Scientific Research Publishing

Journal of Computer and Communications, 2023, 11, 197-207
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

A Collaborative Machine Learning Scheme for Traffic Allocation and Load Balancing for URLLC Service in 5G and Beyond

Andreas G. Papidas[1], George C. Polyzos[2,3]

[1]Mobile Multimedia Laboratory, Department of Informatics, School of Information Sciences and Technology, Athens University of Economics and Business, Athens, Greece
[2]School of Data Science, The Chinese University of Hong Kong, Shenzhen, China
[3]Mobile Multimedia Laboratory, Athens University of Economics and Business, Athens, Greece
Email: papidas@aueb.gr, polyzos@acm.org

How to cite this paper: Papidas, A.G. and Polyzos, G.C. (2023) A Collaborative Machine Learning Scheme for Traffic Allocation and Load Balancing for URLLC Service in 5G and Beyond. *Journal of Computer and Communications*, **11**, 197-207.
https://doi.org/10.4236/jcc.2023.1111011

Received: October 19, 2023
Accepted: November 27, 2023
Published: November 30, 2023

Abstract

Key challenges for 5G and Beyond networks relate with the requirements for exceptionally low latency, high reliability, and extremely high data rates. The Ultra-Reliable Low Latency Communication (URLLC) use case is the trickiest to support and current research is focused on physical or MAC layer solutions, while proposals focused on the network layer using Machine Learning (ML) and Artificial Intelligence (AI) algorithms running on base stations and User Equipment (UE) or Internet of Things (IoT) devices are in early stages. In this paper, we describe the operation rationale of the most recent relevant ML algorithms and techniques, and we propose and validate ML algorithms running on both cells (base stations/gNBs) and UEs or IoT devices to handle URLLC service control. One ML algorithm runs on base stations to evaluate latency demands and offload traffic in case of need, while another lightweight algorithm runs on UEs and IoT devices to rank cells with the best URLLC service in real-time to indicate the best one cell for a UE or IoT device to camp. We show that the interplay of these algorithms leads to good service control and eventually optimal load allocation, under slow load mobility.

Keywords

5G and B5G Networks, Ultra Reliable Low Latency Communications (URLLC), Machine Learning (ML) for 5G, Temporal Difference Methods (TDM), Monte Carlo Methods, Policy Gradient Methods

1. Introduction

The main target of 5G and Beyond (B5G) networks is to extend the capabilities and performance of 4G (LTE) networks based on the three basic 5G use cases, which are: enhanced Mobile Broadband (eMBB), massive Machine Type Communications (mMTC) and Ultra Reliable Low Latency Communications (URLLC) [1] [2]. The most difficult 5G use case to handle is URLLC, which is the key enabler for emerging applications, including tactile Internet and mission-critical applications such as industrial automation, intelligent communications for improved safety and autonomous driving. However, to successfully deploy the URLLC related applications, stringent requirements in terms of latency (1-millisecond) and reliability (99.999% in terms of packet non-drop rate) must be met [1] [2]. It is apparent that satisfying the strict requirements of URLLC is possibly the most challenging issue to tackle.

We believe that an interesting approach towards URLLC service automated handling, is to embed ML algorithms at the next-generation NodeB (gNB) as well as at the UEs and IoT devices based on a collaborative approach among the two, contrary to other approaches that operate centrally without collaboration from the nodes. Our algorithms proposal leads to a Reinforcement Learning (RL) approach that combines Monte Carlo, Policy Gradient and Temporal Difference Methods with the bootstrapping technique leading to a solution able to be trained in real time and continuously interact with the environment, contrary to legacy ML techniques based on static training. Our main targets and contributions in this paper, related to the B5G URLLC use case, are the following: A) We propose a combination of algorithms and techniques for gNB cells and UEs/IoT devices so that URLLC service is ensured in the Radio Access Network (RAN). Two sets of algorithms, running in a distributed manner at the base stations and at the end UEs/IoT devices are proposed, contrary to other recent approaches that focus only on the base station part. B) We propose the usage of a URLLC Overload Flag (OF), which is a very short data packet that includes a red flag, or a green flag header/attribute, used to inform that a cell is URLLC loaded or not. A green flag corresponds to an unloaded cell while a red one to an (over-)loaded one. Conceptually, the URLLC OF embodies the liaison between the algorithm running on the cells and the algorithm running on the UEs/IoT devices, thus our scheme is collaborative. C) We provide simulation results of the system with the developed solution and selected algorithms in a realistic heterogeneous environment including both UEs and IoT devices. As far as related research in URLLC resource allocation and load balancing is concerned the first proposals based on physical and MAC layer proposals evolved with network level approaches such as Mobile Edge Computing (MEC) and Network slicing [1] [2]. The most recent research approaches propose the embedding of ML algorithms such as Supervised Learning (SL), Unsupervised Learning (UL) and mainly the cutting-edge RL and Deep Reinforcement Learning (DRL), able to interact and get trained from the environment in real time [1] [2] [3].

2. Problem Statement and Proposed Solution for Cells and UEs/IoT Devices

If we consider the scenario where URLLC services must be offered in heterogeneous outdoor environments and 5G UEs and IoT devices coexist, extra difficulty is added due to the non-stable outdoor propagation environment and the unpredictable movement of UEs/IoT devices in real time. Our landscape includes a specific outdoor area planned to serve 5G UE and IoT devices with URLLC service and from a RAN aspect the cells covering the specific area might be microcells, femtocells, picocells. We use a realistic scenario and regard one or more UEs or IoT devices that need URLLC service (depicted in **Figure 1**), covered by urban 5G cells. The specific dense urban area was selected since highly loaded and unloaded 5G cells can coexist. More specifically, we consider a cluster of $4 \times 25 = 100$ cells and a UE/IoT device positioned at the center of the area. 25 5G cells with 9 different gNBs exist in each direction (quadrant) so that a 360 degrees coverage area with 100 cells in total is considered.

Our target is to provide the missing URLLC service control, resource allocation and load balancing in this outdoor heterogenous environment already described, thus we propose an ML algorithm running on every cell and a (different, lightweight) ML algorithm running on each UE/IoT device. The combination of the two algorithms and their resulting interaction (cooperation) in addition to the introduced Overload Flag (OF), broadcast by the cells and evaluated by the UEs and IoT devices, which links the two sides, is our proposed approach to URLLC service control. Our scheme can be applied in any quadrant of the scenario depicted in **Figure 1**, thus any cell combination belonging to gNBs of different quadrants is feasible. Consequently, any cells or parameters involved in both algorithms can interact. Finally, a UE/IoT device can be located in either quadrant, but for simplicity, we shall refer to only to the first quadrant (Quadrant 1).

Key ML Algorithms

We provide the key definitions and parameters for the ML algorithms proposed

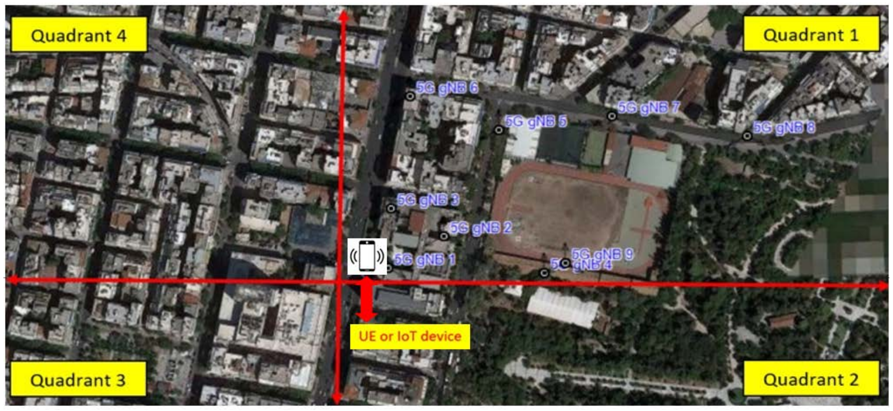


Figure 1. Map of the area divided in four quadrants for the considered scenario including gNBs.

in order to provide a clear understanding of their operation rationale and parameter spaces [3] [4] [5] [6]:

- A Reward (Rt or R) is the response provided to the agent when a specific action is taken. The target is to maximize the sum of rewards in the long-term.
- Action (A) is a choice made, which impacts the future state. E.g., a cell selected for a device to camp.
- Policy (π) is the rationale that the agent follows to decide on the next action according to the current state.
- Q value (Q) or Action Value is an estimation of how good it is to take an action at each state. Trajectory is a collection of rewards and state actions concerning a sequential set of episodes or a single episode.
- Q-Table is a lookup table helping to calculate the best rewards for each action at each state.
- Discount Factor (γ) is a value between zero and one that aims at discounting future rewards compared to current ones, so that the most efficient sum of rewards is identified in a long-term rationale.

Monte Carlo methods operate through repeated random sampling and learn the optimal state values or q-values based on samples collected by the agent that interacts with the environment. Their final target is to collect the most optimum values after the random sampling while the estimate of each state does not depend on the rest, contrary to dynamic programming methods. At the very first step, the agent selects random policies till the end of an episode where a reward from every state is obtained. The value of a state is the expected Return (G) (The sum of rewards that the agent receives from a point in time (t) until an action is completed) following the present policy [3] [4] [5] [6].

Temporal Difference Methods (TDM) combine Monte Carlo methods with dynamic programming methods that enable the segmentation of a single problem into smaller chunks or subproblems. TDM are advantageous compared with other legacy ML techniques since they provide the ability to the agent to learn the optimal values based on experience collected from the environment without receiving an initial model of it, since the agent interacts with the latter in order to the generate a trajectory with the states visited [3] [4] [5] [6]. TDM algorithms can start updating the Q-table immediately after the agent takes the first action, something not possible with Monte Carlo methods, and through this rationale the actions taken at the beginning of the episode start influencing the behavior of the agent immediately and without waiting until the end of the episode. Combined with the bootstrapping technique that is based on random sampling, TDM can result in more effective results. In TDM two different estimates are estimated. The first (the old one prior to each new estimate after each step) and the new one that incorporates real information from the environment [3] [4] [5] [6]. The update process of the Q values can be described by the following equation where the new estimate is α percent of new estimate adding $1 - \alpha$ of the old estimate. As an example, in case α is 30% the old estimate shall represent 70% of the new estimate.

$$Q(S_t, A_t) \leftarrow (1-\alpha) Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) \right] \quad (1)$$

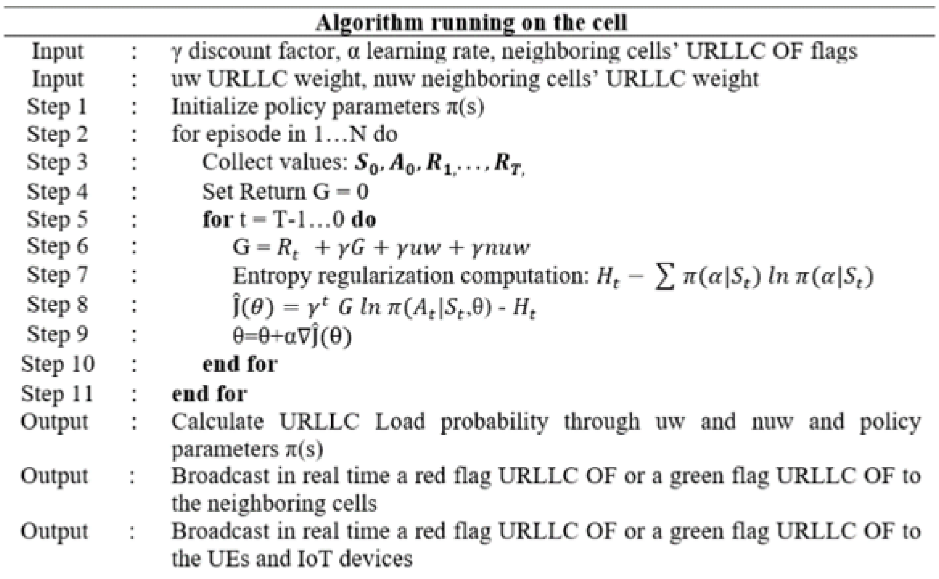
TDM algorithms might face difficulties dealing with complex tasks when the number of states is large such as our scenario. A possible solution to this problem is the combination of TDM with a technique called function approximation leading to a set of methods called Policy Gradient Methods (PGM) that estimate the probability of taking an action in small increments. The basic rationale is that instead of keeping an independent estimate of each value, we use a normalized exponential function (σ) that is modified for each step during the learning process so that we get more accurate results. Each action taken at every step of the algorithm will obtain a probability that can be estimated [3] [4] [5] [6].

Reinforcement Learning (RL) algorithms derive as a combination of Monte Carlo and PGM and have already proved effective for autonomous vehicles, IoT devices and B5G network automation techniques including self-organization [4] [5] [6]. The basic rationale is based on learning through interacting with the neighboring nodes since RL algorithms receive as input a reward function indicating that they operate by providing the most optimum result optimized after algorithm step. The most common examples of are Q-learning and Fuzzy Q-learning that are extensively discussed in the bibliography [3] [4] [5] [6].

3. Algorithms for Cells and UEs/IoT Devices

3.1. Algorithm Running on (5G) gNB Cell

The conceptual target of the gNB cell (agent) algorithm (see **Figure 2**), based on a combination of Monte Carlo and Policy Gradient Methods that leads to a Reinforcement Learning (RL) rationale, is to predict the URLLC load after a few algorithm iterations and inform the neighboring cells and UEs/IoT devices



```
Algorithm running on the cell
Input    : γ discount factor, α learning rate, neighboring cells' URLLC OF flags
Input    : uw URLLC weight, nuw neighboring cells' URLLC weight
Step 1   : Initialize policy parameters π(s)
Step 2   : for episode in 1…N do
Step 3   :     Collect values: $S_0, A_0, R_1, \ldots, R_T$,
Step 4   :     Set Return G = 0
Step 5   :     for t = T-1…0 do
Step 6   :         $G = R_t + \gamma G + \gamma uw + \gamma nuw$
Step 7   :         Entropy regularization computation: $H_t - \sum \pi(\alpha|S_t) \ln \pi(\alpha|S_t)$
Step 8   :         $\hat{J}(\theta) = \gamma^t G \ln \pi(A_t|S_t,\theta) - H_t$
Step 9   :         $\theta = \theta + \alpha \nabla \hat{J}(\theta)$
Step 10  :     end for
Step 11  : end for
Output   : Calculate URLLC Load probability through uw and nuw and policy
           parameters π(s)
Output   : Broadcast in real time a red flag URLLC OF or a green flag URLLC OF to
           the neighboring cells
Output   : Broadcast in real time a red flag URLLC OF or a green flag URLLC OF to
           the UEs and IoT devices
```

Figure 2. Algorithm running on the cell.

about the URLLC load in the cell through the URLLC OF broadcast. Apart from OF we use as input to the algorithm the uw (URLLC weight), which is the ratio of the UEs/IoT device URLLC requests per unit of time (millisecond), and the nuw (neighboring cell URLLC weight), which is the neighboring cells' URLLC load weight. The neighboring cells' URLLC OF flag is collected as well. The collection of (nuw) and OFs is performed all around 360 degrees instead of in a specific direction. uw, nuw and neighboring cells OFs consist of the state (S) of the algorithm Action(a) corresponds to the broadcast of a red OF or a green OF, and Reward(R) corresponds to the fraction of measured green flags/red OF (in a period or iteration).

The output of the algorithm includes: a) The prediction of the URLLC overload probability (uw and nuw) for each cell. b) The Red or Green OF broadcast (not necessarily over the air) in real time to the neighboring cells. If unloaded, it means that it can serve overloaded neighboring cells by offloading them. c) The Red or Green OF broadcast in real time to the UEs and IoT devices. Red means the cell is loaded, avoid it; green means unloaded and can serve UEs/IoT devices.

After a few algorithm iterations the process leads to a heterogenous system consisting of cells and UEs/IoT devices that predict the ideal URLLC serving cell. Moreover, cell balance in a cell's cluster is achieved, URLLC service bottlenecks are avoided and the backhaul latency load is protected. Our proposal is to run the algorithm in a distributed manner on every cell since a possible centralized operation through an orchestrator might be beneficial for larger cell clusters, however extra processing and backhaul latency might be created.

3.2. Algorithm Running on UEs and IoT Devices

The target of the lightweight algorithm (see **Figure 3**) running at each UE/IoT device (agent) based on Temporal Difference Methods (TDM) and the bootstrapping technique, is to rank the cells according to their capability to offer the best latency and URLLC service (in real time) and to determine the best cell for a device to camp, based on the received green vs. red Overload Flag (OF) flags ratio.

We use as input to the algorithm the fraction of measured green to red flags (OF collected from each cell in four directions, φ). Each search/measurement cycle is an episode, or algorithm iteration, and considers 25 cells. These 25 cells can belong to any 90 degrees slice in the coverage area.

Each UE/IoT device camping/attaching to a specific cell comprises an Action (a), the State (s) corresponds to the fraction of measured green flags to red flags for each direction and the Reward (R) equals to the number of green flags collected from the environment in each direction.

The output of the algorithm includes 1) The prediction of Q (s, a) values and the ranking of the surrounding cells according to their Reward (R). 2) The update of the covering cells OF. The algorithm must calculate and rank after a few iterations, what is the reward (R) value of the surrounding cells and which is the ideal to camp. The cells gradually take multiple reward and Q (s, a) values till the

Algorithm running on the UEs and NB-IoT devices

```
Input    :  $\alpha$ learning rate, $\epsilon$ random action probability
Input    :  $\gamma$ discount factor, n bootstrap step, random policy $\pi$ (s)
Input    :  $\varphi$ Fraction of measured green OF flags to the measured red flags collected
            from the environment at each cell direction.
Step 1   :  Initialize $Q(s, \alpha)$ in a random manner
Step 2   :  for episode in 1…N do
Step 3   :      Reset all parameters and check $\mathbf{S_0}$, $\mathbf{A_0}$
Step 4   :      $\mathbf{A_0 \sim \pi (S_0)}$
Step 5   :      While t-n < T do
Step 6   :          If t < T then
Step 7   :              Select action $A_t$ and check $R_{t+1}$, $S_{t+1}$ values and $\varphi$ (green to red OF)
Step 8   :              $A_{t+1} \sim \pi (S_{t+1})$
Step 9   :          end if
Step 10  :          if t ≥ n then
Step 11  :              $G = R_{t-n+1} + \gamma R_{t-n+2} + \cdots + \gamma^{n-1} R_{t+1} + \gamma^n B$
Step 12  :              $Q(S_{t-n}, A_{t-n}) \leftarrow Q(S_{t-n}, A_{t-n}) + a[G - Q(S_{t-n}, A_{t-n})$
Step 13  :          end if
Step 14  :      end while
Step 15  :  end for
Output:  :  Update Q (s, a) values including covering cells URLLC OF
```

Figure 3. Algorithm running on the UEs and IoT devices.

ideal cell to camp with the highest Q (s,a) value is found. After the algorithm ranking process ends, the 5G UE or IoT device can send a registration request which is always accepted from the network side and PDU establishment takes place. The process is similar to the 5G NR Sidelink (SL) introduced in Release 16, where devices communicate directly without packets passing through the core network [7]. The only restriction to this is that a UE/IoT device cannot camp in cells positioned far to avoid adding the propagation delay to the overall latency.

4. Evaluation and Simulation Results

4.1. Algorithm Running on Cells

At each episode, the algorithm can predict URLLC load probability π(s) based on the URLLC load weight (uw) and neighboring load weight (nuw) per millisecond, collected at the end of each episode, in real time. Through this process a table including π(s) values is created at every timestamp. **Figure 4** depicts the algorithms' performance insight based on the returns to episodes as well as the losses to episodes ratio. The stability gained after a specific number of episodes is important since the learning performed at the early stage (beginning of the algorithm) affects the policy during the later episodes, by improving the algorithm decision making process. The fewer the number of episodes/iterations of the algorithm, the better; in our case, 75 episodes can be considered satisfactory.

According to our simulation based on Pytorch, the returns (the sum of rewards that the agent receives from a point in time (t) until an action is completed) improve and become stable after 75 episodes (iterations). Correspondingly, the policy performance follows a similar trajectory, as depicted in the losses graph and the algorithm appears to stabilize after 75 episodes (the loss is calculated through the use of TDM based on previous transitions). Loss consists of the penalty for a bad prediction but there is no perfect algorithm with zero loss. We regard that our algorithm performs well since our loss is low and stable after 75 episodes and our intention is to get a stable graph in less episodes. Same stands for the returns number.

Regarding the OF, **Figure 5** depicts the red and green flags propagation as an output of the algorithm with three UEs (UE6, UE7 and UE8) in the first quadrant of our area of interest and the surrounded gNBs/5G cells. UE6 is within an area with red OF, thus in case URLLC service is needed, it should try to camp on another cell. On the other hand, UE7 and UE8 are within areas where green OF is propagated.

4.2. Algorithm Running on UE and IoT Device

The target of the lightweight algorithm is to rank the cells and determine the best cell for a device to camp. Each search/measurement cycle is an episode or

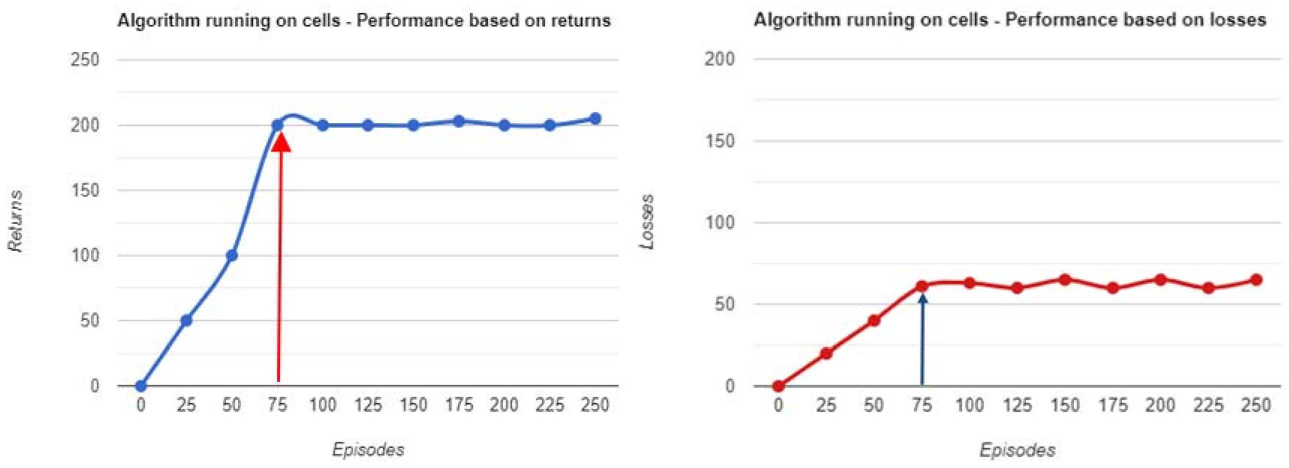


Figure 4. Performance graphs based on returns and losses.

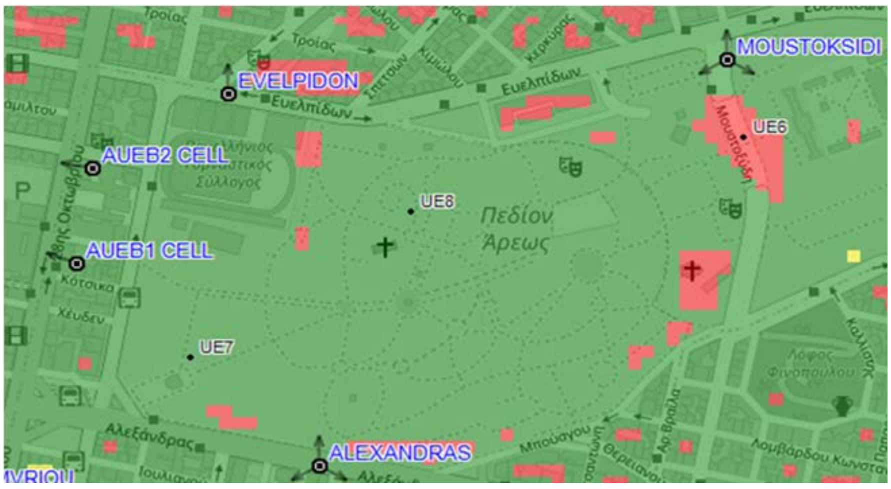


Figure 5. Unloaded and loaded URLLC traffic areas due to OF propagation.

algorithm iteration and considers 25 cells. These 25 cells can belong to any 90 degrees slice in the coverage area as depicted in **Figure 1** that depicts our scenario. The cells gradually take multiple rewards and Q (s, a) values, till the ideal cell to camp with the highest Q (s,a) value is found.

During the first step and the initialization of the algorithm all action values are empty and gradually all the cells get Q (s, a) values and rewards and each cell is ranked by this process as shown in **Figure 6**. The agent picks random actions during some iterations and the higher the Q values the better cell for the device to camp.

At the end of the algorithm, we shall have the following status, which in our case depicts that the target cell to camp is the cell with the four zeros (higher value if compared to the other negative values) as a Q (s, a) value.

Our simulation can be explained by the following table as well (**Table 1**). Each UE/IoT device has a starting cell (cell 1) as depicted in **Table 1**. And the algorithm must calculate and rank after a few iterations, what is the reward (R) value of each cell and which is the ideal to camp. The cells gradually take bad rewards (Red marked cells) and low Q (s, a) values, intermediate rewards (yellow cells) and Q (s, a) values and finally the green marked cell (cell 25) corresponds to the

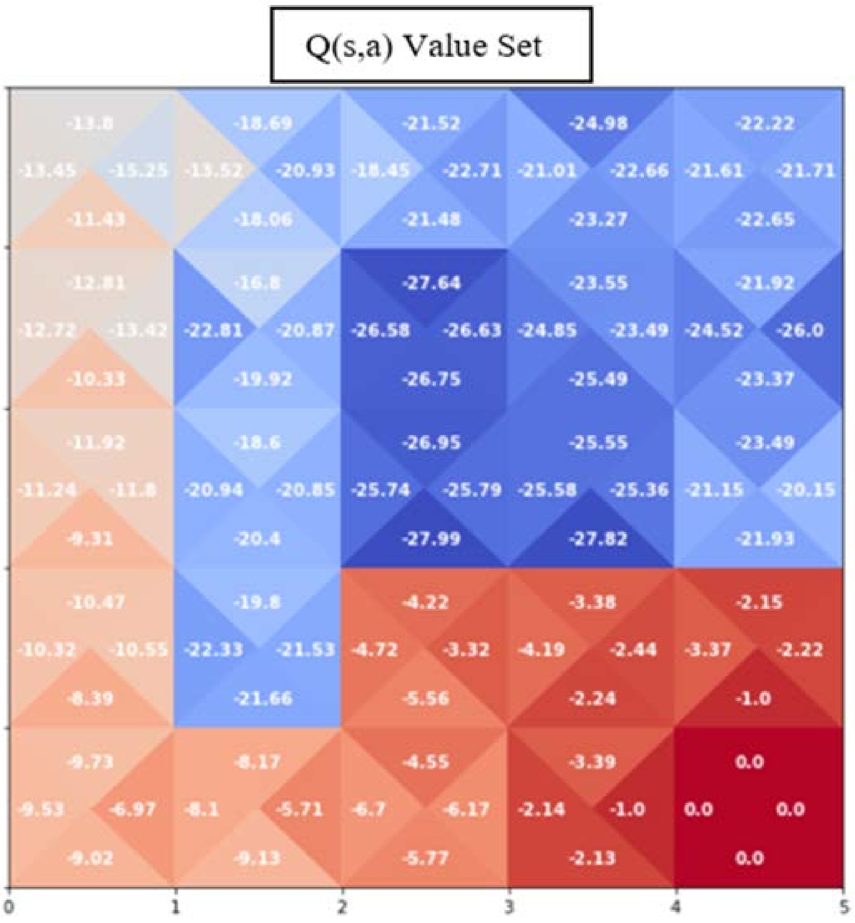


Figure 6. Action values for the scenario.

Table 1. Depicting overloaded cells (red), non-overloaded (yellow), and the target (green, optimal).

cell 1	cell 6	cell 11	cell 16	cell 21
cell 2	cell 7	cell 12	cell 17	cell 22
cell 3	cell 8	cell 13	cell 18	cell 23
cell 4	cell 9	cell 14	cell 19	cell 24
cell 5	cell 10	cell 15	cell 20	cell 25

ideal cell to camp that includes the highest Q (s,a) value.

5. Conclusion and Future Work

In this paper, we describe ML techniques and propose a set of two (different) algorithms to support URLLC service provisioning, for cells/gNBs and UEs or IoT devices. The proposed solution is based on the combination of cutting-edge algorithms and techniques such as Monte Carlo, Policy Gradient Methods, RL, TDM and bootstrapping techniques. In order to provide the cells, the agility to communicate their URLLC load to the devices and neighboring cells, we introduced a short control packet called URLLC OF (Overload Flag) that embodies the liaison between the two sides and their proposed algorithms (for cells and devices). Additionally, we provided simulation results and performance metrics of the selected algorithms and considered a realistic heterogeneous outdoor environment. Possible advancements to our approach include the usage of Deep Reinforcement Learning (DRL) algorithms as an advanced form of RL. The difference between RL and DRL relates to the fact that the former is based on dynamic learning with a trial-and-error method, while the latter is learning from existing knowledge and applying it to a new data set [8] [9]. Directional cell search of URLLC loaded areas through DRL might be a future advancement. Moreover, we believe that our proposed solution can be applied together with other critical network services such as enhanced Mobile Broadband (eMBB) since multiplexing of eMBB/URLLC services is already a realistic scenario [10]. Finally, our proposal can be embedded in Self-Organizing Network (SON) platforms in a form of a SON application assigned to perform resource allocation and load balancing.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Guo, S., Lu, B., Wen, M., Dang, S. and Saeed, N. (2022) Customized 5G and Beyond Private Networks with Integrated URLLC, eMBB, mMTC, and Positioning for Industrial Verticals. *IEEE Communications Standards Magazine*, **6**, 52-57. https://doi.org/10.1109/MCOMSTD.0001.2100041

[2] Sutton, G., Zeng, J., Liu, R., Ni, W., Nguyen, D., Jayawickrama, B., Huang, X.J., Abolhasan, M., Zhang, Z., Dutkiewicz, E. and Lv, T.J. (2019) Enabling Technologies for Ultra-Reliable and Low Latency Communications: From PHY and MAC Layer Perspectives. *IEEE Communications Surveys & Tutorials*. https://doi.org/10.1109/COMST.2019.2897800

[3] Sun, Y., Peng, M., Zhou, Y., Huang, Y. and Mao, S. (2019) Application of Machine Learning in Wireless Networks: Key Techniques and Open Issues. *IEEE Comm. Surveys and Tutorials*, **21**. https://doi.org/10.1109/COMST.2019.2924243

[4] Sutton, R.S. and Barto, A.G. (2018) Reinforcement Learning: An Introduction. The MIT Press Cambridge, Second Edition, Massachusetts London, England.

[5] Sean, M. (2022) Control Systems and Reinforcement Learning. Cambridge University Press, Cambridge. https://doi.org/10.1017/9781009051873

[6] Elsayed, M. and Erol-Kantarci, M. (2019) Reinforcement Learning-Based Joint Power and Resource Allocation for URLLC in 5G. IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA. https://doi.org/10.1109/GLOBECOM38437.2019.9014032

[7] Cintron, F., *et al.* (2021 May) Study of 5G New Radio (NR) Support for Direct Mode Communications. National Institute of Standards and Technology, U.S Department of Commerce. https://doi.org/10.6028/NIST.IR.8372

[8] Brown, B. and Zai, A. (2020) Deep Reinforcement Learning in Action. Manning Shelter Island.

[9] She, C.Y., Sun, C.J., Gu, Z.Y., Li, Y.H., Yang, C.Y., Poor, H.V. and Vucetic, B. (March 2021) A Tutorial on Ultrareliable and Low-Latency Communications in 6G: Integrating Domain Knowledge into Deep Learning. *Proceedings of the IEEE*, **109**, 204-246. https://doi.org/10.1109/JPROC.2021.3053601

[10] João, S., Victor, C., Radoslaw, K., Taufik, A. and Petar, P. (2023) Uplink Multiplexing of eMBB/URLLC Services Assisted by Reconfigurable Intelligent Surfaces.