*Article*

# Secure Mobile Edge Server Placement Using Multi-Agent Reinforcement Learning

**Mumraiz Khan Kasi [1,*], Sarah Abu Ghazalah [2], Raja Naeem Akram [3] and Damien Sauveron [4]**

[1] Department of Computer Science, FICT, BUITEMS, Quetta 87300, Pakistan
[2] Information Security and Cyber Security Unit, King Khaled University, Abha 61421, Saudi Arabia; sabugazalah@kku.edu.sa
[3] Department of Computer Science, University of Aberdeen, Aberdeen AB24 3FX, UK; raja.akram@abdn.ac.uk
[4] Department of Computer Science, University of Limoges, 23204 Limoges, France; damien.sauveron@unilim.fr
* Correspondence: mumraiz.kasi@buitms.edu.pk

**Abstract:** Mobile edge computing is capable of providing high data processing capabilities while ensuring low latency constraints of low power wireless networks, such as the industrial internet of things. However, optimally placing edge servers (providing storage and computation services to user equipment) is still a challenge. To optimally place mobile edge servers in a wireless network, such that network latency is minimized and load balancing is performed on edge servers, we propose a multi-agent reinforcement learning (RL) solution to solve a formulated mobile edge server placement problem. The RL agents are designed to learn the dynamics of the environment and adapt a joint action policy resulting in the minimization of network latency and balancing the load on edge servers. To ensure that the action policy adapted by RL agents maximized the overall network performance indicators, we propose the sharing of information, such as the latency experienced from each server and the load of each server to other RL agents in the network. Experiment results are obtained to analyze the effectiveness of the proposed solution. Although the sharing of information makes the proposed solution obtain a network-wide maximation of overall network performance at the same time it makes it susceptible to different kinds of security attacks. To further investigate the security issues arising from the proposed solution, we provide a detailed analysis of the types of security attacks possible and their countermeasures.

**Keywords:** mobile edge computing; mobile edge server placement; multiagent RL; edge security

check for updates

**Citation:** Kasi, M.K.; Abu Ghazalah, S.; Akram, R.N.; Sauveron, D. Secure Mobile Edge Server Placement Using Multi-Agent Reinforcement Learning. *Electronics* **2021**, *10*, 2098. https://doi.org/10.3390/electronics10172098

Academic Editor: Kevin Lee

Received: 19 June 2021
Accepted: 24 August 2021
Published: 30 August 2021

## 1. Introduction

Widespread deployments of robotics, assembly and production, automation, machine intelligence, and virtual reality applications requires high performance computing resources available close to the point-of-service [1]. Integration of smart services, such as predictive analysis, and delay-intolerant applications, such as healthcare applications, in current cellular architecture with limited battery lifetimes and processing power of edge (mobile and IoT) devices have called for the re-imagination of cloud computing architecture.

The traditional cloud-centric architecture provides flexibility and significant computation power. However, the communication and delay sensitive requirements of the IoT environments place constraints on the centralised cloud—making them less preferable for a robust service platform. To circumvent delay in the traditional cloud-centric architecture, several network architectures have been proposed with the idea of bringing the cloud nearer to user devices [2]. One such architecture is edge computing that provides a virtualized application layer between edge devices and cloud engine in an existing network infrastructure. Edge computing introduces distributed control systems replacing the single remote centralized control-centre or cloud allowing the processing of data near the edge of the network with enhanced scalability.

Cloudlets, multi-access or mobile edge computing (MEC), and fog computing are some of the well-known edge computing architectures. In this work, we utilize the MEC architecture that uses the existing network architecture such as cellular base station or Wi-Fi access point to provide computational resources and data storage at the edge network [3]. In MEC, the edge network serves as mid-tier between edge (mobile and IoT) devices and cloud increasing the network's capability to provide high throughput and offer low latency to edge devices. However, the costly hardware components and limited budget of network operators present some practical complications in implementing mobile edge computing solutions. Due to these constraints, only a limited number of mobile edge servers can be located in networks. This further makes the placement of a limited number of mobile edge servers a challenging problem given the performance requirements of a wireless network. Additionally, finding the optimal placement of mobile edge servers given a large set of possible placement options further increases the complexity of finding optimal placement strategy for mobile edge servers.

The large solution space of mobile edge server placement options can be improved by collocating mobile edge servers with existing cellular network base stations or wireless network cluster heads [4]. In this work, we follow a similar strategy where mobile edge servers are placed within an already existing cellular or wireless network infrastructure disbarring the search space for optimal placement strategy from exploding. Further, the optimal placement strategy should take into consideration the individual mobile edge server's workload, access delay and application specific requirements into consideration. Various solutions have been proposed in literature to solve the edge servers placement problem [5–11]. However, most of these solutions apply heuristic algorithms or some sort of linear or quadratic optimization technique. This has motivated us to propose an online learning based solution for the mobile edge server placement problem with special emphasis on possible security threats and its solutions.

In our proposed solution, an RL agent is employed as mobile edge server that learns the dynamics of the environment and chooses the best placement strategy maximizing the reward which is dependent on the utility function. The number of RL agents is equal to the number of edge servers in the network. This demands for information exchange between the RL agents to maximize network-wide utility. In the proposed work, we implement hysteretic Q-learning for coordination which is a decentralized multi-agent RL technique allowing the agents to adopt independent actions by maximizing a common goal. Further, we analyze the security threats and countermeasures that may arise due to the information sharing between the edge servers acting as RL agents.

The primary contributions of this work are:

- The mobile edge server placement is formulated as multi-objective optimization problem which is then solved using a multi-agent RL approach. The objectives of the proposed solution include reducing the access delay and balancing edge servers workload. Further, experimental results are obtained by applying the proposed solution on base station dataset provided by Shanghai Telecom to analyze the performance of the proposed technique;
- We discuss different scenarios in which the proposed architecture's security can be breached if the exchanged data between RL agents is altered. Further, we discuss the counter strategies to tackle with the arising security issues.

The rest of the paper is organized as follows: In Section 2, we discuss the existing theories on the placement of edge servers. We discuss the preliminaries of RL in Section 3. In Section 4, we provide an overview of our proposed solution using RL. In Section 5, we describe network and RL modeling and its implementation. Section 6 explains our findings for various system configurations and parameters. Section 7 provides details on the identification of the security issues involved in the exchange of information between edge servers. Finally, Section 8 concludes our work and suggests future research directions.

## 2. Literature Review

The literature review related to mobile edge computing can be divided into two parts. The first part inherently deals with efficient resource allocation in a MEC network. Although the second part, which has been trending recently, tackles the mobile edge server placement problem in a MEC network. It is important to note that the solutions provided for resource allocation problems consider an arbitrary placement of edge servers. Although the second part deals with deployment strategies for edge servers such that desired quality of service is met. To that end, this literature review discusses pioneer works in both of these parts followed by the contributions of this work.

In literature belonging to efficient resource allocation, the authors have proposed an online learning based solution for the resource allocation, scheduling or offloading problems in MEC networks [12–15].

The proposed solutions have studied MEC problems in different aspects ranging from computation offloading schemes in MEC to mobile edge application placement. However, they have not explicitly discussed the mobile edge server's positioning as a challenging problem. For example, the authors in [13], have proposed a deep RL solution to allocate computing and network resources adaptively in MEC to reduce the average service time and balance resource usage under varying MEC environment conditions. In [12], the authors propose an RL based management framework to manage the resources at the network edge. They propose a deep Q-learning based algorithm to reduce service migration in MEC aiming at operation cost reduction. In both these proposed approaches and others [14,15], RL has been used to propose a solution to resource allocation challenges in MEC architecture.

On the other hand, the literature belonging to efficient deployment strategies of edge servers includes solving edge server placement problem using genetic, simulated annealing, and hill-climbing algorithms [5,6], k-means clustering with quadratic programming [8], queuing theory and vector quantization technique [16], graph theory [12], cost constrained multi-objective problem [10], and integer programming [11] to optimally place mobile edge servers in a wireless network.

In [5], the authors formulate the problem as a constrained multi-objective problem to balance workloads of mobile edge servers and reduce network access delay. To find the optimal solution, the authors utilize genetic, simulated annealing, and hill-climbing algorithms to show the effectiveness of the proposed solution. The authors in [6] use data mining techniques, such as a non-dominated sorting genetic algorithm to ensure reliability and low latency in social media services using mobile edge computing.

In [7], the authors present an edge provisioning algorithm that can find the ideal edge locations and map them to their physical locations in a MEC network. In [8], the authors make use of k-means algorithms to solve edge placement problems with mixed-integer quadratic programming. The authors in [9] propose a queuing network based solution to find the best position for cloudlets in a cloud computing network. In [10], the authors have proposed an integer programming solution to find the optimal strategy to place mobile edge servers in smart cities.

The authors in [16] propose an optimal edge server deployment strategy using queuing theory and vector quantization technique, with the aim to minimize the service providers cost and service completion time. The authors in [17] uses graph theory to minimize access delay and the number of edge servers in a MEC network. In [18], the authors develop a two-stage solution to optimally place heterogeneous edge servers in MEC using game theory concepts to optimize service response time.

Although the edge server deployment problem has recently received traction from both academia and industry, the proposed solutions assume global information available at a centralized controller which is responsible for the deployment of edge servers. However, in a realistic environment the changing network condition, user mobility, and traffic patterns will make such centralized solution not scalable due to the large amount of processing required at the centralized controller at each transmission time interval. Considering these reported lacking, the proposed solution provides a distributed and learning based solution to mobile edge server placement problem while considering both delay minimization and edge servers load balancing. To the best of our knowledge, the problem of mobile edge server placement has not been solved through RL. This work serves as a primer on distributed placement strategy in MEC in which each edge server on a local set of observations finds its' optimal placement by exchanging limited set of information with other edge servers in the network. The information exchange between the edge servers is an essential part of a distributed edge server placement solution. To that end, this paper investigates the security concerns in implementing a multi-agent RL solution for mobile edge server placement problems and its possible counter-measures.

## 3. Reinforcement Learning

This section briefly discusses the concepts of reinforcement learning (RL) and its extension to a multi-agent RL.

### *3.1. One-Agent RL*

RL algorithms are built on Markov decision processes (MDP) that allow the agent to receive a reinforcement signal from the environment steering it towards an optimal action policy. MDP is defined as $\langle O, A, P, \rho \rangle$, where $O$ is the set of observations or states perceived from the environment, $A$ is the discrete or finite set of actions, $P : O \times A \times p \rightarrow [0,1]$ is the probabilistic state transition function, and $\rho : O \times A \times O \rightarrow R$ is the reward function.

At any time step $i$, the action $a_i \in A$ influences the environment state to change from $o_i$ to $o_{i+1}$ with a transition probability of $P(o_i, a_i, o_{i+1})$. In return of implemented action, the agent receives a scalar reward $r_{i+1} \in R$ according to the expression $\rho : r_{i+1} = \rho(o_i, a_i, o_{i+1})$. The overarching target of an RL agent is to adapt an action policy that maximizes the discounted future expected reward which is given as:

$$Q^{\pi}(o,a) = \mathbb{E}[R_i | o_i = o, a_i = a, \pi], \tag{1}$$

where $R_i = \sum_{j=0}^{\infty} \gamma^j r_{i+j+1}$ is the reward signal, $\gamma \in [0,1]$ is the discount factor and $Q^{\pi} : O \times A \rightarrow R$ is the Q-function representing the discounted future expected return for a state-action pair.

Mathematically, the maximum value for the discounted expected return is characterized as $Q^*(o,a) = \max_{\pi} Q^{\pi}(o,a)$, which can be learned and estimated using Q-learning in the absence of probabilistic state transition and reward functions [19]. It is theoretically proven that the Q-learning algorithm converges to the optimal solution under certain conditions [19]. The Q-learning algorithms allow a RL agent to iteratively learn the estimates $Q^*$ based on its interactions with the environment using the formula:

$$Q_{i+1}(o_i, a_i) = (1-\alpha) Q_i(o_i, a_i) + \alpha(r_{i+1} + \gamma \max_{a_{i+1}} Q_i(o_{i+1}, a_{i+1})), \tag{2}$$

where $\alpha \in [0,1]$ defines the learning rate of the Q-learning algorithm.

### *3.2. Multi-Agent RL*

In a multi-agent RL problem, multiple agents using RL algorithms interact or compete to maximize a well-defined goal. The complexity of multi-agent RL is comparatively more than a one-agent RL solution since the use of multiple RL agents allow the environment to be jointly influenced by the actions of all agents which leads to non-dominance of a specific action policy. Optimal behavior of a multi-agent RL is reached when each agent operates in the Nash equilibrium which is difficult to visualize in a practical applications [20]. In the Nash equilibrium, each RL agent assumes the unvarying behavior from other agents and maximizes its own reward. Due to the complexities involved in implementing Nash equilibrium in practical applications, we discuss a couple of algorithms that can deal with multi-agent RL problems.

#### 3.2.1. Independent Agents

In this method, the RL agent follows a coordination-free strategy with other agents by assuming each agent's independence. This is equivalent to implementing one-agent RL for each agent in the problem without initiating any coordination. The formulation of multi-agent RL problem as independent agents will simplify the solution but it will also make the convergence difficult due to the non-stationarity of independent agents [21].

#### 3.2.2. Indirect-Coordinating Agents

In this method, the RL agent follows a coordination strategy with other agents. The action selection strategy comprises of the joint action of all agents which is made on the reward feedback received from the environment. Although, the learning is still independent but a common objective exists between the RL agents which it tries to maximize. We discuss one such method of indirect-coordination multi-agent RL method which is called hysteretic RL. In hysteretic RL, the agents take actions independently but the reward function is shared between all agents given as [22]:

$$\delta \leftarrow r - Q_k(o, a_k) \tag{3}$$

$$Q_k(o, a_k) \leftarrow \begin{cases} Q_k(o, a_k) + \mu\delta, & \text{if } \delta \geq 0 \\ Q_k(o, a_k) + \sigma\delta, & \text{else} \end{cases} \tag{4}$$

where learning rates $\mu$ and $\sigma$ are between 0 and 1, $r$ is the reward based on the feedback returned by the environment and $Q_k(o, a_k)$ is the $Q$-value of $k$th agent. The core idea behind hysteretic RL is to penalize agents for taking a bad action.

## 4. Multi-Objective Problem Formulation

A mobile edge server positioning problem can be written as an undirected graph, such that the location of base stations in existing cellular architecture makes the vertices of the graph and the base station's distance to mobile edge servers is represented as edge weights. A finite set of mobile edge servers $S$ can be collocated with a set of base station $B$, such that the number of mobile edge servers will always be less than the number of base stations, as shown in Figure 1. As discussed in Section 1, the constraint of collocating mobile edge servers with the existing base station's location is to reduce the virtually infinite solution space of optimal mobile edge server positioning. Assuming a straightforward communication channel between base stations and mobile edge servers, the access delay at edge devices can be defined as the Euclidean distance ($d_b$) between a mobile edge server and base station where $b \in |B|$. The workload of a base station ($t_b$) is defined as the processing of incoming call and flow requests from edge devices.

The desired key performance indicators in MEC are reduced network access delay and balanced load on mobile edge servers which is why the mobile edge server positioning problem in this work is devised to improve these key performance indicators while finding an optimal placement of $S$ mobile edge servers. The goals of the formulated problem are to (i) reduce the access delay or latency between mobile edge servers and base stations, and (ii) balance the workload of mobile edge servers. The key assumptions in formulating the mobile edge server positioning problem considered in this work are:

- A mobile edge server can offload processing and storage requests from more than one base station;
- A base station can offload processing and storage requests to one or more mobile edge servers. If a base station is offloading processing and storage requests to more than one mobile edge server then the workload of incoming mobile call and flow requests at a base station from edge devices will be shared among connected mobile edge servers;
- A mobile edge server is hosted and collocated at a location where a base station in an already existing network infrastructure is present.

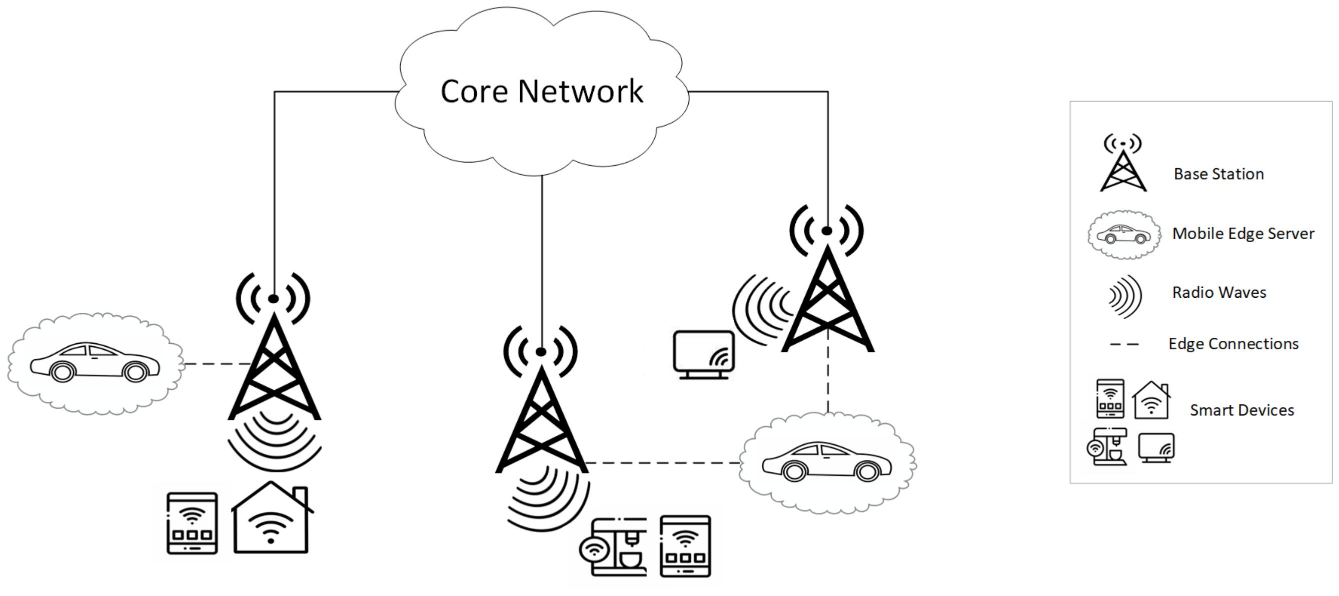


**Figure 1.** Edge servers placement in mobile edge computing.

The workload of a *s*th ($s \in |S|$) mobile edge server ($T_s(\ell)$) is made dependent on the processing and storage requests offloaded from connected base stations, such that $T_s(\ell) = \sum_{b \in |B|} t_b$, $\ell$ is the positioning arrangement of mobile edge servers, $t_b$ is the incoming call and data requests from edge devices to *b*th base station. It is important to note that in MEC, the base station acts as a relay node transferring the incoming call and data requests from edge devices to mobile edge servers. Similarly, access delay is devised as the sum of Euclidean distances from a *s*th mobile edge server to one or more base stations which are offloading incoming requests processing and storage to *s*th mobile edge server, such that, $D(\ell) = \sum_{b \in |B|} \mathbf{d_b}$. Balancing the workload of mobile edge servers ensures that no edge servers is overloaded with offloading requests while some mobile edge server's processing capacity is underutilized. Mathematically, the standard deviation of each mobile edge server's workload is used to devise workload balancing metric ($W(\ell)$) in a MEC, such that,

$$W(\ell) = std(T_j, T_k) \quad \forall j, k \in |S|. \tag{5}$$

Finally, the cost function of multi-objective constrained optimization problem can be defined as:

$$C(\ell) = \beta W^{'}(\ell) + (1 - \beta) D^{'}(\ell), \tag{6}$$

where superscript $z^{'}$ denotes a normalized value of variable $z$, and $\beta \in [0, 1]$ is the weightage parameter.

Therefore, the formulated mobile edge server positioning problem can be defined as:

1. Find mobile edge server positions such that network access delay is minimized; and,
2. Find the edge connections ($x_{bs}, \ \forall\, b \in |B|, s \in |S|$) for which the mobile edge server's workload is balanced where $x_{bs}$ is an indicator function whose value is 1 if a base station is connected to *s*th mobile edge server otherwise 0 if its not connected to *s*th mobile edge server.

Mathematically,

$$\min_{\ell \,\in |B|} \quad C(\ell)$$

such that,

$$\sum_{s=1}^{|S|} x_{bs} \leq |S| \tag{7}$$

where constraint (7) ensures that a base station offloads processing/storage requests to no more than $S$ mobile edge servers. The above formulation of mobile edge servers positioning is a mixed-integer linear program problem that is NP-hard in nature due to the non-linearity of constraint given in (7) [10]. Therefore, this work proposes to solve the mobile edge servers positioning problem using multi-agent RL technique.

## 5. Proposed Solution

Mobile edge computing architecture is beneficial in providing services to a densely deployed network with low-latency and high-throughput requirements [3]. However, there are certain limitations attached to the MEC architecture. First, as explained above, the cost of infrastructure deployment and maintenance is high, therefore, dense deployment of edge servers is not a cost-effective solution. Second, the service requirement of users changes with respect to time, therefore, a certain strategy of mobile edge server's deployment may be optimal for a specific time while it would be sub-optimal for other times. The varying requirements of mobile users due to mobility require that the proposed solution should be able to adapt to the changing scenarios.

One option could be to manually configure the network at different times of the day to make sure that the edge servers deployment is optimal, however, the associated operator expenditure costs may not be feasible for an operator. To circumvent that, an online learning paradigm, such as RL can be effective in dealing with the changing environment conditions. In RL, the environment is modeled as MDP which allows a RL agent to learn the optimal action policy by interacting with the environment. In our proposed approach, each mobile edge server will be working as a RL agent and the environment will be modeled as the mobile edge computing network with base stations, and user devices. Each RL agent will be taking actions independently based on the perceived notion of state from observations and measurements of the environment, however, reward will be computed based on the network-wide delay and workload observed which would require information exchange, as shown in Figure 2. The network-wide utility is defined as the average communication delay and edge server's workload for all edge servers in the network. The objective of the proposed work is to find a mobile edge servers positioning strategy, such that it caters to the needs of data rate requirements of users, as well as it should be able to minimize the delay and maintain workload balancing between edge servers. In this section, we discuss the methodologies adopted for environment and RL agent design.

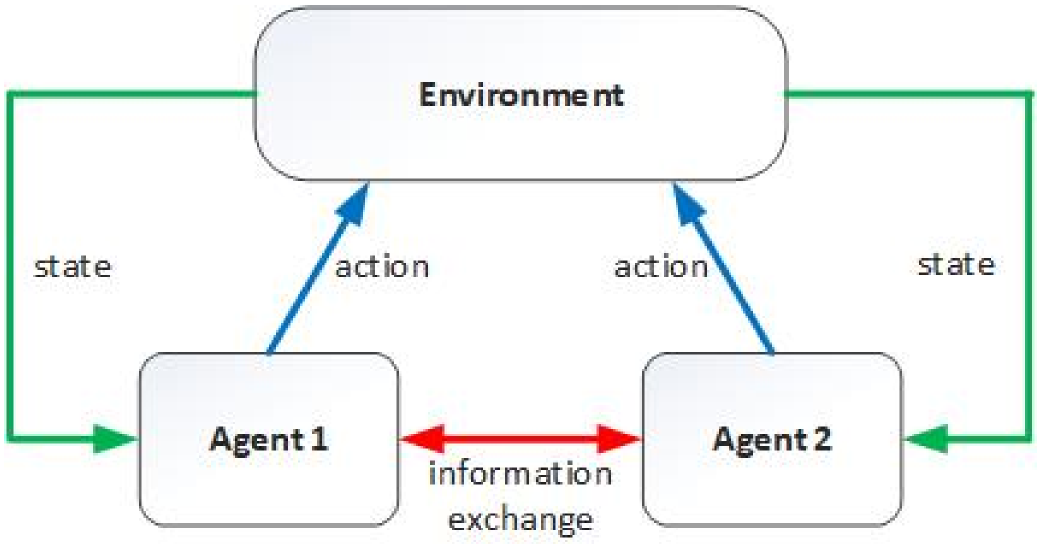


**Figure 2.** Multi-agent RL assisted mobile edge computing.

### *5.1. Environment Design*

To make the proposed environment design realistic, we make use of base station locations and call and data requests dataset from Shanghai Telecom that include record of approximately 7 million call and data requests made through 2766 base stations from 9481 edge devices [8,23,24]. Each call and data record is a tuple of request access time by an device from a base station. Shanghai being a heavily populated city makes it a suitable dataset to implement a mobile edge server placement solution in an ultra-dense MEC network. Figure 3 shows the base station distribution in Shanghai, China where each dot is a location of base station and the color of a dot represent the intensity of incoming call and data requests from edge devices.

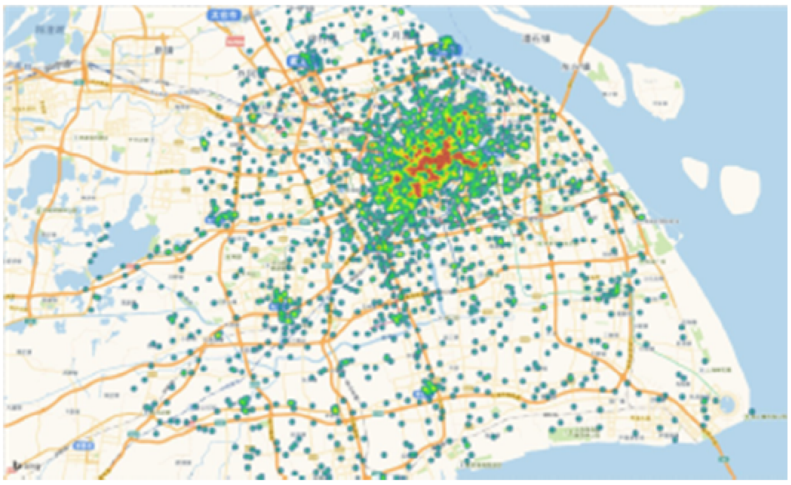


**Figure 3.** Graphical depiction of base station locations in Shanghai Telecom dataset [8,23,24].

The graphical depiction of base station locations is important to realize that a mobile edge server placement solution would require the edge servers to move to any of other base station locations. This means if there are 2766 base stations in the network then a mobile edge server can move to any of these locations. However, there are two problems associated with this assumption. First, the number of locations a mobile edge server can move at each transmission time interval would be dependent on the number of base stations in the network which would make it not scalable if the number of base stations is too high. Second, in a realistic world a mobile edge server would be deployed in a movable object, such as a vehicle, such that limiting the movement of edge servers to only nearby base station locations.

Considering the above two problems discussed, we transform the distribution of base stations given in Shaghai Telecom dataset to a contour line. A contour line links the base station locations with a line joining the two nearest base stations. This transformation of actual locations of base stations to contour line has following benefits:

- Base stations nearest to each other connected with a line allowing the mobile edge servers to move between nearest base station locations in the search of optimal placement strategy;
- The search space of mobile edge server placement becomes scalable. Even if the number of base stations are increased in the MEC network, the solution space will not explode.

In Figure 4a, we show the simulation depiction of base station locations available in Shanghai Telecom dataset. Note that the dataset assumes base station locations in two-dimensional space. Figure 4b shows the contour representation of base station locations, such that each base station is represented a point on a two-dimensional space which is connected to two nearest base stations. The contour line enables the transformation of actual dataset values to so that a mobile edge server can only move between two adjacent locations.

To quantify workload of a mobile edge server and delay experienced by a user equipment, we make use of records available in Shanghai Telecom dataset. The workload of a mobile edge server is quantified by summing up the requested call and data rates from the connected base stations. As a base station offloads its requested computational processing to the connected mobile edge server, therefore, summing up these requested rate is a reasonable assumption [10]. The delay experiences by a user will be proportional to the distance between base station and mobile edge server locations assuming that user equipments are present in close vicinity to base stations [10]. Therefore, the edge device access delay is defined in term of the sum of Euclidean distances from a base station to mobile edge servers to which incoming call and data requests are offloaded for processing or storage.

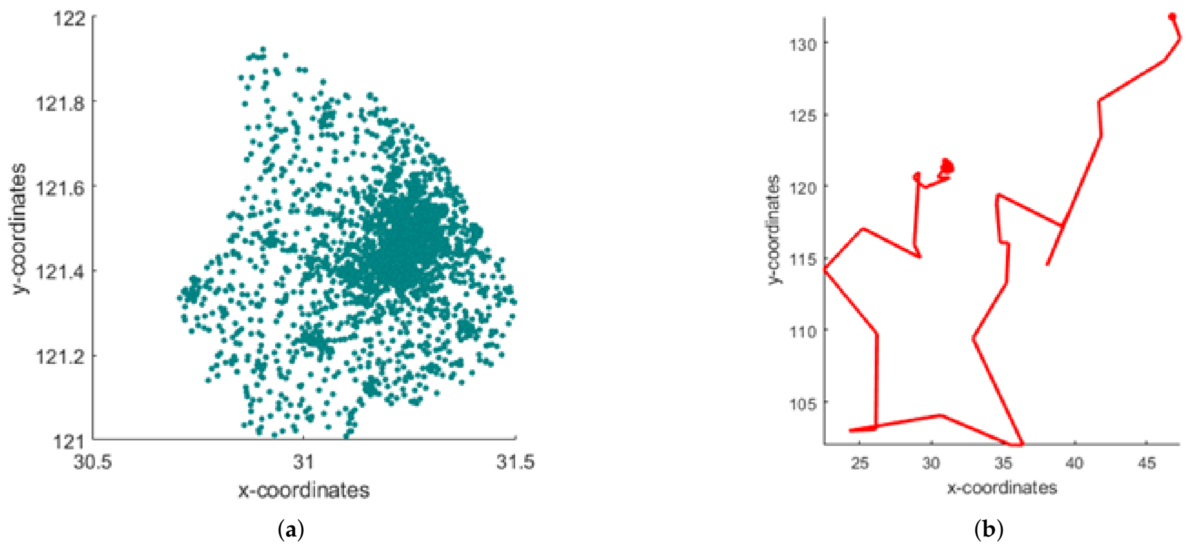


**Figure 4.** Transformation of base station locations in Shanghai Telecom dataset to contour line. (**a**) Simulated depiction of base station locations. (**b**) Contour line joining nearest base stations.

### *5.2. RL Agent Design*

In this part, we aim to solve the optimization problem formulated in Section 4 for each mobile edge server using RL (see Algorithm 1). The proposed approach considers a scenario where each mobile edge server is placed on a movable vehicle that has the ability to move within the network, as shown in Figure 1. The movement of a moving vehicle is controlled by the actions of a RL agent that aims to learn the optimal placement strategy by maximizing the reward. There are three main components involved in the design of RL agent: action space, state space, and reward.

#### 5.2.1. Action Space

Action space in the proposed work is a set of actions by which the mobile edge server change its locations. These actions are updated at the end of an epoch which is dependent on the change in network traffic. In the proposed work, actions are formed to move the mobile edge server between adjacent locations. Since we have formed a contour line from actual base station locations restricting a mobile edge server to move to only two possible neighbor locations. The action space is comprised of a set of three distinct actions by which a mobile edge server can either move to adjacent location on the right or it can move to adjacent location on the left or it stays at the same location. These set of actions will be available for each mobile edge server with the assumption that multiple mobile edge servers can be positioned in the same mobile edge server location.

**Algorithm 1:** Multi-agent RL assisted mobile edge computing.

```
Input: $\alpha$, $\gamma$, $\mu$, $\sigma$, $\epsilon$
$A = \{$left,right,no change$\}$.
$O = \{D(\ell)\}$.
Initialize $|S|$ Q-tables with random values and set $\epsilon = 1$.
Initialize the locations of edge servers randomly.
while converged or aborted do
    For each edge server observe the state (communication delay) of the
    environment $o$.
    For each server choose one of the action from its action space according to
    $arg\ \max_a Q(o, a)$ or randomly with probability $\epsilon$.
    calculate network-wide utility according to Equation (8).
    update Q-tables according to Equation (4).
    linearly decrease $\alpha$, $\mu$, $\sigma$, and $\epsilon$.
end
```

### 5.2.2. State Space

A state in the proposed work is defined by the communication delay between a mobile edge server and base stations that are offloading call and data requests to a mobile edge server. The communication delay as discussed in Section 4 is proportional to the Euclidean distance between a mobile edge server and connected base stations. Delay, as a stand-alone, will be used to infer the state of the environment. Note that other network features, such as location information, data request rate, etc., can also be used to infer on the state of the environment, however, we have made use of a simplified state space model to (i) show the efficacy of proposed solution and (ii) focus on the security aspects that may arise due to the proposed solution.

Even with a simplified state space containing only delay metric as state variable, the number of possible state values can be infinite. For this reason, we quantize the values of delay between maximum and minimum delay which will vary for different MEC networks.

### 5.2.3. Reward

A RL agent learns from the feedback returned by the environment in the form of rewards. In this problem, a reward is a function of cost values, such that:

$$R_t = C(\ell)_{t-1} - C(\ell)_t \tag{8}$$

The expression in Equation (8) drives the RL agent to take actions, such that the cost is minimized from the previous epoch $t$. Note that the cost function is dependent on the global observations. For example, a mobile edge server must be aware of the workload and delay of other edge servers in order to compute the cost function. This transforms the problem into a coordinated multi-agent RL problem in which the reward function is a function of network-wide metrics. This makes it equivalent to hysteretic RL algorithm discussed in Section 3 which is used by each mobile edge server to implement RL in this work.

The state and action spaces are still dependent on the local observations and an agent take actions independent. The sharing of information between edge servers controls the behavior of mobile edge server's placement which, if changed for some reason, would affect the performance of overall implementation. We discuss further on the type of security breaches and its counter solutions in Section 7.

## 6. Results

In this section, we evaluate the performance of multi-agent RL algorithms for mobile edge server positioning problem by experimenting on the Shanghai Telecom's base stations and incoming call and data requests dataset [8,23,24]. The proposed solution is implemented in MATLAB. Multiple RL agents take actions independently to find the best placement strategy, such that the reward returned by the environment is maximized. The proposed solution performance is measured via the cost function value which drives the reward function value. The list of simulation and RL hyperparameters used during the experiments shown in this work are summarized in Table 1. It is important to note that the optimal number of edge servers in a network depends on a number of factors including the network operators budget, and user traffic demand. The proposed model selects an arbitrary number of edge servers (that should be less than the number of base stations), and finds optimal placement for these edge servers, However, the number of edge servers can be found by probability theory and control system rules [25].

**Table 1.** Simulation parameters.

| Parameter | Description | Value |
|---|---|---|
| $BS$ | number of BSs sampled from the Shanghai Telecom's dataset | 120, 240, 360 |
| $ES$ | number of mobile edge servers controlled by RL agents | 20, 30, 40 |
| $\alpha$ | learning rate | 0.35 |
| $\phi$ | learning rate for hysteretic | 0.30 |
| $\beta$ | weightage of delay over workload in utility function | 0.5 |
| $\gamma$ | discount factor | 0.9 |
| $\epsilon$ | random exploration | 0.15 |

The experimentation presented in this work aim to answer the following questions:

A. Does the proposed solution generalize across different random initialized values used in the experimentation?

B. Is the proposed solution effective in finding the best placement for mobile edge servers when different number of base stations are present in the network?

C. Is the proposed solution effective in finding the best placement for mobile edge servers when the number of mobile edge servers present in the network is varied?

**A. Does the proposed solution generalize across different random initialized values used in the experimentation?**

The objective of this experiment is to show the generalizability of the proposed solution across different initial values of parameters used in the experimentation. The RL agent's initial location and other experimentation parameters, such as $\alpha$, $\beta$, and $\epsilon$ require assignment of an initial value which is set to random values. Therefore, using different seeds for random values will change the initial state of each RL agent. Ideally, the average final cost for all these experiments should be same. However, the use of distinct random seeds makes the initial state set for RL exploration strikingly different presenting an entirely different search space for the RL agents to explore and exploit.

In Figure 5, the cost function values across number of epochs are shown where each epoch represents the instant at which all RL agents take actions. The plotted cost function value is the averaged cost function value of each mobile edge server used in the experimentation. We can observe that for different seed values impacting the initial state of each agent, the proposed solution is able to minimize the average cost function values. Another significant observation is the fast convergence of the cost function values for each seed, shown in Figure 5. These results enable us to claim that even with simplified state space, the proposed solution without the use of any complex deep learning models is generalizable for different initial states.

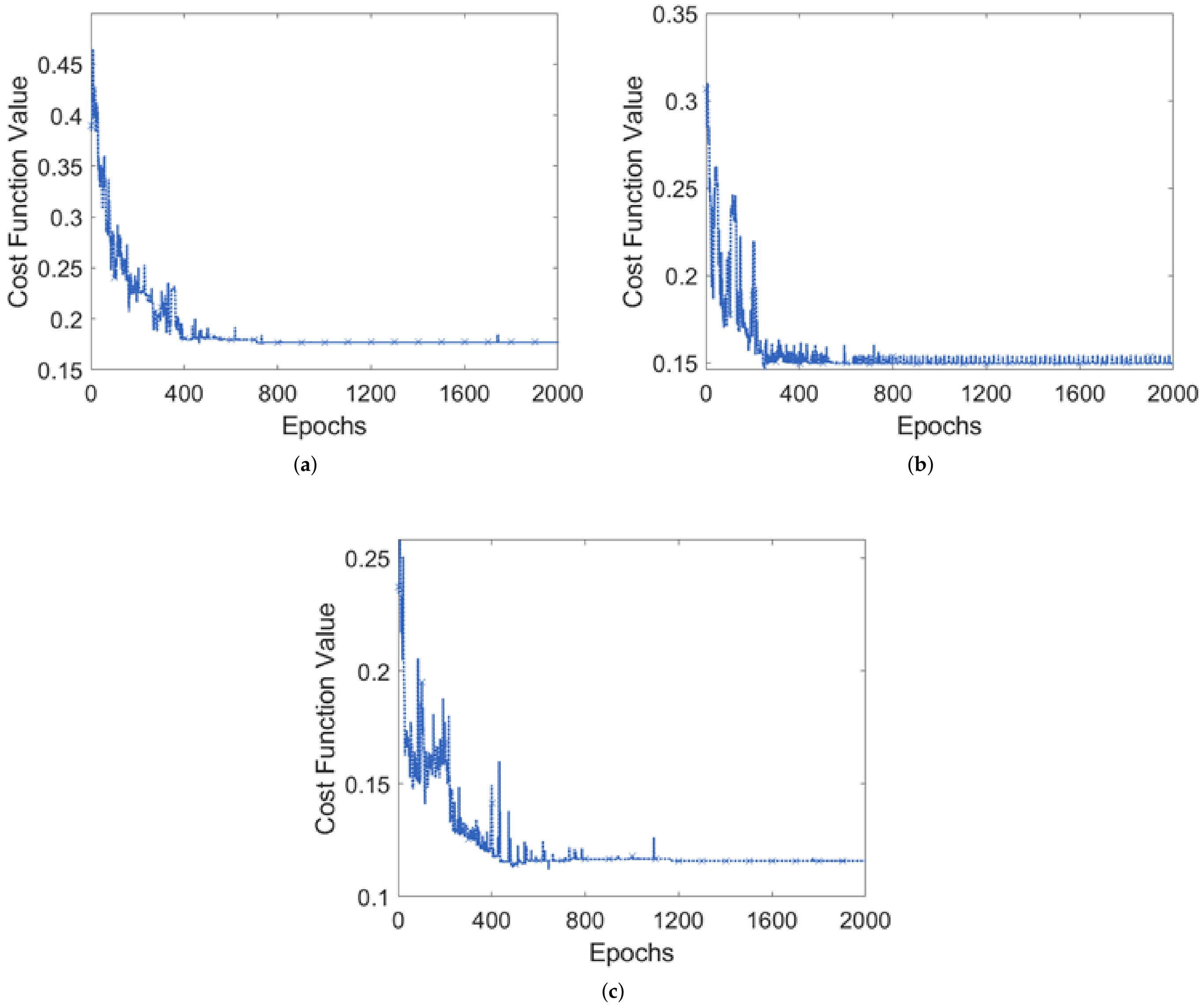
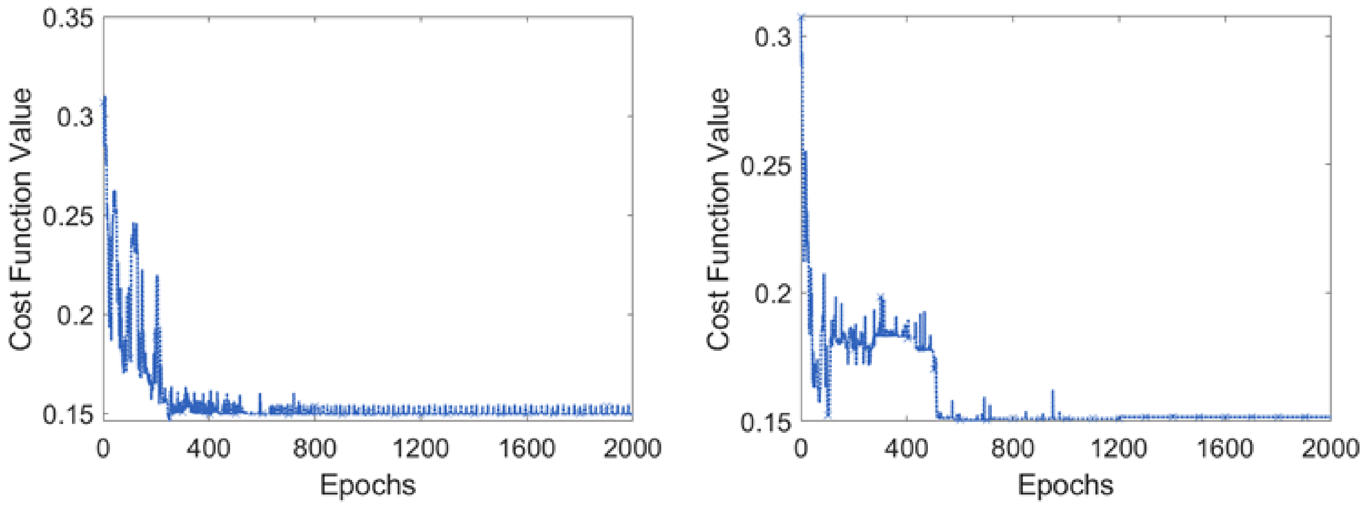


**Figure 5.** Number of BS = 120 and Number of ES = 20. (**a**) Random Seed = 5. (**b**) Random Seed = 8. (**c**) Random Seed = 23.

**B. Is the proposed solution effective in finding the best placement for mobile edge servers when different number of base stations are present in the network?**

In Figure 6, the performance of the proposed solution is shown for varying number of base stations available in the MEC network. Ideally, the change in the number of base stations in the network should not affect the convergence of the proposed multi-agent RL assisted edge servers placement. The results in Figure 6 show that for 120 and 240 base stations available in the network, the cost values converge after a number of epochs. Another significant observation is the increase in cost function value for initial few epochs when number of base stations are 240. This is mainly because RL agents explore the environment by choosing random actions dependent on $\epsilon$ which is reduced at each epoch.

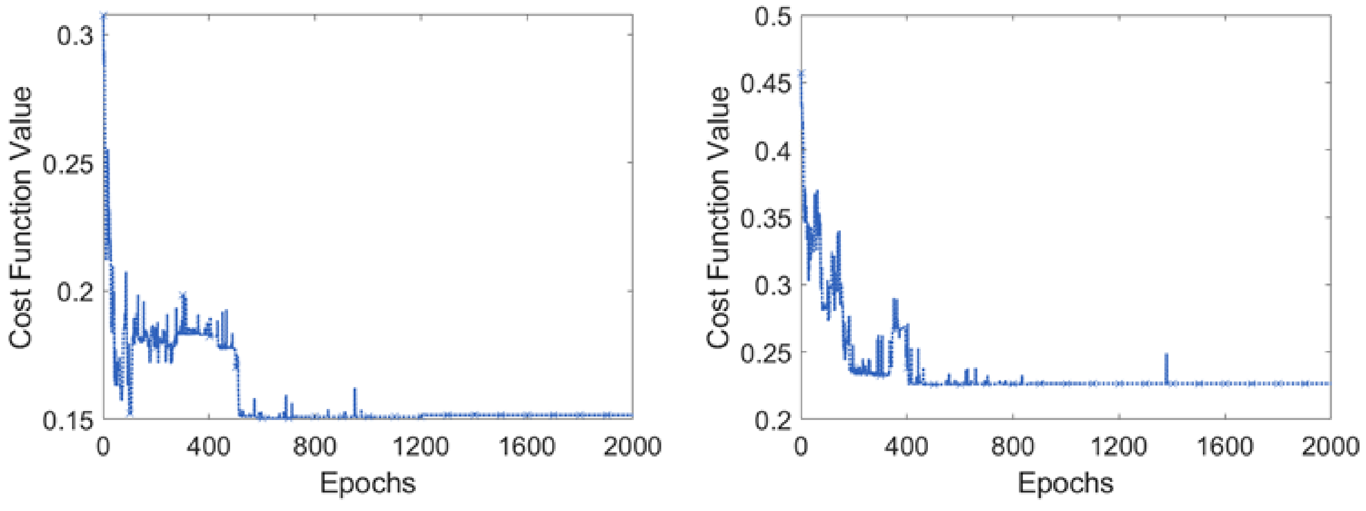


(**a**) (**b**)

**Figure 6.** Number of ES = 20 and Seed = 8. (**a**) BS = 120. (**b**) BS = 240.

**C. Is the proposed solution effective in finding the best placement for mobile edge servers when the number of mobile edge servers present in the network is varied?**

In Figure 7, the performance of the proposed solution is shown for varying number of mobile edge servers. Ideally, varying the number of mobile edge servers in the MEC network should not affect the convergence of the proposed multi-agent RL assisted edge servers placement. The results in Figure 7 show that for 20 and 30 mobile edge servers to be placed in the network, the cost values converge after a number of epochs.



(**a**) (**b**)

**Figure 7.** Number of BS = 240 and Random Seed = 8. (**a**) ES = 20. (**b**) ES = 30.

In Figure 8, we present the results of a toy example in which 03 base stations are placed in a network namely A, B, and C. Base stations A and C are placed in the corners and base station B in the middle. Considering the toy example allows us to evaluate the performance of proposed solution against a numerical solution. Through numerical solution, the optimal locations for edge servers are 'A' and 'C' which can be observed that after a certain number of epochs, both the edge servers converge to its optimal locations.

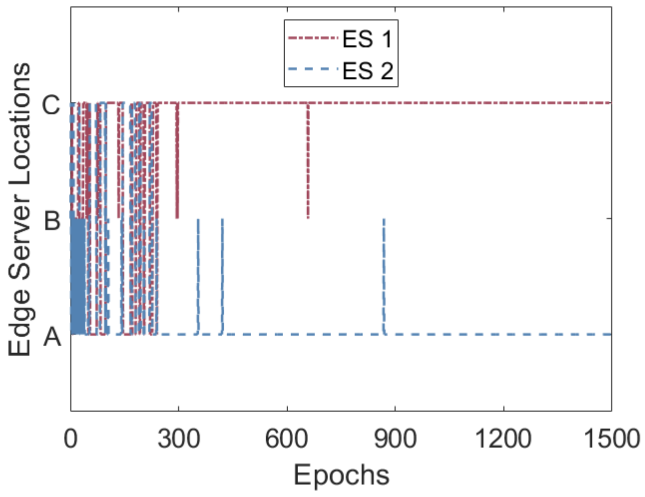


**Figure 8.** Proposed solution comparison with ground truth for a toy example; where three base stations namely A, B, and C, are placed in a network.

## 7. Security Perspective

We demonstrate how the reinforcement learning assisted mobile edge server placement can be performed with multi-agent reinforcement learning coordination techniques. The multi-agent coordination problem may give birth to different security related issues since the working of a reinforcement learning agent is based on the observations from other agents, therefore, if the shared information is modified it will affect the working of entire solution. In the following sections, we present a possible scenario by which security can be breached and present the proposed countermeasures.

### *7.1. Scenarios*

As discussed in earlier sections, workload is the sum of all the data offloaded from connected base stations and delay is the sum of the distance from connected base stations. The agent performs its actions based on the reward function and reward is based on workload balancing and delay.

From a security perspective, the first scenario case we can consider is the when an agent itself or man-in-the-middle (MITM) can alter the information contained in the packets passed between agents. This is done in order to force the agent to change its location to another base station or stay with the same base station despite a need to workload balancing and delay minimization.

Figure 9 presents a scenario of security issues present in the work. Let us assume that the values of workload and delay are increased. This will force the mobile edge server MES1 to move from it current location to a particular location where workload balancing is needed since the agents will assume that it needs to balance the workload and delay by migrating to other base stations. In contrast, if an agent itself or MITM alters the information by decreasing the values of workload and delay, the mobile edge server, MES2, will assume that everything in the network is fine and it does not need to change its location to other base stations.

Since the reward function at the agents makes decision based on the information (workload and delay) received, thus, it will act accordingly. Therefore, in the first scenario, after the information is altered by the malicious node, the mobile edge server MES1 will assume that it needs to move to the MES2 location in order to balance the workload.

The second potential security scenario can be a malicious entity compromised an agent in the network. The attack vector might be different form altering the packet en-route, but the malicious entity can achieve the same impact as the first scenario. Furthermore, a malicious entity compromising an agent in the network may go for eavesdropping with the aim to (a) try to construct a traffic map of the network, (b) build communication patterns between agents, and (c) read communication packets. In the above listed aims, ‘a’ and ‘b’ can help the malicious user to understand the network design and communication patterns between agents. This can assist the malicious entity to mount a network wide attack, for example DDoS. The option ‘c’ allows the malicious entity to read the information communicated between the agents. This might reveal some sensitive information about the agents or applications being executed on these agents.

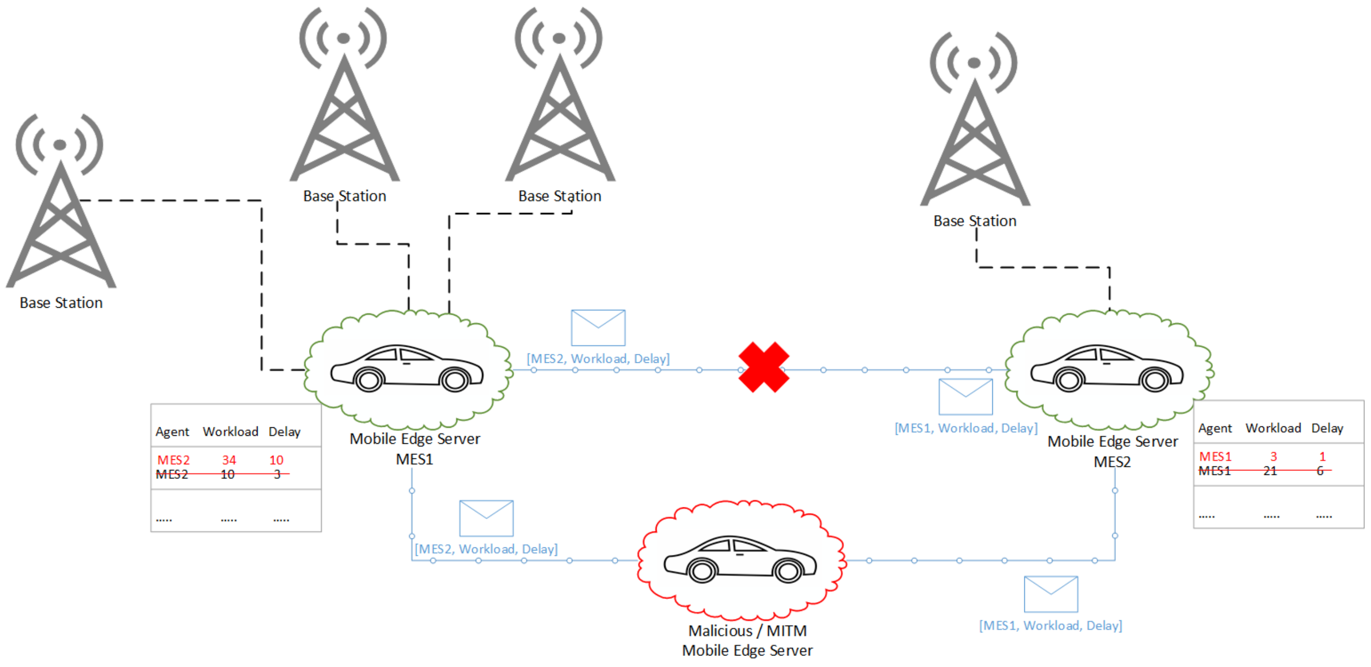


**Figure 9.** Malicious/MITM Agent scenario at a Mobile Edge Server.

The third potential security scenario can be Trojan horse attacks, whether a Trojan horse is embedded in hardware or software. The objectives of such an attacker can be similar to the malicious entity in “second potential security scenario”. The attack objective and impact can also be similar.

The fourth potential security scenario can be insider threat. In this attack, an insider compromises a single node, a collection of the node or the whole network—dependent upon the access of the insider and how senior their role is. The attack objective and impact can be similar to the three scenarios listed before but depending upon the access privilege the impact on the network can be significant.

We are not considering the lack of knowledge or expertise in an organization or a genuine human error as a security threat. As this in most cases leads to the vulnerability that the malicious actors in the above scenarios exploit or the respective impacts.

## *7.2. Countermeasures*

In this section, we explore potential countermeasures to the each of the security scenarios discussed above.

### 7.2.1. Countermeasure to First Security Scenario

As discussed earlier, an agent running on the mobile edge server has global information of workload and delay of all the agents in the network, whereas, the decision is made locally based on the information received and used in the reward function by an agent.

These security issues require verifying the identity of an agent before allowing access to resources in a system. Therefore, an authentication mechanism needs to enable the identity of an agent to be verified and, thus, to prevent it from faking or masquerading.

Additionally, data integrity needs to be ensured to prevent data from being altered or destroyed while being exchanged amongst the agents in an unauthorized manner to maintain consistency. Hence, a secure protocol should withstand such attacks and offer authentication and integrity of the exchanged data.

The cryptosystem we aim to achieve is one where the entities communicate over an insecure network, resulting in both parties needing to provide identity authentication first, and this then proves to the receiver the integrity of the messages. Peer authentication and secure data transmission are vital in our system.

Regarding authentication, public key infrastructure (PKI) provides the means of digital certificate for providing authentication. In our study we assume that all entities have digital certificates generated by the CA.

To achieve integrity between base station and mobile edge server, one may employ integrity encryption techniques, such as HMAC. However, before doing this, both entities should agree first on a secret key. Due to the key distribution problem, key agreement protocols have emerged where the actual key is not transferred on an untrusted channel.

The proposed protocol is divided into four stages:

1. Stage 1 Mutual Authentication Phase:
   We assume that both parties already registered with CA, trust the same CA, and possess their own public key, own private key, own implicit certificate, and CA's public key. Both entities the base station and mobile edge server perform a handshake where both parties exchange their digital certificate to verify the authenticity of each party.
2. Stage 2 Key Agreement:
   After authentication is done in both parties, they should agree on a shared master key. In our protocol, we will use the elliptic curve Diffie Hellman (ECDH) protocol that is most suitable for constrained environments. The elliptic curve cryptosystems are used for implementing protocols such as the Diffie–Hellman key exchange scheme [26] as follows:

   A. A particular rational base point $P$ is published in a public domain.
   B. The base station and mobile edge server choose random integers $\mathrm{k}_A$ and $\mathrm{k}_B$ respectively, which they use as private keys.
   C. The base station computes:
   $$A = k_A * P = (x_A, y_B) \tag{9}$$
   D. The mobile edge server computes:
   $$B = k_B * P = (x_B, y_B) \tag{10}$$
   and then both entities exchange these values over an insecure network.
   E. Using the information, they received from each other and their private keys, both entities compute:
   $$Q = k_A * B = k_A * (k_B * P) \tag{11}$$
   and
   $$Q = k_B * A = k_B * (k_A * P) \tag{12}$$
   respectively. This is simply equal to,
   $$Q = (k_A * k_B) * P \tag{13}$$
   which serves as the shared master key that only base station and mobile edge server possess.
3. Stage 3 Key Derivation:

To reduce the computational complexity on both parties, we assume that the mutual authentication phase is done periodically, only the session secret key is generated from the shared master key for achieving integrity protection algorithm, such as message authentication code (MAC) on each session.
The proposed protocol should use the best option for key derivation function (KDF) that ensures randomness, and we advocate the KDF recommendations in [27], which takes into consideration randomness through the use of random numbers (Nonce) and key expansion. Each peer computes the actual session key PK via the chosen key KDF $\chi$, as:

$$PK = \chi(Q, Nonce) \tag{14}$$

4. Stage 4 Message Exchange:
The exchanged data, such as workload and delay, need to be protected against unauthorized modification, hence HMAC is used to ensure the integrity.
The base station calculates

$$D = HMAC_{PK}((D(\ell), W(\ell)) \tag{15}$$

The base station sends W($\ell$), D($\ell$) and $D$ to the mobile edge server.
The mobile edge server uses the agreed derived session key to calculate HMAC of W($\ell$) and D($\ell$) and verifies its integrity with D.

The performance of the above listed protocol depends on multitude of factors including: the processor speed, availability of specialized cryto-hardware, and communication (network speed). However, to provide a reference performance, we setup a test-bed with each agent node is a Raspberry Pi model B supplied with a Wi-Fi USB dongle TL-WN722N by TP-LINK.

In all the measurements we made, the nodes were configured in ad-hoc mode. Each agent is then connected to a server through an Ethernet connection. The server manages individual agents so as to prepare them for the target scenario and is also in charge of collecting the measurements. In our reference performance measurement, we only consider the scenario of two agents setting up a security communication link directly with each other.

Nevertheless, effective measurement can be done internally on the node initiating the secure channel, called a client, and it can be done at the level of the network data exchanged between the agents of the network and captured with a Wi-Fi card set in monitor mode on the server.

Based on this setup, the stated protocol in this section took 4282 milliseconds (on average) over 100 executions.

### 7.2.2. Countermeasure to Second Security Scenario

Compromising an agent is basically system security problem. Potential countermeasures to this can be hardening the agent environment, pen-testing it before deployment, updating it regularly when new vulnerabilities come alive, having strong access control policies related to the agent configuration, etc.

Another potential solution to this problem can be to have a secure execution environment in individual agents. The secure execution environment can help protect sensitive code during its execution and avoid any malicious entities from interfering with it. Even then, the above listed precautions should be taken.

### 7.2.3. Countermeasure to Third Security Scenario

Protection against Trojan horse attacks, especially related to Trojan horse in the hardware is dependent on secure and reliable supply chains. An organization can test their agents to detect whether they have some non-characteristic behavior. Similar actions can also be taken for the software bases Trojan horse. An effective mechanism can be continuous monitoring of the agent (hardware and software) behavior to detect any stealth Trojan that evades the detection pre-deployment.

#### 7.2.4. Countermeasure to Fourth Security Scenario

Insider threat is a significant challenge to overcome in any large network or organization. A potential countermeasure to such a threat can include limited privilege that only requires single user approval. All sensitive actions should require multiple users to approve and deploy the changes. Employee management and making sure that HR revokes credentials of any employee that is leaving the company. Finally, user network activities and behavior monitor can help minimize any impact from disgruntle employees.

## 8. Conclusions and Future Work

Mobile edge computing facilitates in providing data storage and computational resources to mobile and low-power wireless sensor devices. In this work, we have shown a multi-agent reinforcement learning based solution for the placement of edge services in a mobile network, such that the network latency is minimized and load on edge servers is balanced. The experimental evaluation using Shanghai's Telecom dataset proves that the proposed solution quickly converges. Further, we provided a detailed analysis of the type of security attacks possible in the proposed solution concept. We also listed some of the countermeasures that can be used to deal with the security risks. The effectiveness of the proposed method even with a simple state-space provides a promise to the proposed solution. Much future work remains before the proposed solution can be implemented in the real world, but our findings suggest that this approach has considerable potential. This work serves as the proof of concept for a secure multi-agent RL implementation for edge server placement problem. However, to further validate the results of proposed model, we intend to implement the proposed model in a full-stack emulator such as SIMENA NE5000.

**Author Contributions:** Conceptualization, M.K.K., S.A.G., R.N.A.; investigation, M.K.K., S.A.G., R.N.A.; methodology, M.K.K., S.A.G., R.N.A.; resources, S.A.G.; software, M.K.K., S.A.G., R.N.A.; supervision, S.A.G.; visualization, M.K.K., R.N.A.; writing—original draft, M.K.K., S.A.G., R.N.A.; writing—review and editing, R.N.A., D.S.; funding acquisition, S.A.G. All authors contributed to the final version. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work is supported by King Khaled University under Grant Agreement No. 6204.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Lee, J.; Kim, D.; Lee, J. Zone-based multi-access edge computing scheme for user device mobility management. *Appl. Sci.* **2019**, *9*, 2308. [CrossRef]
2. Bilal, K.; Khalid, O.; Erbad, A.; Khan, S.U. Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers. *Comput. Netw.* **2018**, *130*, 94–120. [CrossRef]
3. Porambage, P.; Okwuibe, J.; Liyanage, M.; Ylianttila, M.; Taleb, T. Survey on multi-access edge computing for internet of things realization. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2961–2991. [CrossRef]
4. Lähderanta, T.; Leppänen, T.; Ruha, L.; Lovén, L.; Harjula, E.; Ylianttila, M.; Riekki, J.; Sillanpää, M.J. Edge server placement with capacitated location allocation. *arXiv* **2019**, arXiv:1907.07349.
5. Kasi, S.K.; Kasi, M.K.; Ali, K.; Raza, M.; Afzal, H.; Lasebae, A.; Naeem, B.; ul Islam, S.; Rodrigues, J.J. Heuristic edge server placement in Industrial Internet of Things and cellular networks. *IEEE Internet Things J.* **2020**, *8*, 10308–10317. [CrossRef]
6. Xu, X.; Shen, B.; Yin, X.; Khosravi, M.R.; Wu, H.; Qi, L.; Wan, S. Edge Server Quantification and Placement for Offloading Social Media Services in Industrial Cognitive IoV. *IEEE Trans. Ind. Inform.* **2020**, *17*, 2910–2918 [CrossRef]
7. Yin, H.; Zhang, X.; Liu, H.H.; Luo, Y.; Tian, C.; Zhao, S.; Li, F. Edge provisioning with flexible server placement. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *28*, 1031–1045. [CrossRef]
8. Guo, Y.; Wang, S.; Zhou, A.; Xu, J.; Yuan, J.; Hsu, C.H. User allocation-aware edge cloud placement in mobile edge computing. *Softw. Pract. Exp.* **2019**, *50*, 489–502. [CrossRef]
9. Jia, M.; Cao, J.; Liang, W. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. *IEEE Trans. Cloud Comput.* **2015**, *5*, 725–737. [CrossRef]
10. Wang, S.; Zhao, Y.; Xu, J.; Yuan, J.; Hsu, C.H. Edge server placement in mobile edge computing. *J. Parallel Distrib. Comput.* **2019**, *127*, 160–168. [CrossRef]

11. Bouet, M.; Conan, V. Mobile edge computing resources optimization: A geo-clustering approach. *IEEE Trans. Netw. Serv. Manag.* **2018**, *15*, 787–796. [CrossRef]
12. Zeng, D.; Gu, L.; Pan, S.; Cai, J.; Guo, S. Resource Management at the Network Edge: A Deep Reinforcement Learning Approach. *IEEE Netw.* **2019**, *33*, 26–33. [CrossRef]
13. Wang, J.; Zhao, L.; Liu, J.; Kato, N. Smart Resource Allocation for Mobile Edge Computing: A Deep Reinforcement Learning Approach. *IEEE Trans. Emerg. Top. Comput.* **2019**, *6750*, 1. [CrossRef]
14. Huang, L.; Bi, S.; Zhang, Y.J. Deep Reinforcement Learning for Online Computation Offloading in Wireless Powered Mobile-Edge Computing Networks. *IEEE Trans. Mob. Comput.* **2019**, *1233*, 1. [CrossRef]
15. Zhai, Y.; Bao, T.; Zhu, L.; Shen, M.; Du, X.; Guizani, M. Toward Reinforcement-Learning-Based Service Deployment of 5G Mobile Edge Computing with Request-Aware Scheduling. *IEEE Wirel. Commun.* **2020**, *27*, 84–91. [CrossRef]
16. Li, B.; Hou, P.; Wu, H.; Hou, F. Optimal edge server deployment and allocation strategy in 5G ultra-dense networking environments. *Pervasive Mob. Comput.* **2021**, *72*, 101312. [CrossRef]
17. Zeng, F.; Ren, Y.; Deng, X.; Li, W. Cost-effective edge server placement in wireless metropolitan area networks. *Sensors* **2019**, *19*, 32. [CrossRef]
18. Cao, K.; Li, L.; Cui, Y.; Wei, T.; Hu, S. Exploring placement of heterogeneous edge servers for response time minimization in mobile edge-cloud computing. *IEEE Trans. Ind. Inform.* **2020**, *17*, 494–503. [CrossRef]
19. Watkins, C.J.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
20. Littman, M.L. Value-function reinforcement learning in Markov games. *Cogn. Syst. Res.* **2001**, *2*, 55–66. [CrossRef]
21. Busoniu, L.; Babuska, R.; De Schutter, B. Multi-agent reinforcement learning: A survey. In Proceedings of the 2006 9th International Conference on Control, Automation, Robotics and Vision, Singapore, 5–8 December 200 ; pp. 1–6.
22. Matignon, L.; Laurent, G.J.; Le Fort-Piat, N. Hysteretic q-learning: An algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA, USA, 29 October–2 November 2007; pp. 64–69.
23. Wang, S.; Guo, Y.; Zhang, N.; Yang, P.; Zhou, A.; Shen, X.S. Delay-aware Microservice Coordination in Mobile Edge Computing: A Reinforcement Learning Approach. *IEEE Trans. Mob. Comput.* **2019**, *20*, 939–951. [CrossRef]
24. Xu, J.; Wang, S.; Bhargava, B.K.; Yang, F. A blockchain-enabled trustless crowd-intelligence ecosystem on mobile edge computing. *IEEE Trans. Ind. Inform.* **2019**, *15*, 3538–3547. [CrossRef]
25. Hajiyev, A. Optimal Choice of Server's Number and the Various Control Rules for Systems with Moving Servers. In *International Conference on Management Science and Engineering Management*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 379–399.
26. Haakegaard, R.; Lang, J. The Elliptic Curve Diffie-Hellman (Ecdh). 2015. Available online: https://koclab.cs.ucsb.edu/teaching/ecc/project/2015Projects/Haakegaard+Lang.pdf (accessed on 10 June 2020)
27. Charan, K.S.; Nakkina, H.V.; Chandavarkar, B.R. Generation of Symmetric Key Using Randomness of Hash Function, In Proceedings of the 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kharagpur, India, 1–3 July 2020; pp. 1–7.