Information Systems Frontiers
https://doi.org/10.1007/s10796-022-10366-2

HRL-Edge-Cloud: Multi-Resource Allocation in Edge-Cloud based Smart-StreetScape System using Heuristic Reinforcement Learning

Arslan Qadeer[1] · **Myung J. Lee**[1]

Accepted: 16 December 2022

Abstract
The Edge Cloud (EC) architecture aims at providing the compute power at the edge of the network to minimize the latency necessary for the Internet of Things (IoT). However, an EC endures a limited compute capacity in contrast with the back-end cloud (BC). Intelligent resource management techniques become imperative in such resource constrained environment. In this study, to achieve the efficient resource allocation objective, we propose HRL-Edge-Cloud, a novel heuristic reinforcement learning-based multi-resource allocation (MRA) framework which significantly overcomes the bottlenecks of wireless bandwidth and compute capacity jointly at the EC and BC. We solve the MRA problem by accelerating the conventional Q-Learning algorithm with a heuristic method and applying a novel linear-annealing technique. Additionally, our proposed pruning principle achieves remarkably high resource utilization efficiency while maintaining a low rejection rate. The effectiveness of our proposed method is validated by running extensive simulations in three different scales of environments. When compared with the baseline algorithm, the proposed HRL-Edge-Cloud achieves 240X, 95X and 2.4X reduction in runtime, convergence time and rejection rate, respectively, and achieves 2.34X operational cost efficiency improvement on average while satisfying the latency requirement.

Keywords Edge cloud · Heuristic reinforcement learning · Task offloading · Resource allocation · Admission control · Smart city · IoT

1 Introduction

Over the years, the concept of smart city has gained attention of the research communities to utilize the technological infrastructure for societies to meet their rapidly growing needs for a high quality of life (Robberechts et al., 2020). Smart cities are expected to transform the way of living by advancing the healthcare (Safitri et al., 2020); the city management services (e.g., smart traffic, electricity consumption and waste management) (Wang et al., 2018; Elhassan et al., 2019); and many other industries (Agbali et al., 2019; Baena et al., 2020). The prevalent 5G wireless communications and edge computing technologies aim to accelerate the development of smart cities to brace the hyper-local awareness and intelligence, allowing real-time, and next-generation mobile & commercial applications to ameliorate the streetscapes (Robberechts et al., 2020; Wang et al., 2018; Yang et al., 2021; Chen et al., 2020). This proposition calls for the co-development of technologies such as low-latency & high-speed wireless communications and edge computing. This underpins the focus of this study on the advancement of streetscape wireless and computational infrastructure which is intended to aid the operations of city assets "(streets, buildings, emergency vehicles, traffic signals, billboards/displays, surveillance cameras, public transport, IoT devices, etc.)".

Next generation mobile applications (e.g. Augmented Reality (AR), Virtual Reality (VR)) and streetscape applications (e.g. swift control-response for emergency vehicles and situation-aware traffic/pedestrian signaling) possess resource-hungry and real-time constraints (Liu et al., 2020). Edge-cloud (EC) architecture is a stepping stone to meet the above compute and real-time constraints by reducing the network latency and providing the

✉ Arslan Qadeer
aqadeer000@citymail.cuny.edu

✉ Myung J. Lee
mlee@ccny.cuny.edu

[1] Department of Electrical Engineering, The City College of New York of CUNY, 160 Convent Ave, New York, 10031, NY, USA

Published online: 16 January 2023

computational resources at the edge of the network (Ning et al., 2019). Furthermore, A three-tier hierarchical EC system integrated with the back-end cloud (BC) provides support for a broad-range of applications with varying QoS requirements in greater extent (Ungureanu et al., 2021).

Edge clouds possess a limited amount of computational resources (Ungureanu et al., 2021) and back-end clouds experience the same in the case of pay-as-you go model (Gong et al., 2017). 5G supports dynamic Radio Access Network (RAN) and a wider frequency spectrum landscape (Habibi et al., 2019). However, in the presence of excessive amount of connected devices in the Edge-cloud environment, a large amount of concurrent traffic can be anticipated. Thus, communication resources, which connect the devices with EC and BC, also become a bottleneck for the system. This multi-resource allocation and system cost reduction challenge is manifold: First, handling the user requests from a wide range of applications at large scale with different QoS requirements. Second, the computational complexity pertaining to optimal resource allocation of the system particularly in dynamic traffic patterns requires innovative solution techniques.

Long-established approaches, such as optimization-based methods and game theory-based techniques (Bi et al., 2019; Tran & Pompili, 2019; Xue & An, 2021; JošCsilo & Dán, 2018; Zheng et al., 2019; Dong & Wen, 2019) are utilized to solve resource management problems. Among Machine Learning (ML) based techniques, Deep Learning (DL) based methods (Cheng et al., 2018; Wei et al., 2019; Peng & Shen, 2020; Nath & Wu, 2020; Chen et al., 2021) have also gained significant popularity to solve the control and management problems in IoT ecosystem. However, such approaches are not suitable to solve multi-resource allocation problem in non-stationary and large scale environments due to heavy computational loads (More details can be found in Section 2).

To solve the multi-resource allocation problem in a comprehensive way, we propose HRL-Edge-Cloud framework, a heuristic reinforcement learning based highly scalable and adaptable resource allocation system. In this paper, we jointly consider the edge-cloud (EC) and back-end cloud (BC) based environment to handle a huge number of user-requests, minimize system cost and rejection rate and improve the quality of experience (QoE) of the users.

The main contribution of our work is summarized as follows:

- We present a simple user job model which takes into consideration both the deadline of the job and data to be processed at the same time. Thereafter, to aptly process these jobs, we present a multi-resource allocation model under an integrated wireless communication, EC and BC environment to handle a large-scale of user requests under constrained resources in real-time.
- We formulate the multi-resource allocation problem for user requests into semi-Markov decision process (SMDP). The reward maximization objective for resource allocation with wireless bandwidth, EC and BC compute resources considers to optimize three fundamental bench-marking points; 1) Minimize system cost; 2) Minimize round-trip time of a user request for better Quality of Experience (QoE); and 3) Minimize rejection rate for enhanced reliability.
- A modified heuristic approach with linear-annealing and a pruning principle (Qadeer et al., 2021) is applied to standard reinforcement learning algorithm to quickly determine the best action policy to procure minimum rejection rate, reduce system cost and improve QoE of the users.
- Compared to the baseline method, which is a heuristic method with fixed exploration rate, our proposed heuristic approach with linear-annealing achieves nearly 95X reduction in convergence time which fortifies the ability of adapting to quick changes in the environment while maintaining lower rejection rate and runtime of the algorithm. And compared to the greedy approach which is known for remarkably high immediate rewards, HRL-Edge-Cloud outperforms in system cost and rejection rate efficiency improvement.

Our extensive simulations verify the practicality of the system in a near real-world scenario. At the same time, performance evaluation also shows that our approach outperforms the existing heuristic method and two greedy algorithms when compared at bench-marking metrics (cost, rejection rate, QoE).

The remainder of this paper is organised as follows: Related works are discussed in Section 2. System model of Edge-cloud based smart streetscape system is described in Section 3. Section 4 contains our core heuristic reinforcement leaning (HRL) algorithm with linear-annealing and pruning principle details about multi-resource allocation system. Section 5 presents the performance evaluation and discusses the stability of the system, followed by the conclusion and future directions in Section 6.

2 Related Work

Resource optimization in the Edge-cloud (EC) is an elementary concern for assuring the efficient network utilization, QoE and reliability (Barakabitze et al., 2020). This can be catered using the information derived from hyper-local movement or user density, occasional events like sports tournaments in a stadium and video content demand in a

particular region. Below we discuss several studies which analyzed the resource management problem by taking into account the constrained resources (CPU, Memory, Bandwidth) along with the notion of applications and service characteristics to magnify the system performance and cut the operational cost for service providers. We categorize existing literature review into two research areas, including optimization-based and machine-learning-based methods, respectively.

2.1 Optimization-Based Methods

The MRA (Multi-Resource Allocation) problem can be formulated as a mixed integer linear programming (MILP) model based on a hierarchical architecture. For example, Bi et al. (2019) designed a data rate based heuristic (DRH) algorithm derived from the resource allocation patterns obtained from the MILP model to minimize the total latency and improve the CPU utilization of EC servers. Tran and Pompili (2019) and Xue and An (2021) decoupled the problem of task offloading and resource allocation into subproblems. They formulated the problem as a mixed integer nonlinear programming (MINLP) problem, and solved using quasi-convex and convex optimization techniques. A novel heuristic algorithm is also proposed in Tran and Pompili (2019) that achieves suboptimal solution in polynomial time. Nevertheless, all of the above consider only EC based environments and lack the three-tier hierarchical architecture integrated with the BC. Further, these approaches usually include heavy computations and exhaustive search space, which is not suitable for addressing MRA problem in large scale scenarios.

Some studies proposed game theory based methods to solve the optimization problem of network and computational resources in EC based environments (JošCsilo & Dán, 2018; Zheng et al., 2019; Dong & Wen, 2019). JošCsilo and Dán (2018) models the resource allocation problem as Stackelberg game and an efficient equilibrium strategy profile is computed for optimal wireless and computation resource allocation to reduce the completion time of resource intensive tasks at the expense of increase in computational complexity. Similarly, Zheng et al. (2019) formulated a multi-user computation problem to allocate wireless channels to mobile users in a dynamic environment and proposed a multi-agent stochastic learning algorithm to achieve Nash Equilibrium among mobile users. However, these algorithms are computationally heavy and do not scale well in large environments. Dong and Wen (2019) leveraged the decentralized strategy based on the evolutionary game theory to address the task offloading and resource allocation problem coordinately at the EC and the central cloud. Although, this strategy minimizes the cost of resource procurement while meeting the delay constraints, but it is only practical in stationary environments where user behavior does not change in time.

Liu et al. (2016) proposed an adaptive multi-resource allocation strategy using semi-Markov decision process (SMDP) modeling and solved with linear programming, which determines an optimal action policy for allocating wireless bandwidth and compute resources in three-tier EC and BC-based environment. Although, it achieves optimal system benefits in a stationary world, but due to the real-time needs and varying traffic patterns this offline approach is not practical. Thereafter, a structured policy table and index based search approach (Qadeer et al., 2020) addressed the above real-time resource allocation problem. Nonetheless, this solution still has legacy scalability issue for a large number of IoT devices such as massive machine type communication (mMTC) (Sharma & Wang, 2020). Li et al. (2018) proposed a new ECIoT (Edge computing for IoT) architecture which considers scalability, big data processing and power consumption of IoT devices. Radio and computation resources, and power management problem is formulated as a cross-layer dynamic stochastic network optimization and solved by utilizing the Lyapunov stochastic optimization approach. This technique needs the prior statistical observation information of IoT devices which may not be practically available in dynamic environments.

2.2 Machine Learning-Based Methods

The proven success of Machine Learning (ML) based techniques has spurred the adoption of ML algorithms to solve control and management problems for IoTs in clouds and 5G wireless networks (Lei et al., 2020; Wang et al., 2020; Feriani & Hossain, 2021). Some recent studies are discussed in this section.

Cheng et al. (2018) utilized the Deep Reinforcement Learning (DRL) with experience replay (ER) and a target network to train the Deep Q-Network in order to solve a resource provisioning and task scheduling problem in a cloud-based environment under the strict QoS requirement. Wei et al. (2019) proposed a natural actor-critic reinforcement learning framework to jointly solve the problem of content caching, computation offloading and radio resource management with the goal of minimizing the end-to-end delay. Peng and Shen (2020) leveraged the deep deterministic policy gradient (DDPG) and hierarchical learning architectures to jointly solve the spectrum, computation and storage allocation problem in an EC based system. Recent studies (Nath & Wu, 2020; Chen et al., 2021) solved the computation offloading and resource allocation problem for multiple mobile users in EC based systems by utilizing the DDPG-based framework and proposing the sate-of-the-art algorithms. However, all of

the above techniques are based on deep-learning methods which is heavily dependent on resource intensive hardware (e.g. GPU/FPGA) for the training of complex and large networks (Wang et al., 2020), which may not be provisioned in a resource constrained practical environment. The transformation of deep-learning based complex theoretical models into real-world systems is another big challenge (Lei et al., 2020).

Motivated from the above discussion, we aim at presenting a lightweight but scalable solution which can also be easily implemented in real-world scenarios like COSMOS testbed (Raychaudhuri et al., 2020). A recent work (Qadeer et al., 2021) proposed heuristic reinforcement learning-based algorithm to solve the bandwidth allocation problem in an EC based system. The presented results are encouraging, which stimulated to extend the current framework (Qadeer et al., 2021) and comprehensively solve the multi-resource allocation problem. This is the rationale behind the introduction of HRL-Edge-Cloud. HRL-Edge-Cloud framework takes into account the wireless and computation resource allocation problem equitably at the Edge-cloud (EC) and back-end cloud (BC). To the best of our knowledge, none of the existing works applied heuristic reinforcement learning with linear-annealing and pruning principle to solve the multi-resource allocation problem jointly in EC and BC with the goal of minimizing overall system cost for providers, meeting the strict QoS requirements of applications and fast self-learning capability.

3 System Model for Edge-cloud based Smart Streetscape

This section presents the system description, users and jobs model and bandwidth model. The computational model is described for both EC and BC. Delay model explains different types of delays, followed by the utility model of the system.

3.1 System Description

We consider the Edge-cloud (EC) based streetscape system as shown in Fig. 1. Our proposed system provides 5G based wireless radio access (including sub-6 GHz and mmWave) to sensors, street signals, vehicles, security cameras, mobile devices and other Internet of Things (IoTs) via software defined radios (SDRs) herein called virtual base stations (BSs). In 5G architecture, one SDR/BS covers a small cell; therefore, multiple BSs can connect to one nearby edge-computing infrastructure via high speed and low-latency software defined fiber links (Raychaudhuri et al., 2020).

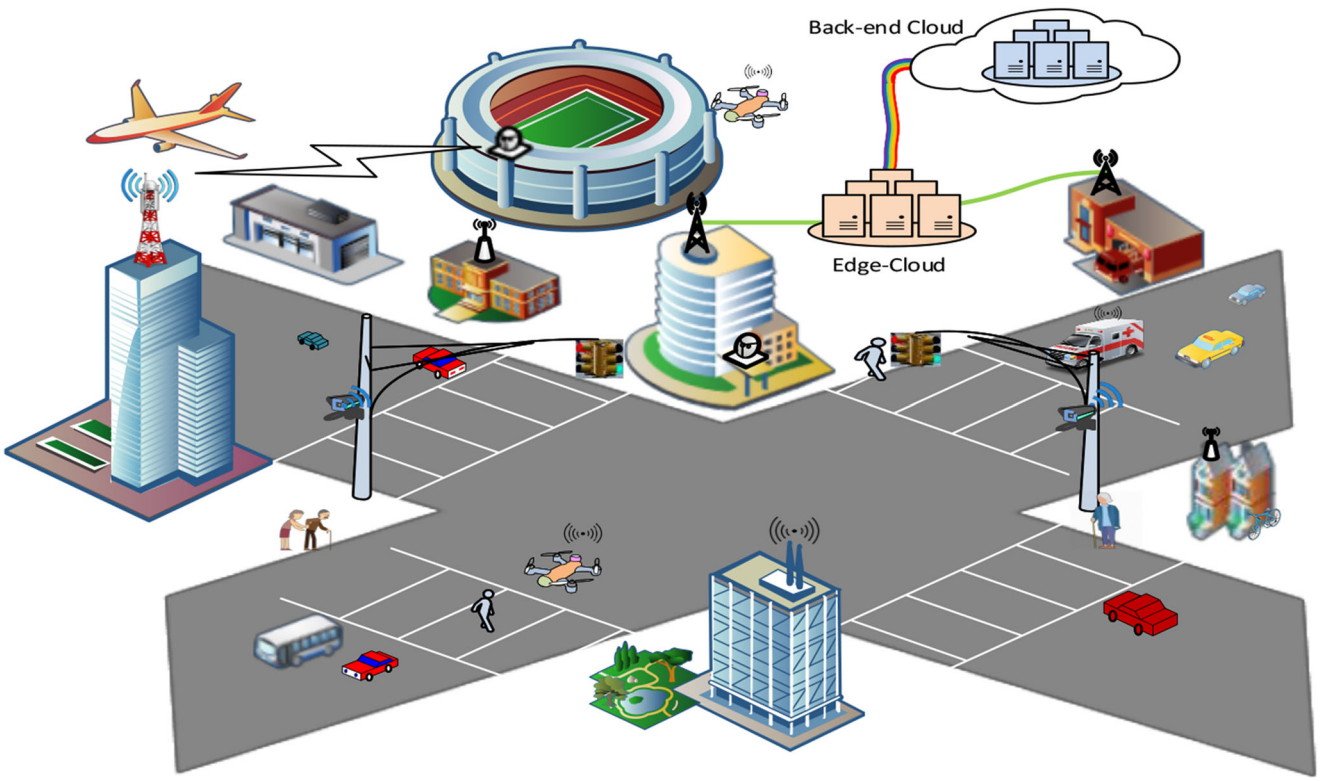

Fig. 1 System model and Structure of Edge-cloud based Smart Streetscape System. The detail about System Model is given in Section 3

For the beyond 5G deployments, mobile network operators (MNO) rely on connected EC and BC for scalability and enhanced services (Arno & Mazur, 2021). Therefore, it is vital to consider EC along with BC in order to propose a realistic system. In our proposed system, EC is interconnected with the BC via high speed fiber links to leverage abundant and always available resources at public clouds (AWS, Google Compute Engine, Microsoft Azure) in order to offload delay-tolerant multimedia tasks or to save data for future uses. The connected mobile and IoT devices offload their resource intensive tasks to nearby EC or BC for fast processing and energy conservation, or to smartly manage/divert the traffic of streetscape in an emergency situation.

Context aware control of resources is main ingredient of the smart streetscape environment such as smart control of pedestrian signal for elderly people and traffic signal control for emergency situations (e.g. fire on a building) or in a logistics unloading case, traffic can automatically be diverted to a safer and smoother street with the help of data sent by the IoTs (Yang et al., 2021).

Multi-media (AR, VR, Video) applications are bandwidth hungry and resource intensive, at the same time require low end-to-end latency (Liu et al., 2020). In such scenarios, proposed Edge-cloud infrastructure plays an important role to supply uninterrupted services to both streetscape and multi-media applications. Upon arrival of a service request, based on it's exigency, the system decides whether to run it on EC or BC, and based on the availability of resources how much bandwidth and compute resources have to be allocated in order to execute the task within the deadline (More details can be found in Section 4).

3.2 Users and Jobs Model

In the proposed framework, we consider all the devices which are connected to the system as Edge-cloud users. Each user offloads its computational task in the form of a distinct service request. The entire workload of the system is a set of J jobs from U users i.e. $J = \{j_1(dl_1, d_1), j_2(dl_2, d_2), ..., j_U(dl_U, d_U)\}$. A job $j_u(dl_u, d_u)$ is a tuple of two variables where dl_u means the hard deadline in milliseconds and d_u represents the data in bytes to be processed in job j_u offloaded by user u. The user job can demand both CPU and Memory for a successful execution but processing vitally involves CPU; therefore, we only consider CPU cycles as a job processing source as proposed by Cheng et al. (2018). Suppose $\mathbb{C}$ represents the number of CPU cycles required to process 1 Byte of data, then L_u is given as the total CPU cycles required to compute data d_u ($L_u = d_u \times \mathbb{C}$). A similar task computation model (with CPU cycles) is proposed in Nath and Wu (2020) and Chen et al. (2021) as well.

3.3 Wireless Bandwidth Model

As shown in Fig. 1, an Edge-cloud system owns W base stations $\{1, 2, 3, ..., W\}$, each of which makes meshed wireless connections with actuators/relays to provide wireless access capacity, load-balancing and failover. The total wireless bandwidth in the form of discrete units that is available at each base station is represented as B_w. Each base station w can support H wireless bandwidth channels $\{ch_1^w, ch_2^w, ch_3^w, ..., ch_H^w\}$, and each wireless bandwidth channel ch_h^w ($h \in [1, H]$) provides a specific amount of bandwidth units (data rate in bps) and costs c_h^w. The directional antenna array in mmWave cellular networks of the COSMOS testbed (Raychaudhuri et al., 2020) is capable of exploiting beamforming, which compensates the increased path-loss at mmWave frequencies and overcomes the additional noise due to the large transmission bandwidth (Habibi et al., 2019). As a bonus, interference isolation is achieved in directional beamforming, which, as a result, reduces the adjacent-cell interference. Therefore, in our case, we deliberately ignore path-loss, channel noise and interference factors, and manage resources at the application layer through well-defined APIs (Ungureanu et al., 2021).

3.4 Computational Model

3.4.1 Edge Cloud

An Edge-cloud (EC) owns M servers $\{1, 2, 3, ..., M\}$. Each server m processes an offloaded job via a set of virtual machines (VMs). Let $K = \{vm_1^m, vm_2^m, vm_3^m, ..., vm_K^m\}$ be the set of VMs that can be assigned by server m and each VM vm_k^m provides a specific amount of computational units (CPU cycles in Hz) to process a job and costs c_k^m. The total compute capacity at an EC server m is given by C_m processing units and these units are allocated to the jobs in the form of VMs (CPU cycles). Chen et al. (2021) proposes a similar computational model, however, they consider EC with unlimited compute resources, which may not be valid for practical scenarios.

3.4.2 Back-end Cloud

Previous work (Liu et al., 2016) considered back-end cloud (BC) as a source of unlimited compute capacity and always allocated the maximum number of VM units to the user jobs that are offloaded to the BC for faster execution. However, we take into account a realistic model to minimize the overall cost of the system by adopting pay-as-you go model. Therefore, just like an EC, we consider N BC servers $\{1, 2, 3, ..., N\}$, which execute a job via a set of $K = \{vm_1^n, vm_2^n, vm_3^n, ..., vm_K^n\}$ VMs, and each of which costs c_k^n. The total of C_n computational units are available at

a BC server n, which are further allocated as VM units. This model can be easily extended to infinite resource model by relaxing N and K sufficiently large.

3.5 Delay Model

We define round-trip time (RTT) as the total time that it takes for a job to upload to the EC or BC via a wireless channel, process the job and then send the result back. This involves propagation, transmission (2-way) and processing time. A similar delay model is also used in Wei et al. (2019).

3.5.1 Propagation Time

Propagation time through fiber or air media is negligible and assumed to be constant. Therefore, we consider a constant delay $t_u(prop_{ec})$ for EC and $t_u(prop_{bc})$ for BC depending on where the resources are allocated for job execution.

3.5.2 Transmission Time

This includes the time that a job takes to upload to the EC or BC and the time to send the result back successfully. It depends upon the amount of data and wireless channel that is allocated. The transmission time $t_u(trans_{ec})$ of a job to the EC can be calculated as follows:

$$t_u(trans_{ec}) = \frac{d_u}{ch_h^w} + \frac{R_u}{ch_h^w}, \tag{1}$$

where R_u is the result which is generated after the job execution and sent back. In general, the result is a control signal and contains only a few kilobytes of data (JošCsilo & Dán, 2018). Nonetheless, the result for AR/VR applications can be substantially large, therefore, we incorporate it in our framework. According to Fig. 1, EC is connected with BC via a high capacity fiber link and considered to guarantee b bandwidth (Raychaudhuri et al., 2020). Thus, the transmission time between a device and BC can be given as:

$$t_u(trans_{bc}) = t_u(trans_{ec}) + \frac{d_u}{b} + \frac{R_u}{b}. \tag{2}$$

3.5.3 Processing Time

This depends upon the number of required CPU cycles L_u and the allocated VM vm_k^m, vm_k^n at EC or BC, respectively:

$$t_u(proc_{ec}) = \frac{L_u}{vm_k^m}, \tag{3}$$

$$t_u(proc_{bc}) = \frac{L_u}{vm_k^n}. \tag{4}$$

To summarize, when a job j_u is offloaded to the EC, the total round-trip time is given as:

$$rtt_{ec}(j_u) = t_u(prop_{ec}) + t_u(trans_{ec}) + t_u(proc_{ec}), \tag{5}$$

similarly, when the job is offloaded to the BC then:

$$rtt_{bc}(j_u) = t_u(prop_{bc}) + t_u(trans_{bc}) + t_u(proc_{bc}). \tag{6}$$

3.6 Utility Model

We evaluate the utility of the system per base station and per server basis. The total usage of the system at any given time t is the sum of all the occupied resources by all the jobs which are being processed. Therefore, the bandwidth utility rate of a base station w is given as:

$$Ur_w(t) = \frac{\sum_{h=1}^{H} ch_h^w . \mu_h^w(t)}{B_w}, \tag{7}$$

where μ_h^w is the total number of ch_h^w channels which are serving the jobs at time t. Similarly, the utility rate of a server at EC and BC can be calculated as:

$$Ur_m(t) = \frac{\sum_{k=1}^{K} vm_k^m . \mu_k^m(t)}{C_m}, \tag{8}$$

and

$$Ur_n(t) = \frac{\sum_{k=1}^{K} vm_k^n . \mu_k^n(t)}{C_n}, \tag{9}$$

respectively. The evaluation of the system at individual base station and server level provides granularity and fine-grained control to the underlying resources.

4 HRL-Edge-Cloud: Multi-Resource Allocation

In this section, we present HRL-Edge-Cloud to solve the multi-resource allocation (MRA) problem for mobile and streetscape based applications. 5G connected edge platforms expose rich APIs for the enterprises and third parties to integrate their operational services or ML-based applications (Open, 2020). As depicted in Fig. 2, MRA-controller is deployed in the EC, which utilizes the well-defined APIs to dynamically control the underlying resources with the help of resource allocation policy. The connected mobile devices and IoTs offload their resource intensive jobs to MRA-controller. The MRA-Controller, using the resource allocation policy (generated by HRL agent), decides where to schedule the job (EC or BC) and how much resources have to be allocated based on the utility model (as proposed in Section 3.6).

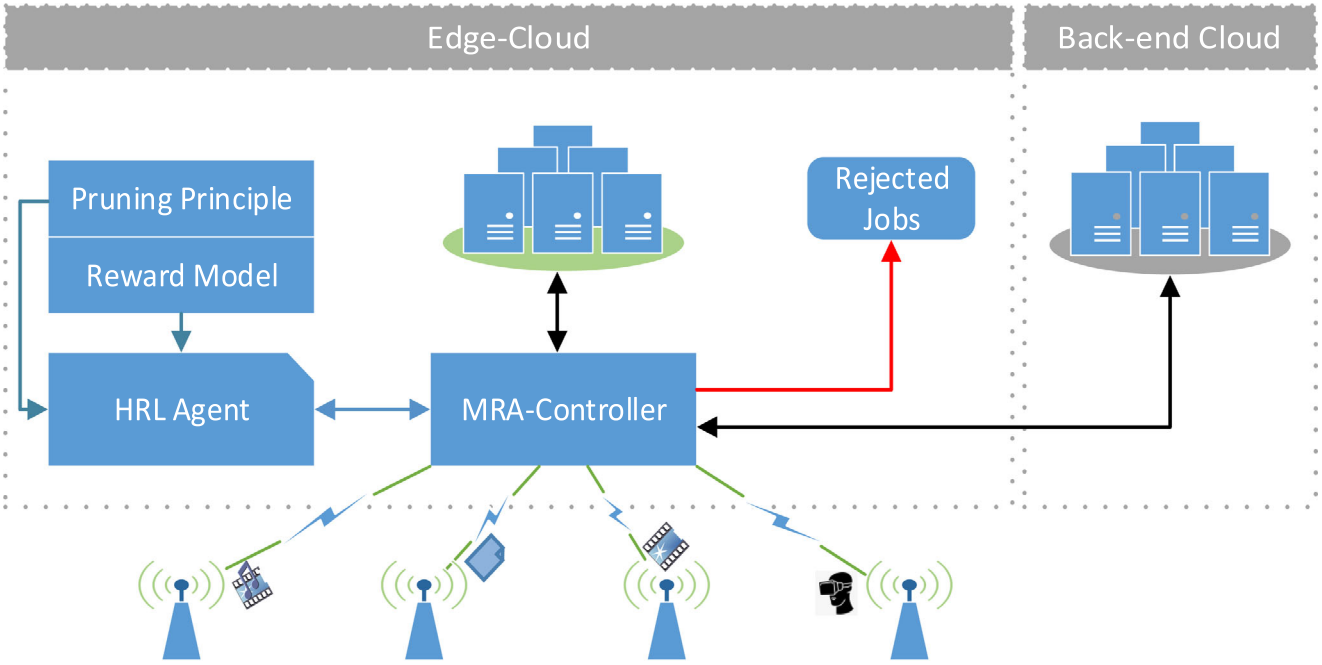

Fig. 2 The structure of HRL-Edge-Cloud framework: the detail of MRA-Controller is described in Section 4

4.1 MRA-Controller

Upon the arrival of a job, MRA-controller rejects it if no more resources are available or if the job cannot be completed within the given deadline even by allocating maximum possible bandwidth and VM units. Otherwise, the MRA-controller, using the optimal policy learnt by heuristic reinforcement learning (HRL) agent, selects an appropriate base station and server (EC or BC) pair, and then decides the amount of bandwidth and VM units to schedule the job that can give maximum system benefits. When execution of the job is completed, the result is sent back to MRA-controller which relays the result to the mobiles/IoTs or sends the control signal to actuators such as traffic control signal, etc.

4.2 Semi-Markov Decision Process (SMDP) Formulation

We first formulate the multi-resource allocation (MRA) problem of wireless bandwidth and VM allocation in the Edge-cloud system into semi-Markov decision process (SMDP). We define state space and action space followed by our unique reward model. At each state, different resource allocation actions yield different rewards. The goal is to maximize the long-term reward (minimize system cost, rejection rate and round-trip time) while meeting the hard deadline of the job.

4.2.1 State Space

In the proposed Edge-cloud based system, the state of the system at any time t is the observation of utility rates of all the base stations and servers, and the current job j_u which has to be scheduled either at EC or BC. Therefore, each distinct state is a sequence of observations of all the resources, $s_t = \{Ur_w, ..., Ur_W, Ur_m, ..., Ur_M, Ur_n, ..., Ur_N, j_u(dl_u, d_u)\}$. In each decision epoch, HRL agent learns optimal resource allocation strategies based on these sequences which leads to a large but finite SMDP chain.

4.2.2 Action Space

For base stations and servers, the MRA-controller is responsible for selecting a base station w and server m or n (depending on the deadline of job and available resources). Whereas for wireless bandwidth and VM, the controller decides how many units have to be allocated in order to successfully execute the job within the deadline. Therefore, in any state, the action is chosen from a set of all possible allocations in that state. Let action set, $A_{s_t} = \{(ch_h^w, vm_k^m), (ch_h^w, vm_k^n), -1, 0\}$, where 0 means rejection of the job, -1 represents departure of the job after successful execution, (ch_h^w, vm_k^m) represents an action to allocate a wireless channel with h units of bandwidth on

base station w and a VM with k units of CPU cycles on server m at the EC. Similarly, (ch_h^w, vm_k^n) represents VM allocation on server n at the BC.

4.2.3 Reward

The goal of HRL-Edge-Cloud multi-resource allocation system is to minimize the overall system cost by taking a sequence of actions in all the states. According to the definition of a real-valued reward function (Cheng et al., 2018; Liu et al., 2016), in order to find an optimal multi-resource allocation policy by taking action a_t at state s_t, system advances into a new state s_{t+1} and receives a reward r_t from the environment. In our model, the reward can be calculated as the sum of lump income and cost of resource procurement in the system:

$$r_t(s_t, a_t) = I_u(s_t, a_t) - cost_u(s_t, a_t) \times rtt(j_u), \tag{10}$$

where $I_u(s_t, a_t)$ is the net income earned by executing the job j_u and $cost_u(s_t, a_t)$ is the cost of resource procurement for $rtt(j_u)$ time (time it takes to process the job) for selecting action a_t at state s_t. In the definitions of $I_u(s_t, a_t)$ and $cost_u(s_t, a_t)$, we take into account the completion time of the job and the utility of wireless and computation resources.

$I_u(s_t, a_t)$ is defined as,

$$I_u(s_t, a_t) = \begin{cases} (dl_u - rtt_{ec}(j_u)).\delta, \ if \ a_t = (ch_h^w, vm_k^m) \\ (dl_u - rtt_{bc}(j_u)).\delta, \ if \ a_t = (ch_h^w, vm_k^n). \end{cases} \tag{11}$$

In Eq. (11), δ represents the revenue that the service provider generates by successfully executing the user job. It can be set on-the-fly which can differ depending on the job completion time. Note that, if the completion time of job $(rtt(j_u))$ exceeds the deadline (dl_u), then the income becomes negative which impacts the overall reward. This effect encourages the system to take the allocation decisions that can complete the jobs before deadlines.

In contrast to the income, $cost_u(s_t, a_t)$ describes the cost of resource procurement per unit time by allocating wireless bandwidth channel and a VM at EC or BC and is given as,

$$cost_u(s_t, a_t) = \begin{cases} Ur_w(t).c_h^w + Ur_m(t).c_k^m, \ if \ a_t = (ch_h^w, vm_k^m)) \\ Ur_w(t).c_h^w + Ur_n(t).c_k^n, \ if \ a_t = (ch_h^w, vm_k^n)), \end{cases} \tag{12}$$

where c_h^w and $\{c_k^m, c_k^n\}$ represent the cost of wireless bandwidth channel and VM allocation per unit time, respectively.

As compared to Liu et al. (2016), we calculate the reward for each individual job, so as the cost and the income. This approach is more meaningful in a way that the QoS requirement in each job may vary and calculating reward for every individual job can better assist the Q-Learning agent to derive an optimal resource allocation policy.

The optimization problem of wireless bandwidth and VM allocation for the user jobs at different base stations and servers, while considering the varying QoS requirements and constrained resources is formulated as below:

$$maximize \sum_{t=1}^{T} r_t(s_t, a_t) \tag{13}$$

subject to:

$$Ur_w(t) \leq 1, \forall t \in T, \forall w \in W \tag{14}$$

$$Ur_m(t) \leq 1, \forall t \in T, \forall m \in M \tag{15}$$

$$Ur_n(t) \leq 1, \forall t \in T, \forall n \in N \tag{16}$$

$$rtt_{ec}(j_u) \leq dl_u, rtt_{bc}(j_u) \leq dl_u, \forall u \in U, \tag{17}$$

where constraints (14, 15 and 16) describe that the utility of any base station and EC/BC server does not exceed from it's total available capacity, respectively. The constraint (17) guarantees the hard deadline requirement of the job offloaded at EC or BC.

4.3 Heuristic Reinforcement Learning Algorithm for MRA

Since our Edge-cloud environment is non-stationary, which means that the traffic patterns can change over time. There is a need to devise an algorithm that can quickly adapt to the changes in the environment. Therefore, we modify our training algorithm from standard Q-Learning by using a heuristic function with linear annealing and a pruning principle to make it scalable with speedy convergence.

4.3.1 Q-Learning

Q-Learning is an Off-Policy based machine learning algorithm that is used to generate an arbitrary target policy (resource allocation policy in our case) and can converge with probability 1 to a close approximation of the action-value function $Q(s, a)$ (Sutton & Barto, 1998). The above given reward model (Eq. (10)) is used by the Q-Learning agent as utility to calculate the value function $Q(s, a)$ by following any policy $(\epsilon - greedy, \epsilon - soft, softmax$ etc.). The optimum action-value function $Q^*(s, a)$ is the maximum expected achievable reward which follows the Bellman equation (Cheng et al., 2018). In state s_t, after

taking action a_t the system receives a reward r_t, thus, we update the Q value estimates as follows:

$$Q_{t+1}(s_t, a_t) = (1-\alpha) Q_t(s_t, a_t) + \alpha(r_t(s_t, a_t) + \gamma \max_{a_{t+1}} Q_t(s_{t+1}, a_{t+1})), \quad (18)$$

where α is the learning rate and γ is the discount factor which is used to weight the impact of the future reward on the current action (Sutton & Barto, 1998).

4.3.2 Heuristic Function

Q-Learning is very time consuming because it is performed through trial-and-error interactions of the agent with the environment. The rate of convergence can be sped up by using a heuristic function for selecting suitable actions to guide the exploration of the state-action space in the direction of useful regions, thus, improving agent's behavior (Bianchi et al., 2008). The heuristic function $H(s_t, a_t)$ is an action policy modifier which specifies the significance of performing an action a_t at time t when visiting state s_t. To accelerate the learning process, prior domain information is extracted from the environment and a heuristic is composed from the extracted structural information. The detail to incorporate the heuristic function as an addition with standard $\epsilon - greedy$ Q-Learning action value function is given below:

$$\pi(s_t) = \begin{cases} \arg\max_{a_t} [Q_t(s_t, a_t) + \xi (H_t(s_t, a_t))^{\beta}], & if\ q \leq \epsilon_t, \\ a_{random} & otherwise. \end{cases} \quad (19)$$

In above Eq., ξ and β are design parameters to control the impact of heuristic value function on action selection policy. q is uniform random value from 0 to 1, ϵ_t is also between 0 and 1 which is the probability of exploration (random action) at time t; and we linearly anneal it which is detailed in the next section. a_{random} is a uniform random action chosen from the possible actions in state s_t. According to Bianchi et al. (2008) and Fang et al. (2012), when $\xi = \beta = 1$, the update formula for the heuristic function is given as:

$$H_{t+1}(s_t, a_t) = \begin{cases} \max_a Q_t(s_t, a) - Q_t(s_t, a_t) + \eta, & if\ a_t = \pi^H(s_t), \\ 0 & otherwise, \end{cases} \quad (20)$$

where a_t is an optimal action influenced by the heuristic $\pi^H(s_t)$, and η is a small positive value usually set to 1 (Bianchi et al., 2008; Fang et al., 2012). According to Bianchi et al. (2008), a heuristic is obtained in two stages, in the first stage, domain information is extracted and heuristic is built only once. In the following stage, the obtained heuristic is used in the action selection policy and is not updated in future to overcome a bad heuristic. Whereas, Fang et al. (2012) proposed a heuristic reinforcement learning based on the state backtracking technique which no longer requires stage one; nonetheless, it causes additional delay to update the action transfer probability function at the end of every training episode. Therefore, we adopt the former approach in order to save the overall training time.

4.3.3 Linear Annealing

Bianchi et al. (2008) used fixed exploration rate which is suitable only for small environments. However, for an EC like large environment, we start exploration with the highest probability and then linearly decay the exploration rate. This strategy gives enough trials to explore a large environment randomly at the beginning and then exploits the already known good policies more often. Further, randomness breaks the correlation of learning data and reduces the variance in the update (Cheng et al., 2018; Chen et al., 2021). The decay frame or in other words the linear annealing frame $F_{LA}(t)$ for ϵ_t is given below:

$$F_{LA}(t) = \frac{\epsilon_s - \epsilon_f}{T} \times t, \quad (21)$$

where T is maximum number of iterations (steps) in one training episode, and ϵ_s and ϵ_f represent the start ϵ and final ϵ, respectively. Thereafter, ϵ_t can be given as follows:

$$\epsilon_t = \max(\epsilon_f, (\epsilon_s - F_{LA}(t))) \quad (22)$$

The Eq. (22) describes that the exploration rate is high at the beginning and annealed with the time, which comes to its final state at the end of first training episode. A novel study, human-level control through deep reinforcement learning (Mnih et al., 2015), used a predetermined approach of linear annealing where ϵ is decayed in a fixed number of iterations. However, our proposed model gives the malleability to adjust annealing rate with respect to the size of the environment and the number of learning iterations.

4.3.4 Pruning Principle

According to our cost function in (12), we know that allocating bandwidth and VM units on a base station and server with least utility will give us more reward (assuming initial investment is fixed). Therefore, during action selection, we apply pruning on BS/EC/BC servers so that we allocate resources on the base station and the server with least utility in order to get the maximum reward.

$$w_t = \arg\min(Ur_w(t)), \forall t \in T, \forall w \in W, \quad (23)$$

$$m_t = \arg\min(Ur_m(t)), \forall t \in T, \forall m \in M, \quad (24)$$

$$n_t = \arg\min(Ur_n(t)), \forall t \in T, \forall n \in N. \quad (25)$$

Intuitively, our proposed pruning principle also helps to balance the load among all the base stations and servers;

which means, it does not lead to overload a single base station or server which could potentially result in higher rejection rate for future jobs. The major contribution of our pruning principle is the reduction of action space at every state by a significant amount. The new action set $a_t = \{ch_h, vm_k, -1, 0\}$ is only responsible for selecting an appropriate wireless channel and VM because the base station and server are already pruned.

The algorithm for Heuristic reinforcement learning with linear annealing and pruning principle to obtain optimal Multi-resource allocation policy in the Edge-cloud environment is summarized in Algorithm 1.

Input : User jobs with varying QoS requirements

Initialize the environment;
Initialize action-value function Q arbitrarily;
Initialize Heuristic function H;
for *episode = 1 to E* **do**
  Reset the environment to initial state;
  for *t = 1 to T* **do**
    Calculate ϵ_t using Eq. (22);
    Prune action space using Eqs. (23, 24, 25);
    Choose action a_t using Eq. (19);
    Execute action a_t, observe next state s_{t+1} and receive reward r_t;
    Update the values of heuristic function using Eq. (20);
    Update the values of Q according to the update rule in Eq. (18);
    $s_t = s_{t+1}$;
    end
  end

Output: Optimal Multi-resource Allocation policy Q^*

Algorithm 1 Heuristic Reinforcement Learning (HRL) with Linear Annealing and Pruning Principle

5 Experimental Results

In this section, we evaluate the performance of our heuristic reinforcement learning with linear annealing and pruning principle based multi-resource allocation algorithm. We develop our HRL-Edge-Cloud framework (Section 3) and proposed HRL training algorithm (Section 4.3) in the Python 3.8.1 to simulate a near real-world environment. We run all the experiments on Dell Desktop Machine with 3.60 GHz Intel Core i7 processor, 8GB memory and Windows 10 Pro 64-bit OS, and discuss the advantages of our proposed algorithm over the alternative baselines.

5.1 Experiment Setup

The list of hyperparameters and their corresponding default values that are used by our system for the simulation are given in Table 1. The three baselines we use to compare with the HRL-Edge-Cloud are described below:

- **HAQL** (Bianchi et al., 2008): Existing heuristic accelerated Q-learning approach to accelerate the learning mechanism in Q-learning. This approach used fixed exploration rate of 10% to calculate the heuristic only once. The rest of the parameters ($\alpha, \beta, \gamma, \eta, \xi$) and reward function are kept same for a fair comparison.
- **Greedy-1** (Liu et al., 2016): In this approach, system tries to preserve resources by allocating one wireless bandwidth channel and one virtual machine with the minimum resources i.e. one bandwidth unit and one VM unit on a base station and EC or BC server which are chosen randomly.
- **Greedy-2** (Liu et al., 2016): System tries to boost QoE by decreasing the delay and allocates maximum number of wireless bandwidth units on a base station and VM units on an EC or BC server to process the job (i.e. largest wireless channel and VM) if available. Note that, like Greedy-1 approach, base station and server pair is chosen randomly in Greedy-2 approach as well.

Convergence speed is compared only with the HAQL approach because this is the closest model related to our model. The rest of the performance indicators such as, system cost, job rejection rate, QoE and runtime are compared with all of the three baselines. We present three streetscape simulation environments: small, medium, and large scale; for performance comparisons. Small-scale environment contains 4 wireless base stations, 10 EC and 10 BC servers. Given the default parameters in Table 1, small-scale environment contains 400 (4×100) wireless bandwidth units, 360 ($(18 \times 10) + (18 \times 10)$) VM units at the EC and BC combined. Medium-scale environment contains 12 base stations, 30 servers at the EC and 30 at the BC, which constitutes 1200 wireless bandwidth units and 1080 VM units (combined), respectively. Likewise, the large-scale environment consists of 20 base stations, 50 EC and 50 BC servers, which constructs 2000 wireless bandwidth units and 1800 VM units, respectively. We conduct experiments on small-scale, medium-scale and large-scale environments with 10,000, 100,000 and 1,000,000 jobs, respectively. The wireless bandwidth and VM per unit rental/usage cost is normalized to 1 cent per second.

The default values of reinforcement learning related hyperparameters such as learning rate $\alpha = 0.1$ and discount factor $\gamma = 0.99$ are chosen from commonly used range (Qadeer et al., 2021; Bianchi et al., 2008; Mnih et al., 2015). The exploration rate ϵ is decreased from 1 to 0.1 using

Table 1 Hyperparameters and corresponding Default Values with Description

Hyperparameter	Value	Description
α	0.1	The learning rate alpha, set between 0 and 1, used in the Q-learning update.
β	1	Design parameter to control the influence of heuristic function.
γ	0.99	Discount factor gamma, set between 0 and 1, used in the Q-learning update.
δ	10	Fixed revenue that is generated by service provider after successful job execution.
ϵ_s	1	Start value of exploration rate ($\epsilon - greedy$).
ϵ_f	0.1	Final value of exploration rate ($\epsilon - greedy$).
η	1	A small positive value used in the heuristic function update.
ξ	1	Design parameter to control the influence of heuristic function.
$\mathbb{C}$	100	Number of CPU cycles required to process one byte of data.
R	10	The result in $kilobytes$ generated after successful job execution.
b	1	Average bandwidth in Gbps available on the fiber link from EC to BC.
B	100	Total wireless bandwidth units available at each wireless base station (1 unit = 10Mbps).
C	18	Number of cores (processing units; 1 unit/core = 1GHz CPU cycles) per server both at EC and BC.
E	10	Number of episodes to train the HRL agent.
H	4	Number of different wireless bandwidth channels that can be allocated on a wireless base station.
K	4	Number of different virtual machines that can be allocated on a server at EC or BC.
$t_u(prop_{ec})$	5	Average propagation time in milliseconds from user u to the EC.
$t_u(prop_{bc})$	50	Average propagation time in milliseconds from user u to the BC.
T	100000	Number of learning steps or iterations per episode.

our proposed linear annealing method (Section 4.3.3). The default values of hyperparameters related to the heuristic function such as $\beta = 1$, $\eta = 1$ and $\xi = 1$ are also commonly used (Bianchi et al., 2008; Fang et al., 2012). All other parameters are experiment specific which can be set according the environment needs.

Mobile users and other IoT devices generate tasks of different sizes with varying QoS demands. We group these user tasks/jobs into two groups; 1) Type-1 tasks are critical tasks with comparably small size (bytes) and deadline. We set the deadline of such tasks to be in the range of [200 ms, 800 ms], and the size to be in the range of [100 KB, 500 KB]; 2) Type-2 tasks are multi-media tasks with large size and relaxed deadline as compared to the Type-1 tasks. The deadline of such tasks is set between [1000 ms, 2000 ms], and the size between [1 MB, 2 MB].

5.2 Convergence

We compare the convergence rate of our proposed strategy with the existing HAQL algorithm. As shown in Fig. 3, HRL-Edge-Cloud converged almost in one episode and obtained the overall reward higher than the HAQL. The recorded time that it took to reach at the convergence point was only 37 seconds in the case of HRL-Edge-cloud and 3492 seconds in the case of HAQL on small scale environment. The improvement of this magnitude is due to our unique pruning principle as described in Section 4.3.4. This shows that our proposed method outperforms the existing approach by reducing the convergence time by nearly 95X and achieving a better cumulative reward. In addition to that, after one episode when the heuristic is applied, the divergence and oscillations in the case of HRL-Edge-Cloud

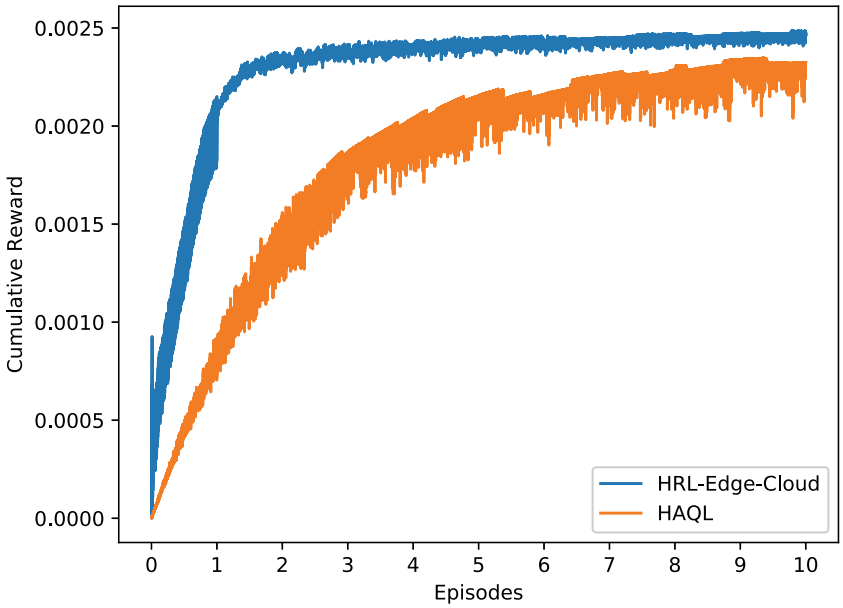

Fig. 3 Convergence speed and cumulative reward comparison between HRL-Edge-Cloud and HAQL

are significantly reduced as compared to HAQL which demonstrates the higher efficiency of the learning procedure of our proposed strategy as detailed in Section 4.3.3.

5.3 Performance Analysis

We analyze the performance of our proposed strategy by comparing the operational cost of the system, job rejection rate, Quality of Experience (QoE) of the user (explained later in this section) and the runtime with three baselines (HAQL, Greedy-1 and Greedy-2) for three different scales of environments. Figure 4(a) shows the operational cost in different scenarios. The cost is calculated after dividing the cumulative cost of jobs by total number of accepted jobs in the respective scale. It is evident from the results that HRL-Edge-Cloud achieves up to 2.34X, 5X and 20X cost efficiency improvement on average as compared to the HAQL, Greedy-1 and Greedy-2 baselines, respectively. Greedy-2 is the most costly strategy due to the fact that it always allocates maximum possible bandwidth and VM units if available. Greedy-1 allocates minimum resources which yields longer delay, in turn, higher cost according to Eq. (10). The rejection rate is calculated by dividing the number of rejected jobs by the total jobs in the respective scale. Our proposed strategy consistently gives rejection rate a multiple of 10^{-3} even in the large scale environment. Compared to HAQL, Greedy-1 and Greedy-2 our strategy reduces the rejection rate by 2.4X, 109X and 60X, respectively, as shown in Fig. 4(b).

In our environment, QoE is inversely proportional to the round-trip time (RTT) of the job. This means, smaller RTT of a job will induce better QoE for the users. With that remark, Fig. 4(c) depicts 120% and 389% gain on average in QoE of HRL-Edge-Cloud compared to HAQL and Greedy-1, respectively. To be noticed, Greedy-2 outperforms all other strategies in view of QoE. This is because Greedy-2 always allocates maximum possible wireless bandwidth and VM units to all the jobs if available, which gives rise to the QoE gain. Notice that the best QoE of Greedy-2 comes at the expense of high operational cost and rejection rate as shown in Figs. 4(a) and 4(b) respectively.

As shown in Fig. 4(d), Greedy-1 and Greedy-2 outperform in runtime because both provide with fixed allocation and do not require any decision making time. Compared to HAQL, HRL-Edge-cloud achieves 240X reduction in runtime on average all due to the proposed pruning principle.

5.4 Stability and Adaptiveness of Our Proposed Strategy

The stability of our proposed strategy is evaluated by conducting experiments to show the balanced distribution of the load to all EC/BC servers and wireless base stations. In Fig. 5(a-c), the vertical axis describes the utility rate, which is given by Eqs. (7, 8 and 9), for all base stations and servers in small-scale environments under different numbers of jobs. We show that the proposed pruning principle helps to evenly distribute the load among

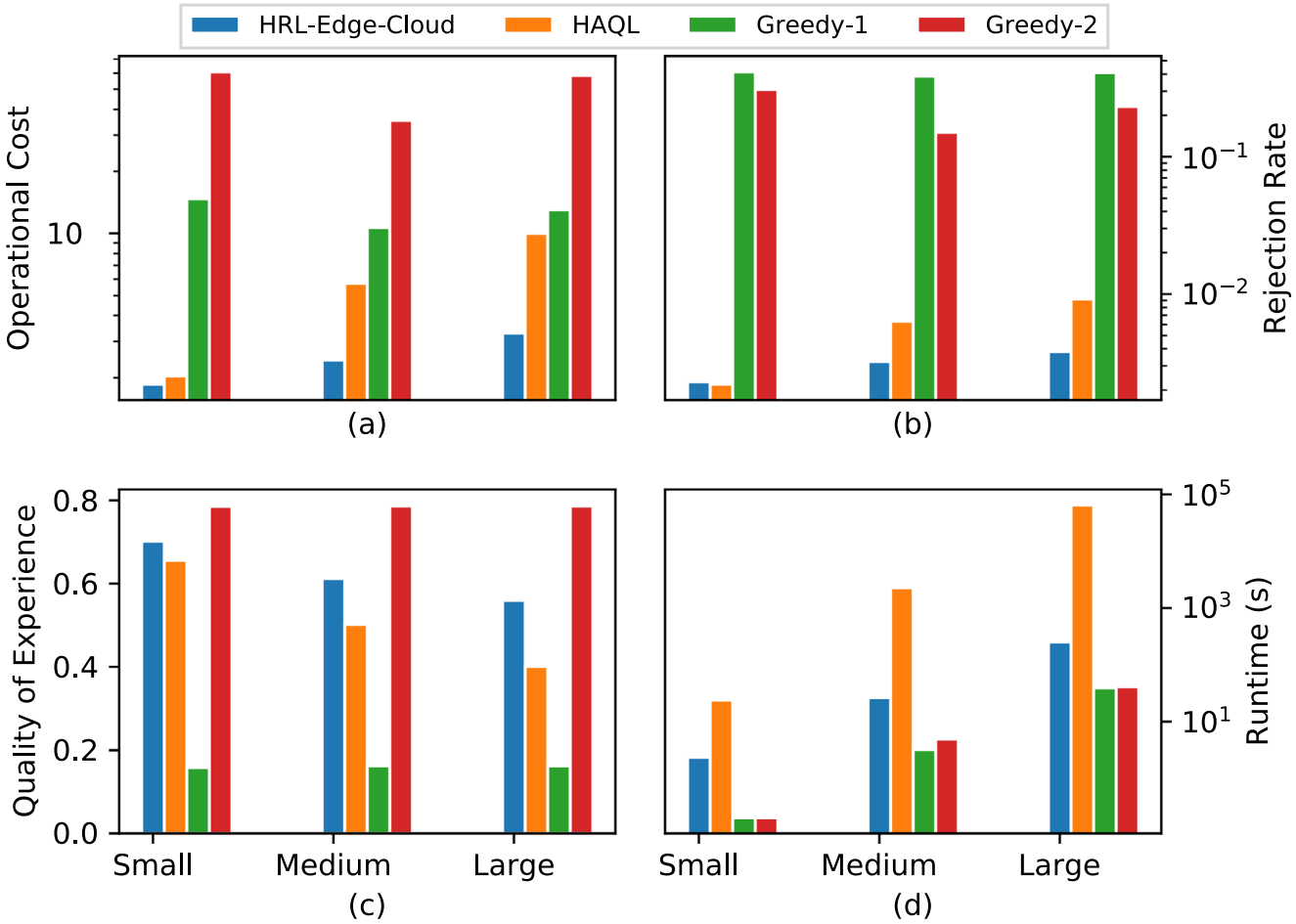

Fig. 4 Performance comparisons of proposed strategy with HAQL, Greedy-1 and Greedy-2 baselines. All four evaluations are conducted on small, medium and large scale environments. In view of (a) Operational cost, (b) Rejection rate, (c) Quality of Experience and (d) Runtime; vertical axes except (c) are in log scale

all servers and base stations and tries to keep a fair balance.

First, we notice that the overall utility rate of BC servers is lower than the EC servers. This phenomenon shows that, if resources are available, our strategy prefers to offload more jobs to the EC servers to induce less delay and contribute to better QoE as discussed above. Second, the graph curves in Fig. 5(a-c) illustrate the stability of both strategies. The utility of resources in the case of HRL-Edge-Cloud is much smoother, whereas the utility rate in the case of HAQL is fluctuating in all three graphs. This balanced distribution of the resources shows the stability of our strategy within the dynamics of a practical environment. Furthermore, as shown in Fig. 4(b), the balanced distribution of load aids in lowering the overall rejection rate.

To evaluate the adaptability of our strategy in a dynamic environment, we traced 100 jobs to observe how VM, wireless channels, and cloud are dynamically allocated for Type-1 and Type-2 jobs (separately). Firstly, it can be seen from the Fig. 6 (a and b) that Type-1 jobs are allocated 1 VM and 1 bandwidth unit in most cases. However, during the experiments, it was observed that some of the Type-1 tasks, which have critical deadlines (around 200 ms) and relatively larger amounts of data (around 400 KB) among other Type-1 tasks, are allocated higher VM and bandwidth units to complete the job within the deadline.

Secondly, Type-2 tasks are allocated higher VM and bandwidth units for faster execution in most cases. However, for some of the Type-2 tasks, which have relaxed deadlines (around 1700 ms) and a smaller amount of

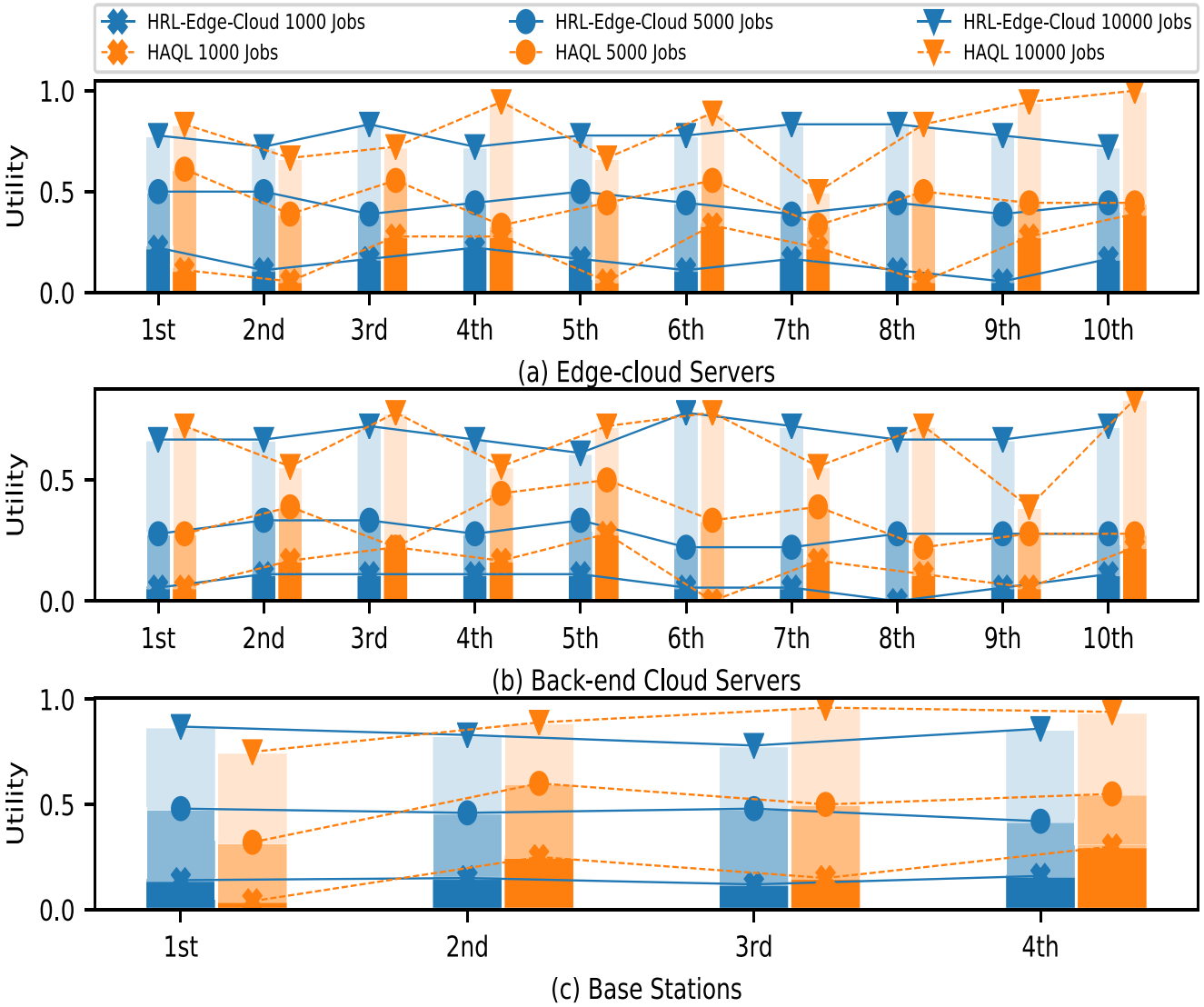

Fig. 5 Stability comparison of HRL-Edge-Cloud with HAQL for (a) Edge-Cloud servers utility, (b) Back-end Cloud Servers utility and (c) Base Stations utility in small scale environment

data (around 1 MB) than other Type-2 tasks, our strategy allocates fewer resources.

Lastly, it is shown in Fig. 6 (c) that Type-1 jobs are mostly offloaded to the EC to meet the deadline of the critical tasks. However, Type-1 jobs are also offloaded to the BC when the deadline is not critical and the amount of data to be processed is relatively smaller. Contrarily, the Type-2 tasks, which have relaxed deadlines, are mostly offloaded to the BC to conserve resources at the EC to execute future critical tasks and provide better QoS to the real-time applications.

To summarize, our proposed strategy adaptively determines the dynamic selection of VMs, wireless bandwidth channels, and cloud for joint and stable optimization of resources in the EC and BC even in a non-stationary environment.

6 Conclusion

This paper optimizes the problem of compute and wireless resource allocation for the IoT and mobile users in a smart streetscape based edge-cloud system jointly with the back-end cloud. To address this problem, we present a novel multi-resource allocation (MRA) framework by introducing HRL-Edge-Cloud, a heuristic reinforcement learning-based algorithm with linear annealing and a

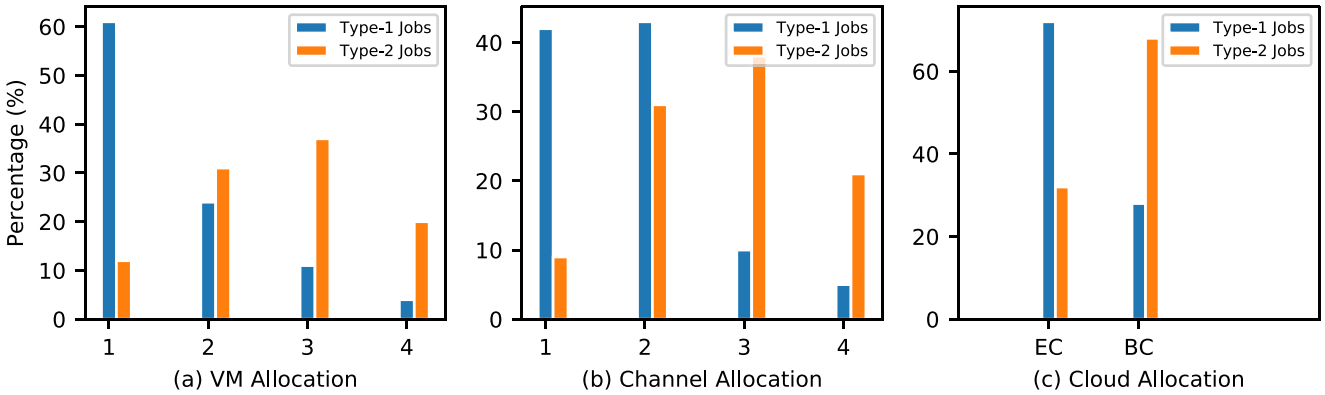

Fig. 6 Adaptive resource allocation. (a) VM allocation, (b) Channel allocation and (c) Cloud allocation for Type-1 and Type-2 jobs

pruning principle. In the proposed MRA framework, we consider the deadline and amount of data of individual jobs to replicate a near real-world environment. The proposed strategy is faster in convergence speed, highly scalable, stable and adaptive to the dynamics of a practical environment. To show the effectiveness of our strategy we develop the proposed framework in Python and conduct extensive experimentation at different scales. As compared to an existing algorithm, our proposed technique achieves 2.34X, 95X and 240X reduction in operational cost, convergence time and runtime on average, respectively, while maintaining the lower rejection rate and ensuring the Quality of Experience (QoE) for the users and applications.

The proposed framework addresses the multi-resource allocation problem in a single streetscape-based edge-cloud environment. A smart city consists of many streetscapes. Therefore, a captivating future direction is to enhance our framework across multiple streetscapes and investigate on multi-agent reinforcement learning-based algorithms to solve the resource allocation problem in a distributed and larger city-scale environment.

Acknowledgements We thank Colin Skow (colinskow@gmail.com) for discussions and help with the world formulation.

Author Contributions Both authors contributed to conceive the design of this study. The system model, problem formulation and algorithm design were carried out by Arslan Qadeer and analyzed by Myung J. Lee. The first draft of the manuscript was completed by Arslan Qadeer. Myung J. Lee read and approved the final manuscript.

Funding This work is supported in part by the National Science Foundation (NSF) IRNC COSMIC (#2029295) and NSF PAWR COSMOS (#1827923).

Availability of data and materials The authors confirm that all the data including numerical results captured and analyzed during the validation of this study are included in this article.

Code Availability The simulation code will be publicly available after the community release of COSMOS testbed. However, the code can be provided by the corresponding author on reasonable request.

Declarations

Conflict of Interests The authors declare no conflict of interests.

References

Agbali, M., Trillo, C., Fernando, T., Oyedele, L., Ibrahim, I. A., & Olatunji, V.O. (2019). Towards a refined conceptual framework model for a smart and sustainable city assessment. In *2019 IEEE international smart cities conference (ISC2), pp. 658–664*. https://doi.org/10.1109/ISC246665.2019.9071697.

Arno, H. V., & Mazur, M. (2021). Telecom infrastructure the open source way. Technical report, Ubuntu, Canonical, United Kingdom.

Baena, B., Cobian, C., Larios, V. M., Orizaga, J. A., Maciel, R., Cisneros, M. P., & Beltran-Ramirez, J.R. (2020). Adapting food supply chains in smart cities to address the impacts of COVID19 a case study from Guadalajara metropolitan area. In *2020 IEEE international smart cities conference (ISC2), pp. 1–8*. https://doi.org/10.1109/ISC251055.2020.9239076.

Barakabitze, A. A., Barman, N., Ahmad, A., Zadtootaghaj, S., Sun, L., Martini, M. G., & Atzori, L. (2020). Qoe management of multimedia streaming services in future networks: a tutorial and survey. *IEEE Communications Surveys Tutorials*, *2*(1), 526–565. https://doi.org/10.1109/COMST.2019.2958784.

Bi, Y., Colman-Meixner, C., Wang, R., Meng, F., Nejabati, R., & Simeonidou, D. (2019). Resource allocation for ultra-low latency virtual network services in hierarchical 5G network. In *ICC 2019 - 2019 IEEE international conference on communications (ICC), pp. 1–7*. https://doi.org/10.1109/ICC.2019.8761272.

Bianchi, R. A. C., Ribeiro, C. H. C., & Costa, A.H.R. (2008). Accelerating autonomous learning by using heuristic selection of actions. *Journal of Heuristics*, *14*, 135–168. https://doi.org/10.1007/s10732-007-9031-5.

Chen, J., Xing, H., Xiao, Z., Xu, L., & Tao, T. (2021). A DRL agent for jointly optimizing computation offloading and resource allocation in MEC. IEEE Internet of Things Journal, pp. 1–1. https://doi.org/10.1109/JIOT.2021.3081694.

Chen, H., Yuan, L., & Jing, G. (2020). 5G boosting smart cities development. In *2020 2nd International conference on artificial intelligence and advanced manufacture (AIAM), pp. 154–157*. https://doi.org/10.1109/AIAM50918.2020.00038.

Cheng, M., Li, J., & Nazarian, S. (2018). DRL-cloud: deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers. In *2018 23rd Asia and*

south pacific design automation conference (ASP-DAC), pp. 129–134. https://doi.org/10.1109/ASPDAC.2018.8297294.

Dong, C., & Wen, W. (2019). Joint optimization for task offloading in edge computing: an evolutionary game approach sensors, vol. 19(3). https://doi.org/10.3390/s19030740.

Elhassan, R., Ahmed, M. A., & AbdAlhalem, R. (2019). Smart waste management system for crowded area : makkah and holy sites as a model. In *2019 4th MEC international conference on big data and smart city (ICBDSC), pp. 1–5*. https://doi.org/10.1109/ICBDSC.2019.8645576.

Fang, M., Li, H., & Zhang, X. (2012). A heuristic reinforcement learning based on state backtracking method. In *2012 IEEE/WIC/ACM international conferences on web intelligence and intelligent agent technology, vol. 1, pp. 673–678*. https://doi.org/10.1109/WI-IAT.2012.187.

Feriani, A., & Hossain, E. (2021). Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: a tutorial. *IEEE Communications Surveys Tutorials*, *23*(2), 1226–1252. https://doi.org/10.1109/COMST.2021.3063822.

Gong, S., Yin, B., Zhu, W., & Cai, K.-Y. (2017). Adaptive resource allocation of multiple servers for service-based systems in cloud computing. In *2017 IEEE 41st annual computer software and applications conference (COMPSAC), vol. 2, pp. 603–608*. https://doi.org/10.1109/COMPSAC.2017.43.

Habibi, M. A., Nasimi, M., Han, B., & Schotten, H.D. (2019). A comprehensive survey of RAN architectures toward 5G mobile communication system. *IEEE Access*, *7*, 70371–70421. https://doi.org/10.1109/ACCESS.2019.2919657.

JošCsilo, S., & Dán, G. (2018). Joint allocation of computing and wireless resources to autonomous devices in mobile edge computing. In *Proceedings of the 2018 workshop on mobile edge communications. MECOMM'18, pp. 13–18. Association for computing machinery*. https://doi.org/10.1145/3229556.3229559.

Lei, L., Tan, Y., Zheng, K., Liu, S., Zhang, K., & Shen, X. (2020). Deep reinforcement learning for autonomous internet of things: model, Applications and Challenges. *IEEE Communications Surveys Tutorials*, *22*(3), 1722–1760. https://doi.org/10.1109/COMST.2020.2988367.

Li, S., Zhang, N., Lin, S., Kong, L., Katangur, A., Khan, M. K., Ni, M., & Zhu, G. (2018). Joint admission control and resource allocation in edge computing for internet of things. *IEEE Network*, *32*(1), 72–79. https://doi.org/10.1109/MNET.2018.1700163.

Liu, Y., Lee, M. J., & Zheng, Y. (2016). Adaptive multi-resource allocation for cloudlet-based mobile cloud computing system. *IEEE Transactions on Mobile Computing*, *15*(10), 2398–2410. https://doi.org/10.1109/TMC.2015.2504091.

Liu, Y., Peng, M., Shou, G., Chen, Y., & Chen, S. (2020). Toward edge intelligence: multiaccess edge computing for 5G and internet of things. *IEEE Internet of Things Journal*, *7*(8), 6722–6747. https://doi.org/10.1109/JIOT.2020.3004500.

Mnih, V., Kavukcuoglu, K., & Silver, D.E.A. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*, 529–533. https://doi.org/10.1038/nature14236.

Nath, S., & Wu, J. (2020). Dynamic computation offloading and resource allocation for multi-user mobile edge computing. In *GLOBECOM 2020 - 2020 IEEE global communications conference, pp. 1–6*. https://doi.org/10.1109/GLOBECOM42002.2020.9348161.

Ning, Z., Kong, X., Xia, F., Hou, W., & Wang, X. (2019). Green and sustainable cloud of things: enabling collaborative edge computing. *IEEE Communications Magazine*, *57*(1), 72–78. https://doi.org/10.1109/MCOM.2018.1700895.

Open, F. (2020). Networking AETHER: private 4G/5G connected edge platform for enterprises. Technical report, open networking foundation, California USA.

Peng, H., & Shen, X. (2020). Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks. *IEEE Transactions on Network Science and Engineering*, *7*(4), 2416–2428. https://doi.org/10.1109/TNSE.2020.2978856.

Qadeer, A., Lee, M. J., & Tsukamoto, K. (2020). Real-time multi-resource allocation via a structured policy table. In L. Barolli, H. nishino, & H. Miwa (Eds.) *Advances in intelligent networking and collaborative systems, pp. 370–379. Springer, Cham*.

Qadeer, A., Lee, M. J., & Tsukamoto, K. (2021). Flow-level dynamic bandwidth allocation in SDN-enabled edge cloud using heuristic reinforcement learning. In *2021 8th International conference on future internet of things and cloud (ficloud), pp. 1–10*. https://doi.org/10.1109/FiCloud49777.2021.00009.

Raychaudhuri, D., Seskar, I., Zussman, G., Korakis, T., Kilper, D., Chen, T., Kolodziejski, J., Sherman, M., Kostic, Z., Gu, X., Krishnaswamy, H., Maheshwari, S., Skrimponis, P., & Gutterman, C. (2020). Challenge: COSMOS: a city-scale programmable testbed for experimentation with advanced wireless. In *Proceedings of the 26th annual international conference on mobile computing and networking. MobiCom '20. Association for computing machinery*. https://doi.org/10.1145/3372224.3380891.

Robberechts, J., Sinaeepourfard, A., Goethals, T., & Volckaert, B. (2020). A novel edge-to-cloud-as-a-service (E2CaaS) model for building software services in smart cities. In *2020 21st IEEE international conference on mobile data management (MDM), pp. 365–370*. https://doi.org/10.1109/MDM48529.2020.00079.

Safitri, W. D., Ikhwan, M., Firmansyah, D., Rusdiana, S., Rahayu, L., & Akhdansyah, T. (2020). Partial distance strategy analysis on city characteristics to improve reliable smart cities services. In *3rd Smart cities symposium (SCS 2020), vol. 2020, pp. 354–358*. https://doi.org/10.1049/icp.2021.0933.

Sharma, S. K., & Wang, X. (2020). Toward massive machine type communications in ultra-dense cellular IoT networks: current issues and machine learning-assisted solutions. *IEEE Communications Surveys Tutorials*, *22*(1), 426–471. https://doi.org/10.1109/COMST.2019.2916177.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: an introduction, sec. edn.* Cambridge: The MIT Press. http://incompleteideas.net/sutton/book/ebook/the-book.html.

Tran, T. X., & Pompili, D. (2019). Joint task offloading and resource allocation for multi-server mobile-edge computing networks. *IEEE Transactions on Vehicular Technology*, *68*(1), 856–868. https://doi.org/10.1109/TVT.2018.2881191.

Ungureanu, O.-M., Vlădeanu, C., & Kooij, R. (2021). Collaborative cloud - edge: a declarative API orchestration model for the NextGen 5G core. In *2021 IEEE international conference on service-oriented system engineering (SOSE), pp. 124–133*. https://doi.org/10.1109/SOSE52839.2021.00019.

Wang, X., Han, Y., Leung, V. C. M., Niyato, D., Yan, X., & Chen, X. (2020). Convergence of edge computing and deep learning: a comprehensive survey. *IEEE Communications Surveys Tutorials*, *22*(2), 869–904. https://doi.org/10.1109/COMST.2020.2970550.

Wang, P., Yang, L. T., & Li, J. (2018). An edge cloud-assisted CPSS framework for smart city. *IEEE Cloud Computing*, *5*(5), 37–46. https://doi.org/10.1109/MCC.2018.053711665.

Wei, Y., Yu, F. R., Song, M., & Han, Z. (2019). Joint optimization of caching, computing, and radio resources for fog-

enabled IoT using natural actor–critic deep reinforcement learning. *IEEE Internet of Things Journal*, *6*(2), 2061–2073. https://doi.org/10.1109/JIOT.2018.2878435.

Xue, J., & An, Y. (2021). Joint task offloading and resource allocation for multi-task multi-server NOMA-MEC networks. *IEEE Access*, *9*, 16152–16163. https://doi.org/10.1109/ACCESS.2021.3049883.

Yang, J., Kwon, Y., & Kim, D. (2021). Regional smart city development focus: the south korean national strategic smart city program. *IEEE Access*, *9*, 7193–7210. https://doi.org/10.1109/ACCESS.2020.3047139.

Zheng, J., Cai, Y., Wu, Y., & Shen, X. (2019). Dynamic computation offloading for mobile cloud computing: a stochastic game-theoretic approach. *IEEE Transactions on Mobile Computing*, *18*(4), 771–786. https://doi.org/10.1109/TMC.2018.2847337.

Arslan Qadeer received the BS degree in 2013 from University of the Punjab and the MS degree in 2017 from National University of Sciences and Technology in Pakistan. From 2015 to 2018, he worked in the telecommunications and software-defined networking industries. He is currently working toward a PhD degree in the Department of Electrical Engineering at the City College of City University of New York. His current research interests are in the area of resource allocation and management in wireless networks, software-defined networks, and edge-cloud computing using machine learning.

Dr. Myung Jong Lee received a B.S and an M.S from Seoul National University in Korea and PhD from Columbia University. He is currently a professor of the department of Electrical Engineering at the City College of City University of New York. His current research interests include multi-resource allocation for mobile/edge cloud computing, deep learning for communications, secure V2X communications, stochastic computing applications, and wireless testbed. He publishes extensively in these areas and 25 U.S & International Patents, and numerous contributions to IEEE standards. Dr. Lee actively contributed to international standard organizations as TG chairs of IEEE 802.15.8 (Peer Aware Communications) and IEEE 802.15.5 (WPAN Mesh). He received the best paper awards from IEEE CCNC 2005 and 2016 EAI SmartGrid, and InCos 2019, also CUNY Performance Excellence Award.