International Journal of Information Communication Technologies and Human Development
Volume 14 • Issue 1

Adaptive Cache Server Selection and Resource Allocation Strategy in Mobile Edge Computing

Michael Pendo John Mahenge, Department of Informatics and Information Technology, Sokoine University of Agriculture, Tanzania*
Edvin Jonathan Kitindi, Department of Informatics and Information Technology, Sokoine University of Agriculture, Tanzania
https://orcid.org/0000-0003-0413-5757

ABSTRACT

The enormous increase of data traffic generated by mobile devices emanate challenges for both internet service providers (ISP) and content service provider (CSP). The objective of this paper is to propose the cost-efficient design for content delivery that selects the best cache server to store repeatedly accessed contents. The proposed strategy considers both caching and transmission costs. To achieve the equilibrium of transmission cost and caching cost, a weighted cost model based on entropy-weighting-method (EWM) is proposed. Then, an adaptive cache server selection and resource allocation strategy based on deep-reinforcement-learning (DRL) is proposed to place the cache on best edge server closer to end-user. The proposed method reduces the cost of service delivery under the constraints of meeting server storage capacity constraints and deadlines. The simulation experiments show that the proposed strategy can effectively improve the cache-hit rate and reduce the cache-miss rate and content access costs.

KEYWORDS

Cache Server Selection, Content Delivery, Content Service Provider, Cost Effective, Deep Reinforcement Learning, End User, Mobile Edge Computing, Resource Allocation

INTRODUCTION

The enormous improvement of smart mobile equipment is considered to be significant in this era of big data development to enable access to delay-critical and resource-intensive mobile applications such as video-on-demand (Tran et al., 2017). While facilitating vast potential for offering anywhere and anytime accessibility, vast amount of data generated by mobile equipment emanates great burden to the core network due to huge increase of data traffic that is expected to grow multi-fold in the future(Cisco, 2016; Jaleel et al., 2010). The enormous increase of data traffic emanates challenges for both Internet Service Providers (ISP) and Content Service Provider (CSP). The ISP strive to provide quality services along with minimizing operational expenses such as internet access costs. On the same vein, CSP strive to enhance quality of experiences (QoE) for end users in-line with achieving cost-efficient content delivery.

Cloud computing as an internet-based computing has been considered important in providing quality services and handling big data processing (Skourlelopoulos et al., 2017). Consequently, large CSP such as YouTube, Facebook, or Twitter store their content in massive data centers in the cloud. Also, the advanced features of the cloud computing such as elastic assignment of resources on-demand, and unlimited resources for processing and storage, guarantees substantial capacity to deal with huge amount of data emanating

DOI: 10.4018/IJICTHD.299412 *Corresponding Author

from mobile applications (Pompili et al., 2016). However, due to multi-hop communication between mobile equipment and remote servers, legacy systems such as mobile cloud computing (MCC) still face performance challenges. In the traditional content delivery network (CDN), the mobile devices form the frontend and the CDN servers are deployed at the backend. Each mobile device is associated with a nearby base station (BS) or access point (AP) for internet access services. Each content request received at the BS, is forwarded to the CDN through core network, retrieve the requested content and respond back to the requesting user. However, the overwhelming evolution of resource-intensive applications with low-latency requirement emanates challenges in the traditional CDN in terms of network overloading, high service utility cost, and inadequate service quality (Tran et al., 2017).

Recently, emerging computing paradigm such as MEC that provides cloud computing facilities at the vicinity of mobile users has been proposed (Hu et al., 2016). MEC has been considered as a significant computing paradigm to mitigate challenges emanating from the immense pressure created by resource-intensive mobile applications (Tran et al., 2017). Meanwhile, mobile edge caching deployed at the BS of mobile network is proposed as a novel and promising architecture that bring contents at the proximity of the content service requesters (Wang et al., 2014). This novel architecture offers substantial opportunity to achieve cost-efficient content delivery through caching mostly accessible contents closer to users (Zhao et al., 2016). Therefore, CSP could benefit through Infrastructure-as-a-service (IaaS) offered by MEC which guarantee scalability, low service delivery cost, high performance, location-awareness, and low delay. While mobile users could benefit from enhanced QoE achieved through content caching at the BS or AP. Moreover, MEC cooperative capability offers potential opportunity to improve QoE through cooperation between BSs and the central cloud (Tran et al., 2017). Despite the unique contributions offered by mobile-edge caching, the limited cache storage capacity at the BS become stumbling block to efficiently deal with the enormous pressure triggered by latency-critical and resource-intensive mobile applications (Tran et al., 2017). Also, varying application and users' preferences, heterogeneity of MEC computing instances, and limited MEC resources such as bandwidth and power, intensify cache server selection and resource allocation problem.

Some existing works proposed deployment of edge-caches without considering the unique characteristics of edge devices such as limited resources and varying computing capacity (Pantisano et al., 2014; Bastug et al., 2015; Bharath et al., 2016). Moreover, the existing works focused on cache placement optimization autonomously among one edge-cache and CDN servers or among several edge-caches. However, the proposed approaches can face challenges such as high cache-miss rate and low probability of cache-hit due to limited caching capacity. Consequently, the chance to utilize backhaul link become high because many requests would be forwarded to the remote CDN thus failing to achieve the required QoE due to high transmission cost and latency especially for delay-sensitive applications.

Therefore, to guarantee cost-efficient service delivery, in this paper an adaptive cache server selection and resource allocation scheme in MEC environment taking into account both caching and transmission costs is proposed. The key objective of this paper is to find out the cache placement design that selects the best cache server for caching taking into account storage cost and transmission cost along with satisfying capacity constraint and delay requirements. To trade-off between content transmission cost and caching resource renting cost, the weighted cost model is formulated. Then, an efficient adaptive strategy based on deep reinforcement learning (DRL) is proposed to achieve optimal content caching decision.

The unique contributions of this paper are three-fold as follows: initially, a cache server and resource allocation in MEC-assisted architecture is formulated as a weighted cost model to balance the transmission cost and caching resource renting cost. Then, a novel adaptive caching decision scheme that leverages the advanced capability of DRL is proposed to guarantee optimal service delivery cost while meeting latency and caching capacity constraints for content delivery in MEC environment. Through simulations, the proposed strategy demonstrates significant improvements in terms of average rewards, cache-hit rate, cache-miss rate, and access latency.

The remaining sections of this paper is structured as follows: section "caching model in mobile edge computing" describes the caching models, and section "the proposed adaptive cache server

selection and resource allocation strategy" expound the proposed methodology. Section "performance evaluation" presents the analysis of experiment results to evaluate the performance of the proposed strategy, and finally, section "conclusion" presents the conclusion and proposes future works.

CACHING MODEL IN MOBILE EDGE COMPUTING

This section demonstrates the system model. The details of the network model, service utility model, and problem formulation are given in the following subsections.

Description of the Caching Scenario

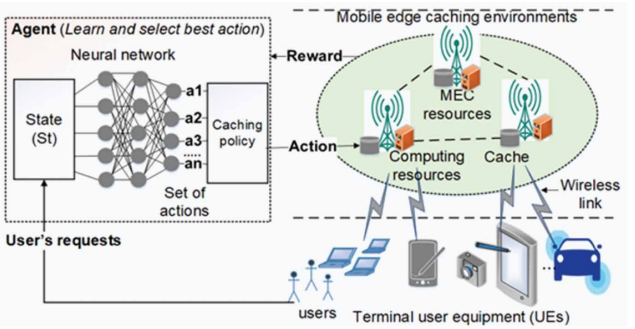
In this paper, the set of cache-capable BSs is denoted as $\mathbb{C} = \{1, 2, \ldots C\}$, where the caching capacity of each BS is φ_i. Furthermore, we assume each BS comprises of controller and cache manager. The controller which can be a powerful machine or server cluster manages edge nodes and requests arriving at each BS. Specifically, it is responsible for handling each request arrival, request scheduling, and communication with cache manager or computing servers in its locality and controllers in other areas within cooperative space. The cache manager is responsible for cache management activities such as content placement and replacement based on the implemented caching policy. Moreover, its critical role is to keep track of resources and cached contents in each BS and update them regularly. The content-caching optimizer is part of the cache manager which deals with cache management decisions making to efficiently utilize the available resources through scheduling and placing contents in suitable locations. Therefore, the controller and content-caching optimizer can exchange messages to one another synchronously to achieve efficiency cache management processes. Additionally, the advanced features of MEC such as cooperative resource allocation for offloading and caching guarantee optimal resource utilization, less costs and service delivery delay, and enhance QoE (Mahenge et al., 2019).

Moreover, as shown in Figure 1, an agent perceives the caching-environment and acquires input signals such as user inquires and network condition. The acquired inputs can be gathered and form the state S_t that is used as the input for training neural network (NN). Based on the learning output and caching policy, an agent chooses an action $\left(a_t\right)$ that indicates caching of content to the subsequent slot. The resultant performance based on the chosen action is then perceived and the reward fed back to the learning agent. The agent utilizes the received feedback to train and enhance the NN model with the objective of maximizing the entire rewards.

Caching Network Model

In the system model, MEC-based caching paradigm with multiple edge-caches deployed at the BSs of mobile network is considered. It is assumed that the BSs connect with one another through backhaul

Figure 1. Components of the proposed system model



connection, while end users are connected to the closest BS through wireless communication link. Let the finite set of users be denoted as $\Phi = \{1,2,\ldots u,\ldots,N\}$, where each user is associated to a closest base stations (BS) or access points (AP) depending on its connectivity strength. The edge-caches in the MEC-based caching system can be configured at the BSs located in various areas. The capacity of each BS to deliver services is limited by the coverage region and resources. The set of cache-capable BSs is denoted as $\mathbb{C} = \{1,2,\ldots C\}$, where the caching capacity of each BS is φ_i. Let $K+1$ indicates the catalogue of the available content provider's in the CDN, $\mathbb{M} = \{1,2,\ldots,\mathcal{M}\}$ represents the finite set of contents with size s_k (MB) that are accessible by users in the Internet.

In the considered scenario, it is assumed that the memory units (in GB) with caching capacity of φ_i are deployed in each BS whereby the CSPs can rent for placement of contents at the proximity of end users. Therefore, the total cache capacity in all BS within the cooperative domain is expressed as:

$$\varphi_t = \sum_{i=1}^{C} \varphi_i, \forall_{i\in\mathbb{C}} \tag{1}$$

Moreover, it is considered that each BS can receive multiple caching inquiries, therefore let r_u be the caching resources required to store content k at the BS $i \in \mathbb{C}$. Furthermore, if μ_{ui} denotes the inquiries for data caching received at a particular BS $i \in \mathbb{C}$, then, the total caching inquiries become $\sum_{u\in\Phi} \mu_{ui,\forall_{i\in\mathbb{C}}}$. According to Nguyen & Vojnovic (2011), the BS can allocate resources to multiple requests grounded on weighted proportional distribution whereby the portion of caching resources allocated for each request can be computed as:

$$r_u = \varphi_t \frac{\mu_{ui}}{\sum_{u\in\Phi} \mu_{ui}}, \forall_{i\in\mathbb{C}} \tag{2}$$

Let χ_{kij} be the cache decision variable with the content placement matrix defined as:

$$\chi_{kij} = \begin{cases} 1, & if\ content\ k\ is\ cached\ at\ the\ BS\ i\ or\ j\ in\ the\ collaborative\ space \\ 0, & otherwise \end{cases}$$

Therefore, given the capacity constraint φ_t, the overall rented cache space must satisfy the achievable policy expressed as:

$$\sum_{k=1}^{\mathcal{M}} \sum_{i=1}^{C} r_u \times \chi_{kij} \leq \varphi_t, \forall_{u\in\Phi}, i,j \in \mathbb{C} \tag{3}$$

Service Utility Cost Model

The service utility cost model is computed by data storage cost, and data transmission cost.

Data Storage Cost

Let C_k denote the unit cost for renting caching resources on the specific location which can be defined as:

$$C_k = \begin{cases} Cost_e, if\ the\ content\ is\ cached\ at\ the\ BS \\ Cost_c, if\ the\ content\ is\ cached\ at\ the\ remote\ servers \end{cases}$$

Let $R(k)$ be the integer such that $R(k) > 0$ which denotes the number of cache servers with the content k. For example if $R(k) = 1$, signifies that the content k is cached in only one cache server. Therefore, the total caching resources rental costs (C_s) can be calculated by:

$$c_s = \sum_{i=1}^{R(k)} \left(\sum_{j \in \mathbb{C}} \chi_{kij} \times C_k \times s_k \right) \tag{4}$$

Content Transmission Cost

Let d_{ke} denotes the unit costs for transmission of content k from the edge-cache servers within the cooperative domain. Also, let d_{kr} denotes the unit costs for transmission of content k from the remote servers in the cloud. Therefore, the transmission cost (C_t) for serving user requests can be calculated as:

$$c_t = \sum_{i=1}^{C} \left(\frac{X_{kij} * d_{ke} * s_k}{\mathcal{B}_{ij}} \right) + \sum_{j=1}^{K+1} d_{kr} \left(1 - X_{kij}\right)\left(1 - \Pi_k\right) \tag{5}$$

where $\mathcal{B}_{ij}$ is the bandwidth, and $\Pi_k \in [0,1]$ is the popularity of each content k which represents the probability of caching content k at the BS. Then, the probability that content k is not cached at the BS is given as $(1 - \Pi_k)$. The content with higher probability has high chance to be cached at the BS.

Problem Formulation

In mobile edge caching application scenarios, the system cost can be contributed by many factors. In this study, the authors considered two metrics that includes first, the cache renting cost for storing contents proximate to users and second, the transmission costs incurred for service delivery. Therefore, in order to trade-off between content transmission cost and caching resource renting cost, the system cost incurred for serving all user requests is defined as the weighted cost model expressed as:

$$\mathcal{Q}(c_s, c_t) = \sum_{j=1}^{n} \left(\omega_s c_{s,j} + \omega_t c_{t,j} \right) \tag{6}$$

where parameter ω is a weight that balance between caching cost and content transmission cost, n is the number of requests received by the system. The CSPs prefer low operational costs such as renting caching resource while satisfying users' preferences. Similarly, users prefer quality services with low service delivery cost and high QoE. The key objective of this chapter is to minimize the overall cost taking into account caching cost and transmission cost that is defined by the following objective function:

$$\min_{\{n\}} \ \mathcal{Q}(c_s, c_t) \tag{7}$$

$$\text{s.t.} \sum_{i=1}^{n} r_u \times \chi_{kij} \leq \varphi_t \tag{8}$$

$$\sum_{i=1}^{n} \chi_{kij} \leq R\left(k\right) \tag{9}$$

$$c_t \leq C_{\max} \tag{10}$$

The objective function (7) maximizes QoE while satisfying the given constraints (8) to (10). The expression (8) is the capacity constraints that ensures that the allocated resources for content caching cannot exceed the available storage capacity, the expression (9) guarantees that selection of cache server to handle user's request should not exceed the number of cache servers available, and expression (10) guarantees that the transmission cost cannot exceed the threshold cost for each request.

THE PROPOPOSED ADAPTIVE CACHE SERVER SELECTION AND RESOURCE ALLOCATION STRATEGY

In this section, cache placement strategy is formulated as Markov Decision Problem (MDP), such that the recent cached content is determined by the preceding state and action. The following subsections present the theoretical background, settings and design of the proposed methodology.

Theoretical Context

This subsection presents the theoretical background of the proposed methodology based on Deep Reinforcement Learning (DRL). In DRL approach, there are three (3) key concepts that include: states, actions, and rewards. The states are the representation of the current situation or tasks that the learning agent observes from environment while the actions are the possible deeds that the learning agent can do to adjust these states to meet the optimization objective. The rewards are the returns that the learning agent receives for executing the right action. In this paper, the set of the state-space is represented by $S = \left\{s_1, s_2, s_3, \ldots, s_t\right\}$, and the set of the action-space is represented $A = \left\{a_1, a_2, a_3, \ldots, a_t\right\}$ respectively. Also, r_t denotes the reward for selecting an action a_i where $i = 1, 2, \ldots t$. The action is a key contributing factor that affect rewarding decision for both the present reward $\left(r_t\right)$ and succeeding reward $\left(r_{t+1}\right)$. The transition from r_t to r_{t+1} can be achieved through reward-function and the state-transition probability $P\left(s_{t+1} \mid s_t, a_t\right)$. The actions that efficiently incline towards the optimization objective has high chance to be selected by an agent. Initially, at the state $s_1 \in S$ the learning-agent performs an action $a_t \in A$ at t^{th} selection epoch and transfers the present state $s_t \in S$ to $s_{t+1} \in S$ with a probability distribution P. Consequently, the agent gains a reward r to assess the performed action. Thus, the transit probability and r can be represented as function expressed as $P\left(s_t, a_t, s_{t+1}\right)$ and $r\left(s_t, a_t, s_{t+1}\right)$ respectively. Assume that the action a_t is performed based on the stochastic strategy defined as $\Pi\left(a \mid s\right) = P_r\left(a_t = a \mid s_t = s\right)$. The agent learns and evaluate the strategy $\Pi\left(a \mid s\right)$ defined as a mapping in the state s_t to a probability of selecting an action a_t. Let $\mathcal{R}_t$ denotes the return which represents the aggregate rewards given by:

$$\mathcal{R}_t = \sum_{k=0}^{\infty} \psi^k r_{t+k} \tag{11}$$

where $\psi \in \left(0,1\right)$ is a discount factor. Given the strategy $\Pi\left(a \mid s\right)$, the state-action expression can be defined as:

$$Q^{\pi}\left(s,a\right) = E\left[\mathcal{R}_{t} \left| \Pi, a_{t} = a \right| s_{t} = s\right] \tag{12}$$

Then, to determine the best Q-value $Q^{*}\left(s_{t},a_{t}\right)$, the Bellman-optimality method proposed in [149] can be adopted. Mathematically, it can be expressed as:

$$Q^{*}\left(s_{t},a_{t}\right) = E\left[r_{t} + \psi \max\nolimits_{a_{t+1}} Q\left(s_{t+1},a_{t+1} \mid s_{t},a_{t}\right)\right] \tag{13}$$

However, due to large space in S, computing all values of Q using Bellman method become challenging. Moreover, Q-learning is an appropriate and effective algorithm adopted to solve Markov-decision optimization problems. However, the key challenge reported in the literature is the limited storage capacity of the Q-table which is considered to be a limiting factor in addressing problems with large S and A (Kan et al., 2019). Therefore, other effective methods such as neural network can be employed to estimate and evaluate the value-function in Q-learning approach. Algorithm 1 presents the main algorithmic steps.

Furthermore, in order to guarantee efficiency, steadiness and improved convergence-speed of the algorithm, deep Q-network approach with improved features such as experience-replay, value-function estimation, and target Q-network is adopted. The algorithm takes the state-values as input to the neural-network, then proceed with the training phase through Multi-Layer Perceptron (MLP). Finally, it returns the Q-value serving as a basis for selection of an actions.

The key procedures of algorithm 1 start with configuration and initialization of parameters such as the feature space s_{t}, weights $\theta\left(t\right)$ of the neural-network, temporary storage ϕ to keep track of experiences, Q-network parameters, and the number of state transition k (Algorithm 1 line 1-6). Then, proceed with iterative process to choose the current action based on the given state and selection strategy, assess the Q-values for the present state-action couple $\left(s,a\right)$, observe the succeeding state s_{t+1} and the reward r_{t} (Algorithm 1 line 7-14). Consequently, store the tuple $\mathcal{L} = s_{t},a_{t},r_{t},s_{t+1}$ in the temporary storage ϕ referred as the experience-replay (Algorithm 1 line 15). The experience-replay technique addresses the fluctuations and deviation problems emanating from correlation among the experience data. Moreover, if samples in ϕ are sufficient, the samples $\mathcal{H}$ of a mini-batch of $\mathcal{L}$ from ϕ are randomly selected to train the DQN algorithm using a gradient-descent method and approximate the Q-values (Algorithm 1 line 16-20). Finally, the algorithm updates the variable θ of the mini-batch sample to optimize the loss function, and terminates the loop if the termination criteria is met (Algorithm 1 line 21-23).

Important Configurations of the Proposed Algorithm

In this paper, the key objective is to reduce the caching cost and content transmission cost simultaneously, thus to tradeoff between the two optimization goals, a linear-weighting-method is used to guarantee accurate weight distribution and improve caching decision. Thus, the entropy-weight-method (EWM) is adopted to govern the weight distribution of caching cost and transmission cost (Delgado & Romero, 2016). Let p denote the number of samples in EWM referred as the number of the chosen actions; q denote the number of objectives considered for optimization, where in this paper the caching cost and transmission cost are considered as indicators, thus the value of q in this case is 2.

Let the $y_{i,j}$ denote the measured value of the i^{th} indicator in the j^{th} sample. Then, the initial step in EWM is to normalize the indicators. In this paper the authors use min-max mathematical model expressed as:

Algorithm 1. Deep-Q network algorithm

Input: State value
Output: Action value
1: Set feature space s_t
2: Set temporary Storage ϕ that keeps track of experiences with capacity N_t
3: set the Q-network variabale with θ
4: set the Q-value function $Q_\theta\left(s,a\right)$ and the Stochastic strategy $\Pi\left(a \mid s\right)$
5: Initialize the weights $\theta\left(t\right)$ of the neural-network where $t = 1,2,\ldots,n$
6: Set k, the number of State-transition to 0 and k_{max} to a large number
7: for episode 1 to n, **do**
8: Initialize the initial state s
9: for $t = 1; t \le T; t++$, **do**
10: Select action a_t, with respect to the present state s_t and strategy $\Pi\left(a \mid s\right)$
11: otherwise
12: choose $a_t^* = argmax_a Q\left(s_t, a; \theta\right)$
13: assess the Q-values for the present state-action couple $\left(s,a\right)$
14: observe the succeeding state s_{t+1} and the reward r_t
15: store the tuple $\mathcal{L} = s_t, a_t, r_t, s_{t+1}$ in ϕ
16: generate the samples $\mathcal{H}$ of a mini-batch of $\mathcal{L}$ from ϕ randomly
17: for each sample $h \in \mathcal{H}$, **do**
18: execute a gradient-descent method on $\left(y_h - Q\left(s_t^h, a_t^h; \theta\right)\right)^2$
19: end for
20: return Q-value
21: update the variable θ of the mini-batch to optimize the loss-function
22: end for
23:end for

$$y_{i,j}^* = \frac{y_{i,j} - \min\left\{y_{1,j},\ldots,y_{p,j}\right\}}{\max\left\{y_{1,j},\ldots,y_{p,j}\right\} - \min\left\{y_{1,j},\ldots,y_{p,j}\right\}} \tag{14}$$

where $y_{i,j}^*$ is a normalized parameter, $i = 1,2,\ldots,p$, and $j = 1,2$. Then, the ratio of individual sample value in different indicators can be defined as:

$$\mathcal{G}_{i,j} = \frac{y_{i,j}^*}{\sum_{i=1}^{p} y_{i,j}^*} \tag{15}$$

The entropy value of each indicator in EWM can be obtain by the following mathematical model:

$$E_j = -\frac{1}{\ln\left(p\right)}\sum_{i=1}^{p}\mathcal{G}_{i,j}\ln\left(\mathcal{G}_{i,j}\right) \tag{16}$$

Finally, the weight $\left(\omega_i\right)$ of each parameter can be obtained using the following mathematical model:

$$\omega_j = \frac{1-E_j}{q-\sum_{j=1}^{q}E_j} = \frac{1-E_j}{2-\sum_{i=1}^{2}E_j} \tag{17}$$

Considering the present requested content and the present cached content as the state-space S, each state s_t is represented as feature vector. Let ξ_k denotes the features of each caching resource, the state vector can be expressed as $s_t = \left\{\xi_0, \xi_1, \xi_2, \ldots, \xi_C\right\}$ where ξ_0 denotes the feature vector of the recently inquired content, and ξ_τ represent a feature vector of the τ^{th} cached content, that is to say $\tau = 1, 2, \ldots, C$.

The DQN algorithm is used to achieve the optimization objective. Therefore some elements of the algorithm are customized and redefined to suit the formulated optimization problem. Subsequently, the weighted sum of caching cost and transmission cost of the requested content from each edge node is considered as the state, and the state-space is redefined as:

$$S = \left\{s_1, s_2, \ldots, s_p\right\}, s_i = \omega_{i,1}c_s + \omega_{i,2}c_t \tag{18}$$

Similarly, since this paper addresses the cache server selection and cache resource allocation problem in MEC, the choice of the cache server is considered as the action where the cache server is located at the BS. The action-space is redefined as, $A = \left\{cnode_1, cnode_2, \ldots, cnode_k\right\}$ where k is the number of cache servers at the BS. Moreover, the rewards is considered in terms of system cost and can be obtained using the following mathematical model:

$$r_t = -\left(\omega_s c_{s,j} + \omega_t c_{t,j}\right) \tag{19}$$

Therefore, define a tuple $\mathcal{L} = s_t, a_t, r_t, s_{t+1}$ as the temporary storage that keeps track of experiences with π^{th} tuple of experience example. Let θ be the network-target variable and ϕ be the experiences stored in $\mathcal{L}$ tuples, the target-value of Q can be obtained as $V_Q^* = r_t + \psi max_{a_{t+1}} Q_\theta\left(s_{t+1}, a_{t+1}\right)$ and $Q\left(s_t, a_t\right)$ is the current-value of Q denoted as V_Q. Therefore, the loss-function, $\mathcal{F}\left(\theta\right)$ can be defined as:

$$\mathcal{F}\left(\theta\right) = E_\phi\left[V_Q^* - V_Q\right]^2 \tag{20}$$

To compute $\mathcal{F}\left(\theta\right)$ for each iteration, first, the neural network and an adaptive learning rate $\lambda_{t,\pi}$ is adopted. Then, a gradient-descent approach is used to update the variable θ as expressed by the following mathematical model:

$$\triangle_\theta = \lambda_{t,\pi}\left[V_Q^* - V_Q\right]\nabla_\theta V_Q \tag{21}$$

where ∇_θ represents the partial derivative with respect to θ.

The advantages of DRL adopted in this paper includes but not limited to: first, by leveraging advanced capacity of neural networks for storage, addresses the challenge of limited storage capacity of the Q-table in traditional Q-learning which is considered to be a limiting factor in addressing problems with large state and actions. Second, using deep Q-network approach with improved features such as experience-replay, value-function estimation, and target Q-network, guarantees efficiency, steadiness and improved convergence-speed of the algorithm Third, adjusting the parameters of Deep Neural Network (DNN) and learning the popularity structure of the contents achieves superlative long-term performance optimization.

PERFORMANCE EVALUATION

This section presents the performance evaluation of the proposed strategy. The remaining subsections portray the experimental design and configurations, comparison algorithm, and experimental results and analysis.

Experimental Design and Configuration

The dataset used in our experiments comprises of two chunks. Initially, we used the analytical content-request model following Zipf-distribution to generate datasets for user's requests. In the simulation experiment, the probabilistic distribution was used to get samples based on the requested content identifiers. It is assumed that a random probability is used for each service requester at the edge requesting content *k* from the system. The probability of selecting *k* from the library determines the popularity of *k*. Specifically, the probability of requesting the k^{th} content (most popular) can be obtained using the expression:

$$P_k = \frac{1/k^\alpha}{\sum_{n=1}^{F} 1/n^\alpha} \tag{22}$$

where α is the Zipf-distribution skewness parameter. The content library is set to 10000 files with similar size set to 3.5 MB for each file. Initially the data sets generated with fixed value of $\alpha = 0.74$, and then, the value of α varied in the range $\left[0.4, 1.4\right]$. The number of user's requests were randomly generated between 100 to 100000 requests. The extracted features (number of requests) used as the inputs of the neural network. In this work, the features represent the latest 10, 100, and 1000 requests. Similar to He et al. (2019), the size of temporary storage ϕ is set as 10000, the min-batch size is set to 32, and 256 neurons were considered for neural network training. The adaptive-learning rate is considered to be $\left\{1.0\times10^{-3}, 1.0\times10^{-4}, 1.0\times10^{-5}\right\}$ and thereafter it was fixed to 1.0×10^{-3} for the remaining simulations. The number of cache-capable BSs is set to 10 and the number of users requesting contents is randomly and evenly distributed with an average of 50 users in each BS. The average access cost for cached content k at the BS $\left(Cost_e\right)$ or remote servers $\left(Cost_c\right)$ are set to 6 and 63ms. The average transmission cost are assigned randomly and evenly distributed between 10 to 30ms for transmission between one MEC server to another, and 60 to 100ms between MEC server and remote cloud. Table 1 presents the parameter setting of the edge devices.

Comparison Algorithm

The performance of the proposed strategy is accessed in comparison with the well-known benchmark algorithms that includes Q-Learning (QL), Least Recently Used (LRU) caching strategy (Mohamed et al., 2018), Least Frequently Used (LFU) caching strategy (Jaleel et al., 2010), and No caching (NC) algorithm.

Table 1. Edge devices parameter setting

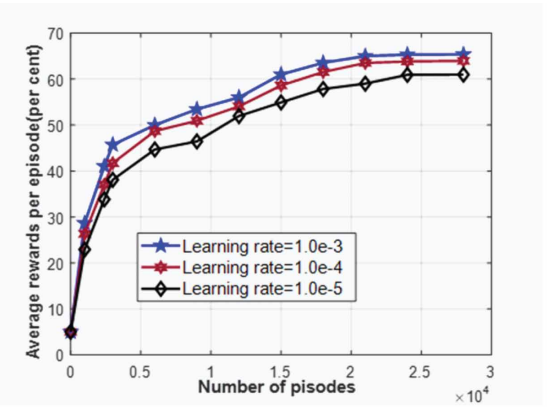
Parameter	Values
Processing rate	10256 MIPS
RAM	4096 MB
Storage	350000 MB
Unit cost of storage	0.15
Unit cost of bandwidth	0.1
Bandwidth	20 MHz

The QL algorithm is a classic reinforcement Learning (RL) algorithm with good learning effects, which is appropriate for solving joint optimization problems that are modeled as MDP, such as task unloading and resource allocation. The algorithm keep the Q-values in bi-dimensional matrix known as Q-table. Nevertheless, the capacity of Q-table is limited in terms of storage which emanates Q-table explosion challenge that cause QL algorithm to be inappropriate for large population size. Therefore, this algorithm competes with the proposed algorithm that leverages the advanced capacity of neural networks to improve storage capacity. The LRU strategy keeps track of the most up-to-date requests for each content in the cache server and evict the least-recently accessed content when the cache is running out of space to free space for storage of the new content. The LFU strategy focuses on the frequency of requests for each cached content and evict the least-frequently accessed content when the cache is running out of space to accommodate the new incoming content. The NC strategy serves users' requests independent of caching policy. There is no any content cached at the local server, therefore, each users' request is served only by the original server. The algorithms are implemented in edgeCloudSim 4.0 simulation environment built on computer with INTEL core i5-7200U @3.10GHz CPU, 4.00GB-RAM, 1TB hard disk, x64 Ubuntu 16.04 LTS. The following sections present the performance metrics and discussion of simulation results.

Experimental Results and Comparative Analysis

Figure 2 illustrates the impacts of the learning-rate on the convergence performance of the proposed strategy in terms of average rewards when the number of episodes varies. As shown in the figure, the average rewards per episode is very low when the number of episode is low. However, for all learning-rates 1.0e-3, 1.0e-4, and 1.0e-5, the proposed scheme exhibits increasing trend of average rewards with increase in the number of episode until it attains a reasonably stable value. In all cases,

Figure 2. The impacts of the learning-rate on convergence performance of the proposed strategy in terms of average rewards when the number of episodes varies



the proposed strategy converges quicker and achieves higher rewards when the learning-rate is large (1.0e-3) than when the learning-rate is small (1.0e-4 and 1.0e-5). Conversely, the larger learning-rate is suitable for local optimization problems while the smaller learning-rate is suitable for global optimization problems [157]. Consequently, the choice of the learning rate depends on the nature of the problem. Therefore, throughout the simulation experiments, the learning-rate used is 1.0e-3.

Figure 3 demonstrates the impacts of the size of the min-batch for individual gradient-update in DRL method on average rewards achieved by an agent. The min-batch size parameter defines the number of samples considered in every training phase. The analysis of simulation results show that, the agent achieves higher average rewards when the size of min-batch sample is small (32) than when is 64 and 128. Therefore, the selection of this parameter can vary depending on the problem. Throughout the simulation experiments for this study, the size of the min-batch is set as 32.

Figure 4 portrays the impact of Zipf-distribution skewness parameter on the performance of the caching schemes in terms of cache-miss rate when the caching capacity is fixed. It is evident that the cache-miss rate of the proposed strategy, QL, LFU and LRU decreases when the Zipf-distribution skewness parameter increases from 0.4 to 1.4. This is because as skewness parameter become large, it intensify the content-popularity that improves caching strategy. Consequently, the probability of cache-miss become low. However, the performance of no caching scheme is constant which implies that Zipf-skewness distribution parameter has no impact on non-caching scheme. Moreover, at first the proposed

Figure 3. The impacts of the size of the min-batch for individual gradient-update in DRL-method on average rewards achieved by an agent

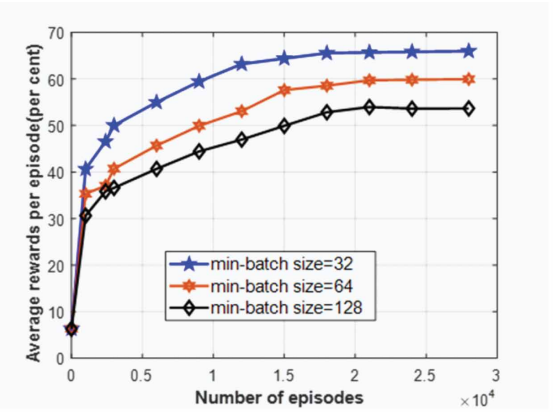
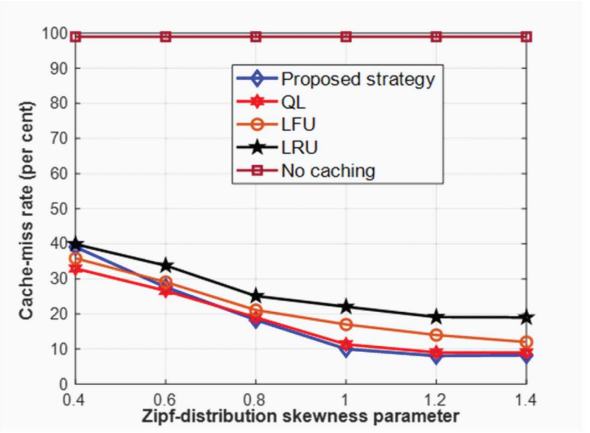


Figure 4. The impacts of Zipf-distribution skewness parameter on cache-miss rate



strategy exhibit no benefits over other schemes, then it outperforms QL, LFU, LRU and no caching schemes with smaller performance gap compared to QL. This is because the proposed scheme begins to learn the popularity of contents, update the neural-network, and adapt to the caching environment.

Figure 5 portrays the impact of Zipf-distribution skewness parameter on the performance of the caching schemes in terms of cache-hit rate when the caching capacity is fixed at 200GB.

It is evident that the performance of the proposed strategy, QL, LFU and LRU increases when the Zipf-distribution skewness parameter increases from 0.4 to 1.4.

This is because as skewness parameter become large, imply that the content is more popular, and consequently the probability of caching such content to the local cache is high. Moreover, the proposed strategy performs better than other schemes. Therefore, the analysis of results provide evidence that the proposed strategy can efficiently learn the popularity of contents from user's requests and make optimal caching decision that addresses long-term cache-miss challenge and improves cache-hit in caching environment.

Figure 6 portrays the impact of cache capacity on the performance of the caching schemes in terms of gain in service cost. As shown in Figure 6, the proposed strategy, QL, LFU and LRU exhibit increasing trend in terms of gain in service cost when the cache capacity increases from 50GB to 500GB. In all cases the proposed strategy achieves better results compared to other strategies saving cost up to 68.5% when the cache capacity is large (500GB). It is obvious that large caching capacity provide high chance to cache more popular content which minimizes the transmission cost to the remote servers.

Figure 5. The impacts of Zipf-distribution skewness parameter on cache-hit rate

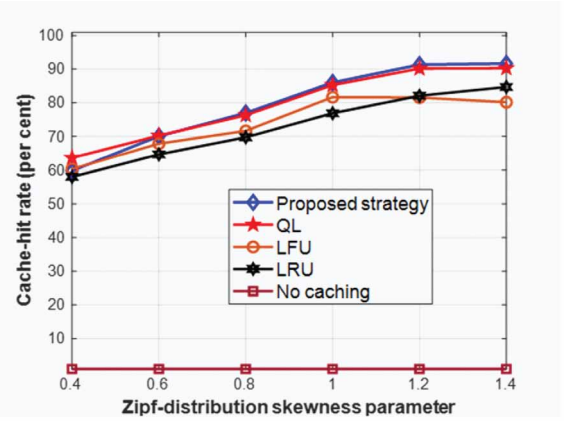
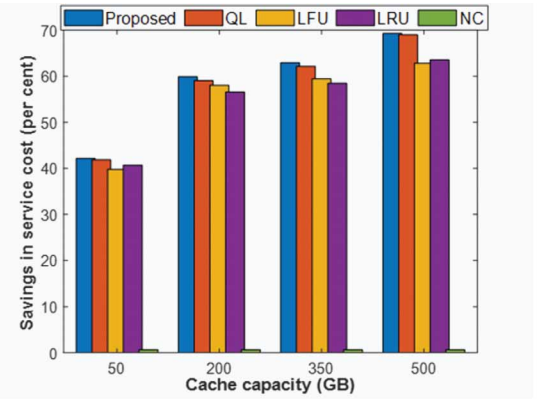


Figure 6. The impacts of cache capacity on the performance of the caching schemes in terms of gain in service cost



CONCLUSION

This paper essentially investigates the cache server selection and resource allocation problem in MEC-supported environment. An adaptive strategy that take into account transmission cost and caching cost while fulfilling capacity and delay constraints is proposed. Initially, the cache server and resource allocation in MEC-assisted system is formulated as a weighted cost model to balance the transmission cost and caching resource renting cost. Then, a novel adaptive caching decision scheme that leverages the advanced capability of DRL is proposed to guarantee optimal service delivery cost while meeting latency and caching capacity constraints for content delivery in MEC environment. Experimental results show that the average rewards increases when the number of episode increases until it attains a reasonably stable value. Also, it is observed that, the agent achieves higher average rewards when the size of min-batch sample for individual gradient-update is small. Moreover, the proposed strategy attains an improved performance and outperforms other baseline approaches in terms of cache-hit rate, cache-miss rate, and savings in service costs. In the future, researchers will largely focus on extensive assessment of the proposed scheme to evaluate its effectiveness in real-working mobile edge caching systems.

ACKNOWLEDGMENT

The authors would like to thank Wuhan university of Technology (WHUT), and Sokoine university of Agriculture for creating supporting environments for the study.

FUNDING AGENCY

The publisher has waived the Open Access Processing fee for this article.

REFERENCES

Bastug, E., Bennis, M., & Debbah, M. (2015). A transfer learning approach for cache-enabled wireless networks Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks. *Proceeding of 13th IEEE International Symposium*, 161-166.

Bharath, B. N., Nagananda, K. G., & Poor, H. V. (2016). A learning-based approach to caching in heterogeneous small cell networks. *IEEE Transactions on Communications*, *64*(4), 1674–1686. doi:10.1109/TCOMM.2016.2536728

Cisco. (n.d.). *Cisco visual networking index: Global mobile data traffic forecast update, 2016-2021*. https://www.cisco.com/c/en/us/solutions/ collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html

Delgado, A., & Romero, I. (2016). Environmental conflict analysis using an integrated grey clustering and entropy-weight method: A case study of a mining project in Peru. *Environmental Modelling & Software*, *77*(1), 108–121. doi:10.1016/j.envsoft.2015.12.011

He, X., Wang, K., & Xu, W. (2019). QoE-Driven Content-Centric Caching With Deep Reinforcement Learning in Edge-Enabled IoT. *IEEE Computational Intelligence Magazine*, *14*(4), 12–20. doi:10.1109/MCI.2019.2937608

Hu, Y. C., Patel, M., Sabella, D., Sprecher, N., & Young, V. (2015). *Mobile Edge Computing: A key technology towards 5G*. ETSI White Paper No. 11. http://www.etsi.org/images /files/ETSI White papers/etsi wp11 mec a key technology towards 5g.pdf

Jaleel, A., Theobald, K. B., Steely, S. C., & Emer, J. (2010). High performance cache replacement using re-reference interval prediction (RRIP). *Proceeding of 37th Annual international symposium on computing architecture*, 60–71. doi:10.1145/1815961.1815971

Kan, N., Zou, J., Tang, K., Li, C., Liu, N., & Xiong, H. (2019). Deep reinforcement learning-based rate adaptation for adaptive 360-degree video streaming. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4030–4034. doi:10.1109/ICASSP.2019.8683779

Mahenge, M. P. J., Li, C., & Sanga, C. A. (2019). Mobile Edge Computing: Cost-Efficient Content Delivery in Resource-Constrained Mobile Computing Environment. *International Journal of Mobile Computing and Multimedia Communications*, *10*(3), 23–46. doi:10.4018/IJMCMC.2019070102

Mohamed, A., Traverso, S., Giaccone, P., Leonardi, E., & Niccolini, S. (2020). *Analyzing the performance of LRU caches under non-stationary traffic patterns*. https://arxiv.org/abs/1301.4909

Nguyen, T., & Vojnovic, M. (2011). Weighted proportional allocation. *Proceeding of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, 173-184. doi:10.1145/1993744.1993760

Pantisano, F., Bennis, M., Saad, W., & Debbah, M. (2014). In-network caching and content placement in cooperative small cell networks. *Proceeding of international conference on 5G for Ubiquitous Connectivity (5GU),* 128–133. doi:10.4108/icst.5gu.2014.258230

Pompili, D., Hajisami, A., & Tran, T. X. (2016). Elastic resource utilization framework for high capacity and energy efficiency in Cloud RAN. *IEEE Communications Magazine*, *54*(1), 26–32. doi:10.1109/MCOM.2016.7378422

Skourletopoulos, G., Mavromoustakis, C. X., & Mastorakis, G. (2017). Big Data and Cloud Computing: A Survey of the State-of-the-Art and Research Challenges. In C. X. Mavromoustakis (Ed.), *Advances in Mobile Cloud Computing and Big Data in the 5G Era* (pp. 23–41). Springer. doi:10.1007/978-3-319-45145-9_2

Tran, T. X., Hajisami, A., Pandey, P., & Pompili, D. (2017). Collaborative Mobile Edge Computing in 5G Networks: New Paradigms, Scenarios, and Challenges. *IEEE Communications Magazine*, *55*(4), 54–61. doi:10.1109/MCOM.2017.1600863

Tran, T. X., Hajisami, A., & Pompili, D. (2017). A Cooperative Hierarchical Caching Strategy for Cloud Radio Access Networks. *IEEE Network*, *31*(4), 35–41. doi:10.1109/MNET.2017.1600307

Wang, X., Chen, M., Taleb, T., Ksentini, A., & Leung, V. (2014). Cache in the air: Exploiting content caching and delivery techniques for 5G systems. *IEEE Communications Magazine*, *52*(2), 131–139. doi:10.1109/MCOM.2014.6736753

Zhao, N., Liu, X., Yu, F. R., Li, M., & Leung, V. C. (2016). Communications, caching, and computing oriented small cell networks with interference alignment. *IEEE Communications Magazine*, *54*(9), 29–35. doi:10.1109/MCOM.2016.7565184

Michael Pendo John Mahenge is a Tanzanian and Lecturer at the Department of Informatics and Information Technology (DIIT), College of Natural and Applied Sciences (CoNAS), Sokoine University of Agriculture, Tanzania. He was awarded a PhD in computer Science and Technology from Wuhan University of Technology, China in May, 2021, Master's degree in Information and Communication Science and Engineering from Nelson Mandela African Institution of Science and Technology (NM-AIST), Tanzania in 2014 and Bachelors' degree (BSc. informatics) from Sokoine University of Agriculture (SUA), Tanzania in 2011. He has co-authored a number of papers, conference proceedings and book chapters in the field of Mobile Computing, E-learning, M-learning and ICT4D published in peer reviewed journals and conference proceeding. His research interest is in Mobile Edge Computing (network resource optimization, offloading, resource scheduling and allocation, and content caching), application of artificial intelligence in plant disease identification and prediction, Mobile Cloud Computing applications particularly E-Learning, M-Learning, Healthcare and ICT4D. He has excellent interpersonal skills.

Edvin J. Kitindi received the B.Sc. degree in ICT management from Mzumbe University Morogoro Tanzania, in 2008. Then M.Eng. degree in electronics and communication engineering and Ph.D. in Communication and Information Systems from Chongqing University, Chongqing, China, in 2012 and 2018, respectively. He has been with the Sokoine University of Agriculture, Tanzania, as an Academic Staff of informatics, since 2008. His current research interests include wireless network virtualization, mobile networks' resources management, and ICT4D.