Software-Defined Networks with Mobile Edge Computing and Caching for Smart Cities: A Big Data Deep Reinforcement Learning Approach

Article *in* IEEE Communications Magazine · December 2017

DOI: 10.1109/MCOM.2017.1700246

CITATIONS
29

READS
2,258

5 authors, including:

Ying He
Dalian University of Technology
26 PUBLICATIONS **291** CITATIONS
SEE PROFILE

F. Richard Yu
Carleton University
640 PUBLICATIONS **14,444** CITATIONS
SEE PROFILE

Nan Zhao
Dalian University of Technology
181 PUBLICATIONS **2,099** CITATIONS
SEE PROFILE

Victor C. M. Leung
University of British Columbia - Vancouver
1,318 PUBLICATIONS **19,915** CITATIONS
SEE PROFILE

Software-Defined Networks with Mobile Edge Computing and Caching for Smart Cities: A Big Data Deep Reinforcement Learning Approach

Ying He, F. Richard Yu, Nan Zhao, Victor C.M. Leung, and Hongxi Yin

ABSTRACT

Recent advances in networking, caching, and computing have significant impacts on the developments of smart cities. Nevertheless, these important enabling technologies have traditionally been studied separately in the existing works on smart cities. In this article, we propose an integrated framework that can enable dynamic orchestration of networking, caching, and computing resources to improve the performance of applications for smart cities. Then we present a novel big data deep reinforcement learning approach. Simulation results with different system parameters are presented to show the effectiveness of the proposed scheme.

INTRODUCTION

In recent years, with the great progress of urbanization, *smart cities* are emerging as a priority for research and development across the world. Using advanced technologies, smart cities are expected to provide their citizens with improved quality of life and a variety of innovative services applications, including transportation, education, healthcare, energy, waste management, tourism, and so on [1, 2].

The development of smart cities are heavily influenced by information and communications technology (ICT). It is through ICT that smart cities are truly turning smart. Recent advances in ICT have fueled a plethora of innovations in various areas, including *networking*, *caching*, and *computing* [1, 3–5]. These three areas have been extensively studied in the development of smart cities. In the area of networking, software-defined networking (SDN) has attracted great interest in both academia and industry. The basic principle of SDN is to separate the control plane from the data plane, enabling the ability to program the network via a centralized software-defined controller with a global view of the network. Another related concept is network functions virtualization (NFV), which enables abstraction of physical networking resources and flexible sharing of resources by multiple users through isolating each other [6]. It is beneficial to apply SDN and NFV to the networking infrastructure of smart cities [7].

Another promising technology, *in-network caching*, as one of the key features of information-centric networking (ICN), can efficiently reduce duplicate content transmission in networks. Particularly, caching content (e.g., videos) at network edge nodes, such as base stations (BSs), roadside units (RSUs), and access points (APs), has been proposed as one of the key enablers in the services of smart cities [8]. Investigation on exploiting caching in the applications for smart cities has shown that access delays, traffic loads, and network costs can be significantly reduced by caching contents in networks.

Recent advances in computing (e.g., cloud/fog/edge computing) will have profound impacts on smart cities as well [4]. *Cloud computing* has become very popular to enable access to a shared pool of computing resources. Nevertheless, as the distance between the cloud and the end device is usually large, cloud computing services may not provide guarantees to low-latency applications for smart cities. To address these issues, *fog computing* and mobile edge computing (MEC) [9] have been studied to deploy computing resources closer to end devices, which can efficiently improve the quality of service (QoS) for applications that require intensive computations and low latency [4].

While some excellent works have been done on networking, caching, and computing for smart cities, these three important enabling technologies have traditionally been studied separately in the existing works on smart cities. In this article, we jointly consider networking, caching, and computing to enhance the performance of applications for smart cities. Specifically, we propose an integrated framework that can enable dynamic orchestration of networking, caching, and computing resources to improve the performance of applications for smart cities. In this framework, we formulate the resource allocation strategy as a joint optimization problem, where the gains of not only networking but also cach-

The authors propose an integrated framework that can enable dynamic orchestration of networking, caching, and computing resources to improve the performance of applications for smart cities, and they present a novel big data deep reinforcement learning approach. Simulation results with different system parameters are presented to show the effectiveness of the proposed scheme.

This research was supported in part by the Xinghai Scholars Program, the Fundamental Research Funds for the Central Universities under DUT17JC43, the National Natural Science Foundation of China (NSFC) under Grant 61671101, and Natural Sciences and Engineering Research Council of Canada (NSERC).

Ying He is with Dalian University of Technology and Carleton University; F. Richard Yu is with Carleton University; Nan Zhao and Hongxi Yin are with Dalian University of Technology; Victor C. M. Leung is with the University of British Columbia.

Digital Object Identifier:
10.1109/MCOM.2017.1700246

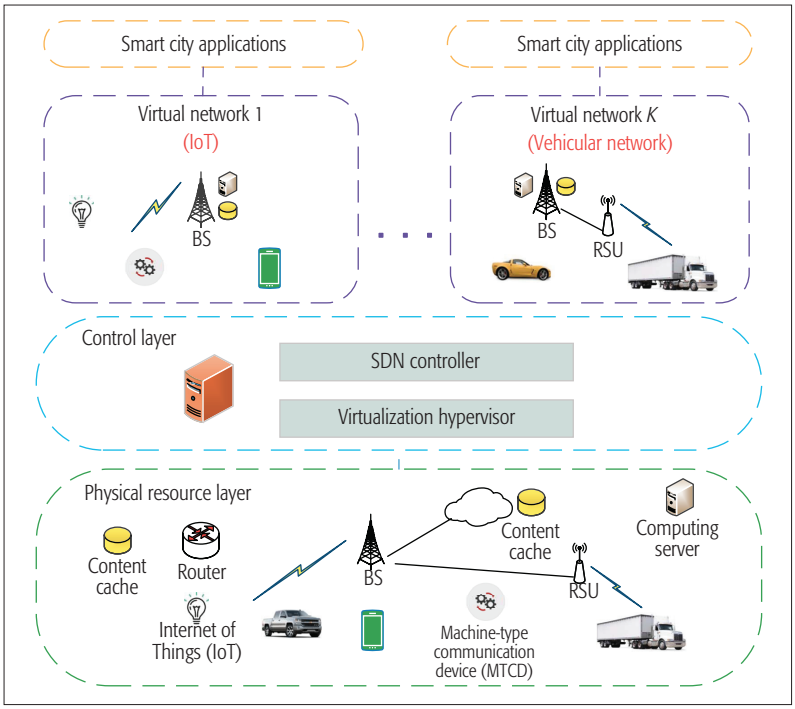

Figure 1. A framework of integrated networking, caching, and computing for smart cities.

ing and computing are taken into consideration. In addition, we propose a novel *big data deep reinforcement learning* approach in this article. Deep reinforcement learning is an advanced reinforcement learning algorithm that uses deep Q-network to approximate the Q value-action function [10]. Google DeepMind has adopted this method in several artificial intelligent projects with big data [10, 11], and has gotten quite good results. Deep reinforcement learning is used in this article to obtain the resource allocation policy in applications for smart cities with integrated networking, caching, and computing. Simulation results with different system parameters are presented to show the effectiveness of the proposed scheme.

The rest of this article is organized as follows. The system description is presented. The proposed framework of integrated networking, caching, and computing for smart cities is described. The deep reinforcement learning algorithm is introduced. A problem formulation is given. Simulation results are discussed. Finally, conclusions are presented.

System Description

In this section, we first introduce recent advances in software-defined and virtulaized networks. Then information-centric networking and mobile edge computing are introduced. Next, we present the proposed framework of integrated networking, caching, and computing for smart cities.

Software-Defined and Virtualized Networks

It is beneficial to extend SDN principles (e.g., flexibility, programmability, and centralized control) to manage networking and communication resources in wireless/wireline networks, for example, to optimize channel allocation and network selection and reduce interference in a multi-channel, multi-radio environment, to improve packet routing decisions in multi-hop environments, and to effectively handle mobility in high-speed scenarios.

In addition, wireless network virtualization has been considered as an effective and promising approach in the management of network architecture and network resources (e.g., spectrum and infrastructure) [6] With different quality of service (QoS), service functions, or network requirements, a common physical wireless network can be virtualized into several virtual ones by the hypervisor. Moreover, since virtualization enables the sharing of network infrastructure and resources, the capital expenses (CapEx) as well as operation expenses (OpEx) of wireless access networks can be reduced significantly.

Figure 1 shows a framework of integrated networking, caching, and computing with software-defined and virtualized networks for smart cities, where the infrastructure (i.e., the substrate network) is abstracted and virtualized into multiple customized virtual networks (energy, health, vehicular, IoT, etc.) with different QoS requirements. With this framework, a variety of smart city applications can be provided, including healthcare assistance, security and safety, intelligent transportation systems, traffic monitoring, waste management, environment management, and so on.

Information-Centric Smart Cities

In the smart city ecosystem, huge amounts of information-rich and safety-critical data will be exchanged by users, devices, and networks in a variety of smart city applications. Due to poor-quality wireless links and high mobility in some applications, it is challenging to deliver huge amounts of data using the traditional host-centric IP-based approach in smart cities [8]. Recent advances in ICN can be extended to traditional networks to address this issue. In-network caching, as one of the key features of ICN, can efficiently reduce duplicate content transmission in networks.

It is beneficial to extend this innovative ICN approach to smart cities, which natively privilege the information (e.g., trusted road traffic information in a specific proximity of a hazard and a specific time period) rather than the node identity. In addition, with in-network caching, ICN helps to address the mobility and sporadic connectivity issues in smart cities.

Smart Cities with Mobile Edge Computing

Recently, cloud computing has been applied in diverse domains, where user data needs to be transmitted to and processed in the data centers. However, since data centers are usually far away from the end users, cloud computing services may not provide guarantees to low-latency applications for smart cities, and transmitting a large amount of data (e.g., in big data analytics [12]) from the device to the cloud may not be feasible or economical. To address these issues, *mobile edge computing* has been proposed to deploy computing resources closer to end devices, which can efficiently improve the QoS for applications that require intensive computations and low latency [13]. Low latency and location

awareness are two obvious characteristics of fog computing. Similar to the concept of fog computing, MEC has recently been proposed, particularly for radio access networks (RANs) [9], in close proximity to mobile devices. It is defined as running IT-based servers (called MEC servers) at the edge of the RAN, which apply the concept of cloud computing in network edge nodes, to facilitate services and applications for smart cities [1, 4, 14].

Proposed Framework of Integrated Networking, Caching and Computing for Smart Cities

In this section, we first present the proposed framework of integrated networking, caching, and computing for smart cities, followed by some smart city use cases using the proposed framework.

Framework

In most existing works on smart cities, networking, caching, and computing are studied separately. However, from the smart city application's point of view, network, cache, and compute are underlying resources enabling smart city applications. How to abstract, allocate, and optimize these resources can have significant impacts on the performance of smart city applications. In this article, we propose a novel framework of integrated networking, caching, and computing in a systematic way for smart cities. Figure 1 shows this framework. Based on the programmable control principle originating from SDN, we incorporate the ideas of information centricity originating from ICN. This integrated framework can enable dynamic orchestration of networking, caching, and computing resources to meet the requirements of different smart city applications.

In the proposed framework, through wireless network virtualization, the physical wireless network can be virtualized into several virtual ones by the hypervisor shown in Fig. 1. Virtual networks are established logically. Each virtual network includes BSs, RSUs, APs, MEC servers, and content caches, which can provide the capabilities of content caching and high-speed computing for smart city applications. Each device will eventually access one of them based on its QoS requirements. As such, multiple virtual networks with different network services could be embodied in the same platform. The virtualization hypervisor in Fig. 1 is an important component. In general, the hypervisor is responsible for slicing and scheduling resources. In addition, the hypervisor allocates the virtual resources to different mobile virtual network operators (MVNOs).

Users access the wireless network. The SDN controller can manage the network nodes equipped with caching and computing capacities. The caching resource is used for caching massive contents including the popular contents requested by users and the data collected by users. With the caching capability, when users want to access contents or some smart city applications want to process the data, the contents may hit the in-network cache. Thus, this approach will decrease latency and alleviate traffic redundancy and in-network burden. In addition, integrating computing resource is very desirable for processing the massive contents or data, reducing transmission delay greatly and guaranteeing the timeliness of applications. Thus, the proposed framework not only provides the essential networking capability, but also caching capability and computing capability.

Smart City Use Cases

One example use case in this framework is as follows. Assume that Joe visits a city and finds an interesting monument. With his augmented reality (AR) glasses, Joe notices that there is an introductory video about this monument. Then he issues a video content request via his AR glasses to a virtual BS (could be a BS, AP, or RSU in the physical wireless network). According to the description of the video content and the information about the user, the virtual BS will check whether or not its associated cache has the requested content. If yes, the cache will further examine if the version of its cached video content can be played on and matches the user's device. If still yes, the virtual BS will directly send the requested content to the user from the cache. If no, the virtual BS will extract the current video content and construct a computation task according to the size of the content involving the input data, codes, and parameters, as well as the number of CPU cycles needed to accomplish the computing/transcoding task. Then the virtual BS will transmit the task to the MEC server to execute the computation. After the computation is finished, the virtual BS sends the transcoded video content to the user. If the cache cannot find the matched video content, the virtual BS has to retrieve the content from the Internet, which will inevitably occupy some of the backhaul resources. The procedure is shown in Fig. 2.

The above use case is for downlink streaming video. Another similar use case for uplink streaming video can be considered as follows. When Joe finds an interesting monument, he takes a video of it, puts some comments on it, and streams it to a remote server via a virtual BS. The virtual BS extracts the video content, and estimates the possibility that other people want to see this video. If the possibility is high, the virtual BS will cache this video locally so that other people who are interested in this video can obtain this video from its cache directly, instead of transmitting it from the remote server to save backhaul resources and reduce latency. If the cached version does not match the new user's device, transcoding will be performed by the MEC server.

While the mentioned use cases focus on tourism services, similar use cases with similar requirements can be derived for other services and applications, such as transportation, education, and healthcare.

When we consider networking, caching, and computing jointly, the complexity of the system is very high, which makes it difficult to solve the resource allocation problem using traditional approaches. In this article, we use a novel deep reinforcement learning approach to solve the optimization problem in smart cities with integrated networking, caching, and computing. In the following section, we describe recent advances in deep reinforcement learning.

Integrating computing resource is very desirable for processing the massive contents or data, reducing transmission delay greatly and guaranteeing the timeliness of applications. Thus, the proposed framework not only provides the essential networking capability, but also caching capability and computing capability.

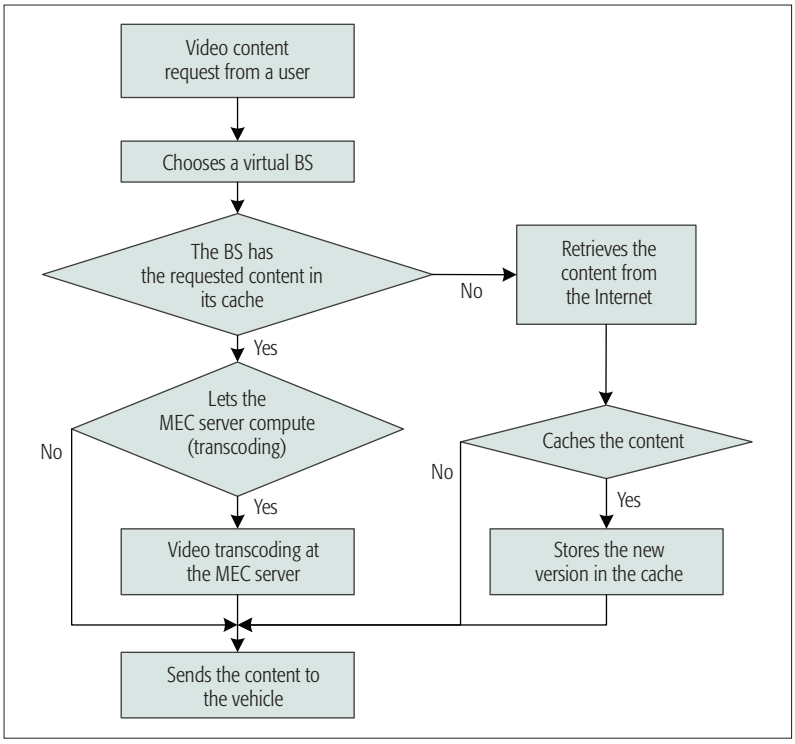

Figure 2. The procedure of tourism streaming services in a smart city network with mobile edge computing and caching.

Deep Reinforcement Learning

In order to better understand a state-of-the-art reinforcement learning agent, Deep *Q*-Network (DQN), a brief review of reinforcement learning and *Q*-learning are first described. Then recent advances in DQN are presented.

Reinforcement Learning

Reinforcement learning is an important branch of machine learning, where an agent learns to take actions that would yield the most reward by interacting with the environment. Different from supervised learning, reinforcement learning cannot learn from samples provided by an experienced external supervisor. Instead, it has to operate based on its own experience despite facing significant uncertainty about the environment. Reinforcement learning is defined not by characterizing learning methods, but by characterizing a learning problem. Any method that is suitable for solving that problem can be considered as a reinforcement learning method. A reinforcement learning problem can be described as optimal control of a Markov decision process (MDP); however, state space, explicit transition probability, and reward function are not necessarily required. Therefore, reinforcement learning is powerful in handling tough situations that approach real-world complexity [10].

Deep Q-Learning

The agent that uses a neural network to represent *Q*-function is called the *Q*-network, which is denoted as $Q(x, a; \theta)$. The parameter θ stands for the weights of the neural network, and the *Q*-network is trained by updating θ at each iteration to approximate the real *Q* values. While neural networks allow for great flexibility, they do so at the cost of stability when it comes to *Q*-Learning, which is interpreted in [10]. A deep *Q*-network that uses deep neural networks instead of approximating the *Q*-function was proposed recently, and it is proven to be more advantageous with greater performance and more robust learning [10]. To transform an ordinary *Q*-network into a deep *Q*-network, three improvements have been implemented:

- Replacing ordinary neural networks with advanced multi-layer deep convolutional networks, which utilize hierarchical layers of tiled convolutional filters to exploit the local spatial correlations, and make it possible to extract high-level features from raw input data.
- Utilizing experience replay, which stores its interaction experience tuple $e(t) = (x(t), a(t), r(t), x(t + 1))$ at time t into a replay memory $D(t) = \{e(1), ..., e(t)\}$, and then randomly samples batches from the experience pool to train the deep convolutional network's parameters rather than directly using the consecutive samples as in *Q*-learning. This allows the network to learn from more various past experiences, and restrains the network from only focusing on what it is immediately doing.
- Adopting a second network to generate the target *Q* values that are used to calculate the loss for each action during the training procedure. One network for both estimations of *Q* values and target *Q* values would result in falling into feedback loops between the target and estimated values. Thus, in order to stabilize the training, the target network's weights are fixed and periodically updated.

Problem Formulation

In this section, we formulate the resource allocation optimization problem as a deep reinforcement learning process.

In our system model, there are K BSs, m MEC servers, and C content caches that are virtualized and managed by an MVNO and provide services for smart cities. Since we consider realistic scenarios, where the BSs' downlink channel conditions, the MEC servers' computation abilities, and the caches' states are all dynamically changing, the MVNO faces a large amount of system states, and it has to make a decision on which virtualized resources will be assigned to a specific user according to the system's current state. It is barely possible to solve this complicated task using a traditional method.

Deep *Q*-learning is a recent advance that is capable of receiving complex high-dimensional data as input, and yielding an optimal action for each input data. By taking advantage of a deep *Q*-network, the MVNO can manage the system in an effective and efficient way.

Here, the MVNO is responsible for collecting the status from each BS, MEC server, and content cache, and then assembling all the information into a system state. Then the MVNO sends the constructed system state to the agent, (i.e., the deep *Q*-network) and gets feedback of the optimal policy for arranging which resources for a certain user. After obtaining the action, the

MVNO will send a notice to inform the user of which virtualized network it can access.

Inside the deep Q-network, the ε-greedy policy is used to balance the exploration and exploitation. The deep Q agent tries to take a random action to explore the unknown environment with a probability ε, and it maximizes the reward based on the knowledge already obtained with a probability $1 - \varepsilon$. Since we introduce double DQN in our algorithm, σ represents the rate that updates the target network toward the primary network. The training algorithm of the deep Q network is described in Algorithm 1.

In order to obtain the optimal policy, it is necessary to identify the system states, actions, and reward functions in our deep Q learning model, which are described as follows. The state of an available BS $k \in \{1, 2, ..., K\}$, the state of an available MEC server $m \in \{1, 2, ..., M\}$, and the state of an available cache $c \in \{1, 2, ..., C\}$ for user u_s in time slot $t \in \{0, 1, ..., T - 1\}$ will be included in the system states. For actions, the agent has to decide which BS is assigned to the user, whether or not the requested content should be cached in the BS, and whether or not the computation task should be offloaded to the MEC server. In addition, the system reward is the MVNO's revenue, and it is formulated as the function of received signal-to-noise ratio (SNR) of the access wireless link, the computation capability, and the cache state. We set the comprehensive revenue of the MVNO as our system's reward. The MVNO rents the wireless spectrum and the backhaul bandwidth from information providers (InPs), and assigns them to virtual service providers (SPs). The MVNO needs to pay for the usage of spectrum to InPs. Furthermore, the MVNO also needs to pay the computation fee as long as a computation task is executed on the MEC server. Moreover, the MVNO charges the users for accessing to the virtual networks. The users who have already paid the fee can access the virtual network for offloading their computation task. The backhaul cost paid by the MVNO can be saved when users call for the contents that have already been cached.

Simulation Results and Discussions

In this section, we use computer simulations to show the performance of the proposed deep reinforcement learning approach to smart cities with integrated networking, caching, and computing. We use TensorFlow [15] in our simulations to implement deep reinforcement learning.

Figure 3 shows the convergence performance of different scenarios in the proposed scheme using the deep reinforcement learning algorithm. From Fig. 3, we can observe that the total utility of different scenarios in the proposed scheme is very low at the beginning of the learning process. With the increase of the number of episodes, the total utility increases until it reaches a relatively stable value, which is around 6600 in the proposed scheme with integrated networking, caching, and computing in Fig. 3. This shows the convergence performance of the proposed scheme. We can also observe that the total utility of other scenarios is less when networking, caching, and computing are not jointly considered. Particularly, the total utility is lowest when virtualization is not considered compared to other scenarios. Figure 4 shows the convergence performance of the proposed scheme with different learning rates in the deep reinforcement learning algorithm. We can observe from Fig. 4 that the learning rate in the AdamOptimizer has effects on the convergence performance in Fig. 4. Specifically, the convergence is faster when the learning rate is 0.001 compared to the case when the learning rate is 0.0001 and 0.00001. However, a larger learning rate will result in local optimum

Initialization.
  Initialize the experience replay buffer.
  Initialize the main deep-Q network with weights θ
  Initialize the target deep-Q network with weights $\theta^- = \theta$
For episode $k = 1, \ldots, K$ do:
  Receive the initial observation s_1, and pre-process s_1 to be the beginning state x_1.
  For $t = 1, 2, 3 \ldots, T$ do:
    Choose a random probability p.
    Choose a_t as,
      if $p \le \varepsilon$
        randomly select an action a_t,
      otherwise,
        $a_t = \arg\max_a Q(x, a; \theta)$.
    Execute action a_t in the system, and obtain the reward r_t, and the next observation s_{t+1}.
    Process s_{t+1} to be the next state x_{t+1}.
    Store the experience (x_t, a_t, r_t, x_{t+1}) into the experience replay buffer.
    Get a batch of U samples (x_i, a_i, r_i, x_{i+1}) from the replay memory.
    Calculate the target Q-value y_i^- from the target deep-Q network,

$$y_i^- = r_i + \varepsilon Q(x_{i+1}, \arg\max_{a'} Q(x_{i+1}, a'; \theta); \theta^-)$$

    Update the main deep Q-network by minimizing the loss $L(\theta)$,

$$L(\theta) = \frac{1}{U}\sum_i (y_i^- - Q(x_i, a_i; \theta))^2,$$

  and perform a gradient descent step on $L(\theta)$ with respect to θ
    Every G steps, update the target deep Q-network parameters with rate σ
      $\theta^- = \sigma\theta + (1 - \sigma)\theta^-$.
  End for
End for

Algorithm 1. Double-Dueling-Deep Q-network algorithm in the virtualized smart city network.

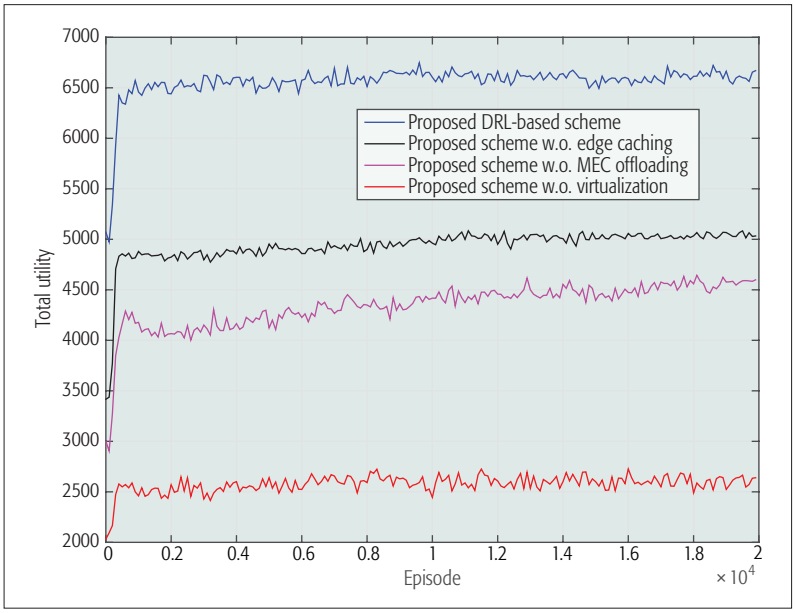

Figure 3. Convergence performance of different schemes.

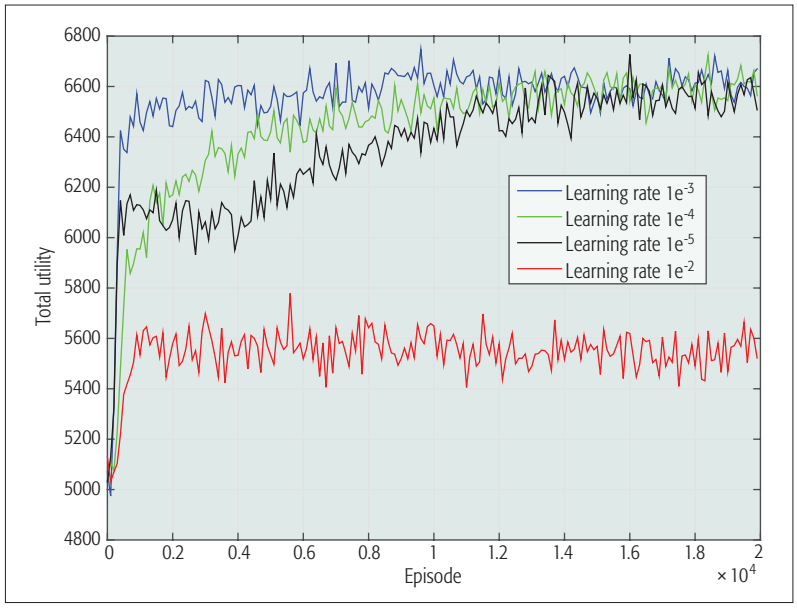

Figure 4. Convergence performance with different learning rates.

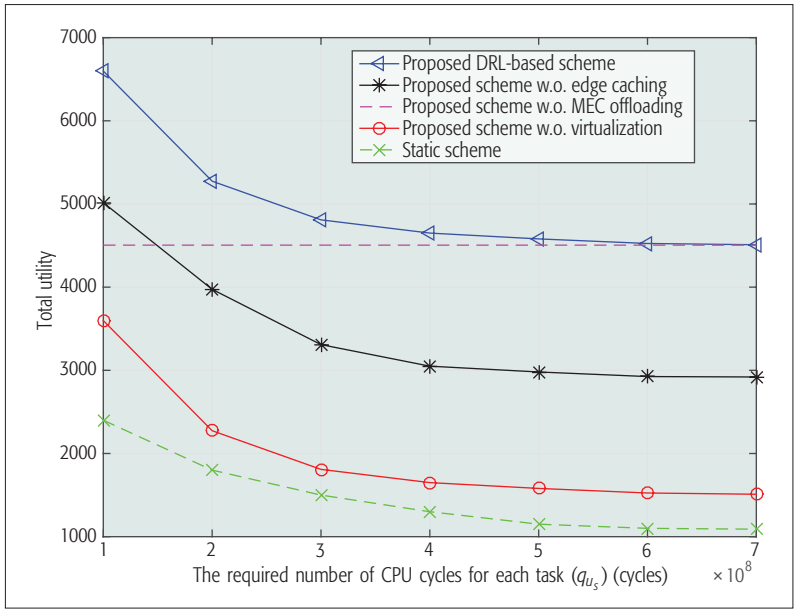

Figure 5. Effects of the required number of CPU cycles for each task.

instead of global optimum. As shown in Fig. 4, when the learning rate is 0.01, the algorithm converges to the total utility of around 5500, which is much lower than the converged values achieved by other learning rates. Therefore, an appropriate learning rate should be chosen for a specific problem. In fact, the best learning rate depends on the problem at hand, as well as on the architecture of the model being optimized, and even on the state of the model in the current optimization process. In the rest of the simulations, we choose the learning rate of 0.001. Figure 5 shows the effects of the required number of CPU cycles for each task. We can see from Fig. 5 that the required number of CPU cycles has no effect on the total utility of the proposed scheme without MEC offloading. This is because the total utility will not be changed with the different required number of CPU cycles if MEC offloading is not available in the proposed scheme. In other scenarios of the proposed scheme and the static scheme, the total utility decreases exponentially with the increase of the required number of CPU cycles for each task. This is because a larger number of required CPU cycles will induce higher consumed computation energy and lower gain of computation utility, and thus the total utility decreases.

Conclusions and Future Work

In this article, we study software-defined and virtualized networks with mobile edge computing and caching for smart cities. Based on the programmable control principle originating from SDN and caching originating from ICN, we present an integrated framework that can enable dynamic orchestration of networking, caching, and computing resources to improve the performance of applications for smart cities. In addition, we formulate the resource allocation strategy as a joint optimization problem, which is solved using a novel big data deep reinforcement learning approach in this article. The visualization of the deep *Q*-network model is presented. We present the convergence performance of different scenarios in the proposed scheme using the deep reinforcement learning algorithm. Simulation results with different system parameters are presented to show the effectiveness of the proposed scheme. Future work is in progress to consider energy efficiency issues in the proposed framework.

References

[1] D. Mazza, D. Tarchi, and G. E. Corazza, "A Unified Urban Mobile Cloud Computing Offloading Mechanism for Smart Cities," *IEEE Commun. Mag.*, vol. 55, no. 3, Mar. 2017, pp. 30–37.

[2] A. Al-Fuqaha *et al.*, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Commun. Surveys & Tutorials*, vol. 17, 4th qtr. 2015, pp. 2347–76.

[3] G. Han *et al.*, "HySense: A Hybrid Mobile Crowdsensing Framework for Sensing Opportunities Compensation Under Dynamic Coverage Constraint," *IEEE Commun. Mag.*, vol. 55, no. 3, Mar. 2017, pp. 93–99.

[4] T. Taleb *et al.*, "Mobile Edge Computing Potential in Making Cities Smarter," *IEEE Commun. Mag.*, vol. 55, no. 3, Mar. 2017, pp. 38–43.

[5] G. Han *et al.*, "Analysis of Energy-Efficient Connected Target Coverage Algorithms for Industrial Wireless Sensor Networks," *IEEE Trans. Industrial Informatics*, vol. 13, Feb. 2017, pp. 135–43.

[6] C. Liang and F. R. Yu, "Wireless Network Virtualization: A Survey, Some Research Issues and Challenges," *IEEE Commun. Surveys & Tutorials*, vol. 17, 1st qtr. 2015, pp. 358–80.

[7] R. Abhishek, S. Zhao, and D. Medhi, "SPArTaCuS: Service Priority Adaptiveness for Emergency Traffic In Smart Cities Using Software-Defined Networking," *Proc. IEEE Int'l. Smart Cities Conf.*, Sept. 2016, pp. 1–4.

[8] A. Shariat, A. Tizghadam, and A. Leon-Garcia, "An ICN-Based Publish-Subscribe Platform to Deliver UAV Service in Smart Cities," *Proc. IEEE INFOCOM Wksps.*, Apr. 2016, pp. 698–703.

[9] ETSI, "Mobile-Edge Computing — Introductory Technical White Paper," Sept. 2014.

[10] V. Mnih *et al.*, "Human-Level Control Through Deep Reinforcement Learning," *Nature*, vol. 518, no. 7540, 2015, pp. 529–33.

[11] D. Silver *et al.*, "Mastering the Game of Go with Deep Neural Networks and Tree Search," *Nature*, vol. 529, no. 7587, 2016, pp. 484–89.

[12] Y. He *et al.*, "Big Data Analytics in Mobile Cellular Networks," *IEEE Access*, vol. 4, Mar. 2016, pp. 1985–96.

[13] X. Hou *et al.*, "Vehicular Fog Computing: A Viewpoint of Vehicles as the Infrastructures," *IEEE Trans. Vehic. Tech.*, vol. 65, June 2016, pp. 3860–73.

[14] F. R. Y. C. Liang, Y. He, and N. Zhao, "Energy-Efficient Resource Allocation in Software-Defined Mobile Networks with Mobile Edge Computing and Caching," *Proc. IEEE INFOCOM '17 Wksps.*, May 2017.

[15] M. Abadi *et al.*, "Tensorflow: Large-Scale Machine Learning on Heterogeneous Systems," arXiv:1603.04467, Nov. 2015.

Biographies

Ying He [S'16] (heying@sce.carleton.ca) received her B.S. degree from Dalian Ocean University, China, and her M.S. degree from Dalian University of Technology in 2011 and 2015, respectively, both in communication and information systems. She is currently pursuing a Ph.D. degree with both Dalian University of Technology and Carleton University. Her current research interests include big data, wireless networks, and machine learning.

F. Richard Yu [S'00, M'04, SM'08] (richard.yu@carleton.ca) is a professor at Carleton University, Canada. His research interests include connected vehicles, security, and green ICT. He serves on the Editorial Boards of several journals, including Co-Editor-in-Chief for *Ad Hoc & Sensor Wireless Networks*, Lead Series Editor for *IEEE Transactions on Vehicular Technology*, *IEEE Transactions on Green Communications and Networking*, and *IEEE Communications Surveys & Tutorials*. He is a Distinguished Lecturer and a member of the Board of Governors of the IEEE Vehicular Technology Society.

Nan Zhao [S'08, M'11, SM'16] (zhaonan@dlut.edu.cn) is an associate professor at Dalian University of Technology. He received his B.S. degree in electronics and information engineering in 2005, his M.E. degree in signal and information processing in 2007, and his Ph.D. degree in information and communication engineering in 2011 from Harbin Institute of Technology, China. His research interests include interference alignment, cognitive radio, green communications, and physical layer security.

Victor C. M. Leung [S'75, M'89, SM'97, F'03] (vleung@ece.ubc.ca) is a professor of electrical and computer engineering and holder of the TELUS Mobility Research Chair at the University of British Columbia, Canada. His has co-authored more than 900 journal and conference papers in the areas of wireless networks and mobile systems. He is a Fellow of the Royal Society of Canada, the Canadian Academy of Engineering, and the Engineering Institute of Canada. He is an editor of *IEEE JSAC-SGCN*, *IEEE Wireless Communications Letters*, and several other journals.

Hongxi Yin (hxyin@dlut.edu.cn) is an associate professor at Dalian University of Technology. He received his B.Sc. degree from Shandong University, Jinan, China, in 1982, his M.Eng. degree from Harbin Institute of Technology in 1988, and his Ph.D. degree from Zhongshan University, Guangzhou, China, in 1998. He finished his postdoctoral research work at Peking University, Beijing, China, in 2000, and worked as an associate professor there from 2000 to 2008.