American Academic Scientific Research Journal for Engineering, Technology, and Sciences

ISSN (Print) 2313-4410, ISSN (Online) 2313-4402

http://asrjetsjournal.org/

A Q-Learning Based Slice Admission Algorithm for Multi-Tier 5G Cellular Wireless Networks

Elizabeth M. Okumu*

Kabarak University, School of Science Engineering and Technology, P. O. Box Private Bag 20157, Nakuru 20100, Kenya

Email: eokumu@kabarak.ac.ke

Abstract

Network slicing enables a 5G infrastructure provider (network infrastructure owner) to create multiple separate virtual networks, each tailored at a specific performance requirement, on a common physical network. In this context, slice admission algorithms are required to process slice requests received by the infrastructure provider. These algorithms are tailored to admit and allocate resources to network slices in a manner that results in the optimization of a given objective. In this paper, a Q-learning slice admission algorithm, which maximizes the infrastructure provider's revenue, is designed. Results show that the designed algorithm learns from its environment, which enables it to acquire knowledge about the multi-tiered cellular network, thus allowing it make optimal slice admission decisions. The results further show that the designed algorithm has superior performance in terms of revenue achieved when compared to algorithms that admit, a) to maximize immediate rewards and b) slices in a random manner.

Keywords: Network slice; 5G; Reinforcement learning; Slice admission; Resource allocation.

1. Introduction

Fifth generation (5G) wireless networks are expected to accommodate a profusion of mobile devices and support a variety of services with unique performance requirements. The one-size-fits-all approach employed by traditional networks, e.g. 4G, is incapable of addressing these varying performance requirements. Software-defined radio (SDN) and network function virtualization (NFV) are enabling technologies for softwarized and virtualized networks [1,2,3]. By leveraging on the aforementioned technologies, 5G networks can address the divergent service requirements using network slicing. With network slicing, the physical network is sliced into multiple logical networks, each customized for a unique service requirement [4].

----

* Corresponding author.

Network resources are dynamically allocated to network slices, depending on their service requirements, enabling the efficient deployment of the various services. Network slicing brings about a division of roles in the 5G network between infrastructure providers, who own and operate the physical network recourses, and slice tenants, who buy slices from the infrastructure providers, and then sell them to the end users. Slice admission algorithms are required for efficient allocation of resources, and to meet predetermined objectives for the infrastructure provider. The objectives can be broadly classified as revenue optimization, QoS control, inter-slice congestion control and slice fairness assurance [5]. The authors in [6] designed a revenue optimizing Q-learning algorithm for elastic and inelastic service types; slice requests that were not serviced were lost. In this paper, reinforcement learning techniques are used to develop an adaptive revenue optimizing slice admission algorithm for a multi-tied 5G wireless network. Network slice requests are placed on queues from where they are processed from. The designed Q-learning algorithm also ensures that the service requirements of the different slices are met.

1.1. Reinforcement Learning

Reinforcement learning (RL) is a machine learning technique that enables an autonomous agent to select optimal actions without prior knowledge of the environment (system) [7]. The agent takes actions, observes the response from the environment and then automatically modifies its approach to achieve the optimal policy. In RL, the environment is modeled as a Markov Decision Process (MDP) [8]. MDPs model decision making in a discrete, stochastic environment under the control of a decision maker or agent. An MDP is defined by

- A finite set of states, S, of the environment.
- A finite set of actions, A
- A transition probability, $P(s, a, s')$, from state s to state s' $(s, s' \in S)$ when action $a \in A$ is executed.
- An immediate reward, r, that is received after action a is performed.

The role of the agent is to learn a control policy π used to select the next action a_t, given the current state s_t, at each discrete time step t; that is $\pi(s_t) = a_t$ For an infinite time horizon MDP, the goal is to determine the optimal policy, π^*, that maximizes the discounted cumulative reward, defined as

$$V^{\pi}(s_t) = \sum_{i=0}^{\infty} \gamma^i r_{t+i}, \forall s \in S$$

$0 \leq \gamma < 1$ is the discount factor. Beginning at the state s_t, the sequence of rewards r_{t+i} are generated by repeatedly selecting actions using the policy π. The optimal policy π^*, that chooses the optimal action for each state, can be realized by learning the optimal value function given by [9]

$$V^*(s) = \max_{a_t}[r(s_t, a_t) + \gamma V^{\pi}(s_{t+1})]$$

In Q-learning, the evaluation function to be learned is $Q(s, a)$, defined as $Q(s, a) = r(s_t, a_t) + \gamma V^{\pi}(s_{t+1})$. Thus, the optimal value function can be expressed as $V^*(s) = \max_{a_t} Q(s, a)$. The Q values are stored in a large table, the Q-table, with a separate entry for each state-action pair. The optimal policy is now determined by finding the optimal values for all state-action pairs. The agent perceives its present state s, executes an action a, receives a reward r and moves to the next state s'. The Q table is then updated in an iterative manner after each action. If $Q_n(s,a)$ denotes the learned value after the nth iteration, then the entries in the Q table are updated using the following rule

$$Q_n(s,a) \leftarrow Q_{n-1}(s,a) + \alpha_{n-1}\left[r_n + \gamma \max_{a'} Q_{n-1}(s',a') - Q_{n-1}(s,a)\right]$$

$\alpha_n \in [0,1]$ is the learning rate which has to satisfy the conditions $\sum_{n=0}^{\infty} \alpha_n = \infty$ and $\sum_{n=0}^{\infty} \alpha^2 < \infty$ [10]. The Q-learning algorithm belongs to the class of temporal difference algorithms, learns by reducing the differences between estimates made by the agent at different time units [9].

2. Q-Learning Slice Admission Algorithm

2.1. System Model

The network model is a multi-tiered 5G network consisting of a single macro-cell overlaid with B small cells. The macro-cell has a capacity of C_{Mac}, while each small cell has a capacity of C_{Sc}. It is assumed that at any given time, a user in the cellular network can only communicate through a single base station, i.e. the macro-cell's or one small cell's. Geographically, it is assumed that the users are uniformly distributed among the small cells; the geographical position of a user is given with respect to the coverage area of the small cell the user is located in. It is assumed that all the small cells have equal coverage area and that the coverage areas do not overlap. In this paper, the focus is on three types of services, ultra-reliable low latency communication (uRLLC), enhanced mobile broadband (eMBB) and internet of things (IoT) services. The IoT devices are assumed to be low data rate sensors. We define an IoT user as a group of d_i IoT devices; each IoT device requires a capacity of C_i/d_i, therefore the IoT user capacity requirement is C_i. The capacity required by uRRLC and eMMB users are C_u and C_e respectively. IoT and eMMB users are serviced through the macro-cell base station while the uRRLC users, due to the latency sensitive nature of uRLLC services, are serviced through the small cell base station whose coverage area they are located in. The network is assumed to be owned and operated by an infrastructure provider who provides network slices to tenants; the tenants in turn provide services to the end users. A network slice is defined as follows;

- Service type $m \in [1,2,3]$; the supported service types IoT, uRRLC and eMMB are represented by $m = 1, 2\ and\ 3$ respectively.
- Size n: this is the number of users the slice can support

2.2. The Q-Learning Algorithm

Markov Decision Process theory (MDP) is used to model the system as a set of S finite states and a set of A finite actions. A state $s \in S$ is defined as $(z_1, \dots, z_m)$ where z_m is an n-tuple $(z_{m1}, z_{m2} \dots\dots z_{mN_m})$ which describes the number slices of different sizes present in the system for service type m. N_m represents the maximum slice size for service type m. The states in the state space S have to satisfy the following constraints

$$C_i \sum_{n=1}^{N_1} nz_{1n} + C_e \sum_{n=1}^{N_3} nz_{3n} \le C_{Mac}$$

$$C_u \sum_{n=1}^{N_2} nz_{2n} \le BC_{Sc}$$

The slice requests submitted to the infrastructure provider are in of the form $\{m, n, t\}$, where t is the number of time units the slice is required for. Incoming slice requests are placed in queues depending on their type and size. A slice request of type m and size n is placed it the queue Q_{mn}. The slice admission algorithm retrieves the slice requests from the queues in a first in first out manner. If resources are available the slice request is processed and resources are allocated to the network slice; the slice request is then removed from the queue. A revenue (reward) of $r = tn\rho_m$, where ρ_m is the revenue (reward) per unit time for service type m, is earned by the infrastructure provider. But if resources are not available, the slice request will not be serviced but will remain in the queue, and no reward will be earned. The possible actions, $a \in A$ are defined as (a_1, a_2, a_3, R), where a_m is an n-tuple $(a_{m1}, a_{m2}, \dots., a_{mN_m})$; the nth element of a_m corresponds to the action of accepting a slice of size n of service type m. Based on the model described, the Q-learning algorithm is described in Table 1.

Table 1: Q-Learning Algorithm

```
For each $(s, a)$ pair, initial the Q table entry $Q(s, a)$ to zero
Observe the current state $s$
for $n = 1$ to $E$
    From state $s$ execute action $a$
    Receive immediate reward $r$
    Observe new state $s'$
    Update the table entry $Q(s, a)$ as follows
        $Q_n(s, a) \leftarrow Q_{n-1}(s, a) + \alpha_{n-1} \left[ r_n + \gamma \max_{a'} Q_{n-1}(s', a') - Q_{n-1}(s, a) \right]$
    Replace $s \leftarrow s'$
End for
```

The learning rate is determined as $\alpha_n = {}^{1}/_{1 + \tau(s,a)}$, where $\tau(s, a)$ is the number of times the action a has been executed from state s [9]. Choosing the action that maximizes $Q(s, a)$, exploits the knowledge of already visited states to maximize the overall revenue. Relying on exploitation will result in an algorithm that utilizes only (s, a) determined in the early stages, while failing to explore other (s, a) that may potentially yield higher rewards. In order to tradeoff between exploration and exploitation, a probabilistic approach is used to select actions. The probability of selecting action $a_i$when in state s is determined as follows [9]

$$P(a_i \backslash s) = \frac{k^{Q(s,a_i)}}{\sum_j k^{Q(s,a_j)}}$$

The value the constant $k > 0$ is varied (from small to large) so as to favor exploration in the early stages of the learning process, which then gradually shifts to exploitation. E is used to indicate the end of an episode.

3. Simulation and Results

In this section, the performance of the Q-Learning algorithm is evaluated via simulations, based on the following network settings and parameters.

- The number of small cells is $B = 3$
- $C_{Mac} = 3C_{Sc}$
- $C_u = C_{Sc}/2$, $C_e = 1.5C_u$ and $C_i = C_u/2$
- $N_1 = C_{Mac}/C_i$, $N_2 = BC_{Sc}/C_u$ and $N_3 = C_{Mac}/C_e$
- Maximum slice size of 2, that is $n \in [1,2]$
- $\rho_2 = 4\rho_1$, $\rho_3 = 3\rho_1$ and $\rho_1 = 5$
- The arrival rates for IoT, eMMB and uRLLC slice requests are 0.2, 0.35 and 0.35 respectively
- Number of queues is 6, each queue is stores a maximum of 100 slice requests

The performance of the Q-Learning slice admission algorithm is compared to 1) a greedy algorithm that admits slices that maximize the immediate reward, 2) an algorithm that admits slices in a random manner. Figure 1 shows the average cumulative revenue gained per slice request.

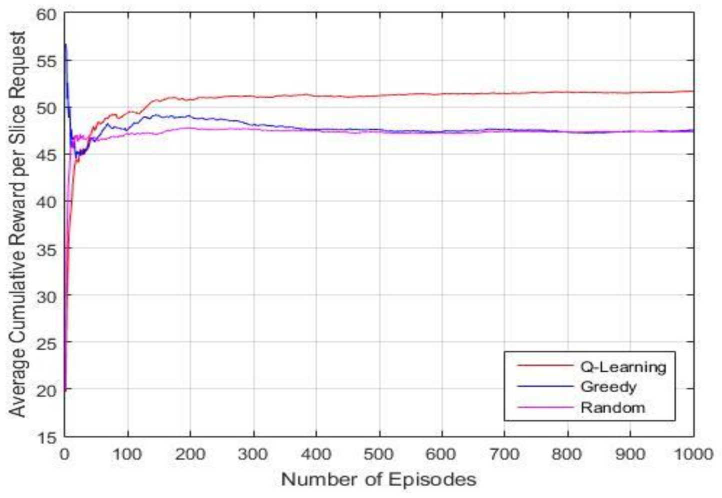
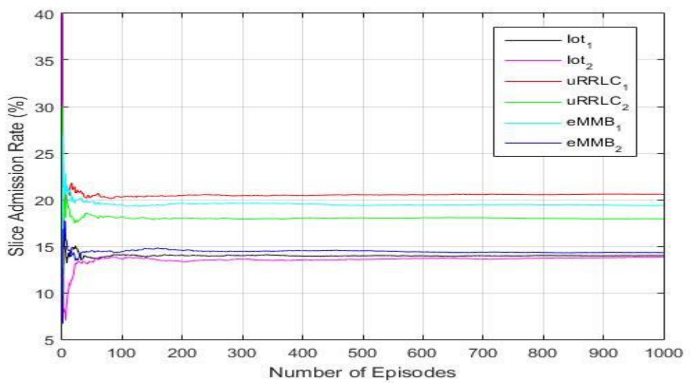
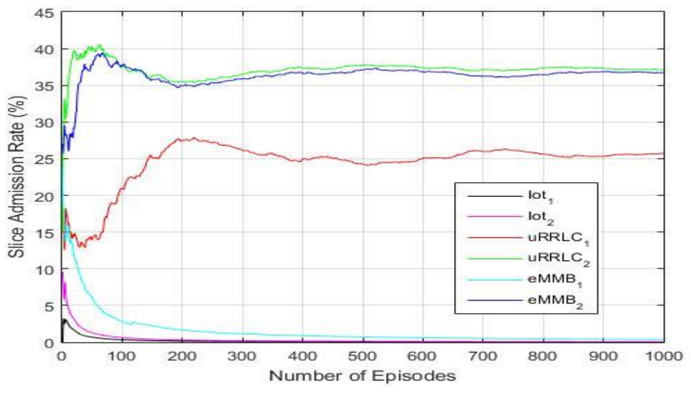


Figure 1: Plot of average cumulative revenue against number of episodes

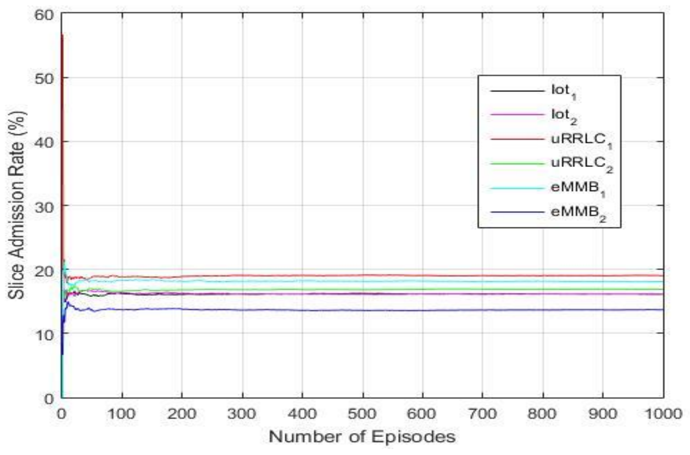
It is observed that the Q-learning algorithm outperforms the greedy and random algorithms for episodes > 50. For episodes $1 - 150$, the Q-learning algorithm has not converged, this results in low average cumulative revenue value. But after convergence, the revenue value stabilizes to a maximum value that is greater than that of the other two algorithms. Figures 2(a), (b) and (c) show the slice admission rates for Q-learning, greedy and random algorithms, respectively.



(a)



(b)



(c)

Figure 2: Average slice admission rate for (a) Q-learning (b) Greedy and (c) Random

The plots for admission rates for, (i) IoT slice requests of size 1 and 2 (IoT_1 and IoT_2), (ii) eMMB slice requests of size 1 and 2 ($eMMB_1$ and $eMMB_2$) and (ii) uRRLC slice requests of size 1 and 2 ($uRLLC_1$ and $uRLLC_2$). From Figure 2(a), it is observed that the Q-learning algorithm, after convergence, maintains the admission rates for all slice request above 14%. Admission rates are dependent on the reward and slice size; a larger reward translates to higher admission rate while a bigger slice size reduces the admission rate. It is more difficult to allocate resources to a slice request with a larger size. It is evident that the Q-learning algorithm learns from the environment, enabling it to prioritize slice requests in a manner that results in maximized cumulative revenue and fair admission rates for the slices. The greedy algorithm, as seen in Figure 2(b), which maximizes immediate rewards, prioritizes $uRLLC_2$, $eMMB_2$ and $uRLLC_1$ at the expense of the other slice requests which have lower immediate rewards. This results in cumulative revenue that is high for initial episodes, and then decreases as the number of episodes increases. The random algorithm, as seen in Figure 2(c), maintains the admission rates between 14% and 19%. Since the algorithm selects slice requests in a random manner, the admission rate for each slice should be approximately 16.6%. But a selected slice will not necessarily be admitted because the resources may not be available, therefore the slice request do not have the same admission rate.

4. Conclusions

In this paper, Q-learning based slice admission algorithm, that maximizes the revenue for a multi-tier 5G cellular wireless network infrastructure was designed and implemented. Results show that the algorithm's capability to learn from the environment (network) allowing it to adaptively allocate resources, this enabled it provide revenue gains when compared to greedy and random slice admission algorithms.

5. Recommendations

The Q-learning algorithm used in this study learns by maintaining and updating Q-values for all state-action pairs. If the wireless network is large in scale, as 5G networks tend to be, then the number of states can become quite large or even unlimited, negatively impacting algorithm efficiency. This issue can be overcome if Deep Q-learning, which combines Q-learning and deep learning techniques, is used.

References

[1]. ONF TR-526, "Applying SDN Architecture to 5G Slicing," Apr. 2016.

[2]. ETSI GS NFV 002, "Network Functions Virtualization (NFV); Architectural Framework," v. 1.1.1, Dec. 2014.

[3]. N. M. M. K. Chowdhury and R. Boutaba, "A Survey of Network Virtualization," Computer Networks, vol. 54, no. 5, Apr. 2010, pp. 862–76

[4]. NGMN Alliance, "Description of Network Slicing Concept," Public Deliverable, 2016

[5]. M. O. Ojijo and O. E. Falowo, "A survey on slice admission control strategies and optimization schemes in 5g network", IEEE Access, vol. 8, pp. 14977-14990, 2020.

[6]. D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, K. Samdanis, and X. Costa-Perez, ''Optimising

5G infrastructure markets: The business ofnetwork slicing,'' inProc. IEEE INFOCOM, Atlanta, GA, USA, May 2017,pp. 1–9.

[7]. R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. MIT press Cambridge, 1998.

[8]. R. Bellman, "A markovian decision process," DTIC, Tech. Rep., 1957.

[9]. T. Mitchell. Machine Learning, McGraw-Hill, 1997.

[10]. E. Even-Dar and Y. Mansour, "Learning rates for Q-learning," Journal of Machine Learning Research, vol. 5, pp. 1–25, Dec. 2003.