Mohammed, Thaha; Albeshri, Aiiad; Katib, Iyad; Mehmood, Rashid

UbiPriSEQ—Deep reinforcement learning to manage privacy, security, energy, and QoS in 5G IoT hetnets

Published in:
Applied Sciences (Switzerland)

DOI:
10.3390/app10207120

Published: 02/10/2020

Document Version
Publisher's PDF, also known as Version of record

applied sciences

MDPI

Article

UbiPriSEQ—Deep Reinforcement Learning to Manage Privacy, Security, Energy, and QoS in 5G IoT HetNets

Thaha Mohammed [1], Aiiad Albeshri [2], Iyad Katib [2] and Rashid Mehmood [3,*]

[1] Department of Computer Science, Aalto University, 02150 Espoo, Finland; thaha.mohammed@aalto.fi
[2] Department of Computer Science, King Abdulzaziz University, Jeddah 21589, Saudi Arabia; aaalbeshri@kau.edu.sa (A.A.); iakatib@kau.edu.sa (I.K.)
[3] High Performance Computing Center, King Abdulzaziz University, Jeddah 21589, Saudi Arabia
* Correspondence: RMehmood@kau.edu.sa

Received: 1 September 2020; Accepted: 1 October 2020; Published: 13 October 2020

Abstract: 5G networks and Internet of Things (IoT) offer a powerful platform for ubiquitous environments with their ubiquitous sensing, high speeds and other benefits. The data, analytics, and other computations need to be optimally moved and placed in these environments, dynamically, such that energy-efficiency and QoS demands are best satisfied. A particular challenge in this context is to preserve privacy and security while delivering quality of service (QoS) and energy-efficiency. Many works have tried to address these challenges but without a focus on optimizing all of them and assuming fixed models of environments and security threats. This paper proposes the UbiPriSEQ framework that uses Deep Reinforcement Learning (DRL) to adaptively, dynamically, and holistically optimize QoS, energy-efficiency, security, and privacy. UbiPriSEQ is built on a three-layered model and comprises two modules. UbiPriSEQ devises policies and makes decisions related to important parameters including local processing and offloading rates for data and computations, radio channel states, transmit power, task priority, and selection of fog nodes for offloading, data migration, and so forth. UbiPriSEQ is implemented in Python over the TensorFlow platform and is evaluated using a real-life application in terms of SINR, privacy metric, latency, and utility function, manifesting great promise.

Keywords: 5G networks; Internet of Things (IoT); fog computing; edge computing; cloud computing; Deep Reinforcement Learning (DRL); privacy; security; energy; quality of service (QoS)

1. Introduction

We have come a long way towards realizing Mark Weiser's 1991 vision of ubiquitous computing, where anytime, anywhere technologies will penetrate into our daily lives to such an extent that they become indistinguishable from the environments we live in. Cyber-physical systems (CPSs) such as smart cities and societies are examples of such environments [1,2]. The vision is to make systems and applications coexist in these environments, interacting with each other, producing and consuming historical and real-time information emanating from humans, sensors, and machines, and improving quality of life in many areas, for example, transportation [3–6], healthcare [7], and others [8].

The Internet of Things (IoT) is a vital technology to enable sensing, actions, and other interactions among various "entities" in these ubiquitous environments [9–12]. The number of "things" connected to the IoT is expected to reach 50 billion by 2020 due to the massive influx of diverse "things" emerging rapidly [13]. This will enable us to monitor and control our environments at micro-levels, in time and space, advancing the path to providing interactive, personalized, and smarter services. However, it will also require solving challenges related to transport, management and analysis of data generated

from IoT, particularly the challenges related to the 4Vs of big data analytics (Volume, Velocity, Variety, and Veracity) [9,14]. Any movement of data requires energy and could increase latency. The data, analytics, and other computations would have to be optimally migrated and placed in these environments—on the device, or in the cloud, fog, or edge layers—such that various energy efficiency and QoS demands of these applications and the environments are best satisfied. The ubiquitous environments hence will need reliable, high speed, and low latency wireless networks.

5G networks with their significantly higher speeds (50 Mbps—Gbps), lower-latency, reliability, capacity, and other benefits offer a powerful communication platform for ubiquitous environments [15]. However, privacy and security of users and data will pose paramount challenges in the adoption of these ubiquitous environments by governments, industry, and common people. A particular challenge in this context would be to preserve privacy and security while delivering quality of service (QoS) and energy-efficiency.

Several works have tried to address these challenges (QoS, energy efficiency, security, and privacy) in distributed environments [16–21]. Despite these efforts, a number of gaps exist (see Section 2 for a detailed literature review). Firstly, the focus of the existing approaches is on improving or optimizing one or two of the design parameters—QoS, energy efficiency, security or privacy. Optimizing one or more parameters without measuring and counteracting their impact on other parameters is no more acceptable in the emerging ubiquitous environments. Secondly, existing approaches assume a fixed network configuration and attack models, and use fixed strategies in devising solutions. However, real environments are unpredictable and hence a priory identification of all design and operations parameters is not feasible. Solutions are needed that dynamically and adaptively devise strategies and make decisions in rapidly changing ubiquitous environments and holistically optimize performance subject to varying sets of policy constraints.

This paper proposes UbiPriSEQ, a framework that adaptively, dynamically, and holistically optimizes QoS, energy-efficiency, security, and privacy of IoT devices and 5G heterogeneous networks (HetNet) based environments. The Framework is built on a three-layered model of ubiquitous environments (IoT Devices Layer, FRAN Layer, and Core Layer). It comprises two modules (Device Module and Network Module). Device Module executes within the IoT Devices Layer, and Network Module runs in both the FRAN Layer and Core Layer. See Figure 1. For background on Core Layer, FRAN Layer, and IoT Devices Layer, see References [17,22–28].

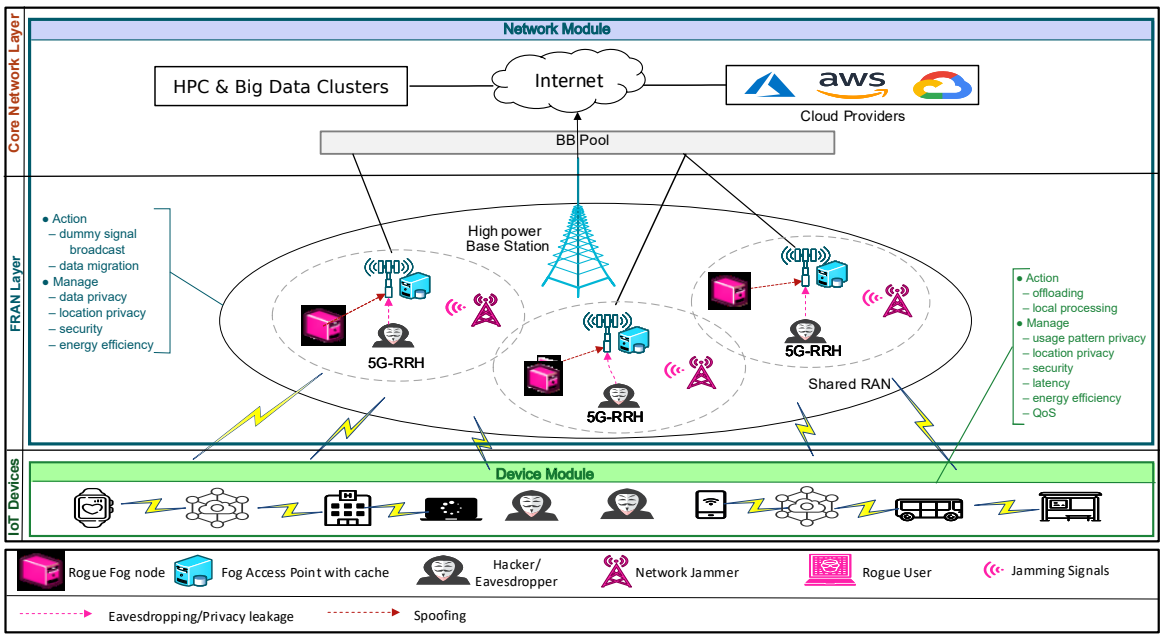

Figure 1. The network and architectural overview of UbiPriSEQ.

UbiPriSEQ allows various IoT devices and other entities to fulfil their computing and data requirements while also managing their privacy and security needs. For example, a smartphone may need to offload a deep learning (DL) computation and associated data to FRAN Layer due to its low-latency constraints and the lack of computing resources to execute the computation on board. Larger and latency-tolerant jobs that require bigger resources are offloaded to Core Layer, which comprises clouds, high performance computing (HPC) clusters, big data analytics facilities, and other infrastructures. The data and computations may also be offloaded or migrated due to security and privacy reasons, or perhaps because the data required by a computational job at a device resides in FRAN Layer. Many other possibilities also exist.

Device Module decides whether to offload computations to the fog or cloud, or run locally. This is done while managing privacy, priority, energy-efficiency, and QoS requirements of the computations. Network Module devises policies for data migration between fog nodes to preserve privacy. It also broadcasts dummy signals to confuse eavesdroppers to address privacy and security concerns. The two modules continuously learn from each other and adaptively optimize the ubiquitous environment. All in all, UbiPriSEQ devises policies and makes decisions related to a number of important parameters including local processing and offloading rates for data and computations, radio channel states, transmit power, task priority and selection of a fog node for offloading, data migration, and others. These decisions are made dynamically, adaptively, and holistically, as needed, and based on the environment to optimize QoS, energy efficiency, security and privacy. The operational intelligence in UbiPriSEQ is provided by Deep Reinforcement Learning (DRL).

The UbiPriSEQ framework proposed in this paper is novel for several reasons. As mentioned above, the earlier works have focused on one or two of the network design parameters while we attempt to holistically optimize QoS, energy efficiency, security, and privacy (data privacy, user location privacy, and user pattern privacy). Secondly, UbiPriSEQ adaptively and dynamically deals with the environment by continuous learning, and does not need a priory complete specification of the environment, security attack models, and other design parameters. Thirdly, we use DRL to provide adaptive intelligence which has not been investigated before in 5G networks. Fourthly, most existing works use simulated data for performance evaluation while we have used SpMV, a real-life application to study UbiPriSEQ performance, and this has not been reported before. The SpMV operation is central to deep learning algorithms and this per se would create a significant impact.

The rest of the paper is organized as follows. Section 2 reviews the relevant works. Section 3 discusses system design requirements. Section 4 describes the UbiPriSEQ framework and Section 5 evaluates the system. Section 6 concludes and gives future directions.

2. Related Work

Many works have focused on offloading computations to fog or edge devices in order to address latency and energy consumption. Zhang et al. [29] and Liu et al. [30] address latency optimized and energy optimized offloading respectively. Sun et al. [31] optimizes the computational energy efficiency and latency on offloading to the mobile edge. Jiao et al. [20] and Dong et al. [19] discuss multi-user task offloading, which involves multiple users offloading tasks to an edge server. Partial offloading which involves both local processing and edge processing to reduce latency and energy have been discussed by Ren et al. [32] and Kuang et al. [33] respectively. Cooperative offloading, i.e., offloading computation tasks with the help of relay nodes to MEC servers far from the users to meet the computational demands, have been discussed in Reference [34]. Task offloading with energy-harvesting have been discussed by Mahmood et al. [35] and Xu et al. [36]. Extensive literature on this work can be found in References [21,37]. However, these works does not address the privacy or security issues faced by the users and the user devices.

Privacy (especially location-based privacy) have been explored in various networking environments such as wireless sensor networks [38], wireless networks [39], and location-based services [40]. Data privacy has been studied extensively leading the development of several

new encryption techniques and distributed multi-party computational solutions; see for example, References [41–43]. Large number of works have discussed the privacy and security issues while offloading computations in an MEC network (see the surveys in References [17,18]). However, most of these works address privacy and security issues derived from older cloud based systems. There are only a few works that address privacy and security issues specific to edge and fog networks. He et al. [44] have proposed Constrained Markov Decision Process (CMDP) based technique for offloading computations to the mobile edge to reduce energy consumption and delay with a goal to maintain a pre-specified level of privacy (location and usage pattern). Preserving location privacy while optimizing the energy consumption in presence of a compromised edge service provider have been discussed in Reference [45], where a deep post decision state (PDS) learning algorithm that exploits information about energy harvesting process, has been proposed to offload to various edge devices so that the energy consumption is minimal while maintaining specified privacy. He et al. [46] propose privacy-preserving and cost-efficient (PEACE) task offloading scheme that can preserve user location privacy when there is a user presence inference attack that invades user privacy when offloading tasks to the edge nodes. PEACE is developed based on Lyapunov optimization framework. These works have mainly focussed on offloading with privacy without considering security and other design challenges.

Even though physical layer security techniques such as noisy signals and receiver jamming ([47]) have been utilized to promote the security and privacy of the conventional wireless communication systems [48], there are very few recent works that utilize such techniques. A novel secure offloading scheme based on physical layer has been proposed in Reference [49]. In this work, the edge server broadcasts jamming signals and utilizes full duplex communication to impede the eavesdroppers and reduce the self interference. The optimal jamming signal and offloading ratio is solved as a bilevel optimization problem. Efficient offloading algorithm is developed based on this technique. Similarly, Zhao et al. [50] utilize a physical layer based technique and massive multiple input multiple output (mIMO) to provide security while the user's total energy consumption is minimized by jointly optimizing the user's offloading data bits, transmission power, and offloading rate. A user side secure sub-carrier allocation has been studied in Reference [50]. Unmanned aerial vehicle (UAV) secure offloading for single antenna mobile edge systems have been studied in Reference [51]. A block chain based cooperative computation offloading and resource allocation scheme considering security and privacy have been discussed in Reference [52]. These works have mainly focussed on offloading with security without considering privacy and other design challenges (except the last work [52] that considers both security and privacy).

The literature review shows that the focus of the existing approaches is on improving or optimizing one or two of the design parameters while we propose to holistically optimize QoS, energy efficiency, security, and privacy. Also, the way our proposed framework UbiPriSEQ adaptively and dynamically deals with the environment by continuous learning is not seen before. Moreover, the use of DRL to provide adaptive intelligence in 5G networks is also novel. Finally, no work similar to UbiPriSEQ exists that have used a real-life application (SpMV) to implement and study the system performance.

3. UbiPriSEQ: System Requirements

We discuss here the design challenges for 5G IoT HetNets [53–56], which we have used as the software design requirements for the UbiPriSEQ system.

3.1. QoS

Satisfactory QoS is the primary goal for network design. QoS metrics include throughput, packet loss, bit error rate, throughput, latency, jitter, and others. While all QoS metrics are important, latency has become a bottleneck due to emerging interactive, real-time, and safety-critical applications, for example, online gaming, augmented reality, 4K video streaming, drone control, and remote surgery.

3.2. Energy-Efficiency

Energy-efficiency (using less energy to perform the same task) has become a key goal in designing any systems, vehicles, buildings, software, and so forth. The drivers include reducing greenhouse gas emissions for planet sustainability, costs, and power requirements. For instance, offloading computations to fog instead of cloud could reduce energy usage significantly.

3.3. Privacy

We have addressed the following types of privacy in UbiPriSEQ design.

3.3.1. Location Privacy

Service providers, eavesdroppers, and other attackers can estimate the location of the users with ease via analyzing the offloaded data. Specifically, the user location can be inferred in several ways. Analysis of offloaded data pattern, including the size and channel state, enables the service providers (fog nodes) as well as eavesdropping attackers to deduce the distance to the mobile user. Moreover, the service providers with multiple fog nodes or attackers can use techniques like triangulation to estimate the exact location of the user when the data is offloaded to multiple fog nodes. Hence, the offloading strategy should grantee location privacy.

3.3.2. Usage Pattern Privacy

This is a critical issue missing in the majority of existing research [44]. The fog nodes or attackers can estimate the usage pattern of the user by analyzing the number of tasks generated and the size of each task. As the mobile devices offload all tasks (including queued tasks and newly generated tasks) to fog node to enhance the computational latency when in good channel radio state, the personal information about the mobile device's usage pattern can be obtained by observation. Task offloading history and other related network statistics of the mobile user can be analyzed by the fog nodes to infer and identify the user in the network. This enables the fog node as well as eavesdropper to deduce the specific applications and usage times of a given user.

3.3.3. Data Privacy

Ensuring the confidentiality and privacy of the stored data at the fog nodes constitutes data privacy. With the advent of 5G networks, a huge volume of data from safety-critical applications such as banks and health care is generated and stored in the fog nodes as well as the clouds. Protecting this data during transmission, computation, and storage requires privacy and security policies.

3.4. Security

We have addressed the following critical security challenges in UbiPriSEQ design.

3.4.1. Jamming

Jammers utilize noisy signals to interrupt radio communication between the fog nodes and devices and prevents offloading of computations and data, affecting the network resources including bandwidth, computational and energy resources.

3.4.2. Rogue Fog Nodes and Users

Rogue nodes, devices, or users can masquerade as a genuine entity and deceive others into connecting to it, gaining access to confidential and personal information. They can perform a man in the middle attack, which results in data loss and even loss of device control.

4. UbiPriSEQ: The Proposed Framework

The UbiPriSEQ framework aims to adaptively, dynamically, and holistically optimize QoS, energy-efficiency, security, and privacy of 5G IoT HetNet-based environments. We describe here UbiPriSEQ, its architecture (Section 4.1), its brain (Section 4.2), and the two functional modules (Sections 4.3 and 4.4).

4.1. UbiPriSEQ: Architectural Overview

UbiPriSEQ is built on a three-layered model of ubiquitous environments comprising **IoT Devices Layer (dLayer)**, **FRAN Layer (fLayer)**, and **Core Layer (cLayer)**. Figure 1 provides its overview, and Figure 2 accentuates its two functional components, Device Module (DevM) and Network Module (NetM).

dLayer [17,25–28] comprises devices (smartphones and other smart "things" in IoT) that produce or consume data or take certain actions. **fLayer** consists of a macro base station and several micro base stations along with several fog nodes. FRAN (Fog Radio Access Networks) is a paradigm for 5G networks to provide high spectral and energy efficiency [23]. The idea behind FRAN is to take full advantage of local radio signal processing, cooperative radio resource management, and distributed storing capabilities in edge devices, which can decrease the heavy burden on fronthaul and avoid large-scale radio signal processing in the centralized Baseband Unit (BBU) pool [24]. The fog nodes near the micro base stations can locally store and process the data. Both the fog nodes and the micro base stations have caches; however, the data stored in them are not the same. Fog caches have more locally relevant data than the micro base station caches. **cLayer** [22] consists of the BBU pool connected through the wider Internet with clouds and HPC clusters, dealing with larger, delay-tolerant, applications that require large compute and storage resources. DevM executes in the dLayer on all devices and NetM executes in both fLayer (not in all fog nodes, rather centralized) and cLayer (see Figures 1 and 2). The two modules continuously learn from each other and adaptively optimize the ubiquitous environment.

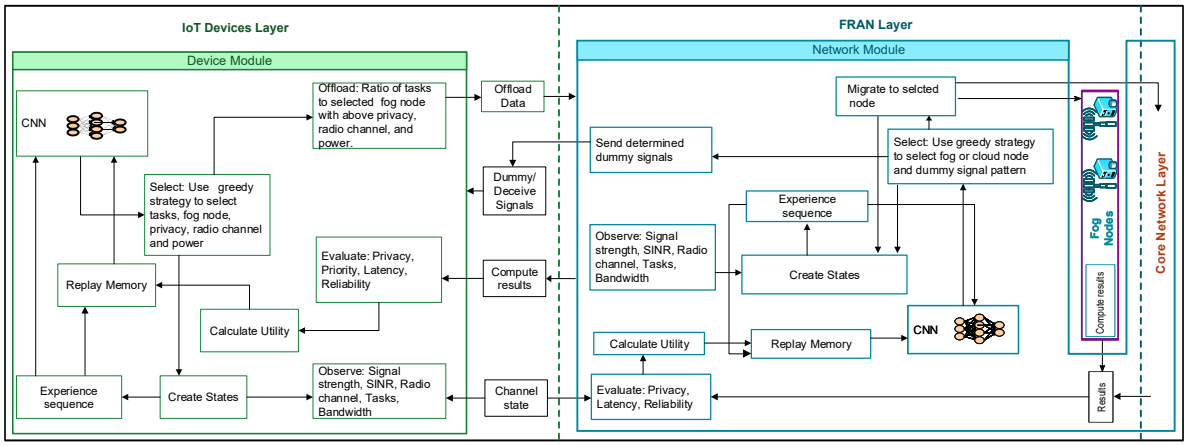

Figure 2. The functional architecture of Privacy Aware High Capacity Low-Latency Offloading (Device Module) and Defense against Attacks and Data migration (Network Module) components in UbiPriSEQ.

4.2. UbiPriSEQ: Adaptive Learning Architecture

Deep Reinforcement Learning (DRL) is UbiPriSEQ's brain. Reinforcement Learning (RL) enables agents to take optimal decisions through observations and feedback received from the system in terms of positive or negative rewards [57]. *Q*-Learning is an RL technique in which the agent tries to estimate the *Q* value-function iteratively, based on the reward obtained from an action, which indicates the optimality of the action in a given state of the system. Many techniques exist to find the *Q*-value.

We use here **Deep Q-Network (DQN)**, a DRL technique that uses a Deep Learning Network for estimating the Q-values. DQNs leverage Deep Neural Networks (DNNs) to train the learning process by the involved agents thereby improving the learning speed and the performance of Q-Learning algorithms. We discuss the DQN in detail, later in Section 4.3.5.

In UbiPriSEQ, a **learning agent** could be an IoT device, or edge or fog device, situated near the remote radio head at the micro base stations. An **action** corresponds to various privacy policies, security policies, defending mechanisms against attacks, and offloading policies. A **state** is the state of the agent, other devices, fog nodes, various attacks, and privacy characteristics observed by an agent. A search for the best Q-function terminates when an **optimal strategy** is found. The fog and other devices need to explore and exploit various actions to discover the best action providing an **optimal reward**. Sufficient interaction with various privacy leaking states and various attack states enable it to reach the **optimum action** for each state.

4.3. Device Module (DevM)

The main functions of Device Module (DevM) are to offload computations to the fog or cloud, or run locally, while managing privacy, priority, energy-efficiency, and QoS. The algorithm for Device Module is given in Algorithm 1. The explanation of Device Module and its algorithm follows in Section 4.3.1 to Section 4.3.5.

Algorithm 1 Device Module (DevM)

Input: *system state* $\boldsymbol{s_t}$
Output: *action* $\boldsymbol{a_t} = [a_t^l, a_t^t, a_t^b, f]$

Set $Q(s_t, a_t)$ with random weights θ_0
while ¬converged **do**
  observe $d_t, b_t,$ and l_t
  compute ζ_t
  estimate h_t
  $s_t \leftarrow \{d_t, b_t, l_t, h_t, \zeta_t, sinr_t\}$
  if $t \leq K$ **then**
    Select $\boldsymbol{a_t} \in \mathbf{A}$ at random
  else
    Form experience sequence
    $\Phi_t = ((\boldsymbol{s_{t-K}}, \boldsymbol{a_{t-K}}), \cdots, (\boldsymbol{s_{t-1}}, \boldsymbol{a_{t-1}}), \boldsymbol{s_t})$
    Input Φ_t to the DNN
    $[Q(\Phi_t, \boldsymbol{a_t})] \leftarrow$ get Q-value for $\mathbf{A}$ using DNN
    $\epsilon \leftarrow$ `random(0,1)`
    take action $\boldsymbol{a_t}$ with probability ϵ^t
    $a_t(a_t^l, a_t^t, a_t^b, f) \leftarrow \boldsymbol{a_t}$
  end if
  locally compute a_t^l; keep a_t^b in buffer; and migrate a_t^t tasks to fog node f
  $Pri_d(\boldsymbol{s_t}, \boldsymbol{a_t}) \leftarrow |g_t - t_t| + w_l$
  obtain the latency t_t
  $uf_n \leftarrow Pri_n(\boldsymbol{s_t}, \boldsymbol{a_t}) - \mu \cdot \zeta_t \cdot t_t - \rho + \eta \cdot sinr_t$
  $\mathcal{P} \leftarrow \mathcal{P} \cup (\Phi_t, \boldsymbol{a_t}, ud_d^{(t)}, \Phi_{t+1})$
  for $x \in$ DNN mini-batch **do**
    Select $\boldsymbol{\Phi_t} \in \mathcal{P}$ at random
    Calculate DNN loss as per Equation (6)
  end for
  Update DNN weights θ_t using gradient descent
  $t \leftarrow t + 1$
end while

4.3.1. DevM: Workflow

DevM observes the current channel state and task buffer, and takes a set of **actions** (i.e.,which tasks to process locally, and which ones to be offloaded, to which fog/cloud node, etc.) based on the current Q**-value** (obtained from the training phase) or at random with a small probability (see Section 4.3.5). A **reward** (also known as **utility**) is associated with each set of actions taken. The utility could be positive or negative, depending on whether DevM is able to improve the functional parameters, that is, privacy, QoS, and energy-efficiency. Subsequently, DevM computes the new Q-value using DQN, and based on it decides the set of actions. This process is repeated iteratively until an optimal Q-value is found. Figure 2 in the left block depicts DevM's process flow.

We now explain how DevM computes its functional parameters—privacy (Section 4.3.2), QoS and energy-efficiency (Section 4.3.3)—utility (Section 4.3.4), and Q-value (Section 4.3.5).

4.3.2. DevM: PrivacyMetric

The PrivacyMetric $Pri_d(s_t, a_t)$ for the state s_t and action a_t at time t is defined in Equation (1). The variable g_t is the number of tasks generated, o_t is the number of tasks to be transmitted, w_l (**PrivacyWeight**) is the weight that determines the relative importance of the location privacy compared to the usage-pattern privacy, and h_t indicates the channel state. The channel state, h_t, is modeled as a Markov process with two states $\{0, 1\}$. The variable w_l takes the value w, $\forall w \in \mathbb{R}^+$ (the required level of usage pattern privacy), when the channel state is "bad", otherwise it is zero for "good" state (see Reference [44]). Simply put, under 'good' channel state, PrivacyMetric is the absolute difference of all the computational tasks produced by a device and the tasks offloaded to fog or cloud.

$$\begin{aligned} Pri_d(s_t, a_t) &\triangleq |g_t - o_t| + w_l \\ w_l &= \begin{cases} 0, \text{if } h_t == 0 \\ w, \text{if } h_t == 1, \ \forall w \in \mathbb{R}^+ \\ . \end{cases} \end{aligned} \tag{1}$$

DevM maintains **usage pattern privacy** and breaks any patterns in PrivacyMetric value over time by adding PrivacyWeight and intentionally delaying offloading to avoid attackers detecting a usage pattern. DevM is able to hide usage pattern and hence attackers are unable to detect the device's location which could have been compromised due to the device's connection with a known or rogue node or base station (using distance or trilateration). DevMPrivacyMetric estimates the **usage pattern privacy** and **location privacy**.

4.3.3. DevM: QoS and Energy-Efficiency

DevM provides higher QoS by selecting the fog or cloud nodes that have SINR higher than a threshold. Moreover, it improves QoS further by utilizing latency constraints of tasks in setting task priorities, and offloading lower-priority tasks to fog or cloud only if needed. Energy-efficiency is improved by DevM by attempting to execute tasks locally if possible, otherwise on fog, and then on cloud.

4.3.4. DevM: Reward Function (Utility)

DevM in each time t observes various parameters from the device and network (newly generated tasks by the device, SINR, etc.), and stores them as the `state` in time t. This state is used to compute the utility or the rewards (ud_d), which is a weighted function given by Equation (2). The reward/utility function provides a higher reward when the task offloading meets the objectives (privacy, latency, QoS) and a lower reward when an offloading is far from our objectives. It is computed by adding PrivacyMetric (given by Equation (1)) and SINR ($sinr_t$), and subtracting tasks' computational costs (c_t),

tasks' communication and queuing delays (b_t), and the loss to the device due to the tasks' execution delays (ρ). All the parameters are weighted (μ, ω, η) based on their relative importance. DevM tries to achieve a higher utility, so parameters such as SINR are added while others such as costs are subtracted. DevM adds each utility for time t, and its associated action (offload policy) to an Experience Sequence (i.e., the accumulated knowledge).

$$ud_d \triangleq Pri_d(\boldsymbol{s_t}, \boldsymbol{a_t}) - \mu \cdot c_t - \rho - \omega \cdot b_t + \eta \cdot sinr_t. \tag{2}$$

4.3.5. DevM: Q-Value

DevM comprises a DQN that consists of an input layer, two hidden convolutional layers, a rectifier linear unit layer, and two fully connected output layer. The weights in the CNN are updated using gradient descent optimization [58] based on the previous attack experiences.

The state transits from the state $\boldsymbol{s_t}$ to $\boldsymbol{s_{t+1}}$ when the agents offload data to the selected fog node at state $\boldsymbol{s_t}$. Based on different offloading experiences denoted by $(\boldsymbol{s_t}, \boldsymbol{a_t}, ud_d^{(t)}, \boldsymbol{s_{t+1}})$, the Q-function update of the state-action pair, using general RL based technique can be computed as

$$Q(\boldsymbol{s_t}, \boldsymbol{a_t}) \leftarrow (1-\alpha)Q(\boldsymbol{s_t}, \boldsymbol{a_t}) + \alpha\big(ud_d^{(t)} + \gamma \max_{\overline{\boldsymbol{a_t}} \in \boldsymbol{A}} Q(\boldsymbol{s_{t+1}}, \overline{\boldsymbol{a_t}})\big), \tag{3}$$

where $\alpha \in (0, 1]$ is the learning rate denoting the weight of the current experience and $\gamma \in [0, 1]$ denotes the view of the agents regrading the future rewards. However, the DQN computes the Q-value for each time t by learning from the replay memory (Experience Sequence). The experience sequence (replay memory) Φ_t consists of the current system state $\boldsymbol{s_t}$ and the previous K state-action pairs, given by:

$$\Phi_t = ((\boldsymbol{s_{t-K}}, \boldsymbol{a_{t-K}}), \cdots, (\boldsymbol{s_{t-1}}, \boldsymbol{a_{t-1}}), \boldsymbol{s_t}). \tag{4}$$

The experience sequence Φ_t is fed as input to the first convolutional layer in DQN. The output of the first convolutional layer is then passed through a rectified linear unit (ReLU) as an activation function. The output of ReLU is passed through the second convolutional layer. The outputs from second convolutional layer is then passed to fully connected layers to obtain the Q-value for all the possible actions $|A|$ (offloading strategies) at current sequence Φ_t. Subsequently, in time t, DevM takes an action for offloading based on the Q-value with a probability $1 - \epsilon$, or takes a random action with a minute probability ϵ (to explore various rewards and reach global optimum reward).

Once the results are obtained from the offloaded computation the agents update their reward using Equation (2) and the offloading experience $(\Phi_t, \boldsymbol{a_t}, ud_d^{(t)}, \Phi_{t+1})$ is stored in the memory pool denoted by $\mathcal{P}$. As per the experience replay technique, the agents randomly select an experience from the pool to update the CNN weight θ_t. The mean squared error is used to calculate the loss between the networks output and the optimal Q-value. The Q-function for each state-action pair in a DQN is given below:

$$Q(\Phi_t, \boldsymbol{a_t}, \theta_t) = \mathbb{E}\left[ud_d^{(t)} + \gamma \max_{\overline{\boldsymbol{a_t}} \in \boldsymbol{A}} Q(\overline{\Phi_t}, \overline{\boldsymbol{a_t}}) | \Phi_t, \boldsymbol{a_t}\right], \tag{5}$$

where $\overline{\Phi_t}$ is the next state sequence based on the selection of the offloading policy $\boldsymbol{a_t}$ at state Φ_t and θ_t is the weight of DNN at time t. Furthermore, every T time steps, the Q-network is duplicated to obtain a target network to obtain the target Q-values so that updated DNN weights can be obtained. The target optimal Q-function $Q_{\text{opt}}(\Phi_t, \boldsymbol{a_t}, \theta_t)$ is given as follows:

$$Q_{opt}(\Phi_t, \boldsymbol{a_t}, \theta_t) = \left[ud_d^{(t)} + \gamma \max_{\overline{\boldsymbol{a_t}} \in \boldsymbol{A}} Q(\overline{\Phi_t}, \overline{\boldsymbol{a_t}}, \theta_{t-1}) | \Phi_t, \boldsymbol{a_t}\right] \tag{6}$$

and hence the loss is computed as follows [59]:

$$\mathcal{L}(\theta_t) = Q(\Phi_t, \boldsymbol{a_t}, \theta_t) - Q_{opt}(\Phi_t, \boldsymbol{a_t}, \theta_t). \tag{7}$$

This explains the loss calculation and weight updating in DQN as well as Q-value compuations using DQN.

4.4. Network Module (NetM)

The data sent to the fog nodes for task processing and other cached data in fog needs to be protected against privacy leakages and eavesdroppers. Network Module (NetM) devises policies for data migration between fog nodes to preserve privacy and broadcasts dummy signals to confuse eavesdroppers to address privacy and security concerns. The algorithm for Network Module is given in Algorithm 2 and is explained in the following.

Algorithm 2 Network Module (NetM)

Input: *system state* s_t
Output: *action* $\boldsymbol{a_t} = [a_t^{cl}, a_t^{cf}, a_t^{cc}, c, f, fr]$

while ¬converged **do**
  observe a_t^{cl}, a_t^{cf}, and a_t^{cc}
  compute ζ_t
  estimate h_t
  $s_t \leftarrow \{c_t, h_t, \zeta_t, sinr_t, \delta\}$
  if $t \leq K$ **then**
    Select $\boldsymbol{a_t} \in \mathbf{A}$ at random
  else
    Form experience sequence
    $\Phi_t = ((\boldsymbol{s_{t-K}}, \boldsymbol{a_{t-K}}), \cdots, (\boldsymbol{s_{t-1}}, \boldsymbol{a_{t-1}}), \boldsymbol{s_t})$
    Input Φ_t to the DNN
    $[Q(\boldsymbol{s_t}, \boldsymbol{a_t})] \leftarrow$ get Q-value for $\mathbf{A}$ using DNN
    $\epsilon \leftarrow$ `random(0,1)`
    take action $\boldsymbol{a_t}$with probability ϵ^t
    $a_t(a_t^{cl}, a_t^{cf}, a_t^{cc}, c, f) \leftarrow \boldsymbol{a_t}$
  end if
  locally store a_t^{cl}; transmit a_t^{cf} to fog node f
  migrate a_t^{cc} from fog to cloud node c
  send dummy signal with frequency f
  $Pri_n(\boldsymbol{s_t}, \boldsymbol{a_t}) \leftarrow \frac{1}{p(\mathcal{K}(D_i)\in M) - p(\mathcal{K}(D_j)\in M)}$
  obtain the latency t_t
  $uf_n \leftarrow Pri_n(\boldsymbol{s_t}, \boldsymbol{a_t}) - \mu \cdot \zeta_t \cdot t_t - \rho + \eta \cdot sinr_t$
  $\mathcal{P} \leftarrow \mathcal{P} \cup (\Phi_t, \boldsymbol{a_t}, ud_a^{(t)}, \boldsymbol{\Phi_{t+1}})$
  for $x \in$ DNN mini-batch **do**
    Select $\boldsymbol{\Phi_t} \in \mathcal{P}$ at random
    Calculate DNN loss using Equation (6)
  end for
  Update DNN weights θ_t using gradient descent
  $t \leftarrow t + 1$
end while

NetM workflow is the same as of DevM (see Section 4.3.1), i.e., it observes its environment and takes a set of actions (see Figure 2). The difference lies in the specific observations and set of actions it takes, the way it calculates the PrivacyMetric and Utility, and the parameters it tries to improve. The **observations** include cache content, channel state, task priority, and SINR. The **actions** include which data to keep, and which ones to be offloaded, to which fog/cloud node, and which basestations the dummy signal is broadcasted to. The **PrivacyMetric** used by NetM, $Pri_n(\boldsymbol{s_t}, \boldsymbol{a_t})$, defined in Equations (8), is based on differential privacy, i.e., the difference in hamming distance between two data blocks (from two arbitrary devices) present at a fog node should not exceed a certain threshold (i.e., required privacy) because a higher difference could allow attackers to identify types and content of data, and the device. NetM PrivacyMetric estimates the **data privacy** unlike the PrivacyMetric in DevM.

$\mathcal{K}$ is a privacy operation (Laplace mechanism) that operates on two data sets D_i and D_j received at a fog node from two arbitrary devices, i and j. M is the local memory of the fog node and ϵ_{diff} is the privacy parameter in differential privacy. $Pri_n(\boldsymbol{s_t}, \boldsymbol{a_t})$ can be considered as the lower bound differential probability parameter $\exp(\epsilon_{diff})$ [42].

$$Pri_n(\boldsymbol{s_t}, \boldsymbol{a_t}) = \frac{p(\mathcal{K}(D_i) \in M)}{p(\mathcal{K}(D_j) \in M)}. \tag{8}$$

The utility function (reward) ud_n is computed by Equation (9). $Pri_n(\boldsymbol{s_t}, \boldsymbol{a_t})$ represents the data privacy metric for NetM. c_t is the tasks' aggregated computational costs and $sinr_t$ is SINR at time t; μ and η are the weights giving their relative importance, respectively. ζ_t is the priority of the task and ρ is the loss to the device when the computations are not performed in the given time threshold.

$$uf_n \triangleq Pri_n(\boldsymbol{s_t}, \boldsymbol{a_t}) - \mu \cdot \zeta_t \cdot c_t - \rho + \eta \cdot sinr_t. \tag{9}$$

5. UbiPriSEQ: System Evaluation

We now evaluate the performance of UbiPriSEQ using SpMV computations (see Section 1). We have implemented the UbiPriSEQ framework in Python over the TensorFlow platform. The UbiPriSEQ system is evaluated using a real-life application of prime significance, Sparse Matrix-Vector product (SpMV). SpMV computations are fundamental to many scientific, engineering, business, and social applications that provide timely intelligence for the design, operations, and management of ubiquitous environments [60–67]. SpMV is also utilized in newer compressed Deep Neural Networks with pruned layers for computations [68]. The sparse matrices used for our experiments are from the University of Florida Sparse Matrix Collection consisting of a varied number of applications such as thermal, economics, optimization, and structural design (https://sparse.tamu.edu/). We have studied the performance of the UbiPriSEQ system for a range of SpMV computations and metrics including SINR, privacy metric, latency, and utility function and have compared it with constrained Markov Decision Process (CMDP) [44].

The experiments were run on a machine with 12 core Intel Xeon CPU with clock frequency of 3GHz, 32 GB RAM, and an Nvidia Quadro GPU. The edge nodes based on Jetson Nano has a Quad-core ARM A57 CPU with 1.43 GHz frequency, 4 GB RAM, and Ethernet connectivity. The storage of fog nodes is using microSD. The experiments were performed based on one-second time slots. The number of devices in IoT Devices Layer to the number of fog nodes is 1:3. The ϵ for the greedy selection in DRL is x^t, where x is a real number in the range $(0, 1)$, and t is the time [57]. The default network parameters used in the experiments have been depicted in Table 1. The jamming attacks are modeled as interfering signals/noise in the experiment. Some devices and fog nodes are modeled as "rogue" and UbiPriSEQ detects and defend itself from these.

Table 1. UbiPriSec: Experimental Network Parameters.

Parameter	Value
System bandwidth	10 MHz
Transmit power of user nodes	10 dBm
Number of fog nodes	30
Number of user nodes	90
Processing Density	4 double-precision FLOPS/cycle [69]
Fog compute power	10 GHz
Sparse Matrix Size	[10, 40,000] KB
Learning Rate	0.8
Discount Factor	0.7
Privacy weight	5
Task arrival rate	[0, 5]
Selection probability: ϵ	$x^t, \forall x \in (0, 1), t \in \mathbb{R}$

Figure 3 compares performance of UbiPriSEQ (both DevM and NetM combined) with CMDP in terms of PrivacyMetric, SINR, latency, and utility against clock-time. The utility is the average of all DevM and NetMutilities (Sections 4.3.4 and 4.4). The PrivacyMetric is also an average (Sections 4.3.2 and 4.4). The latency is the average latency including the tasks' aggregated computational costs, and the queuing and transmission delays. Figure 3a compares provided privacy levels and shows that UbiPriSEQ outperforms CMDP by 25% on average (computed by the average difference between the two privacy levels). UbiPriSEQ achieves a higher-level of privacy at a faster rate due to the data migration policies not seen in Reference [44]. We achieve a higher QoS due to better defenses against the attacks. Figure 3b plots SINR and shows an average improvement in the network QoS by 22% in the presence of jamming signals and eavesdroppers. Figure 3c shows that UbiPriSEQ minimizes the average latency at a faster rate then CMDP with the maximum difference against CMDP of 74% 2.19×10^5 seconds (lower difference in latency is seen later after the systems stabilize). Figure 3d shows better average utility for UbiPriSEQ compared to CMDP, 33% on average.

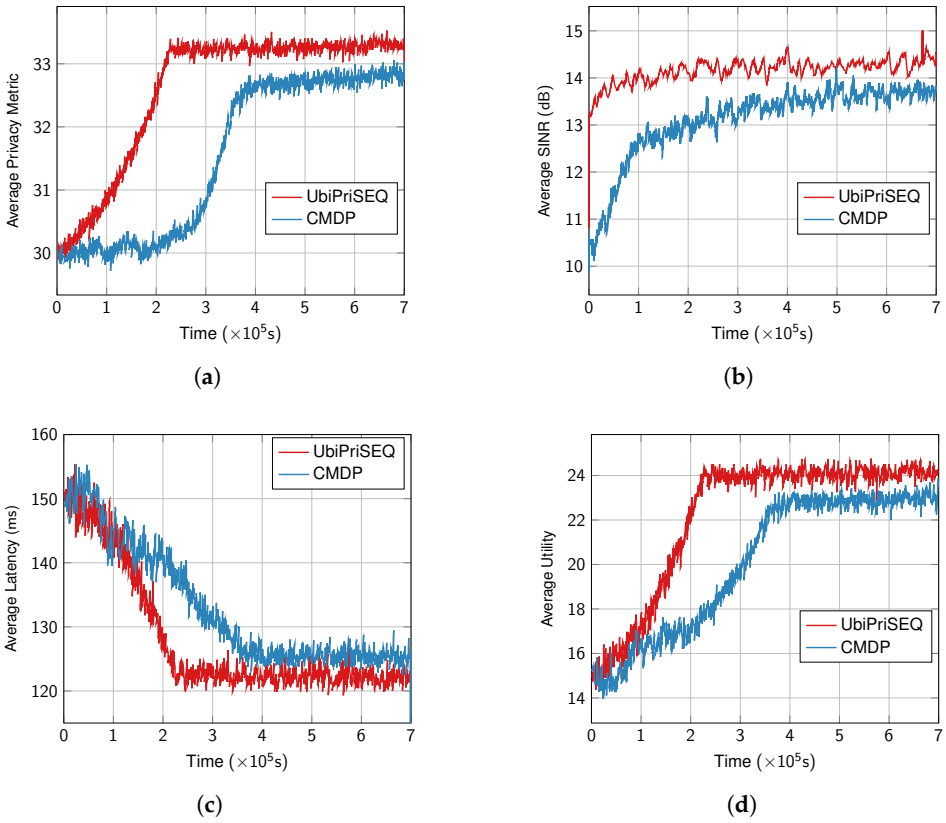

Figure 3. UbiPriSEQ: Performance vs. Time (**a**) PrivacyMetric, (**b**) SINR, (**c**) Latency, (**d**) Utility.

Figure 4 depicts the performance of UbiPriSEQ and CMDP for the same four metrics as in Figure 3 against the task and data sizes (SpMV computations for matrices of sizes ranging between 10 KB and 40 MB). UbiPriSEQ outperforms CMDP, on average, for PrivacyMetric by 1.12×, SINR by 1.25×, latency by 1.20×, and utility by 1.18×.

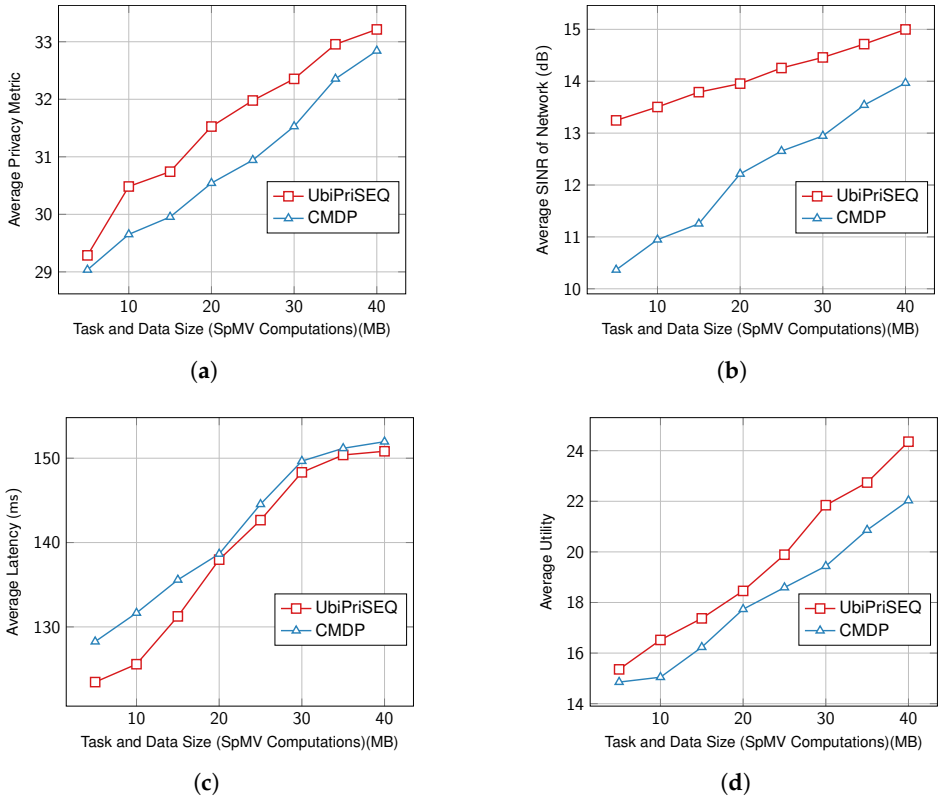

Figure 4. UbiPriSEQ: Performance vs. Task and Data Sizes (**a**) PrivacyMetric, (**b**) SINR, (**c**) Latency, (**d**) Utility.

These results illustrate that UbiPriSEQ was able to defend and recover from security issues in the network. UbiPriSEQ is successful in selecting the best offload and migration policies that ensure improved privacy, security, and QoS including latency, energy-efficiency, and reliability. The dynamic learning process in both the modules enables both the fog devices as well as the edge devices to optimize the offloading strategy while preserving user privacy and security. This multi-agent learning enables the devices to learn and decide the offloading decisions based on the strategies taken by the network module to provide security. Similarly, at the same time, the network module learns from the effects of the strategies taken by the device module to provide a better security (jamming, noise, etc.) strategy while maintaining the required privacy. This process of learning by modules from each other provides theUbiPriSEQ framework a better performance compared to the CMDP-based approach [44]. Moreover, as we have mentioned earlier, the CMDP-based approach [44] has only focussed on privacy, while we propose a holistic framework to optimise multiple design goals, and this differentiates our work from all the state-of-the-art relevant approaches.

6. Conclusions and Future Work

5G networks with their significantly higher speeds, lower-latency, reliability, capacity, and other benefits offer a powerful communication platform for ubiquitous environments. The complexity of today's ubiquitous environments have become increasingly complex due to the emergence of collaborative edge, fog, and cloud paradigms. However, privacy and security of users and data will pose paramount challenges in the adoption of these ubiquitous environments by governments, industry, and common people. A particular challenge in this context would be to preserve privacy and security while delivering quality of service (QoS) and energy-efficiency. Several works have tried to address one or more of these challenges (QoS, energy efficiency, security, and privacy) in distributed environments.

Despite these efforts, a number of challenges exist. Firstly, the focus of the existing approaches is on improving or optimizing one or two of the design parameters; QoS, energy efficiency, security or privacy. Optimizing one or more parameters without measuring and counteracting their impact on other parameters is no more acceptable in the emerging ubiquitous environments. Secondly, existing approaches assume a fixed network configuration and attack models, and use fixed strategies in devising solutions. The ultimate aim is to optimally migrate and place data, analytics, and other computations in these environments such that security, privacy, energy efficiency and QoS demands of these applications and the environments are best satisfied [8,70–73].

In this article, we proposed UbiPriSEQ that uses DRL to adaptively, dynamically, and holistically optimize QoS, energy-efficiency, security, and privacy. The Framework is built on a three-layered model and comprises two modules. UbiPriSEQ devises policies and makes decisions related to a number of important parameters including local processing and offloading rates for data and computations, radio channel states, transmit power, task priority, selection of fog nodes for offloading, and data migration. We evaluate UbiPriSEQ and compare it with the CMDP-based approach [44]. The results of our modular approach show clear performance gain over the CMDP-based approach as well as addresses other challenges that the CMDP-based approach does not address (e.g., security).

Our UbiPriSEQ framework is novel for several reasons. In contrast to the existing state-of-the-art, we attempt to holistically optimize QoS, energy efficiency, security, and privacy. Also, UbiPriSEQ is designed using a modular approach. It adaptively and dynamically deals with the environment by continuous learning, and does not need a priory complete specification of the environment, security attack models, and other design parameters. Moreover, DRL has not been used before in 5G networks to provide adaptive intelligence. Finally, most existing works use simulated data for performance evaluation while we have used SpMV, a real-life application to study UbiPriSEQ performance, and this has not been reported before. The SpMV operation is central to deep learning algorithms and this per se would create a significant impact.

We would like to note here that while the experimental results reported in this paper show that UbiPriSEQ outperforms CMDP [44], the excellence of one scheme over the other cannot be proved by a set of simulated experiments. Although we have given some valid justifications for UbiPriSEQ outperforming CMDP (see the last paragraph in Section 5), this is not a mathematical and absolute proof that UbiPriSEQ is better than CMDP. Many deeper investigations and actual deployments of the proposed work are needed to understand and improve the proposed approach. Nevertheless, we believe that UbiPriSEQ does provide overall a better approach for network design and operations due to its holistic and modular approach.

Going forward, we aim to explore integrating blockchains with UbiPriSEQ, which would enable a highly secure and private network for data and task offloading. We also aim to improve privacy metrics to improve better integration of various types of privacy. While UbiPriSEQ proposed in this paper provides a first step towards holistic optimization of 5G-based ubiquitous environments, this is just the beginning of an exciting quest; much more needs to be done in perfecting the various dimensions of UbiPriSEQ system.

Author Contributions: Conceptualization, T.M. and R.M.; methodology, T.M. and R.M.; software, T.M.; validation, T.M. and R.M.; formal analysis, T.M., R.M., I.K. and A.A.; investigation, T.M. and R.M.; resources, R.M., I.K. and A.A.; data curation, T.M.; writing—original draft preparation, T.M.; writing—review and editing, R.M.; visualization, T.M. and R.M.; supervision, R.M.; project administration, R.M., A.A. and I.K.; funding acquisition, R.M., A.A. and I.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Deanship of Scientific Research (DSR) at the King Abdulaziz University, Jeddah, under grant number RG-10-611-38.

Acknowledgments: The authors acknowledge with thanks the technical and financial support from the Deanship of Scientific Research (DSR) at the King Abdulaziz University (KAU), Jeddah, Saudi Arabia, under Grant No. RG-10-611-38. The experiments reported in this paper were performed on the Aziz supercomputer at KAU.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Yigitcanlar, T.; Butler, L.; Windle, E.; Desouza, K.C.; Mehmood, R.; Corchado, J.M. Can Building "Artificially Intelligent Cities" Safeguard Humanity from Natural Disasters, Pandemics, and Other Catastrophes? An Urban Scholar's Perspective. *Sensors* **2020**, *20*, 2988. [CrossRef] [PubMed]
2. Mehmood, R.; See, S.; Katib, I.; Chlamtac, I. (Eds.) *Smart Infrastructure and Applications: Foundations for Smarter Cities and Societies*; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; p. 665. [CrossRef]
3. Suma, S.; Mehmood, R.; Albeshri, A. Automatic Event Detection in Smart Cities Using Big Data Analytics. In *Smart Societies, Infrastructure, Technologies and Applications*; Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 111–122.
4. Alomari, E.; Katib, I.; Mehmood, R. Iktishaf: A Big Data Road-Traffic Event Detection Tool Using Twitter and Spark Machine Learning. *Mob. Networks Appl.* **2020**, *5*, 9533–9554. [CrossRef]
5. Aqib, M.; Mehmood, R.; Alzahrani, A.; Katib, I.; Albeshri, A.; Altowaijri, S.M. Rapid Transit Systems: Smarter Urban Planning Using Big Data, In-Memory Computing, Deep Learning, and GPUs. *Sustainability* **2019**, *11*, 2736. [CrossRef]
6. Alomari, E.; Mehmood, R. Analysis of Tweets in Arabic Language for Detection of Road Traffic Conditions. In *Smart Societies, Infrastructure, Technologies and Applications*; Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 98–110.
7. Alotaibi, S.; Mehmood, R.; Katib, I.; Rana, O.; Albeshri, A. Sehaa: A Big Data Analytics Tool for Healthcare Symptoms and Diseases Detection Using Twitter, Apache Spark, and Machine Learning. *Appl. Sci.* **2020**, *10*, 1398. [CrossRef]
8. Bosaeed, S.; Katib, I.; Mehmood, R. A Fog-Augmented Machine Learning based SMS Spam Detection and Classification System. In Proceedings of the 2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC), Paris, France, 20–23 April 2020; pp. 325–330.
9. Sun, Y.; Song, H.; Jara, A.J.; Bie, R. Internet of Things and Big Data Analytics for Smart and Connected Communities. *IEEE Access* **2016**, *4*, 766–773. [CrossRef]
10. Atzori, L.; Iera, A.; Morabito, G. The Internet of Things: A survey. *Comput. Netw.* **2010**, *54*, 2787–2805. [CrossRef]
11. Sun, Y.; Zhang, J.; Bie, R.; Yu, J. Advancing researches on IoT systems and intelligent applications. *Pers. Ubiquitous Comput.* **2018**, *22*, 449–452. [CrossRef]
12. Alam, F.; Mehmood, R.; Katib, I.; Albogami, N.N.; Albeshri, A. Data Fusion and IoT for Smart Ubiquitous Environments: A Survey. *IEEE Access* **2017**, *5*, 9533–9554. [CrossRef]
13. Evans, D. How the Next Evolution of the Internet Is Changing Everything. 2011. Available online: http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf (accessed on 12 October 2020).
14. Usman, S.; Mehmood, R.; Katib, I. Big Data and HPC Convergence for Smart Infrastructures: A Review and Proposed Architecture. In *Smart Infrastructure and Applications: Foundations for Smarter Cities and Societies*; Mehmood, R., See, S., Katib, I., Chlamtac, I., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 561–586. [CrossRef]
15. Marsch, P.; Silva, I.D.; Bulakci, O.; Tesanovic, M.; Ayoubi, S.E.E.; Rosowski, T.; Kaloxylos, A.; Boldi, M. 5G Radio Access Network Architecture: Design Guidelines and Key Considerations. *IEEE Commun. Mag.* **2016**, *54*, 24–32. [CrossRef]
16. Wan, S.; Li, X.; Xue, Y.; Lin, W.; Xu, X. Efficient computation offloading for Internet of Vehicles in edge computing-assisted 5G networks. *J. Supercomput.* **2019**. [CrossRef]
17. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2322–2358. [CrossRef]
18. Mukherjee, M.; Matam, R.; Shu, L.; Maglaras, L.; Ferrag, M.A.; Choudhury, N.; Kumar, V. Security and Privacy in Fog Computing: Challenges. *IEEE Access* **2017**, *5*, 19293–19304. [CrossRef]

19. Dong, L.; Satpute, M.N.; Shan, J.; Liu, B.; Yu, Y.; Yan, T. Computation Offloading for Mobile-Edge Computing with Multi-user. In Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, 7–10 July 2019; pp. 841–850. [CrossRef]
20. Jiao, L.; Yin, H.; Huang, H.; Guo, D.; Lyu, Y. Computation Offloading for Multi-user Mobile Edge Computing. In Proceedings of the 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Exeter, UK, 28–30 June 2018; pp. 422–429. [CrossRef]
21. Jiang, C.; Cheng, X.; Gao, H.; Zhou, X.; Wan, J. Toward Computation Offloading in Edge Computing: A Survey. *IEEE Access* **2019**, *7*, 131543–131558. [CrossRef]
22. Taleb, T.; Samdanis, K.; Mada, B.; Flinck, H.; Dutta, S.; Sabella, D. On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1657–1681. [CrossRef]
23. Zhang, H.; Qiu, Y.; Chu, X.; Long, K.; Leung, V.C.M. Fog Radio Access Networks: Mobility Management, Interference Mitigation, and Resource Optimization. *IEEE Wirel. Commun.* **2017**, *24*, 120–127. [CrossRef]
24. Habibi, M.A.; Nasimi, M.; Han, B.; Schotten, H.D. A Comprehensive Survey of RAN Architectures toward 5G Mobile Communication System. *IEEE Access* **2019**, *7*, 70371–70421. [CrossRef]
25. Parwez, M.S.; Rawat, D.B. Resource Allocation in Adaptive Virtualized Wireless Networks with Mobile Edge Computing. In Proceedings of the 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018. [CrossRef]
26. Rost, P.; Banchs, A.; Berberana, I.; Breitbach, M.; Doll, M.; Droste, H.; Mannweiler, C.; Puente, M.A.; Samdanis, K.; Sayadi, B. Mobile network architecture evolution toward 5G. *IEEE Commun. Mag.* **2016**, *54*, 84–91. [CrossRef]
27. Kekki, S.; Featherstone, W.; Fang, Y.; Kuure, P.; Li, A.; Ranjan, A.; Purkayastha, D.; Feng, J.; Frydman, D.; Verin, G.; et al. MEC in 5G networks. *ETSI White Pap.* **2018**, *28*, 1–28.
28. ETSI GS MEC 003 V1.1.1. Mobile Edge Computing (MEC): Framework and Reference Architecture. 2016. Available online: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/01.01.01_60/gs_MEC003v010101p.pdf (accessed on 12 October 2020).
29. Zhang, K.; Mao, Y.; Leng, S.; Zhao, Q.; Li, L.; Peng, X.; Pan, L.; Maharjan, S.; Zhang, Y. Energy-Efficient Offloading for Mobile Edge Computing in 5G Heterogeneous Networks. *IEEE Access* **2016**, *4*, 5896–5907. [CrossRef]
30. Liu, J.; Mao, Y.; Zhang, J.; Letaief, K.B. Delay-optimal computation task scheduling for mobile-edge computing systems. In Proceedings of the 2016 IEEE International Symposium on Information Theory (ISIT), Barcelona, Spain, 10–15 July 2016; pp. 1451–1455. [CrossRef]
31. Sun, H.; Zhou, F.; Hu, R.Q. Joint Offloading and Computation Energy Efficiency Maximization in a Mobile Edge Computing System. *IEEE Trans. Veh. Technol.* **2019**, *68*, 3052–3056. [CrossRef]
32. Ren, J.; Yu, G.; Cai, Y.; He, Y.; Qu, F. Partial Offloading for Latency Minimization in Mobile-Edge Computing. In Proceedings of the GLOBECOM 2017—2017 IEEE Global Communications Conference, Singapore, 4–8 December 2017; pp. 1–6. [CrossRef]
33. Kuang, Z.; Li, L.; Gao, J.; Zhao, L.; Liu, A. Partial Offloading Scheduling and Power Allocation for Mobile Edge Computing Systems. *IEEE Internet Things J.* **2019**, *6*, 6774–6785. [CrossRef]
34. Ning, Z.; Dong, P.; Kong, X.; Xia, F. A Cooperative Partial Computation Offloading Scheme for Mobile Edge Computing Enabled Internet of Things. *IEEE Internet Things J.* **2019**, *6*, 4804–4814. [CrossRef]
35. Mahmood, A.; Ahmed, A.; Naeem, M.; Hong, Y. Partial Offloading in Energy Harvested Mobile Edge Computing: A Direct Search Approach. *IEEE Access* **2020**, *8*, 36757–36763. [CrossRef]
36. Xu, J.; Chen, L.; Ren, S. Online Learning for Offloading and Autoscaling in Energy Harvesting Mobile Edge Computing. *IEEE Trans. Cogn. Commun. Netw.* **2017**, *3*, 361–373. [CrossRef]
37. Mach, P.; Becvar, Z. Mobile Edge Computing: A Survey on Architecture and Computation Offloading. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1628–1656. [CrossRef]
38. Ward, J.R.; Younis, M. Cross-layer traffic analysis countermeasures against adaptive attackers of wireless sensor networks. *Wirel. Netw.* **2019**, *25*, 2869–2887. [CrossRef]
39. Xiao, X.; Chen, C.; Sangaiah, A.K.; Hu, G.; Ye, R.; Jiang, Y. CenLocShare: A centralized privacy-preserving location-sharing system for mobile online social networks. *Future Gener. Comput. Syst.* **2018**, *86*, 863–872. [CrossRef]

40. Zhang, S.; Choo, K.K.R.; Liu, Q.; Wang, G. Enhancing privacy through uniform grid and caching in location-based services. *Future Gener. Comput. Syst.* **2018**, *86*, 881–892. [CrossRef]
41. Ni, J.; Lin, X.; Shen, X.S. Toward Edge-Assisted Internet of Things: From Security and Efficiency Perspectives. *IEEE Netw.* **2019**, *33*, 50–57. [CrossRef]
42. Wagner, I.; Eckhoff, D. Technical Privacy Metrics: A Systematic Survey. *ACM Comput. Surv.* **2018**, *51*, 57:1–57:38. [CrossRef]
43. Sun, Y.; Zhang, J.; Xiong, Y.; Zhu, G. Data Security and Privacy in Cloud Computing. *Int. J. Distrib. Sens. Netw.* **2014**, *10*, 190903. [CrossRef]
44. He, X.; Liu, J.; Jin, R.; Dai, H. Privacy-Aware Offloading in Mobile-Edge Computing. In Proceedings of the GLOBECOM 2017—2017 IEEE Global Communications Conference, Singapore, 4–8 December 2017; pp. 1–6. [CrossRef]
45. He, X.; Jin, R.; Dai, H. Deep PDS-Learning for Privacy-Aware Offloading in MEC-Enabled IoT. *IEEE Internet Things J.* **2019**, *6*, 4547–4555. [CrossRef]
46. He, X.; Jin, R.; Dai, H. Peace: Privacy-Preserving and Cost-Efficient Task Offloading for Mobile-Edge Computing. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 1814–1824. [CrossRef]
47. Zheng, G.; Krikidis, I.; Li, J.; Petropulu, A.P.; Ottersten, B. Improving Physical Layer Secrecy Using Full-Duplex Jamming Receivers. *IEEE Trans. Signal Process.* **2013**, *61*, 4962–4974. [CrossRef]
48. Mukherjee, A.; Fakoorian, S.A.A.; Huang, J.; Swindlehurst, A.L. Principles of Physical Layer Security in Multiuser Wireless Networks: A Survey. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1550–1573. [CrossRef]
49. He, X.; Jin, R.; Dai, H. Physical-Layer Assisted Secure Offloading in Mobile-Edge Computing. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 4054–4066. [CrossRef]
50. Zhao, W.; Wang, B.; Bao, H.; Li, B. Secure Energy-Saving Resource Allocation on Massive MIMO-MEC System. *IEEE Access* **2020**, *8*, 137244–137253. [CrossRef]
51. Bai, T.; Wang, J.; Ren, Y.; Hanzo, L. Energy-Efficient Computation Offloading for Secure UAV-Edge-Computing Systems. *IEEE Trans. Veh. Technol.* **2019**, *68*, 6074–6087. [CrossRef]
52. Feng, J.; Yu, F.R.; Pei, Q.; Chu, X.; Du, J.; Zhu, L. Cooperative Computation Offloading and Resource Allocation for Blockchain-Enabled Mobile-Edge Computing: A Deep Reinforcement Learning Approach. *IEEE Internet Things J.* **2020**, *7*, 6214–6228. [CrossRef]
53. Sun, Y.; Cao, J.; Ma, M.; Li, H.; Niu, B.; Li, F. Privacy-Preserving Device Discovery and Authentication Scheme for D2D Communication in 3GPP 5G HetNet. In Proceedings of the 2019 International Conference on Computing, Networking and Communications (ICNC), Honolulu, HI, USA, 18–21 February 2019; pp. 425–431. [CrossRef]
54. Bangerter, B.; Talwar, S.; Arefi, R.; Stewart, K. Networks and devices for the 5G era. *IEEE Commun. Mag.* **2014**, *52*, 90–96. [CrossRef]
55. Carvalho, G.H.S.; Woungang, I.; Anpalagan, A.; Jaseemuddin, M.; Hossain, E. Intercloud and HetNet for Mobile Cloud Computing in 5G Systems: Design Issues, Challenges, and Optimization. *IEEE Netw.* **2017**, *31*, 80–89. [CrossRef]
56. Bogale, T.E.; Le, L.B. Massive MIMO and mmWave for 5G Wireless HetNet: Potential Benefits and Challenges. *IEEE Veh. Technol. Mag.* **2016**, *11*, 64–75. [CrossRef]
57. Servin, A.; Kudenko, D. Multi-Agent Reinforcement Learning for Intrusion Detection: A Case Study and Evaluation. In *Multiagent System Technologies*; Bergmann, R., Lindemann, G., Kirn, S., Pěchouček, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 159–170.
58. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
59. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
60. Usman, S.; Mehmood, R.; Katib, I.; Albeshri, A. ZAKI+: A Machine Learning Based Process Mapping Tool for SpMV Computations on Distributed Memory Architectures. *IEEE Access* **2019**, *7*, 81279–81296. [CrossRef]
61. Muhammed, T.; Mehmood, R.; Albeshri, A.; Katib, I. SURAA: A Novel Method and Tool for Loadbalanced and Coalesced SpMV Computations on GPUs. *Appl. Sci.* **2019**, *9*, 947. [CrossRef]
62. Usman, S.; Mehmood, R.; Katib, I.; Albeshri, A.; Altowaijri, S. ZAKI: A Smart Method and Tool for Automatic Performance Optimization of Parallel SpMV Computations on Distributed Memory Machines. *Mob. Netw. Appl.* **2019**, 1–18. [CrossRef]

63. Alyahya, H.; Mehmood, R.; Katib, I. Parallel Iterative Solution of Large Sparse Linear Equation Systems on the Intel MIC Architecture. In *Smart Infrastructure and Applications: Foundations for Smarter Cities and Societies*; Mehmood, R., See, S., Katib, I., Chlamtac, I., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 377–407. [CrossRef]
64. AlAhmadi, S.; Muhammed, T.; Mehmood, R.; Albeshri, A. Performance Characteristics for Sparse Matrix-Vector Multiplication on GPUs. In *Smart Infrastructure and Applications: Foundations for Smarter Cities and Societies*; Mehmood, R., See, S., Katib, I., Chlamtac, I., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 409–426. [CrossRef]
65. Mehmood, R.; Crowcroft, J. *Parallel Iterative Solution Method for Large Sparse Linear Equation Systems*; Technical Report UCAM-CL-TR-650; University of Cambridge, Computer Laboratory: Cambridge, UK, 2005.
66. Mehmood, R.; Crowcroft, J.; Elmirghani, J.M.H. A Parallel Implicit Method for the Steady-State Solution of CTMCs. In Proceedings of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation, Monterey, CA, USA, 11–14 September 2006; pp. 293–302.
67. Mehmood, R. *A Survey of Out-of-Core Analysis Techniques in Stochastic Modelling*; Technical Report CSR-03-7; School of Computer Science, University of Birmingham: Birmingham, UK, 2003.
68. Davis, T.A.; Hu, Y. The University of Florida Sparse Matrix Collection. *ACM Trans. Math. Softw.* **2011**, *38*, 1–25. [CrossRef]
69. Dolbeau, R. Theoretical peak FLOPS per instruction set: A tutorial. *J. Supercomput.* **2018**, *74*, 1341–1377. [CrossRef]
70. Arfat, Y.; Aqib, M.; Mehmood, R.; Albeshri, A.; Katib, I.; Albogami, N.; Alzahrani, A. Enabling Smarter Societies through Mobile Big Data Fogs and Clouds. *Procedia Comput. Sci.* **2017**, *109*, 1128–1133. [CrossRef]
71. Muhammed, T.; Mehmood, R.; Albeshri, A.; Katib, I. UbeHealth: A Personalized Ubiquitous Cloud and Edge-Enabled Networked Healthcare System for Smart Cities. *IEEE Access* **2018**, *6*, 32258–32285. [CrossRef]
72. Tawalbeh, L.A.; Mehmood, R.; Benkhlifa, E.; Song, H. Mobile Cloud Computing Model and Big Data Analysis for Healthcare Applications. *IEEE Access* **2016**, *4*, 6171–6180. [CrossRef]
73. Tawalbeh, L.A.; Bakhader, W.; Mehmood, R.; Song, H. Cloudlet-Based Mobile Cloud Computing for Healthcare Applications. In Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016; pp. 1–6.