**Open Access Library Journal**
2018, Volume 5, e4854
ISSN Online: 2333-9721
ISSN Print: 2333-9705

# Hybrid Delay Optimization and Workload Assignment in Mobile Edge Cloud Networks

**Abdul Rasheed Mahesar, Abdullah Lakhan, Dileep Kumar Sajnani, Irfan Ali Jamali**

Southeast University, Nanjing, China
Email: abdulrasheed_mahesar@yahoo.com, Abdullah@seu.edu.cn, sajnani.dileep@gmail.com, jamali.irfan@yahoo.com

**How to cite this paper:** Mahesar, A.R., Lakhan, A., Sajnani, D.K. and Jamali, I.A. (2018) Hybrid Delay Optimization and Workload Assignment in Mobile Edge Cloud Networks. *Open Access Library Journal*, **5**: e4854.
https://doi.org/10.4236/oalib.1104854

**Received:** August 20, 2018
**Accepted:** September 8, 2018
**Published:** September 11, 2018





## Abstract

Nowadays, the usage of mobile devices is progressively increased. Until, delay sensitive applications (Augmented Reality, Online Banking and 3D Game) are required lower delay while executed in the mobile device. Mobile Cloud Computing provides a rich resource environment to the constrained-resource mobility to run above mentioned applications, but due to long distance between mobile user application and cloud server introduces hybrid delay (*i.e.*, network delay and process delay). To cope with the hybrid delay in mobile cloud computing for delay sensitive applications, we have proposed novel hybrid delay task assignment (HDWA) algorithm. The preliminary objective of the HDWA is to run the application on the cloud server in an efficient way that minimizes the response time of the application. Simulation results show that proposed HDWA has better performance as compared to baseline approaches.

## Subject Areas

Cloud Computing

## Keywords

Offloading, Machine Learning, Process Delay, Transmission Delay, Delay Constraint, Augmented Reality, Video Analytics

## 1. Introduction

An increasing proportion of mobile cloud application is developing day by day. Still, cloud computing provides the richer environment to run the computation workload of application for execution. Furthermore, more than thousands of running applications are by managing by rich resource cloud servers. While, each individual application is unparalleled and has different resource require-

ments. Nevertheless, due to long WAN far away cloud services from UE devices, it incurred hybrid latency (*i.e.*, transmission and process delay). The central requirements of latency bound application must meet lower response time. The new paradigm mobile edge computing (MEC) has been proposed by European Telecommunications Standards Institute [1]. Moreover, the purpose behind MEC is to enable the remote cloud service proximity closer to the mobile network and improve the application performance with lower response time [2]. The MEC architecture includes fog computing, cloudlet server and edge network bring cloud resources such as computing and memory capabilities at radio access network layer with lower network delay.

In this paper, we are studying the response time minimization problem of latency bound application, and task scheduling over hybrid computing server (cloudlet and remote cloud) infrastructure simultaneously [3]. Whereas, the response time is the amalgam of hybrid latency (*i.e.*, transmission and process delay). Furthermore, we divided the response time problem into three sub problems; first, application task scheduling; it is challenging to partition the application tasks dynamically and allocate the resources such as processing and memory for local execution or bandwidth for offloading computational tasks to the local cloudlet server for execution; second, optimal network route selection; every smart applications including vehicle traffic signal, security cameras and video analytics generated stream of data at network edge; it incurred the heavy burden on network for processing the data exceeding its limit. It is challenging to optimize the route selection for computational task offloading to the local cloudlet server with minimum transmission delay; third, resource allocation for offloaded task; while cloudlet is a collection of tiny data centers, if requested workload exceeds its limits it incurred by hybrid latency (queue and process delay). It is challenging to allocate the resource optimally at a cloudlet server with minimum hybrid latency.

- Proposed the dynamic application task scheduling framework, the ultimate goal of proposed framework which application task components either task execution locally or remaining part offloading to the cloudlet server, in order to minimize average execution time in this paper we have following contribution in this paper.
- Proposed the machine learning (reinforcement learning) based algorithm to choose the efficient, optimize network route with lower transmission delay for computational task offloading.
- Proposed the **HDWAH** heuristic which always allocates the optimal cloudlet to offloaded task with minimize response time including hybrid latency.

The rest of the paper is organized as follows. Section 2 elaborates related work and Section 3 explains the problem description and formalizes the problem under study. A heuristic is proposed for the considered problem in Section 4 which describes the propose algorithm and sequences. Section 5 evaluates the simulation part. Section 6 is about conclusions.

## 2. Related Work

Moreover, offloading cost is the important mechanism nowadays for mobile cloud application. Existing literature research has been proposed their frameworks to improve the offloaded cost at the mobile run time environment.

In [4] [5] [6] Mobile Assistance User Interface (MAUI) proposed the framework in which the objective was to minimize the device energy during computation offloading. MAUI framework is consisted two run time environments (such as mobile run time and cloud run time) to support mobile cloud application. MAUI has considered only mobile end run time performance it did not consider cloud end offloading cost. CloneCloud run time environment framework for computational offloading is proposed in [7]. The objective is to minimize the device energy consumption. It is highly dynamic technique for computational offloading in which mobile application services wrapped into a virtual machine. Application run time manager captured the full image of the application and offload to the cloud server for execution. However, mobile application and cloud server both have a virtual environment for the offloading and execution [8]. CloneCloud run time focused on mobile device execution time performance and did not consider the cloud server end. Focused on mobile device execution time performance and did not consider the cloud server end. Whereas, [9] [10] [11] [12] [13] and [14] proposed their based partial offloading whether compute intensive tasks are offloaded to the remote cloud based on network condition.

## 3. System Model

In this composition, the scenario is restricted by two cloudlets as shown in **Figure 1**. Whereas, different kinds of application are deployed with specific application services (*i.e.*, augmented reality, virtual reality, and real time video analytics). $u \in U$ Set of mobile users, $ck \in CK$ is cloudlet servers, $\xi k$ is the cache capacity of cloudlet server. All specified applications are wrapped into associated virtual machines (VMs). Local cloudlet are located beside mobile user base station, it is proximity closer to mobile users (UEs). The unit delay between UE and base stations is negligible. Two phases of unit delay in this account are calculated such as $\in U_{CK}$ and $\in CK_{RC}$. Furthermore, the response time minimization problem is the combination transmission delay and process delay is describing into different parts.

### 3.1. Task Characterization

A set a cloudlet server $CK = \{ck1, ck2, \cdots, ckn\}$ whereas, $ck \in CK$ and individual cloudlet has similar service rate and capacity $C_P$. A cloudlet is the collection of homogeneous virtual machines which are deployed at cloudlet data centers. Resources (*i.e.*, memory, CPU processing, storage and bandwidth) are similar in all set of cloudlets. If the requested user works excessive cloudlet capacity, then some tasks forwarded to the remote cloud for further execution.

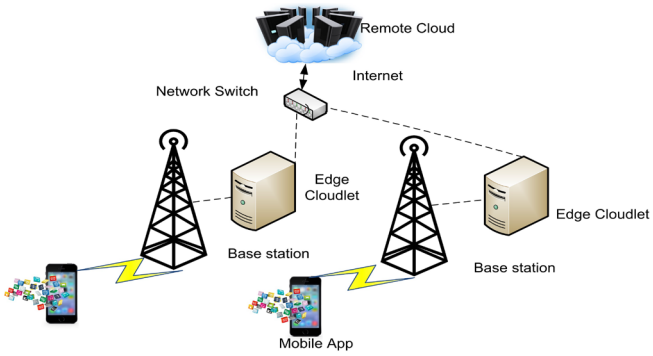


**Figure 1.** MECC cloudlet architecture.

### 3.2. Latency Analysis

The response time of application is calculated as input from user mobility and gets back their result; it is the round trip time. The hybrid latency (transmission delay and process delay) has whipped influence on task offloading. The term latency defines many kinds of delay such transmission delay, process delay (queue and execution delay) and propagation delay between inter-servers [15].

### 3.4. Transmission Delay

Transmission delay has thumping in influence on mobile data offloading, network congestions occurred due to many factors such as noise, interference and network traffic [16]. Most of the time preceding elements is not in our control. The dynamic of network traffic is destructive for data offloading, and take in the offloading benefits less efficient due to high transmission delay. In this process integer mixed programming language is used to identify the decision variable, for instance, The decision variable one shows the mobile user offloaded the computation tasks at local cloudlet with lower transmission delay otherwise zero. $X=\{x_{uck}:ck\in CK,u\in U\}$ with $x_{uck}=\{0,1\}$. Furthermore, $X$ is the decision variable and identify either binary variable $x_{uck}$ is accessed associated cloudlet ck for task execution or not and expressed as follows

$$Xuck=\begin{cases}1, \text{if } X_{uck}=1\\ 0, \text{otherwise}\end{cases} \tag{1}$$

To end with the average transmission delay is followed:

$$\sum_{u\in U}\sum_{ck\in CK}\in U_{CK}+\in CK_{RC} \tag{2}$$

### 3.5. Process and Queuing Delay

The system offloading takes randomly generated input with different arrival rates for I/O System, after being processed I/O System, the result will dispatch as departure [17]. The vector $Y=\{x_{uck}:ck\in CK,u\in U\}$ process delay and queue has signification important in mobile cloud application. The user application generated the workload randomly and followed by Poisson Process as shown in **Figure 2**. The arrival rate is represented by $\lambda_i$ and before scheduling the ap-

propriate resources to the requested, we are prioritizing the all workloads according to the given sequences (*i.e.*, earliest deadline first, short workload first, least slack time). Furthermore, workload scheduler schedule them on the optimal cloud server $k$, the cloud service rate is represented by the exponential distribution $\mu_k$. All the cloud servers are geographically distributed.

$$y_{uck} = \sum_{0}^{1} \tag{3}$$

By using queuing model [9] [18] cloudlet server has two important elements; Request and arrival rate of single user make single request, and the rate at which requests are generated at $\lambda_u$ arrival rates. Second, service rate and server, this is part of a cloudlet server system which has service rate $u_c k$. To evaluate the task queue wait time and process time, a system adhere by the Little's law [5] as follows:

$$C_{ck}^{u} = \frac{y_{uck} cm_u \lambda_u}{u_{ck} - \sum_{u=1}^{U} y_{uck} cm_u \lambda_u} \tag{4}$$

This equation $y_{uck} cm_u \lambda_u$ computation task offload to the cloudlet server whereas, $\frac{1}{u_{ck} - \sum_{u=1}^{U} y_{uck} cm_u \lambda_u}$ is the process and queue time in the system. Furthermore, average service rate of cloudlet must meet the service the result will dispatch as departure; the vector latency bounds such as $u_{ck} - \sum_{u=1}^{U} y_{uck} cm_u \lambda_u > 0$. If requested workload exceeds the capacity of cloudlet $C_p$ few of the tasks migrate to remote cloud for further execution as shown below:

$$RC_u^{ck} = \frac{\left(1 - \sum ck \in CK\right) cm_u \lambda_u}{RCu - \left(1 - \sum ck \in CK\right) \lambda_u}, \tag{5}$$

$\left(1 - \sum ck \in CK\right) cm_u \lambda_u$ workload allocation for service at remote cloud, and it must be $\left(1 - \sum ck \in CK\right) cm_u \lambda_u > 0$. However, $\theta_u$ is the processing capacity of local device to execute some small tasks on the local device? UEs workload offloaded to the proximity closer cloudlet server, the unit transmission delay follows:

$$T_U^{CK} = \sum_{ck \in CK} \in U_{CK} \left(y_{uck} cm_u + dw_u\right), \tag{6}$$

Local cloudlet located on the edge network and remote cloud situated at long WAN distributed network, therefore, the unit propagation delay describes as follows:

$$T_{CK}^{RC} = \sum_{ck \in CK} \in CK_{RC} \left(1 - x_{uck}\right) dw_u, \tag{7}$$

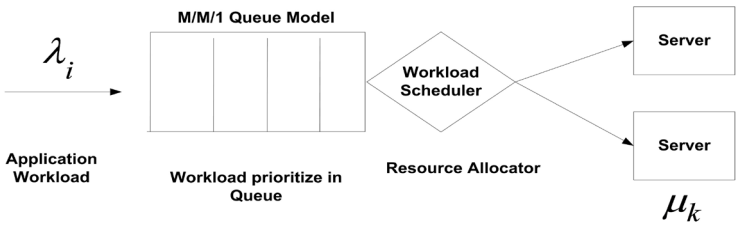


**Figure 2.** Process delay and queuing delay.

The final road map of application response time is the combination of transmission delay and process delay

And depict as go behind:

$$T = \sum_{u \in U} \left( C_{ck}^{u} + RC_{u}^{ck} + T_{C}^{CK} + T_{CK}^{RC} \right). \tag{8}$$

$$\lambda(x) = \lambda_0 1_x \le T \tag{9}$$

### 3.6. Problem Formulation

The response time minimization problem has decision variables and set constraints, system follow feasible solution in linear integer programming as follows:

$$\text{Minimize } T = \sum_{u \in U} \left( C_{ck}^{u} + RC_{u}^{ck} + \mathrm{T}_{\mathrm{C}}^{\mathrm{CK}} + \mathrm{T}_{\mathrm{CK}}^{\mathrm{RC}} \right) \tag{10}$$

$$\text{s.t. } \sum_{u=1}^{U} dw_u x_{uck} cm_u \le C_p, \forall ck \in CK, \forall u \in U \tag{11}$$

$$u_{ck} - \sum_{u=1}^{U} y_{uck} \lambda_u > 0, \forall ck \in CK, \forall u \in U \tag{12}$$

$$x_{uck} = \{0,1\}, \forall ck \in CK, \forall u \in U \tag{13}$$

$$\sum_{u \in U} x_{uck} = 1, \forall ck \in CK \tag{14}$$

Equations (10) (11) illustrate that the arrival rate UEs with computational workload must be less than the service rate and capacity of cloudlet as system become more and more stable. Whereas, Equations (5) (12) (13) exemplify single workload if only allocated to single cloudlet as minimizing the chances of overhead at the system is the minimization problem; it is well known as NP-hard problem.

## 4. Hybrid Delay Workload Assignment (HDWA)

To satisfy the constraints (13), (14) and (15), we have designed the efficient **HDWA** to solve the problem. The primary objective of **HDWA**, iteratively chooses the optimal network path and cloudlet, and allocates the user computational task according to the given deadline. In algorithm 1, first we initialize end user workload assignment Z to cloudlet in step-2-3, in step-5 iteratively select the optimal cloudlet, which has not been influenced by disk bound, and allocate user with that optimal one cloudlet, in order to reduce the average response time of computational tasks; in step-7-8 allocate the resources offloaded tasks at the server which has minimum lateness and completed within deadline; step-9 terminate the loop, if all requested user has been allocated with their respective cloudlet.

The time complexity of this algorithm $o(I \mid \log n \mid k)$ whereas $I$ expressed the user workload, $K$ the number of iterations in the system, till allocated all user workloads in the system.

**Algorithm 1: HDWA Algorithm**

**Input:**

$$T = \sum \mu \in U\left(C_{ck}^{u} + RC_{u}^{ck} + T_{U}^{CK} + T_{CK}^{RC}\right);$$

Arrival rate $\lambda u$
Service rate of cloudlet $\mu_c k$;
Output: Minimize T of application S
1 begin $u^* = 1$
2 Initialization T=0
3 EUs Workload arrival rate organized in descending Order with respect ot service rate;
4
5 While $(u^* \lhd |U|)do$
6 choose the optimal way to minimize the application response time
7 $$T^* = \sum u \in U\left(C_{ck}^{u} + RC_{u}^{ck} + T_{U}^{CK} + T_{CK}^{RC}\right) \le b;$$
8 assign $\mu^* = \mu^* + 1$
$x_{u^* ck^*} = 1;$
9 return T

Algorithm 1 is iterative in nature. Whereas, in step 1 we initialize the application workload zero; in steps 2-3, in step-5 iteratively select. The optimal cloudlet, which has not been influenced by disk bound, allocate user with that optimal one cloudlet, in order to reduce the average response time of computational tasks. In steps 7-8 we allocate the resources offloaded tasks at the server which has minimum lateness and completed within deadline, step 9 terminate the loop, if all requested user has been allocated with their respective cloudlet. The time complexity of this algorithm $o(I|\log n|k)$ whereas $I$ expressed the user workload, $K$ the number of iterations in the system, till allocated all user workloads in the system. The application response time $T$ must be less than application deadline $b$.

## 5. Experiment Analysis

In the results analysis, we are comparing the performance and evaluation of proposed Latency Aware Task Assignment (**HDWA**) with the conventional approach [9] [10] [11] [18] and [19] Baseline approach [1] [2] [4] [10]. Based in **Table 1** numerical simulations evaluated the performance of the proposed algorithm with conventional and base-line approach of giving parameters. We have used core i7 dual core processor, 24-GB Ram, 2000 GB for communication and computation in our experiment.

### 5.1. Simulation Environment

We have designed a simulation framework based on SimPY python API [14] as shown in **Figure 3**, and moving everywhere dataset user application [14] as shown in **Figure 6**. Online cloud server purchased for application installation, and local cloudlet install at local city on two places. The distance between two cloudlet is 2 Km, We have examined the Pokemon (A.R) and Holo (A.R) mobile applications are on our system. These applications installed by many university students and played during peak and normal hours. Our monitoring system

checks the system performance in every 1 hour. In the framework we have managed two kinds of queues (network and process) simultaneously. Overall system monitors the transmission delay, process delay and server capacity handles the different arrival of users.

**Table 1.** Experiment parameters.

| Notation | Explanation |
|---|---|
| $\lambda_i$ | [1; 20] = s |
| Mobile users (MUs) | [100; 1000] Numbers |
| $\psi U_i$ | [0:2; 1:5] $10^2$ |
| $C_i$ | [1024; 5120] KB |
| $d_i$ | [1024; 10; 240] KB |
| $\in UK$ | $2_m$s = MB Augmented Reality |
| $KC$ | $4_m$s = MB Propagation Latency |
| $\mu_k$ | [2; 3:5] $10^4$ |
| Bandwidth (B) | [50; 100] = mbps |
| $\xi_k$ | [2000] GB |

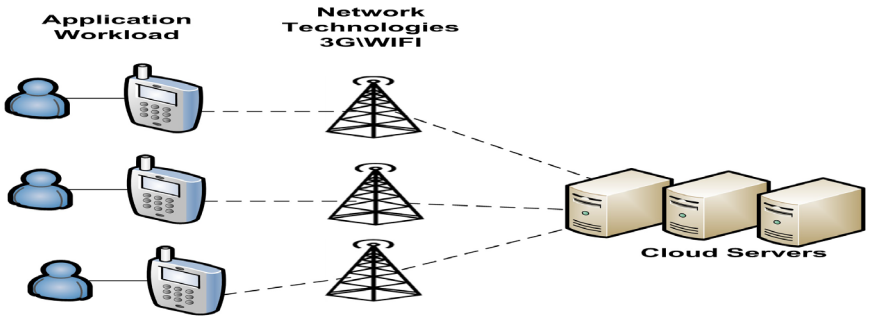


**Figure 3.** Task process scheduling.

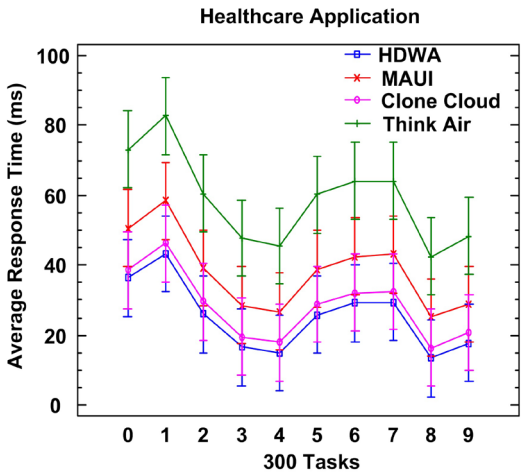


**Figure 4.** Application response time.

### 5.2. Proposed Algorithm versus Existing Approaches

Comparison of proposed algorithm **HDWA** and existing approaches are followed.

#### 5.2.1. Conventional Approach

The conventional approach has a basic offloading scheme either executes of computation task locally on the device or offloaded to remote cloud for additional processing. This approach does not handle the end to end latency due to cloud services are multiple hops away from mobile device [1].

#### 5.2.2. Baseline Approaches

Baseline Approaches brings the computing resources at the mobile network edge, application scheduler randomly decides either computation task execute on the mobile device or offload to the server. This approach attempts to reduce the application response time at some extent, but requested computation workload increase its capacity, it leads lateness in systems such as queue and process delay.

#### 5.2.3. Proposed Approach (HDWA)

The proposed approach randomly takes the advantage of the local task computation at the device and local cloudlet server. If the requested computation tasks exceed the limit of cloudlet dynamically it migrates some computation tasks on the remote server for further execution.

### 5.3. Performance Evaluation of Different Algorithms

In order to achieve the objective of proposed strategy works better as compared to existing approaches. Figure 7 demonstrates the performance and comparison of process delay at local device and the server. We have monitored mobile users' range 50 to 500 with computational tasks. Simulation parameters are declared in Table 1 and evaluated the performance of the proposed algorithm; it shows that **HDWA** 50% has a lower delay as compared to conventional and baseline approach. Figure 4 and Figure 5 show that proposed algorithm **HDWA** minimizes process delay as compared to baseline approaches. Figure 6 shows that our proposed scheme is more optimized rather than existing baseline schemes. Figure 7 demonstrates the performance and comparison of transmission delay during peak hours and normal hours. We have tested user mobile users randomly arrival rates with $\lambda_i$ with computational tasks, as we mentioned the bandwidth (B) parameter in Table 1 and evaluated the performance of the proposed algorithm, it shows that **HDWA** 60% lower transmission delay as compared to Conventional and Baseline approach during peak hours.

Figure 7 shows that **HDWA** works better during peak and normal hours, whereas, task deadline completion demonstrate that **HDWA** has a better response time related to test applications; we have examined two applications with different latency threshold (0.3 and 0.6). The average response time of application is the sum of heterogeneous latency (transmission and process delay) and local and server execution time.

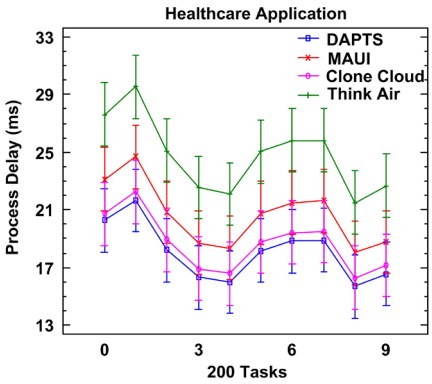


**Figure 5.** Process delay.

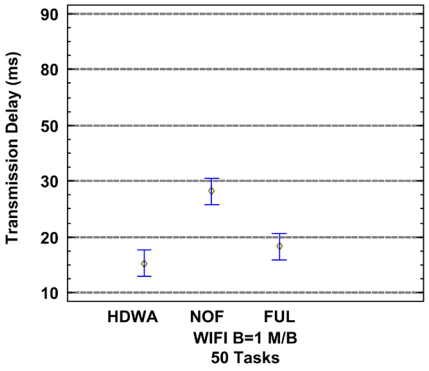


**Figure 6.** Average responses on WIFI network.

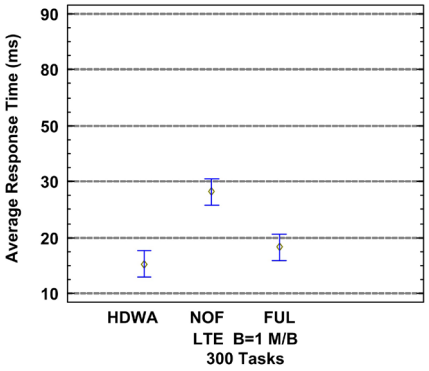


**Figure 7.** Average Response on LTE network.

## 6. Conclusion

In this study, we have examined the problem response time minimization problem with heterogeneous latency (transmission and process delay). We have focus to support the latency sensitive application with lower response time. Our proposed methods and framework has better performance and progress as compared to existing methods and frameworks. In this paper, we have studied the minimization problem, divided it into three sub problems and efficiently solved all sub problems with minimum response time. This problem is NP-complete problem. By integer linear programming we proof it, and our proposed solution

always satisfies the feasible solution. In future includes the detail study of above latency aware computation offloading problem with more detail constraints such as (load-balancing latency, communication energy, transmission energy) via geographically distributed cloudlet network.

## Acknowledgements

The team members Abdul Rasheed Mahesar, Abdullah Lakhan, DIleep Kumar Sajnani and Irfan Ali Jamali work hard to complete this paper, and proposed a novel idea as compared to an existing offloading scheme. Hope In future we will achieve many milestones in our lives.

## Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

## References

[1] Kumar, K. and Lu, Y.-H. (2010) Cloud Computing for Mobile Users: Can Offloading Computation Save Energy? *Computer*, **43**, 51-56. https://doi.org/10.1109/MC.2010.98

[2] Chen, X. (2015) Decentralized Computation Offloading Game for Mobile Cloud Computing. *IEEE Transactions on Parallel and Distributed Systems*, **26**, 974-983. https://doi.org/10.1109/TPDS.2014.2316834

[3] Chen, M., Yin, Z., Li, Y., Mao, S.W. and and Leung, V.C.M. (2015) EMC: Emotion-Aware Mobile Cloud Computing in 5G. *IEEE Network*, **29**, 32-38. https://doi.org/10.1109/MNET.2015.7064900

[4] Waseem, M., Lakhan, A. and Jamali, I.A. (2016) Data Security of Mobile Cloud Computing on Cloud Server. *Open Access Library Journal*, **3**, 1-11.

[5] Deng, S.G., Huang, L.T., Taheri, J. and Zomaya, A.Y. (2015) Computation Offloading for Service Workflow in Mobile Cloud Computing. *IEEE Transactions on Parallel and Distributed Systems*, **26**, 3317-3329. https://doi.org/10.1109/TPDS.2014.2381640

[6] Zhang, W.W., Wen, Y.G. and Wu, D.O. (2015) Collaborative Task Execution in Mobile Cloud Computing under a Stochastic Wireless Channel. *IEEE Transactions on Wireless Communications*, **14**, 81-93. https://doi.org/10.1109/TWC.2014.2331051

[7] Kosta, S., Andrius, A., Hui, P., Mortier, R. and Zhang, X.W. (2012) ThinkAir: Dynamic Resource Allocation and Parallel Execution in the Cloud for Mobile Code Offloading. 2012 *Proceedings IEEE INFOCOM*, Orlando, 25-30 March 2012, 945-953. https://doi.org/10.1109/INFCOM.2012.6195845

[8] Chun, B.-G., Ihm, S., Maniatis, P., Naik, M. and Patti, A. (2011) CloneCloud: Elastic Execution between Mobile Device and Cloud. *Proceedings of the Sixth Conference on Computer Systems*, Salzburg, 10-13 April 2011, 301-314. https://doi.org/10.1145/1966445.1966473

[9] Soomro, A.A., Lakhan, A.R. and Khan, A. (2011) The Secure Data Storage in Mobile Cloud Computing. *Journal of Information and Communication Technolog*, **6**, 69-76.

[10] Rahimi, M.R., Jian, R., Liu, C.H., Vasilakos, A.V. and Venkatasubramanian, N.

(2014) Mobile Cloud Computing: A Survey, State of Art and Future Directions. *Mobile Networks and Applications*, **19**, 133-143. https://doi.org/10.1007/s11036-013-0477-4

[11] Shiraz, M. and Gani, A. (2014) A Lightweight Active Service Migration Framework for Computational Offloading in Mobile Cloud Computing. *The Journal of Supercomputing*, **68**, 978-995. https://doi.org/10.1007/s11227-013-1076-7

[12] Chen, X., Lei, J., Li, W.Z. and Fu, X.M. (2016) Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing. *IEEE/ACM Transactions on Networking*, **24**, 2795-2808. https://doi.org/10.1109/TNET.2015.2487344

[13] Nkosi, M.T. and Mekuria, F. (2010) Cloud Computing for Enhanced Mobile Health Applications. 2010 *IEEE Second International Conference on Cloud Computing Technology and Science*, Indianapolis, 30 November-3 December 2010, 629-633.

[14] Yao, Y.-C. (1987) Approximating the Distribution of the Maximum Likelihood Estimate of the Change-Point in a Sequence of Independent Random Variables. *The Annals of Statistics*, **15**, 1321-1328.

[15] Barbarossa, S., Sardellitti, S. and Lorenzo, P.D. (2014) Communicating While Computing: Distributed Mobile Cloud Computing over 5G Heterogeneous Networks. *IEEE Signal Processing Magazine*, **31**, 45-55. https://doi.org/10.1109/MSP.2014.2334709

[16] Sardellitti, S., Scutari, G. and Barbarossa, S. (2015) Joint Optimization of Radio and Computational Resources for Multicell Mobile-Edge Computing. *IEEE Transactions on Signal and Information Processing over Networks*, **1**, 89-103.

[17] Huang, D.J., Xing, T.Y. and Wu, H.J. (2013) Mobile Cloud Computing Service Models: A User-Centric Approach. *IEEE Network*, **27**, 6-11. https://doi.org/10.1109/MNET.2013.6616109

[18] Kovachev, D., Cao, Y.W. and Klamma, R. (2011) Mobile Cloud Computing: A Comparison of Application Models. arXiv:1107.4940v1.

[19] Flores, H. and Srirama, S. (2013) Adaptive Code Offloading for Mobile Cloud Applications: Exploiting Fuzzy Sets and Evidence-Based Learning. *Proceeding of the Fourth ACM Workshop on Mobile Cloud Computing and Services*, Taipei, 25-28 June 2013, 9-16. https://doi.org/10.1145/2497306.2482984