Revenue-Model Learning for a Slice Broker in the Presence of Adversaries

Muhidul Islam Khan
Dept. of Electrical Engineering and Computer Science
University of Stavanger
Stavanger, Norway
email: md.m.khan@uis.no

Gianfranco Nencioni
Dept. of Electrical Engineering and Computer Science
University of Stavanger
Stavanger, Norway
email: gianfranco.nencioni@uis.no

***Abstract*—Multi-Access Edge Computing (MEC) and network slicing two of of the key enabling technologies of the Fifth Generation (5G) of cellular network. MEC helps to reduce latency, offload the cloud, and allow context-awareness. Network slicing allows to create heterogeneous services on top of shared infrastructures. Slice brokers are emerging intermediate entities that take the resources from the infrastructure providers and make slices for the tenants. In this scenario, a slice broker needs to manage the resource and create the slices in order to maximize its revenue to cover the cost and increase the profit. In this work, we consider that the demand of the slice tenant is depending on the price of the slices. Therefore, we formulate a slice allocation problem that consider this demand-price dynamic. Moreover, we consider the presence of adversary that want to compromise the decision process. In order to solve the problem, we propose a multi-agent environment, where some agents cooperate to learn the revenue model and maximize the revenue. Finally, we evaluate the effectiveness of the proposed solution by comparing it with reference solutions. The results highlight that a notable increment of the revenue can be obtained by using our solution.**

***Index Terms*—Reinforcement learning, Network slicing, Slice broker, Revenue model**

I. Introduction

The Fifth Generation (5G) of cellular network provides an improved architecture and fills the demand of increased capacity, improved data rate, decreased latency, and better quality of service [1]. Multi-Access Edge Computing (MEC) is a critical enabling technology supporting 5G with this increased demand and low latency. Network Slicing is another enabling technology that allows to create multiple virtual networks with heterogeneous requirements on top of a shared infrastructure and enables industry vertical market players to request and lease resources from Infrastructure Providers (InPs) dynamically according to needs [2]. In this context, a new intermediate entity, called slice broker, can create and manage the network slices. For resource allocation, the slice broker cares about the economic aspects. Economical aspects can include cost, revenue, and profit. The cost is something that an entity needs to pay; in this context, a slice broker buys resources from one or multiple InPs, the money spent by slice broker is the cost. The revenue is what one receives when one sells something; in this context, the slice broker sells network slices to service tenants, the money received by the slice broker is the revenue [3]. The profit, instead, is the net benefit that the broker has received, which is the revenue from the service tenants minus the cost paid to the InPs.

Our work focuses on maximizing the revenue of the slice broker in a context where the demand and price of the slice is dynamic. In order to do this, we consider a multi-agent environment, where some agents cooperate to learn the revenue model and maximize the revenue. Learning the revenue model means learning how the revenue varies with respect to the slice demand and the slice price (which are interdependent). Given this knowledge, the slice price can be set in order to maximize the revenue of the slice broker. Moreover, in the addressed problem we include security aspects by considering the presence of adversaries that can compromise the multi-agent environment. The adversaries can inject non-cooperative agents or noise in order to mislead the slice broker about the actual revenue.

In the recent years, several works have addressed the problem of slice allocation in 5G-MEC systems with economical targets. In [4], the authors propose a framework for network slicing in MEC systems, including slice request admission, and investigate the operator's profit escalation problem while considering traffic variations. Their system model is mainly composed of longer-term profit and short-term profit. They jointly optimize slice request admission in the long-term and resource allocation in the short-term to maximize the operator's average profit. They apply the Lyapunov optimization technique to solve the problem. However, their system is not adaptive, and security issues are not considered. In [5], the authors address the problem of allocating network slices. Their target is to minimize the total cost of the slice broker to acquire the resources from the InPs. They propose a heuristic solution to the problem, evaluate the proposed heuristic's behavior in various scenarios, and compare it with a benchmark solution. However, their proposed method is not adaptive, and there is no consideration of dynamic price-demand, and also there is no consideration of security perspectives for resource allocation. In [6], the authors propose an adaptive approach for allocating resources in a distributed fashion. Their proposed method dynamically distributes the network resources among the active information flows according to the characteristic of the slices to which they belong. However, their work did not focus on brokers' profit, and also, they did

not have any consideration of injecting adversaries/noises into the environment. In [7], the authors develop an optimization framework for network slice dimensioning, in which the Slice Customer's Problem (SCP) maximizes the Slice Consumer's (SC) profit, and the Slice Provider's Problem (SPP) maximizes net social welfare (resource efficiency). However, they did not specifically focus on any revenue model for Slice Provider/Slice Broker, and the system is also not adaptive. They also did not consider any security issues. None of the above works consider any revenue model learning for the slice brokers, dynamic behavior of price-demand, and also do not consider any adversaries in the environment. In [8], the authors propose an auction-based mechanism for optimal and revenue-based allocation of radio resources in a slice-based 5G network. However, their proposed method is not adaptive, does not consider the cooperation among multiple agents, and does not try to learn the revenue model considering the adversaries in the system.

Another important aspect in network slicing is the security [9]. In the recent years, a few works, have focused on slice allocation by considering security aspects. In [10], the authors focus on secure slicing for resource allocation under massive network traffic. The authors propose traffic-aware scheduling for secure slicing and resource allocation in 5G networks. In their approach, user devices are authenticated using a password-based key derivation function. Secure network slicing and resource allocation are implemented using deep Reinforcement Learning (RL) models. They apply RL for resource allocation and to predict the Distributed Denial-of-Service (DDoS) attackers. However, their focus was not on learning the revenue model with adversaries.

In summary, the contributions of our paper are the following:

- We address a new problem of network slice allocation with dynamic slice demand (dependent on the price) and in the presence of adversaries in order to maximize the revenue of the slice broker by dynamically setting the slice price.
- We propose a solution based on cooperative RL with a consensus mechanism to detect the adversaries. In [11], the authors propose a trust-based consensus in RL systems in a multi-agent-based system. However, they consider binary consensus, which is poor in convergence. Their definition of trust is also very generic, not specific application-based.
- We compare our solution with reference solutions that are non-cooperative, or that ignore the injection of adversaries and noise.

The paper is structured as follows. Section II describes the problem of maximizing the revenue in the presence of adversaries. Section III introduces the proposed method to solve the presented problem. Section IV presents the results of the comparison of the proposed method with reference methods. Finally, Section V concludes the paper.

II. Problem Description

In our problem, we consider a 5G-MEC system that includes the following business entities: InPs, a slice broker, and a slice tenant.

We assume that each InP has one MEC Host (MEH), but the problem can be generalized to have multiple MEHs belonging to the same InP.

A MEH is a computing platform, which has computational resources (i.e., processing power in vCPU). The slice broker buys the computational resources from the InPs and it resells the computational resources to multiple slice tenants. In this work, for sake of simplicity we assume only one tenant but the problem and the related solution can be generalized to include multiple tenants.

We also assume that the slice broker has already bought an amount of computational resource from each InP. Therefore, in each MEH the broker has one *chunk* of computational resources. The set of chunks is denoted as $\mathcal{M}$. For each chunk $m \in \mathcal{M}$, the amount of bought computational resources is denoted as μ_m. The tenant dynamically requests to the broker an amount of computational resources that we call as *slice demand* and denote as d^t, where t identifies the *time interval*.

At each time interval t, the broker decides the amount of computational resources from each chunk to allocate to the tenant in order to fulfill the slice demand. The portion of chunk allocated to the tenant is denoted as *subchunk*. The amount of computational resources of the subchunk from the chunk $m \in \mathcal{M}$ at the time interval t is denoted as δ_m^t. The broker is also deciding the *subchunk price* (in €/vCPU), which is denoted as c_m^t.

In our system, we consider that the broker has one software agent for each chunk $m \in \mathcal{M}$ helping to decide the amount of computational resources, δ_m^t, and price, c_m^t, for the related subchunk at each time interval t. Figure 1 represents the investigated 5G-MEC system. The adversary and noise injections will be explained in the second part of this section.

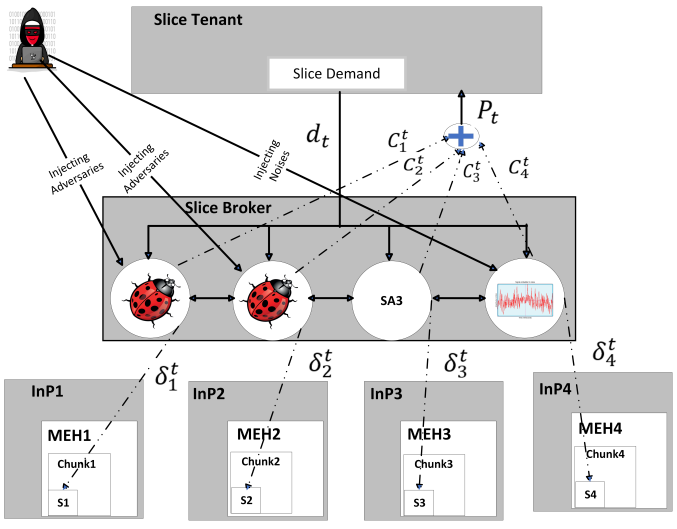

Fig. 1. 5G-MEC system under investigation

We use a practical demand model to get the slice demand of the slice tenant, d^t. In the real world, a demand changes over time and depends not only on the current price but can also be impacted by the magnitude of recent price changes. A price decrease can create a temporary demand increment, wherever a price increase can cause a fall in demand. The impact of price changes can also be asymmetric, so price increases have a much bigger or smaller impact than decreases. In our case, at every time interval, the slice demand depends on the weighted average price, which is computed from the prices of each subchunk as follows.

$$p^t = \frac{1}{d^t} \sum_{m \in \mathcal{M}} \delta_m^t \cdot c_m^t \tag{1}$$

Based on the price-demand function in [12], we compute the slice demand for next time interval $t+1$ as follows.

$$d^{t+1} = d^0 - k \cdot p^t - a \cdot s((p^t - p^{t-1})^+) + b \cdot s((p^t - p^{t-1})^-), \tag{2}$$

where

$$(p^t - p^{t-1})^+ = \begin{cases} p^t - p^{t-1}, & \text{if } p^t > p^{t-1} \\ 0, & \text{otherwise} \end{cases},$$

$$(p^t - p^{t-1})^- = \begin{cases} p^t - p^{t-1}, & \text{if } p^t < p^{t-1} \\ 0, & \text{otherwise} \end{cases},$$

and where p^t is the price for the current time interval t and p^{t-1} is the price for the previous time interval. The first two terms of Eq. (2) correspond to a linear demand model with intercepting d^0 and slope k. The second two terms model the response to a price change between two intervals. Coefficients a and b define the sensitivity to positive and negative price changes, respectively, and s is a shock function that can be used to specify a non-linear dependency between the price change and demand. We assume $s(\cdot) = \sqrt{\cdot}$.

The problem of maximizing the revenue of the slice brokers to serve the slice demand at the time interval t can be formulated by basing on Bertnard model [13]:

$$P : \max \Phi^t = \max \sum_{m \in \mathcal{M}} c_m^t \cdot \delta_m^t, \tag{3}$$

subject to

$$\begin{aligned} &C1 : \delta_m^t \leq \eta_m \qquad \forall m \in \mathcal{M}, \\ &C2 : \sum_{m \in \mathcal{M}} \delta_m^t \leq d^t, \end{aligned} \tag{4}$$

where Φ^t is the defined revenue function and the objective function of the problem, $C1$ is the constraint that limits the size of the subchunk to the size of the related chunk, and $C2$ is the constraint that limits the cumulative size of all subchunks to the slice demand.

A. Environment

Our proposed method is based on RL. In RL, the agents learn over time intervals by performing a particular action, and it shifts from one state to another. After performing an action, the agents receive a reward for the performed action. The environment of the RL determines the states, actions and the reward function [14]. In our case, the states are the slice demand at time t, d^t, the size of the subchunks at time interval $t-1$, δ_m^{t-1} and all the previous prices of the subchunk m, c_m^τ $\forall \tau \in [1, t-1]$. The actions are the size and the price of the subchunks m at t, δ_m^t and c_m^t, respectively. The reward function denotes the revenue for the allocation of the subchunk m at the time interval t.

$$r_m^t = c_m^t \cdot \delta_m^t \tag{5}$$

B. Presence of Adversarial Agents

According to the European Network and Information Security Agency (ENISA) 5G threat landscape [15], one of the potential threats related to 5G MEC is the compromised supply chain (i.e., vendor and service providers). Since the tampering of network product (creating adversaries and added noises) can result in service unavailability, information destruction or misinformation generation. For example, in Figure 1, the attacker compromises the software agents SA1 and SA2 for making them selfish or uncooperative. In this case, SA1 and SA2 will try to maximize their own reward without cooperation, eventually causing less revenue for the broker.

Let $\mathcal{M}^+$ and $\mathcal{M}^-$ denote the set of cooperative agents and adversaries, respectively, where $\mathcal{M} = \mathcal{M}^+ \cup \mathcal{M}^-$. Since we have assumed that there is one agent per chunk (and subchunk), we use $\mathcal{M}$ to indicate the set of agents or the set of chunks, interchangeably.

The objective of agents $m \in \mathcal{M}^+$ is to maximize a team-average objective function given as follows.

$$\max_{c_m^t, \delta_m^t} J^+ = \max_{c_m^t, \delta_m^t} \mathbb{E}\left[\sum_{\tau=0}^{\infty} \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} \gamma_m \cdot r_m^\tau \right] \tag{6}$$

where τ is the time under investigation, γ_m is a discounted factor, has a value between 0 and 1, and indicates how much the RL agents cares about rewards in the distant future with respect to those in the immediate future.

The cooperative agents are unaware of the presence of an adversarial agent that seeks to maximize a different objective function. We define the objective function for $m \in \mathcal{M}^-$ as follows.

$$\max_{c_m^t, \delta_m^t} J^- = \max_{c_m^t, \delta_m^t} \mathbb{E}\left[\sum_{\tau=0}^{\infty} \gamma_m \cdot r_m^\tau \right] \tag{7}$$

It is important to note that the adversarial agent can compromise the rewards $r_m^\tau, m \in \mathcal{M}^-$, to incentivize it's malicious behavior [16]. Furthermore, once the agents establish communication, the adversary can spread false information about the performance of the entire network embedded in

the compromised rewards r_m^τ. This may eventually lead to incentivizing bad behavior in the cooperative agents.

C. Presence of Noisy Agents

An attacker may try injecting noise into any component, e.g., software agent, to compromise the system. For example, in Figure 1, the attacker compromises the software agent SA4. Noise may hamper the cooperation among agents in a way that the broker may have an idea that it has enough resources to allocate or sometimes get the knowledge of not allocating the resources, which is harmful to the revenue-making of a broker. We consider few agents with noise. Here, we consider Additive White Gaussian Noise (AWGN). The noise can be normally distributed as follows:

$$N_m \sim \mathcal{N}\left(0, \left(\frac{A_m}{3}\right)^2\right) \tag{8}$$

where A_m is the magnitude of the noise.

III. Proposed Method

We propose a resource allocation method based on a generic Deep Q Network (DQN) algorithm [17] to learn the revenue model. We use the original DQN as it is simple considering other variants of the learning mechanism. We could not apply classical RL, e.g., Q-learning, State-Action-Reward-State-Action (SARSA) learning, as our states are continuous. The learning mechanism like Actor-Critic learning is not applied for its complexities.

RL considers the setup where an agent interacts with the environment in discrete time intervals to learn a reward-maximizing behavior policy. At each time interval t, with a given state s, the agent takes an action a according to its policy $\pi(s) \rightarrow a$ and receives the reward r moving to the next state s'. We define our environment considering the RL terms as follows.

The goal of the algorithm is to learn an action policy π that maximizes the total discounted cumulative reward/return earned during the episode of T time intervals:

$$R = \sum_{t=0}^{T} \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} \gamma_m \cdot r_m^t \tag{9}$$

Such a policy can be defined if we know a function that estimates the expected return based on the current state and next action, under the assumption that all subsequent actions will also be taken according to the policy:

$$Q^\pi(s,a) = \mathbb{E}_{s,a}[R] \tag{10}$$

Assuming that this function (known as the Q-function) is known, the policy can be straightforwardly defined as follows to maximize the return:

$$\pi(s) = \arg\max_a Q(s,a) \tag{11}$$

We can combine the above definitions based on the Bellman equation as follows:

Algorithm 1 Proposed Method based on Deep Q Learning

Parameters and Initialization:
ϕ - Parameters of the policy network Q_ϕ
ϕ_{targ} - Parameters of the target network $Q_{\phi_{targ}}$
α - Learning rate
B - Batch size
T_u - Period of target updates
Initialization: $\phi_{targ} = \phi$
for t = 1 to T **do**
  Choose the action based on $Q_\phi(s_t, a_t)$
  Execute the action and save transition (s_t, a_t, r_t, s_t') in the buffer
  Calculate the Q-value
  Consensus Step:
  Send the Q-value to the neighbors
  Update the Q-value of agent m as follows considering the neighbor's impact:
  $Q_m(t+1) = Q_m(t) + \Gamma \sum_{j \in \mathcal{M}_m^G} (Q_j(t) - Q_m(t))$
  End of Consensus Step
  Calculate target Q-values for each sample in the batch:
  $y_m = r_m + \gamma_m \cdot \max_{a'} Q_{\phi_{targ}}(s', a')$
  where $Q(s,a) = 0$ for last states of the episodes (initial condition)
  Calculate the loss:
  $L(\phi) = \frac{1}{\mathcal{M}} \cdot \sum_m (y_m - Q_\phi(s_m, a_m))^2$
  Update the network's parameters:
  $\phi = \phi - \alpha \cdot \gamma_m \cdot L(\phi)$
  if t mod $T_u = 0$ **then**
    Update the target network:
    $\phi_{targ} \leftarrow \phi$
  end if
end for

$$Q^\pi(s,a) = r + \gamma \max_{a'} Q(s', a') \tag{12}$$

where s' and a' are the next state and the action taken in that state, respectively. If we estimate the Q-function using approximator, then the quality of the approximation can be measured using the difference as follows:

$$L(\phi) = \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} (y_m - Q_\phi(s_m, a_m))^2 \tag{13}$$

This value is called the temporal difference error, which is the loss function.

Algorithm 1 shows the procedure step by step about how the system works and training has been done.

Consensus Procedure:
An agent i communicates the Q-value to all its neighbors $j \in \mathcal{M}_m^G$. $\mathcal{M}_m^G$ denotes the set of neighbors of agent m. All agents update their Q-values through a linear combination of their own values and the information of neighbors received in the previous step. The procedure can be written as:

TABLE I
SIMULATION PARAMETERS AND THEIR VALUES.

Parameter	*Symbol*	*Value*
Available chunks	$\|\mathcal{M}\|$	4
Number of agents	$\|\mathcal{M}\|$	4
Magnitude of the noise	A_m	5
Learning rate	α	0.5
Number of Episodes	T	5000
Batch size	B	512
Period of target updates	T_u	20
Discount factor	γ_m	0.5
Degree of a node	d_{max}	3
Size of the chunks	η_m	5000 vCPU
Neighboring factor	Γ	0.3
Intercept	d_0	5000 vCPU
Slope	k	20 vCPU/€
Response coefficient for price increase	a	300 vCPU/$€^{1/2}$
Response coefficient for price decrease	b	100 vCPU/$€^{1/2}$

$$Q_m(t+1) = Q_m(t) + \Gamma \sum_{j \in \mathcal{M}_m^G} (Q_j(t) - Q_m(t)) \tag{14}$$

So, in other words, each agent update its state by using the disagreement of states with all its neighbors, scaled by a factor of Γ. Thus the convergence rate of this algorithm depends on the scaling factor used. Convergence is guaranteed as long as the following constraint is met [18]:

$$0 < \Gamma < \frac{1}{d_{max}} \tag{15}$$

where d_{max} denotes the maximum degree among all nodes in the network. The constraint can be fulfilled by realizing an upper bound on the maximum possible neighbors for any node.

IV. Results and Discussions

We consider a network of four software agents which are connected with each other as a mesh. First, we consider the four cooperative agents for learning the revenue model. After that, we consider that two agents out of four are adversary nodes (caused by the attacker), which try to maximize the revenue with own objective function. We also have consensus steps in our proposed work, which identifies the effect of adversary and tried to normalize the outcome.

Table I shows the considered parameters for the simulation and their values. The values are set by empirical studies.

As we are applying a method based on deep Q learning, we consider a total of 150 hidden nodes with three layers (50 nodes per layrer). We exploit the Adaptive Moment Estimation (Adam) algorithm to optimize the Neural Network (NN) weights. Rectified Linear Unit (ReLU) acts as activation function to activate a particular input.

Figure 2 refers to the revenue (€) over the time intervals. We can observe that the agent tries to learn the revenue model over the time intervals. There are some increments and decrements over time. We can observe that after an initial rise for revenue and then saturation over time interval. We

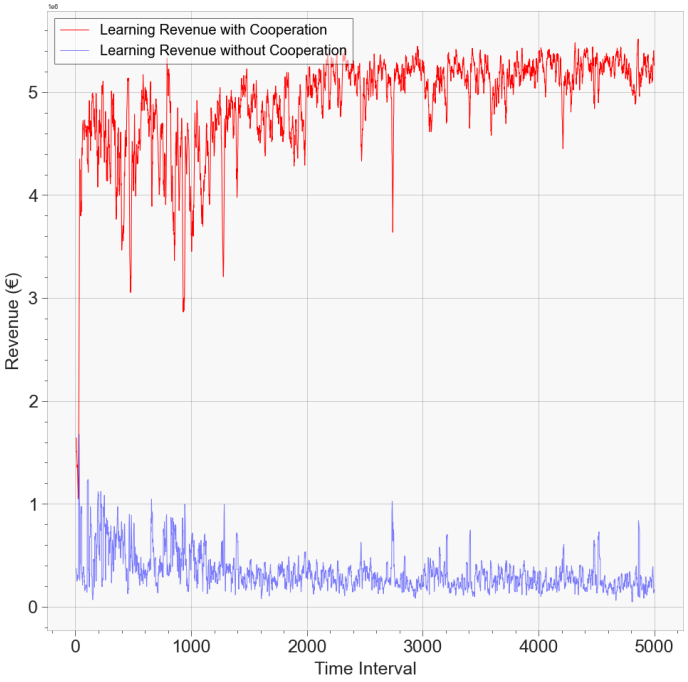

Fig. 2. Revenue over Time Intervals with and without Cooperative Learning

can then observe a significant decrease close to the 1000 time intervals. Then there is a saturation/convergence for the revenue after the 3000 time intervals. These changes are due to the exploration/exploitation and training process. After learning, the cooperative agents are converged for learning the revenue model. The agents explore actions during the learning process, and the revenue may vary over time. On the other hand, for independent learning/without cooperation, the revenue varies significantly over time. Here independent means that there is no cooperation/exchange of values, every agent tries to maximize the revenue independently, which varies over time. There is convergence over the time interval for independent learning, but the revenue is almost on average 48% less compared to the cooperative one.

Figure 3 shows the average revenue over time intervals with the presence of adversary agents, which try to maximize their own revenue without cooperation based on Equation 7 and become selfish. With the adversaries, we can observe that only for a few time intervals the revenue rises very rapidly as adversary agents try to maximize its own revenue and provide an impact, then when the consensus is achieved for the revenue, which becomes saturated and converges after 3000 time intervals. We also observe the revenue model without learning about adversaries/without consensus mechanism in the system, where we can see the revenue is very low. Our proposed method outperforms the revenue model learning by 38% compared with without learning about the adversaries.

Figure 4 shows the average revenue learning over time intervals considering noises in the environment applying our proposed method. The Additive White Noise (AWN) is added for two agents. Here, we can see that at first, there is an

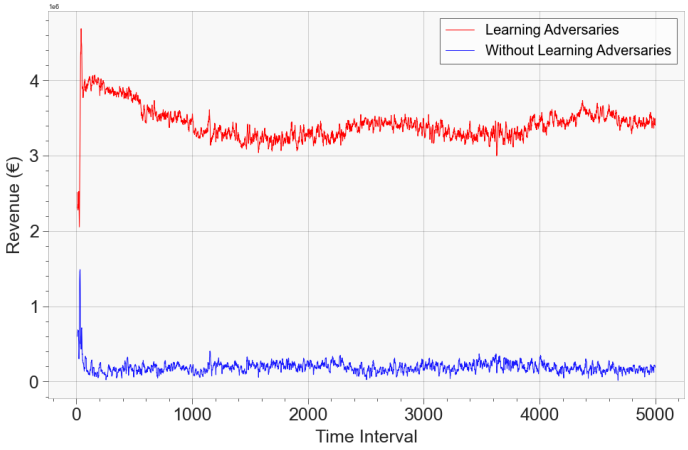

Fig. 3. Revenue over Time Intervals with and without Learning Adversaries

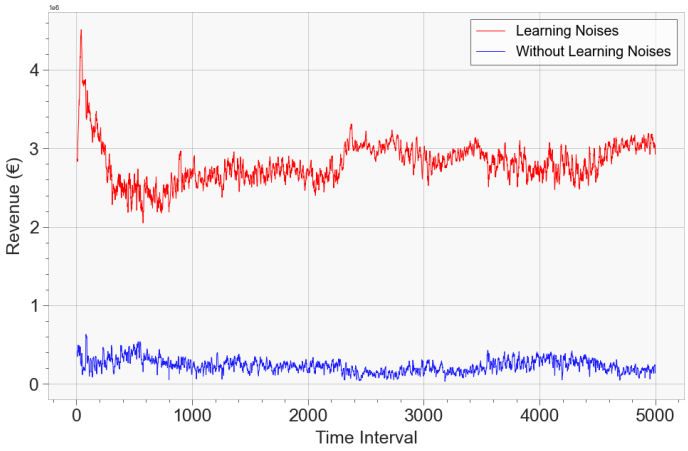

Fig. 4. Revenue over Time Intervals with and without Learning Noises

increment in the overall revenue, and then we can see the downfall, which is for added noises by the agents in the environment. Finally, we can see that the results are converged. But here, we can see that the revenue goes low for added noises compared with the adversary. The noises have been identified for the consensus part of our proposed method, and then the system converged with the revenue level. We can observe that in our proposed method by learning noises, the average revenue outperforms on average 32% compared with the one without learning the noises.

So, in both cases (added adversary and added noises), the agents behave with less revenue, whereas, without having any adversaries, the agents have achieved more robust revenue over time. We can also observe that the cooperation among software agents also has a cooperative impact on the system, which makes the system more effective in learning the revenue model for the brokers.

V. Conclusions

In a scenario where there are dynamic demands from the slice tenants, the slice broker needs to learn the revenue model to decide the slice price that maximize its revenue. Moreover, the slice broker needs to take into account the presence of adversaries that aim to compromise the decision process. In this context, our multi-agent environment implements cooperative RL with a consensus mechanism in order to maximize the revenue while considering the presence of selfish agents and noise. Our evaluation highlights the economical benefit of using our solution.

Acknowledgment

This work was funded by Norwegian Research Council through the 5G-MODaNeI project (no. 308909).

References

[1] K. Samdanis, X. Costa-Perez, and V. Sciancalepore, "From network sharing to multi-tenancy: The 5g network slice broker," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 32–39, 2016.

[2] M. Jiang, M. Condoluci, T. Mahmoodi, and L. Guijarro, "Economics of 5g network slicing: optimal and revenue-based allocation of radio and core resources in 5g," *Kings collections London*, 2016.

[3] A. Gupta and R. K. Jha, "A survey of 5g network: Architecture and emerging technologies," *IEEE access*, vol. 3, pp. 1206–1232, 2015.

[4] J. Feng, Q. Pei, F. R. Yu, X. Chu, J. Du, and L. Zhu, "Dynamic network slicing and resource allocation in mobile edge computing systems," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7863–7878, 2020.

[5] A. Gohar and G. Nencioni, "Minimizing the cost of 5g network slice broker," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2021, pp. 1–6.

[6] F. Mason, G. Nencioni, and A. Zanella, "Using distributed reinforcement learning for resource orchestration in a network slicing scenario," *IEEE/ACM Transactions on Networking*, 2022.

[7] G. Wang, G. Feng, W. Tan, S. Qin, R. Wen, and S. Sun, "Resource allocation for network slices in 5g with network resource pricing," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–6.

[8] N. Tadayon and S. Aissa, "Radio resource allocation and pricing: Auction-based design and applications," *IEEE Transactions on Signal Processing*, vol. 66, no. 20, pp. 5240–5254, 2018.

[9] R. F. Olimid and G. Nencioni, "5g network slicing: A security overview," *IEEE Access*, vol. 8, pp. 99 999–100 009, 2020.

[10] A. J. Ramadhan, "T-s3ra: traffic-aware scheduling for secure slicing and resource allocation in sdn/nfv enabled 5g networks," *arXiv preprint arXiv:2107.05056*, 2021.

[11] H. L. Fung, V.-A. Darvariu, S. Hailes, and M. Musolesi, "Trust-based consensus in multi-agent reinforcement learning systems," *arXiv preprint arXiv:2205.12880*, 2022.

[12] S. K. Vishnoi, T. Bagga, A. Sharma, and S. N. Wani, "Artificial intelligence enabled marketing solutions: A review," *Indian Journal of Economics & Business*, vol. 17, no. 4, pp. 167–177, 2018.

[13] W. W. Sharkey and D. S. Sibley, "A bertrand model of pricing and entry," *Economics Letters*, vol. 41, no. 2, pp. 199–206, 1993.

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

[15] D. Sabella, A. Reznik, K. R. Nayak, D. Lopez, F. Li, U. Kleber, A. Leadbeater, K. Maloor, S. B. M. Baskaran, L. Cominardi *et al.*, "Mec security: Status of standards support and future evolutions," *ETSI White Pap.*, vol. 46, no. 46, pp. 1–26, 2021.

[16] M. Figura, K. C. Kosaraju, and V. Gupta, "Adversarial attacks in consensus-based multi-agent reinforcement learning," in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 3050–3055.

[17] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A theoretical analysis of deep q-learning," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 486–489.

[18] M. Siami, S. Bolouki, B. Bamieh, and N. Motee, "Centrality measures in linear consensus networks with structured network uncertainties," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 3, pp. 924–934, 2017.