8e). The detailed proof under traditional NOMA has been provided in [38]. However, by jointly considering the resource allocation and the finite packet conditions, the problem in (8a) becomes more complex. As a solution, the modern artificial intelligence techniques are known to provide long-term reliable solutions for such type of problems. Therefore, we propose the following model to handle this issue.

III. Reliable Resource Allocation for URLLC-NOMA

In this section, we propose an intelligent Deep SARSA$-\lambda$ based reliable resource allocation algorithm to assist the BS for minimizing the error probability for uplink NOMA-URLLC.

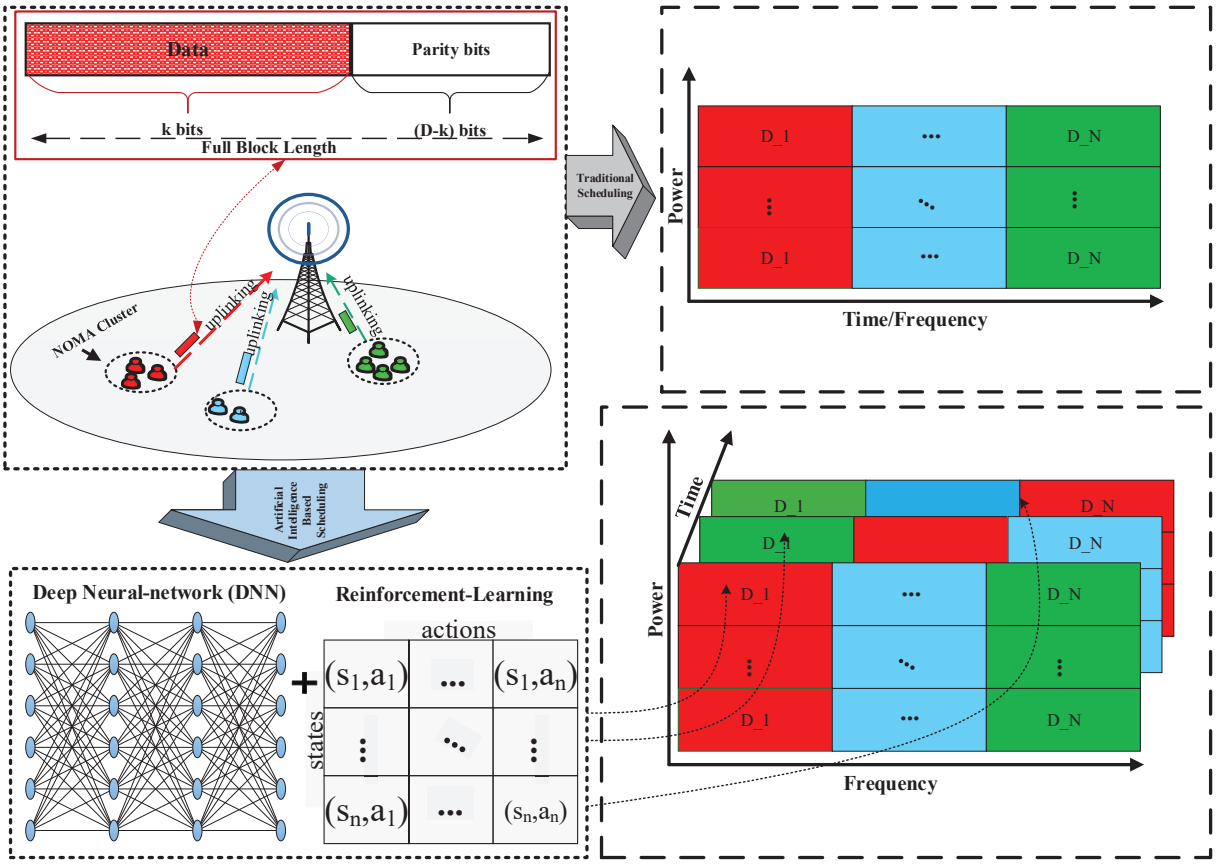


Fig. 1. Illustrating NOMA-URLLC uplink resource allocation as a model-free optimization problem. The top-left sub-figure shows details for the wireless network (uplink NOMA-URLLC) including NOMA clusters highlighted in (red, green and blue colours), a base-station, NOMA users, and the URLLC packet structure. The top-right sub-figure presents fixed resource allocations under traditional strategies. The bottom sub-figures shows the resource allocation strategies (on the right) using the deep reinforcement learning methods (on the left).

Now, we present the basic concepts and problem formulation for the deep SARSA-λ followed by reliable estimations, algorithmic details, and analysing the proposed algorithm.

A. The Reliable Learning Policy with SARSA-λ

1) Multi-step introduced reliability: Unlike the traditional one step RL methods such as, Q-learning and SARSA Q-learning. The deep SARSA$-\lambda$ algorithm performs (n)-step optimization process for dynamically changing networks (such as NOMA-URLLC with simultaneous user connectivity). This property ensures that deep SARSA$-\lambda$ is more reliable than conventional policy learners, which is discussed in **Remark 1**. Before that, we first introduce the Markov Reward process.

Definition 1. *(Markov Reward Process:) A Markov Reward Process (MRP) is based on the tuple* $\{\mathbb{S}, \mathbb{A}, \mathbb{R}, \mathbb{P}\}$. *For every time slot* t, *the element* $\{\mathbb{P}\}$ *is a probability distribution for the movement of an agent from state* $s(t) \in \mathbb{S}$ *to* $s(t+1)$ *that is conditioned on the action* $a(t)$ *to maximise the associated reward* $D_r(t)$, *that is expressed as* $D_r(t)$: $\mathbb{S} \times \mathbb{A} \times \mathbb{P} \Rightarrow \mathbb{R}$.

Remark 1. *For the reliable policy* π^*: $\mathbb{S} \Rightarrow \mathbb{A}$, *to balance the learning process for huge* $\mathbb{S}$ *in the time varying networks. The SARSA-*λ *algorithm adopts deep neural network (DNN) and the dynamic* λ *to keep track of* $(t-n)$ *allocations. Due to this* (n)*-step tracing capability, the SARSA-*λ *achieves the efficient path and enhanced reward performance by choosing the most efficient set of actions in the future.*

Therefore, based on the one-step learning mechanism, the traditional agent/s cannot guarantee the efficient policy by depending only on $a(t), s(t), r(t)$ and the current allocation strategy. The detailed reason is offered in the following remark.

Remark 2. *For the time varying network environments, to enhance the network reliability, the* (n)*-step MRP's SARSA-*λ *eliminates the less reliable allocations updating trace table with '0', as shown in* (12). *However, for the long-term time varying communication environments, conventional one-step expectation return MRP's Q-learning and SARSA Q-learning perform allocations without any prior knowledge. Therefore, the final allocation strategy for conventional approaches is not efficient.*

2) Dynamic λ *introduced reliability evaluations:* The dynamic λ determines the reward performance of each step of an agent leading towards the effective policy, that is enabled by trace table $\tau(s, a)$. Different from one-step expectation return MRP (Q-learning and SARSA Q-learning), the (n)-step

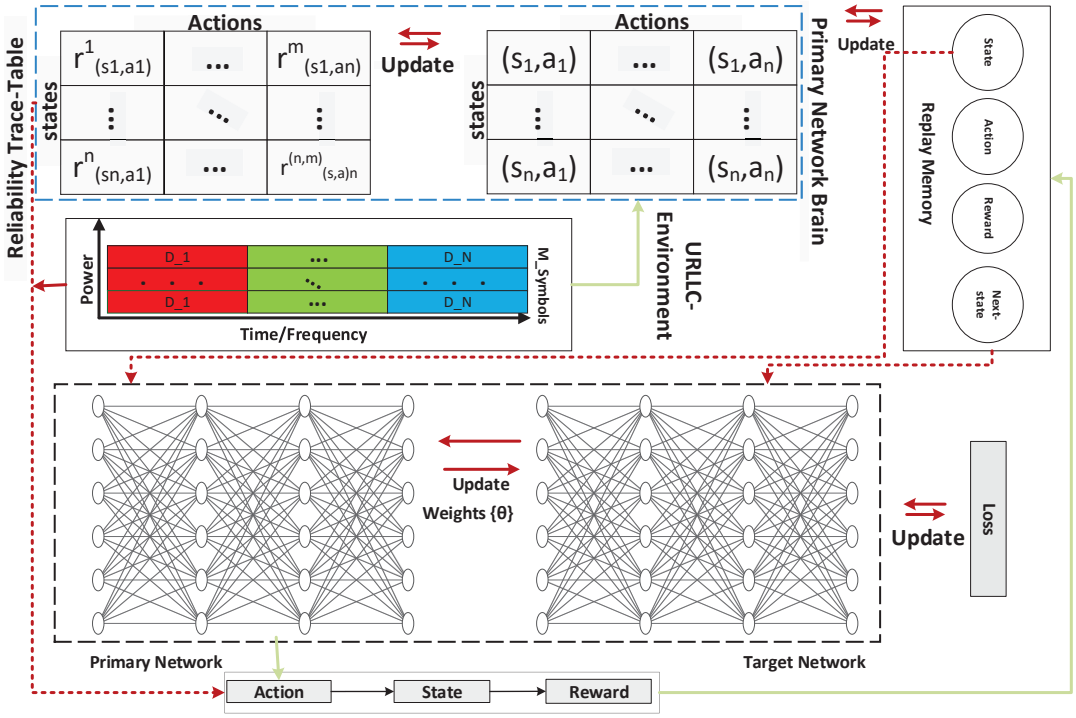


Fig. 2. Presenting the structure of the proposed reliable resource allocation algorithm. Illustrating the communication environment where the Deep SARSA$-\lambda$ is invoked. Additionally, it shows the reliability assistance mechanism between primary DNN and the target DNN to maximise the reward by utilizing replay memory to minimise the DNN loss function.

MRP (SARSA-λ) benefits from combined properties of the dynamic λ and the tracing table $\tau(s,a)$. Because the tracing table $\tau(s,a)$ is updated using dynamic λ over (n)-steps, that is defined as $\lambda \in [0,1]$. When the value of λ is large, it indicates the importance/reliability of current state and action pairs is high, and vice versa. Similarly, due to the dynamicity of λ and tracing property, the agent/s becomes (n)-step learner/s. Therefore, the magnitude of parameter λ assists the agent/s (BS) at time slot (t) to prioritise its short-term experience. Hence, it ensures that the agent (BS) is more reliable by careful assistance. Furthermore, if the $\lambda = 0$ then the agent decides action $a(t)$ without any experience that results in less reliability for the future actions.

Remark 3. *For the fixed λ, the agent performs MRP based on the current Q-values and current reward only. However, when the λ is dynamic the agent performs short-term experience tracing based on multiple time-slots to allocate resources in the future. Hence, by keeping the λ dynamic the agent knows the summary of the past (n)-steps for the long-term future allocation strategies.*

3) DNN introduced long-term control: As it is known that general MRP consists of observations/states, actions, rewards, policies, environment, and RL agent/s. The agent consists of Q-table and DNN θ weights that represent the brain of one-step learning agents. Similarly, the probabilities for each state and action pair are learned via Q-table $\mathbf{Q}(s,a)$ and DNN θ weights. Therefore, Q-table $\mathbf{Q}(s,a)$ and DNN θ weights are updated during the MRP. The overall MRP is achieved in two main parts namely, short term and long-term learning.

In the short-term learning process, the agent performs actions to maximise the immediate reward feedback. That helps the agent/s to learn the long-term policy in the form of a set of actions for each state of an agent, represented with the help of probabilities stored in Q-table $\mathbf{Q}(s,a)$ and DNN θ weights. Consequently, in the result of the long-term optimal policy, the RL agent is expected to gain more long-term rewards. This process is only for the traditional one-step design where the agent can only learn and update probabilities once for each timestep (t_s). Due to this, the one step learning agent can revisit the useless states during the MRP that costs poor convergence. On the other hand, different from one step learning the (n)-step design combines the brain of the agent with trace table $\tau(s,a)$ to learn multiple actions and update multiple the probabilities at once. Hence, on the cost of the trace table $\tau(s,a)$ the (n)-step learning agent increases the speed of convergence and the reliability for the long-term learning policy.

Remark 4. *To speed-up learning the reliable allocation policy π^*, in addition to (θ) weights and tracing table $\tau(s,a)$ the Deep SARSA-λ agent (BS) is equipped with the short-term replay memory. The replay memory consists of online learning traces from the real-time network environment and tracing table $\tau(s,a)$. Consequently, an experienced Deep SARSA-λ agent performs efficient actions $a(t)$. As a result, the future*

action $a(t+1)$ selection policy π^* becomes more reliable for an agent to converge faster by avoiding exploration visits to less reliable states s.

Based on the above discussions, the following sub-section provides the design details for proposed resource allocation algorithm. The design consists of the basic design of RL agent, wireless network environment, state space $\mathbb{S}$, action space $\mathbb{A}$, reward function $D_r(t)$, transition probability $T(s(t), a(t), s(t+1), a(t+1))$, DNN design, and tracing mechanism.

B. Design of the SARSA-λ

- State Space $\mathbb{S}$: is a state space consists of finite set having dimensions N_s containing $N_u^{N_s}$ total number of states. Each state represents one sub-set of 2D associations among users and sub-channels for the BS.
- Action Space $\mathbb{A}$: is an action space consists of a finite set of actions to move the agent in a specific environment, the size of $\mathbb{A} = (N_s - 1) \times N_s$. Actions in this model are $[-1, 0, +1]$. The '-1' is to reduce one user from the current state of the agent (BS). Similarly, '$+1$' shows an increment in one user in the current state of the agent (BS). The last action '0' represents no change in the current state of the agent (BS). For example for one swap operation for both sub-channels and power levels, if the current state of an agent is indicating that N' users are connected via sub-channel j with $\rho_{N'}$ power levels used from it's pool ρ_j and the sub-channel $j+1$ consists of M' users with $\rho_{M'}$ power levels used from it's pool ρ_{j+1}. Note that the size of each pool is same and denoted by $\rho_{j+N_s} = N_u$. Similarly, when the agent performs action '1' for the sub-channel j and '-1' for the sub-channel $j+1$ to swap one user from sub-channel $j+1$ to sub-channel j and also swap and update the power pools from $\rho_{M'-1}$ to $\rho_{N'+1}$. The result of this action is that in the sub-channel j, the number of users is from N' to $N'+1$ with new power level utility $\rho_{N'+1}$, while in the sub-channel $J+1$, the number of users is from M' to $M'-1$ with released power level $\rho_{M'-1}$. The last action '0' means no swap operation. Therefore, in results of the last action '0' the current state of the agent does not change (add/subtract). In this way the agent finds the final combination of state and action pair.
- reward Function $D_r(t)$: In the MRP, the agent tries to maximize the sum reward for each time slot (t) for the transition from current state action pair $\{s(t), a(t)\}$ to the future state and action pair $\{s(t+1), a(t+1)\}$. We use instantaneous sum rate as a reward for an agent during each time slot (t). The reward function is as follows:

$$D_r(t)=\begin{cases}\sum\limits_{k=1}^{N_u^j} R_k^j(t), & if\ \mathbb{E}\left[\sum\limits_{j=1}^{N_s}\sum\limits_{k=1}^{N_u^j}\varepsilon_k^j(t-1)\right]\geq\mathbb{E}\left[\sum\limits_{j=1}^{N_s}\sum\limits_{k=1}^{N_u^j}\varepsilon_k^j(t)\right] \\ & and\ \sum\limits_{j=1}^{N_s}\sum\limits_{k=1}^{N_u^j}c_k^j(t-1)=\sum\limits_{j=1}^{N_s}\sum\limits_{k=1}^{N_u^j}c_k^j(t) \\ 0, & \text{otherwise.}\end{cases} \tag{9}$$

where the reward $D_r(t)$ is equal to instantaneous sum rates $\sum\limits_{k=1}^{N_u^j} R_k^j(t)$ if the expectation sum of the function $\varepsilon_k^j(t)$ for the time slot (t) is less than the previous time slot $(t-1)$ and 0 for rest of the cases. While keeping all the connected with $\sum\limits_{j=1}^{N_s}\sum\limits_{k=1}^{N_u^j} c_k^j(t-1) = \sum\limits_{j=1}^{N_s}\sum\limits_{k=1}^{N_u^j} c_k^j(t)$ conditions.

- Value Function V_π: In MRP the value function V_π shows the expectation return in result of each state and action pair.

$$V_\pi = \mathbb{E}\{D_r(t)|s(t)=s, a(t)=a\}, \tag{10}$$

where V_π denotes the value function that is similar to long-term expected return.

- Transition Function $T(s(t), a(t), s(t+1), a(t+1))$: The state and action transition function shows the expectation probabilities, where the agent changes its current state to the future for the search of optimal state and action pair. As we can see from the Fig. 2, the replay memory is updated after each transition of current state of an agent to the future state in result of the action a. The transition function for the proposed method is as follows.

$$p(s', D_R|s, a) = \Pr\{s(t)=s', \tag{11}$$
$$D_r(t) = D_R(t)|s(t-1)=s, a(t-1)=a\},$$

where Pr is the probability distribution for the transition of state s and action a to the next state s' of the environment. The $(|)$ is to show the conditional probability.

- Q-Table $\mathbf{Q}(s,a)$: The Q-Table is a primary brain of an agent (BS), that keeps track of all the Q-values for each state and action pair, denoted by $\mathbf{Q}(s,a)$. The rows of a table means total number of states that is 2D associations in our case and the total number of actions are presented with the number of columns, as different swap operations.
- State and Action Trace-Table $\tau(s,a)$: The discount value γ and λ are mainly used to enhance the packet transmission reliability with the help of Trace-Table $\tau(s,a)$, due to multi-step return capability. The trace memory helps to improve the learning of an agent/s at each time slot (t), the trace memory is denoted with $\tau(s,a)$ that is initialised with 0 for every episode. Similarly, in this way an agent assisted with the trace memory $\tau(s,a)$ enhances the long-term reliability of the URLLC system. To reflect fully online behaviour, in this model we use replacing reliability traces and the reliability trace value update, which can be expressed as follows:

$$\tau(s,a) = \begin{cases}\gamma\lambda\tau(s(t-1), a(t-1)) + 1, \\ if\ s(t) = s(t+1)\ and\ a(t) = a(t+1) \\ 0, \text{otherwise,}\end{cases} \tag{12}$$

where γ and λ are used to discount the sequence of rewards for multiple state and action pairs. Using

$T(s(t), a(t), s(t+1), a(t+1))$ and $\mathbb{E}[\Sigma_{t=0}^{\infty}\gamma(t)D_r(t)]$ the agent/s (BS) targets to maximise the expected future return based on the discount factor ($\gamma \in [0, 1]$), because γ determines the importance between $D_r(t)$ and $D_r(t+1)$ at each timestep(t_s).

To combine the whole concept, in following sections we discuss the reliability from two significant aspects, the first aspect is the tracing method to remember the reliable path as reliable states $s(t)$ for the future visits at the time slot $(t+1)$. Similarly, the second aspect is based on the DQN part that is assisted by the trace-table $\tau(s, a)$ to further enhance the reliability. Further details are in following sections:

1) Trace and Allocation: The optimal policy of the aforementioned parameters can be discovered by an agent using following function:

$$\pi^*(s(t)) = \arg\max_{a(t)} Q(s(t), a(t)), \forall s(t) \in \mathbb{S}, \quad (13)$$

where $\pi^*(s)$ represents the optimal policy. This function provides the optimal policy value for each state s from the finite state set after taking appropriate action a. For a better understanding, the optimal policy can be defined:

$$V_{\pi^*}(s(t)) = \quad (14)$$

$$\max_{a(t)} \left[r(s(t), a(t)) + \gamma \sum_{s(t+1)} P_{s(t)\to s(t+1)} V_{\pi^*}(s(t+1)) \right]$$

$$q_n(t) = D_r(t+1) + \gamma D_r(t+2) + ... + \gamma_{n-1} D_r(t+n) + \gamma_n Q(s(t+n)), \quad (15)$$

where n denotes the length of the long-term reliability trace, $P_{s(t)\to s(t+1)}$ shows the transition probabilities from state $s(t)$ to the next state $s(t+1)$ and the following equitation illustrates the dynamic $\lambda(t) = \gamma\tau(s(t), a(t)) + \nabla V_\pi(s(t), a(t); \theta)$ based general value policy return (without-$\tau(s, a)$):

$$Q(s(t), a(t)) \leftarrow Q(s(t), a(t)) + \alpha[q_n(t) - Q(s(t), a(t))], \quad (16)$$

$$q^\lambda(t) = (1-\lambda)n \sum_{n=1}^{\infty} \lambda^{n-1} q_n(t), \quad (17)$$

where $(1-\lambda(t))$ is the normalisation factor to ensure that the maximum weights are not more than 1. Similarly, the term λ^{n-1} indicates the weighted proportionality of λ-returns over n-steps taken by the agent. The following a-values are based on (n)-step reliable trace mechanism, in (18) and (19):

$$Q(s(t), a(t)) = Q(s(t), a(t)) \leftarrow Q(s(t), a(t)) + \alpha\delta(t)\tau(s(t), a(t)), \quad (18)$$

$$\delta(t) = D_r(t+1) + \gamma Q(s(t+1), a(t+1)) - Q(s(t), a(t)), \quad (19)$$

The equation (20) shows the initialisation of $\tau(s, a)$ at initial timestep when $t_s = 0$.

$$\tau(s, a) = 0, \quad (20)$$

MRP (evolution from $s(t)$ to $s(t+n)$), the overall mapping process for MRP tuple is defines as: $\{(s(t), a(t), D_r(t)) : s(t+1) \sim \mathbb{P}(\cdot|s(t), a(t)), a(t) \sim \pi(\cdot|s(t)), D_r(t) \sim D_r(\cdot|s(t), a(t))\}, T \in [0, \infty]$.

2) Deep Q Network: The deep Q network is equipped with the reliability trace table $\tau(s, a)$, therefore the training network is trained to learn more reliable polices with the help of short experience memory. Similarly, following loss function indicates the accuracy to allocate resources with a reliable policy.

$$l(\theta_{train}^{DQN}) = 1/T \sum_{t=1/t\in T} [\omega^{DQN}(t) - Q(s(t), a(t); \theta_{train}^{DQN})]^2, \quad (21)$$

where

$$\omega^{DQN}(t) = D_r(t) + \gamma \max_{a(t+1)} Q(s(t+1), a(t+1); \theta^{target}(t+1)), \quad (22)$$

and ω_t^{DQN} is the target Q-values from target DNN. For the improved training, in general the update frequency of the target network θ^{target} is performed in slow manner. Due to this reason the target network remains fixed for the target network update threshold $T_e^{'}$.

The DRL agent uses gradient decent method as in (21) to reduce the prediction error by minimizing the loss function. The updating of θ is provided in (16), which is based on the outcome of new experience. The updating function for θ is defined in (27), namely DRL Bellman equation.

$$\theta^{DQN} \leftarrow \theta^{DQN} - [\omega^{DQN}(t+1) - Q(s(t), a(t); \theta^{DQN})] \times \nabla Q(s(t), a(t); \theta^{DQN}), \quad (23)$$

$$q_\pi(s(t), a(t)) = D_r(s(t+1), a(t)) + \gamma \sum_{s(t+1)} \sum_{a(t+1)} P_{s(t)\to s(t+1)}(a) q_\pi(s(t+1), a(t+1)), \quad (24)$$

$$q_{\pi^*}(s(t), a(t)) = D_r(s(t), a(t)) + \gamma \sum_{s(t+1)} P_{s(t)\to s(t+1)}(a) \max_{a(t+1)} q_{\pi^*}(s(t+1), a(t+1)), \quad (25)$$

where $\nabla Q(s(t), a(t); \theta^{DQN})$ shows the periodic changes in DNN weights and Q-values. Similarly, the function $q_{\pi^*}(s(t), a(t))$ shows Q-values and the long-term reward calculations for the DRL algorithm. That is based on the discount

factor γ and below mentioned optimal DRL policy π^*.

$$\pi^*(s(t)) = \arg\max_{a(t)} \left[q_{\pi^*}(s(t), a(t))\right], \ \forall \ s \in \mathbb{S}, \tag{26}$$

where $\pi^*(s(t))$ represents the optimal policy for the DRL algorithm. This function provides the optimal policy value for each state s from finite sate set after taking appropriate action a.

$$Q(s(t), a(t)) \leftarrow (1-\alpha)Q(s(t), a(t)) \tag{27}$$
$$+\alpha \left[r(s(t), a(t)) + \gamma \max_{a(t+1)} Q(s(t+1), a(t+1))\right],$$

where $Q(s, a)$ is showing Q-value update according to DRL Bellman equation and the following equation represents the inputs of DNN.

$$s(t) = \{a(t)(D_r(t), D_r(t-1)), \tag{28}$$
$$a(t)(D_r(t), D_r(t-1)), \ldots, a(T_t)(D_r(T_t), D_r(T_t-1))\},$$

$$\varrho(\vartheta) : \sum_{h=0/h\in d_h} W_h \times I_h(s(t)) + B_h, \tag{29}$$

where ϱ represents activation function with the input as (ϑ), that is being processed by the hidden layer h with density of neurons d_h. Secondly, W_h is weight of the hidden layer h and I shows the inputs plus B is bias term. It is worth noting that the maximum reward budget for each BS is defined as $D_r(\max) \in [0, \inf]$.

C. Algorithmic Details and Descriptions

This subsection, presents the deep SARSA-λ algorithm in **Algorithm. 1** to provide (n)-step reliable return for the uplink NOMA-URLLC, while guaranteeing the minimum average system utility as well. In the deep SARSA-λ base model, the central control is done by the BS that acts as a deep SARSA-λ agent. At each trial the agent (BS) starts observing the current state of a network environment. Which contains the initial 2D association among users, sub-channels for BS denoted by $s(t)$. According to the $s(t)$ and current policy (initial), the agent takes an action $a(t)$ from the action space $\mathbb{A}$. Which represents swapping operation of users from one resource block to an other. After taking the action $a(t)$ by following the greedy policy $\epsilon - greedy$ the agent (BS) receives a feed back reward $D_r(t)$ based on the long-term URLLC objectives (8a). Therefore, at each trial after observing the state s, taking an action a and feed back $D_r(t)$ the Q-value $\mathbf{Q}(s, a)$ and $\tau(s, a)$ is calculated. In this way, to find the long-term policy the reliability trace and Q-values are updated for each trial. Similarly, each episode represents the long-term optimality of state and action pairs with Q-values and reliability trace $\tau(s, a)$ as an optimal policy. The main target of the agent is to minimise the long-term error and maximise the data-rate in the form of $D_r(t)$. That is based on the instantaneous average system utility $D_r(t)$.

Finally for the deep SARSA-λ, according to $\epsilon - greedy$ and the $\tau(s, a)$ the action $a(t)$ may not be the best choice but for the long-term the optimality of an action is high to achieve long-term reliable allocation policy. In this work, the long-term allocations are considered under continuously changing the URLLC constraints such as packet size$-D$ and user connectivity. Therefore, the exploited scheme is fully online on policy. Base on the $s(t), a(t), D_r(t), s(t+1), a(t+1)$ tuple, the updates of $\mathbf{Q}(s, a)$ are mainly dependent on two significant parts described in the following sections.

1) DNN: The proposed framework is based on uplink NOMA-URLLC scenarios, therefore the state space is huge $\mathbb{S}$ from which state observations are used as an input to DNN's. The frequent error propagation happens, that results in low reliability. In order to increase the reliability with the minimum complexity, the output of the neural network as θ weight approximation is performed with following DNN components:

- Structure of DNN: We adopted the fully connected DNN structure, the fully connected model is a mapping function of $\mathbb{R}_{layer-n} \rightarrow \mathbb{R}_{layer-m}$. In this way, a fully connected DNN structure learns from all the features of neural network in distributed manner. Therefore, DNN model is applied to explore the wireless network environment efficiently. Lastly, we use sparse activation of neurons to avoid non useful neuron activations [39].
- Sparse-Activations: We use ReLU activation function fully connected DNN structure to achieve sparsity. Sparsity is one of the main advantage of neuron activations. With this design we minimise the useless neuron activations that helps to reduce the training randomness of neural networks. Secondly, ReLU activations also suffer very less from the log likelihood of the inputs because ReLU saturate in only one way (forward tracing) [40].
- Experience-Memory-κ: In conventional DNN's the experience is initialised with randomly taken actions to store in the memory κ as an experience. These experiences are based on $s(t), a(t), D_r(t)$ and used as training samples to minimise the training loss between target networks and training networks (21).
- Optimiser-(Loss): The adaptive momentum (ADAM) optimiser is used to approximate the neural network weight training problem by minimising the MSE loss. It is known that ADAM is computationally efficient loss function approximation. Additionally, the proposed (n)-step return method produces long-term (n)-step gradients with sparsity. The step-size annealing property od ADAM is suitable for the proposed design.

In (29) a basic neuron activation is represented using activation function ϱ that is based on ReLU activations then for each neuron the sparse output will be as:

$$\varrho(\vartheta) = \begin{cases} \vartheta, & \vartheta \geq 0, \\ 0, & \vartheta < 0, \end{cases} \tag{30}$$

The convergence of ADAM can be expressed with the following regret function:

$$\kappa(T) = \sum_{t=1}^{T} [\psi(t)(\omega_{adam}(t)) - \psi(t)(\tilde{\omega}_{adam})], \tag{31}$$

where $\tilde{\omega}_{adam} = \arg\min_{\omega\in\chi'} \sum_{t=1}^{T} \psi(t)(\omega_{adam}(t))$ and $\psi(t)$ is

the loss value to calculate the adam weights and gradients.

2) Reliability Tracing: The proposed framework is initialised with the random network parameters to get initial inputs for deep SARSA-λ learning. That includes hyperparameters for learning process, such as T, T_t, γ and $\epsilon - greedy$. After random initial user, BS and sub-channel associations, the reliable deep SARSA-λ framework begins to learn the online traces using (20). We use the $\tau(s,a)$ mechanism to allocations mechanism and to fill the memory κ for the future deep SARSA-λ experience replay initialisations instead of random initialisations. Therefore, when the batch is full of reliable experiences from the wireless network environment, then for the further allocations DNN is invoked to predict and allocate. This process is adopted to minimise/avoid the error propagation in DNN's due to the randomness of the initial policy. Therefore, our designed model is based on two networks namely, target network θ^* and primary network θ. The function of a target network is to train the primary network online by providing the best allocation policy that assists the primary network to learn and compare the correctness of current and future allocations. Similarly, to validate the current allocation policy error minimisation is performed to get the final long-term outcomes. The memory κ updates are based on reliable actions $\tau(s,a)$ then the loss for (n)-step experience-based DNN's can be expressed as follows:

$$l(\theta^{DQN}_{train}) = (s(t), a(t), D_r(t), s(t+1), a(t+1), \tau(s,a)) \sim (\kappa) \{1/T \sum_{t=1/t\in T} [\omega^{DQN}(t) - Q(s(t), a(t); \theta^{DQN}_{train})]^2\}. \quad (32)$$

D. Analysing Proposed Framework

1) Convergence analysis: Prior to start discussion on convergence analysis of the proposed deep SARSA-λ, it is necessary to discuss/know the convergence details for traditional reinforcement learning algorithms. Subsequently, following by the discussion on convergence of traditional reinforcement learning algorithms the convergence for the proposed model can be easily verified. The traditional reinforcement learning, that is (1)step learning algorithm converges by satisfying $\alpha \in \{0,1\}$ to the wel-known Q-function based on bellman equations. The detailed prove is already given in [41].

The deep SARSA$-\lambda$ algorithm uses λ and $\tau(s,a)$ to learn the best path for the future allocation policy that results in faster learning. Similarly, combining the additional power of DNN proposed model is capable to converge even faster to the optimal state and action pair.

2) Complexity analysis: The complexity of proposed framework is based on the number of user communicating and the total number of clusters/resource blocks. The experiments in our propose framework are performed with two different examples, for the first case $\{N_s = 5, N_u = 5\}$ and for the second case $\{N_s = 5, N_u = 7\}$. The computation complexity for our framework is based on the trace-table $\mathcal{O}(\tau(\mathbb{S} \times \mathbb{A}))$, $\mathbb{S} = \{N_u^{N_s}\}$ total number of states, and $\mathbb{A} = (N_s - 1) \times N_s$ represents total number of actions for each state. Therefore, the total complexity is as (trace-table + Weighted matrix of DQN), as $\mathcal{O}(\tau(\mathbb{S} \times \mathbb{A}) + \mathbf{Q}(s,a;\theta^{DQN}))$. Where the complexity

Algorithm 1 Deep SARSA$-\lambda$ with Reliability Tracing Mechanism

1: Inputs:
   1) T Max-Episodes
   2) T_t Max-Trial
   3) α Step-size
   4) γ Discount-factor
   5) $\epsilon - greedy$ Policy
2: Initialization parameters:
   1) Communication system $(M, D, \mathbf{C}, \mathbf{P}, N_s, s_j, N_u, u_k)$
   2) Q-Table $\mathbf{Q}(s,a)$
   3) Trace-Table $\tau(s,a)$
   4) Memory κ
   5) $\lambda = 0.99$
3: $\mathbb{S}$, $\mathbb{A}$
4: Random user association to BS and any c
5: **for** t_s = 1:T **do**
6: $s_1 = rand()$
7: Random power allocations P
8: **for** j = t:T_t **do**
9: $s(t), a(t)$
10: Update $D_r(t)=\begin{cases} \sum_{k=1}^{N_u^j} R_k^j(t), & if\ \mathbb{E}\left[\sum_{j=1}^{N_s}\sum_{k=1}^{N_u^j} \varepsilon_k^j(t-1)\right] \geq \mathbb{E}\left[\sum_{j=1}^{N_s}\sum_{k=1}^{N_u^j} \varepsilon_k^j(t)\right] \\ & and\ \sum_{j=1}^{N_s}\sum_{k=1}^{N_u^j} c_k^j(t-1) = \sum_{j=1}^{N_s}\sum_{k=1}^{N_u^j} c_k^j(t) \\ 0, & \text{otherwise.} \end{cases}$
11: compute $q_{\pi^*}(s(t),a(t)) = D_r(s(t),a(t)) + \gamma \sum_{s(t+1)} P_{s(t)\to s(t+1)}(a) \max_{a(t+1)} q_{\pi^*}(s(t+1), a(t+1))$.
12: compute $\omega^{DQN}(t) = D_r(t) + \gamma \max_{a(t+1)} Q(s(t+1), a(t+1); \theta^{target}(t+1))$, and $l(\theta^{DQN}) = 1/T \sum_{t=1}^{T} [\omega^{DQN}(t) - Q(s(t),a(t);\theta^{DQN})]^2$,
13: $(s(t), s(t+1)), (a(t), a(t+1))$
14: **if** $T_t > State - size$ **then**
15: fetch $(s(t), s(t+1)), (a(t), a(t+1))$ from previous Experience $\tau(s,a)$
16: **end if**
17: **end for**
18: **end for**
19: **for** i = 1:κ **do**
20: **for** j = 1:κ **do**
21: $s(t), a(t)$
22: Update $D_r(t)=\begin{cases} \sum_{k=1}^{N_u^j} R_k^j(t), & if\ \mathbb{E}\left[\sum_{j=1}^{N_s}\sum_{k=1}^{N_u^j} \varepsilon_k^j(t-1)\right] \geq \mathbb{E}\left[\sum_{j=1}^{N_s}\sum_{k=1}^{N_u^j} \varepsilon_k^j(t)\right] \\ & and\ \sum_{j=1}^{N_s}\sum_{k=1}^{N_u^j} c_k^j(t-1) = \sum_{j=1}^{N_s}\sum_{k=1}^{N_u^j} c_k^j(t) \\ 0, & \text{otherwise.} \end{cases}$
23: Calculate $Q(s(t),a(t)) \leftarrow Q(s(t),a(t)) + \alpha[q_n(t) - Q(s(t),a(t))]$
24: update $\lambda(t) = \gamma\tau(s(t),a(t)) + \nabla V_\pi(s(t),a(t);\theta)$
25: Calculate $\tau(s,a) = \begin{cases} \gamma\lambda\tau(s(t-1),a(t-1)) + 1, \\ if\ s(t) = s(t+1)\ and\ a(t) = a(t+1) \\ 0, \text{otherwise,} \end{cases}$
26: Search $\pi^*(s)$
27: $T(s(t),a(t),s(t+1),a(t+1))/p(s', D_r|s,a) = \Pr\{s(t) = s', D_R(t) = D_r(t) | s(t-1) = s, a(t-1) = a\}$
28: $\tau(s,a) \leftarrow \gamma\lambda\tau(s,a)$
29: $(s(t), s(t+1)), (a(t), a(t+1))$
30: **end for**
31: $\tau(s,a) \rightarrow DQN(\theta)$
32: Add new Experience$\rightarrow \kappa$
33: **end for**
34: $\tau(s,a)$
35: Return Reliable Weighted Allocation θ

of trace table is based on the number of states and action that is calculated as $\mathcal{O}(\tau(\mathbb{S} \times \mathbb{A}))$. The weighted matrix is experience memory that is based on the experiences from (28). Therefore the worst case complexity of the weighted matrix is computed as $\mathcal{O}((\theta))$. Furthermore, the complexity of the bias and weighted terms are calculated separately as $\mathcal{O}(|\mathbb{S}|d_h(|\mathbf{B_h}| + |\mathbf{W_h}|))$, with d_h total number of neurons in each layer.

TABLE II
NETWORK SETTINGS

Max Trials	500
Max Time steps	500
Total BSs	1
Total resource blocks	5
Experience Memory	500
Batch size	500
Optimiser	Q-table, ADAM
Hidden layers	2
Number of neurons in each layers	500
Neuron activation	ReLu
Type of DNN structure	Fully-Connected (Fig.2)
DNN-performance	Loss-function
Dynamic-Reward	Data-rate (t)
Loss-calculations	MSE (mean squared error)
Optimiser-batch	500
Max user for each resource block	(5), (7)
Bandwidth	1 MHz
α	4
Traffic Types based on D	8
σ^2	-174
γ_δ	0.6
Learning rate α_r	0.75
$\epsilon - greedy$	0.01
(n)-step λ	Dynamic

IV. SIMULATION RESULTS

The main aim of this section is to show the validity of a proposed framework that is based on the Deep SARSA$-\lambda$. Similarly, the convergence of the proposed Deep SARSA$-\lambda$ algorithm illustrates the validity of the performance enhancement. An efficient convergence of the Deep SARSA$-\lambda$ agent proves the long-term reliable resource allocation assistance in uplink NOMA-URLLC networks. Furthermore, to illustrate the effectiveness we test the performance of the proposed algorithm with 4 communication scenarios. The first scenario is based on NOMA-URLLC network with bursty network traffic, in which the packet size for each user is variable. The second scenario is a fixed packet NOMA-URLLC network that consists of a fixed packet size for each user. Similarly, the third and fourth scenarios are based on different network loads, such as the different number of users connected to each resource block. Therefore, for the third scenario, we allow $(2-5)$ and in the fourth scenario, the network load is $(2-7)$ for each resource block. Additionally, in simulations at the initial timestep, the random allocation is adopted to start the allocation enhancement framework. Lastly, TABLE II contains the details for the simulation parameters.

Our simulation setup was based on Intel core i7-7700 CPU with 3.60 GHz frequency having 16 GB of RAM (Random Access Memory) and 64-bit operating system (windows-10). All the experiments are simulated using Python 3.6. From TABLE II for both the algorithms we used 500 timesteps with 500 timeslots for each episodes.

A. Convergence Rate (D_r)

Fig. 3 characterizes the inter-correlations between achieved rewards and the number of episodes. In this figure, we compare the proposed algorithm with conventional Q-learning and SARSA Q-learning. It is apparent from the figure that the convergence of the traditional Q-learning and SARSA Q-learning algorithm is poor. This is due to the extremely huge state and action space for the time varying wireless network model. Therefore, the traditional RL agents are presenting unreliable random outcomes for the bursty network traffic. Due to the power of DNN, $\tau(s, a)$, and dynamic λ, the proposed deep SARSA-λ algorithm is able to converge with better rewards in 200 episodes for NOMA network environments. In the start, the agent is trying to explore the environment by taking random actions. Due to this, the performance is worst in this case. Similarly, in 100 episodes the agent has gained significant experience to exploit the environment. Therefore, in 100 episodes the reward increases rapidly and after 200 episodes the reward is more stable. Which represents the stable performance of the proposed agent with better rewards after gaining further experiences from the environment. Additionally, this figure also shows the inverse effect of the bursty network traffic on convergence. On the other hand, convergence is much faster and better for static network traffic with fixed packet size. This is because a dynamically changing network increases the overall allocation complexity. Lastly, the proposed reward function consists of real-time reward feedback that is represented by the average rate for all NOMA-URLLC users. Moreover, the scale of the state and action space is based on the maximum connections allowed for each resource blocks and total number of available resource blocks. Therefore, it is denoted by $\mathbb{S} = \{N_u^{N_s}\}$ and $\mathbb{A} = (N_s - 1) \times N_s$ represents total number of actions for each state.

B. Long-Term Error Performance With respect to The Symbol Rate

In Fig. 4 obtained outcomes summarise some of the main characteristics of the long-term relationship among network error probability, symbol rate (M), network traffic density, type of traffic, OMA, NOMA, benchmark schemes, and the proposed algorithm. The most striking result to emerge from the figure is that, for the network density of (5) users, the average error is less than 7 users. Similarly, for the bursty type of traffic when the symbol rate is smaller than 110 the error rate is efficient for 5 users. For 7 users, the error rate is higher for the bursty traffic for all the cases except $M \leq 130$. Therefore, the results obtained from the preliminary analysis of these parameters indicate that the mean error of the FBL transmission is directly proportional to the network density and inversely proportional to the symbol rate (M). It is apparent from this figure that for all settings when the symbol rate (M), is increasing then the average error is decreasing and

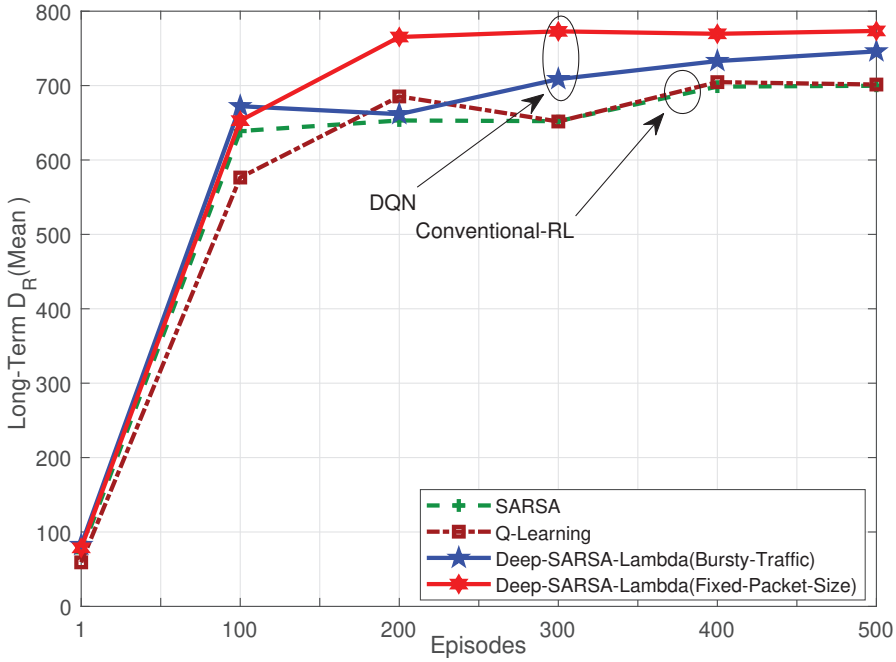


Fig. 3. Illustrating the comparison between two convergence parameters, long-term mean decoding error and long-term reward (expectation-sum-rate) over the number of episodes. Presenting the convergence of deep SARSA-λ framework with variable network traffic (bursty) in blue coloured plot, static (fixed packet size) network traffic in red coloured plot, traditional Q-learning in dotted maroon coloured line and SARSA Q-learning with green line. Where the network is based on 1 BS, $N_s = 5$ sub-channels, and each sub-channel can serve maximum of $N_s = 5$ network users at most.

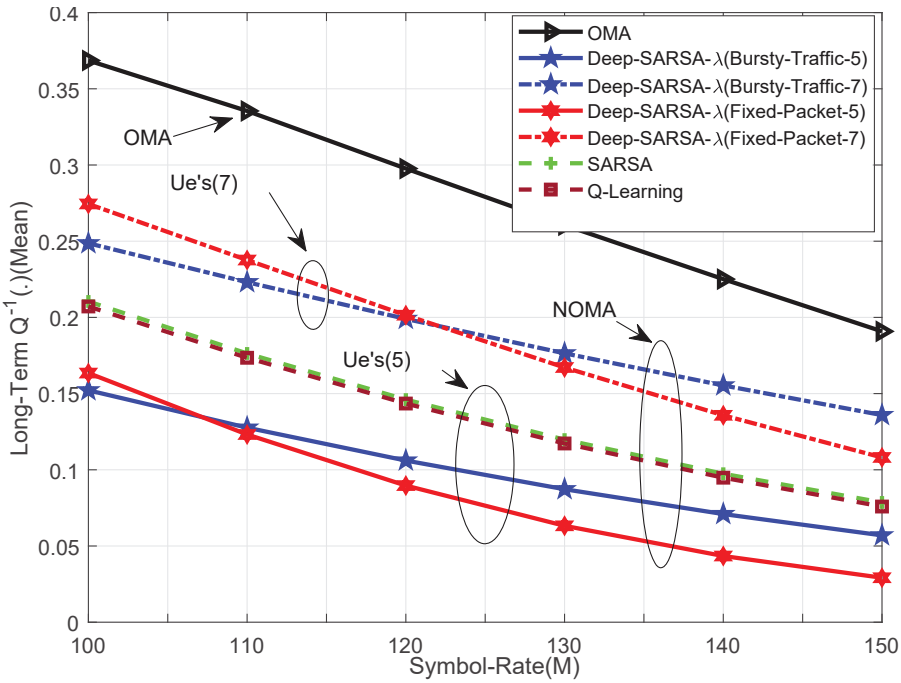


Fig. 4. Showing the comparison among decoding error (mean), symbol rate (M), type of network traffic, network density, NOMA and conventional OMA for deep SARSA-λ, Q-learning and SARSA Q-learning.

vice versa. Interestingly, the performance for the long-term mean error probability of the deep SARSA-λ algorithm is more efficient than traditional Q-learning methods. Similarly, as compared to conventional OMA for all network settings the proposed learning-based NOMA system is performing efficiently.

C. Long-Term Error Rate For The Packet Size and Symbol Rate

In Fig. 5 we present the breakdown of long-term error, variable packet size (D), and symbol rate (M). This figure is quite revealing in several ways. Firstly, the average error rate for the NOMA-URLLC is much better than the conventional

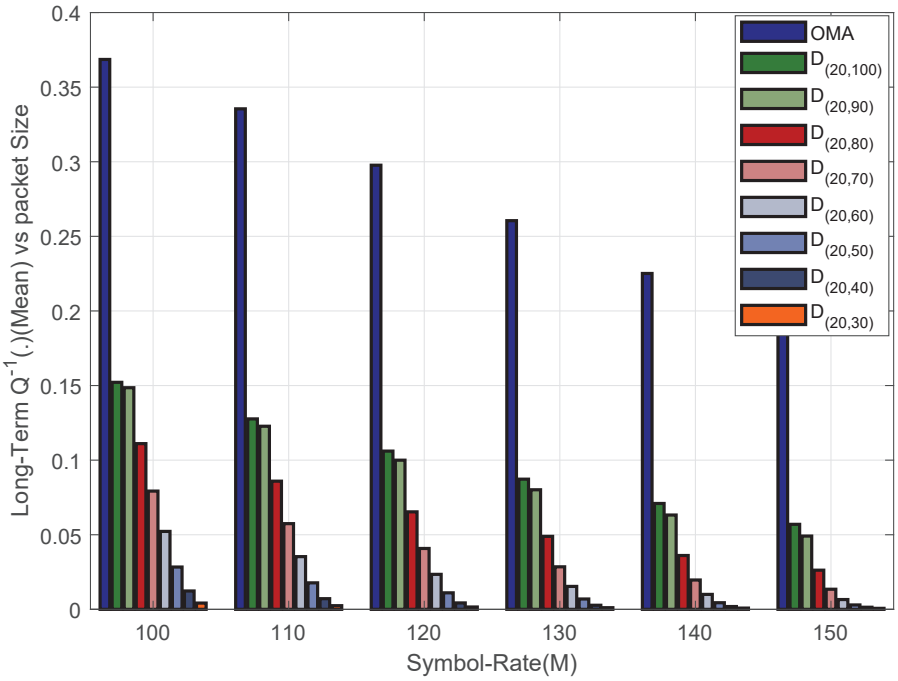


Fig. 5. Presenting the comparison among, packet size D, symbol rate (M), decoding error (mean) with NOMA and OMA techniques.

OMA-URLLC scheme. Secondly, the average error rate is decreasing when the symbol rate is increasing. Similarly, when the symbol rate is decreasing, then the average error starts increasing. Lastly, the packet size D is also inversely proportional to the long-term error. Due to this, when the packet size (D) increases from $(20, 30)$ to $(20, 100)$ bits then the error rate starts increasing.

D. Long-Term Error Performance With σ^2 and Symbol rate

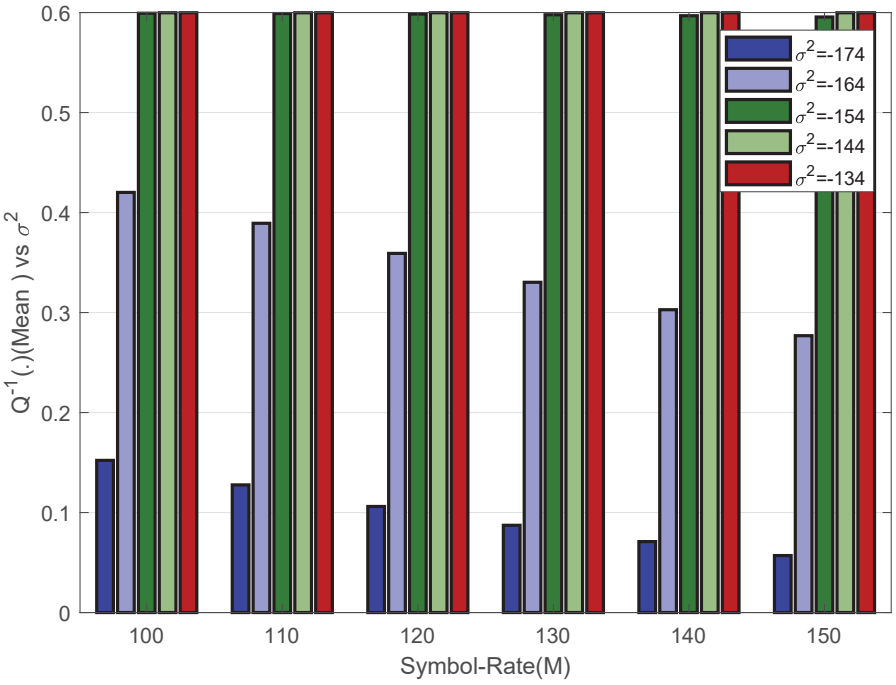
In Fig. 6 we plot the summary statistics for the comparison among long-term error,σ^2, and Symbol rate. As the figure portrays, there is a significant difference between the two groups, when $(\sigma^2 = -174)$ and $(\sigma^2 \geq -164)$. The average error is small for the case when $\sigma^2 = -174$ but it increases more than 50% for $\sigma^2 \geq -164$ case. Therefore, if the noise is stronger then -164, it results in high error rates. Therefore, the results from this figure indicate that regardless of the increased symbol rate (M) the blockage starts when $\sigma^2 \geq -164$ as a result, the average error increases significantly.

E. Loss Function

The results in Fig. 7 compares the preliminary analysis of DNN loss function (MSE), episodes, network density and type of traffic. From this figure, we can see that the prediction loss for the static network traffic with low density is less as compared to the other cases. Therefore, by converging within 100 episodes, the loss stays below 42 during the whole simulation time. Similarly, the DNN prediction loss starts increasing when the type of traffic switches to burst traffic, which is due to the increased network complexity. Which also becomes higher for the increased network density. Therefore, users at each resource block are increased from (5) users to (7) users. The DNN prediction loss is increasing. However, due to the power of DNN reliable trace $\tau(s, a)$ and sparse activations, the proposed DNN structure is capable to handle network dynamics at the cost of negligible performance. The proposed framework provides an efficient long-term allocation policy for different types of networks with (109) maximum average loss.

F. Clustering Time

In Fig. 8 we plot the comparison of the clustering time, network density and type of traffic. Similarly, the top sub-figure compares the clustering time between network density and episodes. In the top sub-figure, the clustering time starts increasing when the network load for each resource block is increased. Hence, there is proportional behaviour between network density and clustering time. The bottom sub-figure illustrates the clustering time comparison between episodes and type of traffic. It is quite revealing that the bursty network traffic scenario costs more time due to complex resource allocations. However, static network traffic scenarios are showing opposite behaviour with less clustering time. Therefore, the type of traffic is inversely proportional to the DNN loss (MSE) and clustering time, as shown in Fig. 7. Lastly, when we compare both of the sub-figures, another interesting trend depicts that the clustering time increases simultaneously by changing the type of traffic. However, the clustering time is low for increased network loads. Therefore, we can say that change in network traffic costs more clustering time and allocation complexity as compared to the variable network loads.



(a)

Fig. 6. Illustrates the comparison of decoding error (mean) with σ^2 and symbol rate (M).

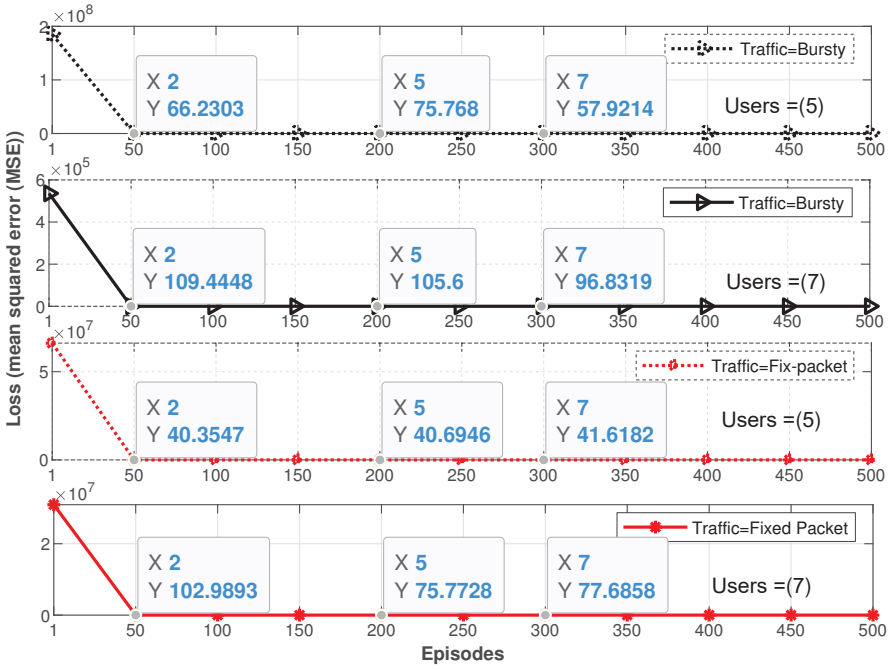


Fig. 7. Shows the mean squared error (MSE) as a DNN loss function for two different network densities having (5) and (7) maximum users for each sub-channel with two different types of network traffic.

V. Conclusion

In this paper, we proposed a long-term mean error minimization framework for uplink NOMA-URLLC networks. For all considered URLLC settings, NOMA is a more efficient scheme than traditional OMA. In particular, NOMA achieved 70% long-term mean error performance gain over the OMA case. With the aid of the reward function, the proposed DRL approach outperforms traditional models within 200 episodes. To prove the efficiency of our solution, we compared the long-term mean error performance of the proposed deep SARSA-λ with two well-known reinforcement learning algorithms namely, Q-learning and SARSA Q-learning. The results indicated that the proposed approach achieves 37% long-term mean error performance gain over the traditional Q-learning

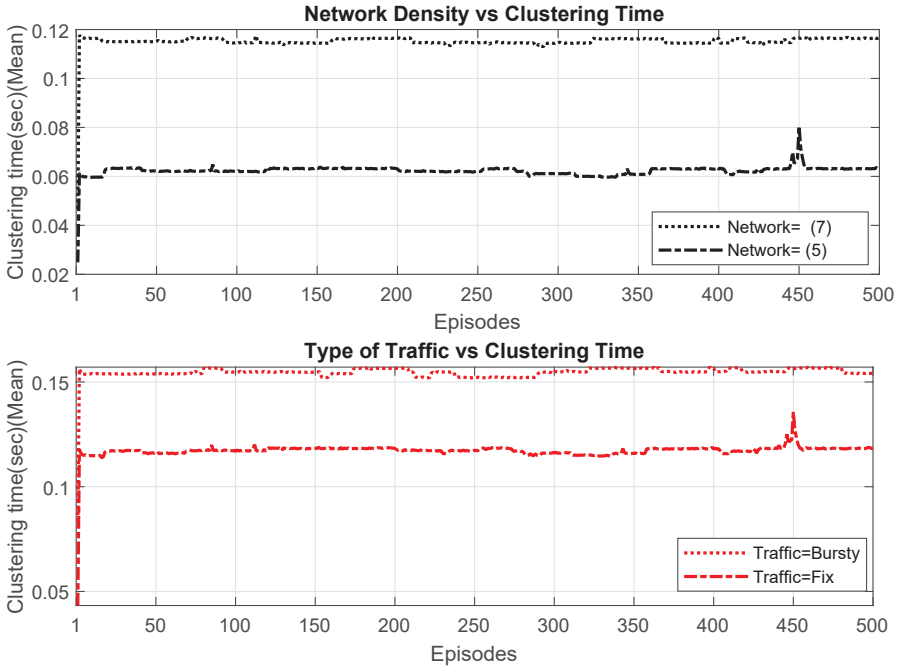


Fig. 8. Shows the mean clustering time (sec) for two different network loads and two different types of network traffic.

and 38% better than the SARSA Q-learning algorithm.

REFERENCES

[1] W. Ahsan, W. Yi, Y. Liu, and A. Nallanathan, "Reliable reinforcement learning based NOMA schemes for URLLC," in *IEEE Glob. Commun. Conf. (GLOBECOM)*, Dec. 2021.

[2] M. Elbayoumi, M. Kamel, W. Hamouda, and A. Youssef, "NOMA-assisted machine-type communications inUDN: State-of-the-art and challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 1276–1304, 2020.

[3] Z. Xiang, W. Yang, Y. Cai, Z. Ding, Y. Song, and Y. Zou, "NOMA-assisted secure short-packet communications in IoT," *IEEE Wireless Commun.*, vol. 27, no. 4, pp. 8–15, 2020.

[4] X. Zhang, L. Yang, Z. Ding, J. Song, Y. Zhai, and D. Zhang, "Sparse vector coding-based multi-carrier NOMA for in-home health networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 2, pp. 325–337, 2021.

[5] L. Qian, Y. Wu, F. Jiang, N. Yu, W. Lu, and B. Lin, "NOMA assisted multi-task multi-access mobile edge computing via deep reinforcement learning for industrial internet of things," *IEEE Trans. Ind. Informat.*, pp. 1–1, 2020.

[6] W. Yi, Y. Liu, A. Nallanathan, and M. Elkashlan, "Clustered millimeter-wave networks with non-orthogonal multiple access," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4350–4364, Jun. 2019.

[7] G. Gui, M. Liu, F. Tang, N. Kato, and F. Adachi, "6G: Opening new horizons for integration of comfort, security and intelligence," *IEEE Wireless Commun.*, vol. 27, no. 5, pp. 126–132, 2020.

[8] X. Jiang, Z. Pang, M. Zhan, D. Dzung, M. Luvisotto, and C. Fischione, "Packet detection by a single OFDM symbol in URLLC for critical industrial control: A realistic study," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 4, pp. 933–946, 2019.

[9] M. Khoshnevisan, V. Joseph, P. Gupta, F. Meshkati, R. Prakash, and P. Tinnakornsrisuphap, "5G industrial networks with CoMP for URLLC and time sensitive network architecture," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 4, pp. 947–959, 2019.

[10] L. Xiang, R. G. Maunder, and L. Hanzo, "Concurrent OFDM demodulation and turbo decoding for ultra reliable low latency communication," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 1281–1290, 2019.

[11] C. She, C. Yang, and T. Q. Quek, "Radio resource management for ultra-reliable and low-latency communications," *IEEE Commun. Mag.*, vol. 55, no. 6, pp. 72–78, 2017.

[12] Y. Polyanskiy, H. V. Poor, and S. Verdú, "Channel coding rate in the finite blocklength regime," *IEEE Trans. Inf. Theory*, vol. 56, no. 5, pp. 2307–2359, 2010.

[13] W. Yang, G. Durisi, T. Koch, and Y. Polyanskiy, "Quasi-static multiple-antenna fading channels at finite blocklength," *IEEE Trans. Inf. Theory*, vol. 60, no. 7, pp. 4232–4265, 2014.

[14] A. Shahraki, M. Abbasi, M. Piran, M. Chen, S. Cui *et al.*, "A comprehensive survey on 6G networks: Applications, core services, enabling technologies, and future challenges," *arXiv preprint arXiv:2101.12475*, 2021.

[15] C. Xiao, J. Zeng, W. Ni, X. Su, R. P. Liu, T. Lv, and J. Wang, "Downlink MIMO-NOMA for ultra-reliable low-latency communications," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 4, pp. 780–794, 2019.

[16] M. Amjad and L. Musavian, "Performance analysis of NOMAfor ultra-reliable and low-latency communications," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, 2018, pp. 1–5.

[17] R. Kotaba, C. N. Manchon, N. M. K. Pratas, T. Balercia, and P. Popovski, "Improving spectral efficiency in URLLC via NOMA-Based Retransmissions," in *Proc. IEEE Int. Commun. Conf. (ICC)*, 2019, pp. 1–7.

[18] J. Chen, L. Zhang, Y.-C. Liang, and S. Ma, "Optimal resource allocation for multicarrier NOMA in short packet communications," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2141–2156, 2019.

[19] H. Ren, C. Pan, Y. Deng, M. Elkashlan, and A. Nallanathan, "Joint power and blocklength optimization for URLLC in a factory automation scenario," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 1786–1801, 2020.

[20] O. Maraqa, A. S. Rajasekaran, S. Al-Ahmadi, H. Yanikomeroglu, and S. M. Sait, "A survey of rate-optimal power domain NOMA schemes for enabling technologies of future wireless networks," *arXiv preprint arXiv:1909.08011*, 2019.

[21] H. Huang, Y. Yang, Z. Ding, H. Wang, H. Sari, and F. Adachi, "Deep learning-based sum data rate and energy efficiency optimization for MIMO-NOMA systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 8, pp. 5373–5388, 2020.

[22] N. Ye, X. Li, H. Yu, A. Wang, W. Liu, and X. Hou, "Deep learning aided grant-free NOMA toward reliable low-latency access in tactile Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 15, no. 5, pp. 2995–3005, 2019.

[23] L. Huang, L. Zhang, S. Yang, L. P. Qian, and Y. Wu, "Meta-learning based dynamic computation task offloading for mobile edge computing networks," *IEEE Commun. Lett.*, vol. 25, no. 5, pp. 1568–1572, 2021.

[24] N. Ye, X. Li, H. Yu, L. Zhao, W. Liu, and X. Hou, "DeepNOMA: A unified framework for NOMA using deep multi-task learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2208–2225, 2020.

[25] Y. Fu, W. Wen, Z. Zhao, T. Q. Quek, S. Jin, and F.-C. Zheng, "Dynamic power control for NOMA transmissions in wireless caching networks," *IEEE Wireless Commun. Lett.*, vol. 8, no. 5, pp. 1485–1488, 2019.

[26] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5G: RAN, core network and caching solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3098–3130, 2018.

[27] A. Azari, M. Ozger, and C. Cavdar, "Risk-aware resource allocation for URLLC: Challenges and strategies with machine learning," *IEEE Commun. Mag.*, vol. 57, no. 3, pp. 42–48, 2019.

[28] C. She, C. Sun, Z. Gu, Y. Li, C. Yang, H. V. Poor, and B. Vucetic, "A tutorial of ultra-reliable and low-latency communications in 6G: Integrating theoretical knowledge into deep learning," *arXiv preprint arXiv:2009.06010*, 2020.

[29] J. F. Hernandez-Garcia and R. S. Sutton, "Understanding multi-step deep reinforcement learning: A systematic study of the DQN target," *arXiv preprint arXiv:1901.07510*, 2019.

[30] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[31] 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, pp. 529–533, 2015.

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

[33] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[34] W. Ahsan, W. Yi, Z. Qin, Y. Liu, and A. Nallanathan, "Resource Allocation in Uplink NOMA-IoT Networks: A Reinforcement-Learning Approach," *IEEE Trans. Wireless Commun.*, 2021.

[35] Y. Shao, A. Rezaee, S. C. Liew, and V. W. Chan, "Significant sampling for shortest path routing: A deep reinforcement learning solution," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2234–2248, 2020.

[36] A. Kiani and N. Ansari, "Edge computing aware NOMA for 5G networks," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1299–1306, 2018.

[37] F. Fang, Z. Ding, W. Liang, and H. Zhang, "Optimal energy efficient power allocation with user fairness for uplink MC-NOMA systems," *IEEE Wireless Commun. Lett.*, pp. 1–1, 2019.

[38] J. Cui, Y. Liu, and A. Nallanathan, "Multi-agent reinforcement learning-based resource allocation for UAV networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 2, pp. 729–743, 2020.

[39] D. C. Nguyen, P. Cheng, M. Ding, D. Lopez-Perez, P. N. Pathirana, J. Li, A. Seneviratne, Y. Li, and H. V. Poor, "Enabling AI in Future Wireless Networks: A Data Life Cycle perspective," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 553–595, Firstquarter 2021.

[40] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *AISTATS*, 2011, pp. 315–323.

[41] F. S. Melo, "Convergence of Q-learning: A simple proof," *Institute Of Systems and Robotics, Tech. Rep*, pp. 1–4, 2001.