KYBERNETIKA — VOLUME *59* (2023), NUMBER 1, PAGES 88–109

CONSTRAINED K-MEANS ALGORITHM FOR RESOURCE ALLOCATION IN MOBILE CLOUDLETS

Rasim M. Alguliyev, Ramiz M. Aliguliyev, and Rashid G. Alakbarov

With the rapid increase in the number of mobile devices connected to the Internet in recent years, the network load is increasing. As a result, there are significant delays in the delivery of cloud resources to mobile users. Edge computing technologies (edge, cloudlet, fog computing, etc.) have been widely used in recent years to eliminate network delays. This problem can be solved by allocating cloud resources to the cloudlets that are close to users. The article proposes a clustering-based model for the optimal allocation of cloud resources among cloudlets. The proposed model takes into account user activity, usage frequency of cloud resources, the physical distance between users and cloud resources, as well as the storage capacity of cloudlets for optimal allocation of cloud resources in cloudlets. The proposed model was formalized as a constrained k-means method and an algorithm was developed to solve it. The MATLAB 2022a toolkit was used to evaluate the efficiency of the proposed algorithm. The obtained results revealed that the algorithm is promising.

Keywords: mobile cloud computing, edge computing, cloudlet, cloud resources, constrained k-means

Classification: 90B80, 62H30, 90B18

1. INTRODUCTION

Recently developed mobile applications require more computing resources. Mobile devices have limited computing and memory resources due to their portable size. Limited resources of mobile devices (processing power, battery life, and storage capacity) create problems for users in using software applications that require large computing and memory resources. Cloud computing is used to solve these problems. Cloud computing systems are an affordable computing model that provides a rich range of applications and services, in addition to meeting users' requirements faster. Cloud technologies provide solutions to the problems of mobile users on cloud servers with sufficient computing and memory resources. Mobile cloud computing (MCC) is widely used to provide computing and memory resources to mobile devices. MCC was a new paradigm created from the integration of network and computing clouds, providing computing resources for solving problems of mobile users requiring large computing resources [30]. MCC is a network infrastructure that processes user applications on remote cloud servers. MCC is a new

DOI: 10.14736/kyb-2023-1-0088

distributed computing system that provides processing power and memory transfer to cloud servers located away from mobile devices. MCC can also be defined as a combination of a mobile network and cloud computing where users can access applications and services over the Internet. In addition, MCC technologies have helped to address many of the limitations of mobile devices. MCC eliminates the following problems:

- provides mobile users with necessary computing resources;
- allows users to use mobile applications that require high computing resources;
- provides prolongation of battery life of mobile devices;
- provides rapid data processing;
- reduces the probability of data loss due to the use of cloud servers;
- increases the security capabilities of user data;
- etc.

In traditional centralized cloud computing, issues are solved on remote cloud servers, causing the Internet to overload and delays in delivering results to the user. At the same time, various applications are used on mobile devices (for example, watching movies, online shopping, interactive games, social networking services, email services, browsing web pages, etc.). Extensive use of these services by mobile users leads to the rapid depletion of power sources (batteries) of mobile devices. On the other hand, when issues are resolved on remote servers, mobile devices run out of power even more quickly. Recently, edge computing systems (cloudlets, edge, fog computing, etc.) based on the infrastructure of cloud computing systems have been used to solve the above-mentioned problems. Edge (periphery) computing systems are established near places or mobile equipment where data is generated. Edge computing systems expand the boundaries of cloud computing systems by directing computing resources to the nearest point where problems arise [15].

Edge computing is presented as a paradigm that expands the cloud computing platform. The use of fog computing in conjunction with cloud technologies helps reduce the processing load at the data processing center. Servers that are close to users, process data and send only the most important data to data processing centers (remote servers). Edge computing servers are close to the end user and provide them with the necessary computing resources. Edge calculations are closer to the user, so less time is required to process and transmit data. As services and features are closer to users in fog computing, users can also achieve a higher quality of service (QoS).

In edge computing, the data is processed on servers close to the user, which in turn eliminates delays in the network infrastructure. The purpose of edge computing is to increase efficiency by processing data where they are generated and to reduce the amount of data transmitted to remote cloud servers [28]. This also improves security issues that cause problems in the network infrastructure. Multiple user applications are processed on remote servers and the transfer of results to them is inefficient. Mobile Edge Computing (MEC) systems have recently been used to overcome these problems.

MEC is a newly developed paradigm to meet the computing needs of mobile devices. MEC delivers computing and memory resources to the boundaries (edges) of the mobile network, which allows mobile users to quickly process applications that require high computing resources [21].

The processing devices (cloudlets) used in edge computing are usually located near the base stations of mobile networks (network access point-AP), and mobile users solve problems by connecting to them via a wireless communication channel (4G/5G, Wi-Fi). Non-homogeneous computers (servers, laptops, tablets, etc.) are used to create cloudlets. Compared to cloud computing, servers in mobile edge computing systems are located on the edge of the network, which provides faster solutions to mobile users' problems and faster delivery of results to mobile devices. During the widespread use of 5G communication technologies, a large volume of data uploaded to the main network will cause serious network delays. Mobile edge computing allows data to be processed over the network, which can effectively reduce network latency while protecting the security of user data. Therefore, mobile edge computing systems are a solution developed to reduce power consumption, and network delays and provide high reliability in mobile devices. The main goal of the MEC is to reduce network delays by bringing computing resources from the wide area network (WAN) to edge network resources [1].

In Mobile Edge Computing (cloudlet-based mobile networks) networks, Resource Management Center (RMC) directs user applications to any cloudlet that has free resources and they're executed there. However, the issue of proximity between users and cloudlets is not considered. On the other hand, if the applications in cloudlets are not optimally distributed in a balanced manner (some of the cloudlets are fully loaded, and some are left empty), the execution time of the task is extended and the power supply of the mobile device is quickly depleted.

Thus, the above-mentioned problems increase the power consumption of mobile devices and network delays. Mentioned problems can be solved by optimally placing frequently used applications between cloudlets with high user activity. If frequently used applications are solved in cloudlets close to users, both the time spent on solving the problem and the delays are reduced. The proposed tasks scheduling strategy provides reducing power consumption and delays by loading the resources to appropriate cloudlets. Thus, the article proposes a clustering model to solve the above-mentioned problems. The major feature of the proposed model is that taking into account the activity of users, usage frequency of cloud resources, and the physical distance between users and cloud resources it optimally allocates the cloud resources among multiple cloudlets. The model allows a balance of the workload between multiple cloudlets and therefore, minimizes the energy consumption and latency in mobile devices.

The rest of the paper is organized as follows. Section 2 describes a literature review. Section 3 presents the architecture of cloudlet-based MCC. The problem statement, all the notations, and definitions are given in Section 4. Section 5 presents the proposed constrained k-means algorithm. The experimental result is given in Section 6. Finally, the conclusion is provided in Section 7.

2. RELATED WORKS

The systematic and comprehensive review of resource planning in mobile edge computing networks was provided by Luo et al. [20]. Yuyi et al. [34] first of all analyzed the problems of reducing power consumption and network latency in the MEC systems built between users and cloud servers and then presented a comprehensive review of the state-of-the-art MEC research from the communication perspective. Another comprehensive overview and research outlook of MEC was provided by Mach and Becvar [21] with a focus on architecture and computation offloading. The main part of the paper was focused on works dealing with computation offloading to the MEC. They first described major use cases and reference scenarios where the MEC is applicable, after that reviewed existing concepts integrating MEC functionalities to the mobile networks and discussed standardization of the MEC.

Yang et al. [33] studied a distributed computation offloading strategy for a multi-device and multi-server system in small-cell networks and proposed an optimization model for joint minimization of energy consumption and latency in mobile devices. By leveraging the Lyapunov technique, a novel framework OPEN (**O**nline **PE**er offloadi**N**g) was proposed by Chen et al. [14], for the optimization of edge computing performance under energy constraints of small-cell networks. The framework OPEN assumes that the task's arrival rates are precise, which may not hold for all network systems. Hu et al. [16] formulated the task offloading problem as a constrained NP-hard optimization problem. To achieve a near-optimal solution the authors designed a task-offloading algorithm named MEFO (Maximum Efficiency First Ordered).

To compute offloading decision-making problems among users for mobile-edge cloud computing, Chen et al. [13] modeled the problem as a multi-user computation offloading game. Multi-user computation offloading game formulation, analysis of computation offloading game properties and distributed computation offloading algorithm design are the main results of the paper [13]. Lin et al. [19] considered computation offloading and resource allocation in an edge computing network with the objective to minimize the long-term average response time delay under the constraints of long-term averages of computation and power usage. The problem was formulated as an upper-bound optimization problem and a distributed an algorithm based on the branch-and-bound method was developed to solve the optimization problem.

To minimization of the average task completion time for each device, Wang et al. [32] designed a multi-agent imitation learning-based computation an offloading algorithm in pervasive edge computing networks. The authors modeled the task scheduling issue as an optimization problem by considering both the communication and computation abilities of edge devices. Mach and Becvar [21] proposed the edge computing architecture, where task offloading schemes, resource allocation, and user mobility factors are taken into account based on QoS parameters. Shen et al. [27] presented a task scheduling mechanism to optimize energy consumption in a cloud environment. This mechanism allowed for more efficient use of resources by finding a correlation between the solution time and energy consumption. In [17], Liao et al. considered the scenario of waiting in line for a mobile application tasks and the selection of reasonable computing offloading decisions, when mobile devices' power is limited in the edge computing system, to reduce the energy consumption of mobile devices and improve the endurance of mobile devices.

For minimization of latency and energy mobile consumption, Sachula et al. [25] considered multilayer MCC architecture consisting of a remote cloud server, cloudlet, and mobile terminal. They formulated the wireless bandwidth and computing resource allocation model as a triple-stage Stackelberg game and developed an iterative algorithm to obtain Stackelberg equilibrium. Zhang et al. [35] formulated the integrated resource allocation problem for delay-sensitive and delay-tolerant applications as a multi-directed acyclic graph scheduling problem and proposed a heuristic algorithm called LA-RATS (**L**oad-**A**ware **R**esource **A**llocation and **T**ask **S**cheduling strategy) for resource allocation and task scheduling in the MCC system. In [26], Sajnani et al. proposed a multi-layer latency-aware workload assignment strategy to tackle the optimal allocation of mobile users' workloads into cloudlets with lower Service Delay. For efficiently delivering the processed data to the user device based on the prediction of the user's current location, Shreya et al. [29] proposed a real-time cloud-fog-edge IoT architecture, named Mobi-IoST (Mobility-aware Internet of Spatial Things). Three-layer (Sensors as IoT devices; Edge device and fog device; Local and remote cloud servers) architecture IoT-F2N (IoT using Femtolet-based Fog Network), to reduce delay and energy consumption was proposed by Mukherjee et al. [23]. The major feature of this architecture is that the data obtained from sensors were processed and maintained inside the edge and fog devices. The experiment results showed that IoT-F2N is energy-efficient as well as reduces delay than the existing cloud-centric IoT architectures. Lin et al. [18] analyzed the task scheduling problems for edge computing in clouds and proposed the Petrel algorithm.

To solve cloudlet scheduling problems in a cloud computing environment, Ala'anzy et al. [6] proposed a novel locust-inspired metaheuristic optimization algorithm. The main goal of the authors in this paper was that the tasks should be allocated on the virtual machines (VMs) in order to minimize the makespan and waiting time and maximize resource utilization. Azad and Navimipour [9] proposed a combination of cultural and ant colony optimization algorithms for an efficient task allocation algorithm taking into consideration two important criteria, the makespan, and energy consumption. Nasr et al. [24] introduced a two-step algorithm for scheduling VMs in a cloud computing environment for load balancing on VMs and minimization of processing time. To achieve desired task scheduling, minimize executing time, and improve the QoS of the clouds, Zhang and Zhou [36] presented a two-stage strategy for task scheduling among heterogeneous VMs. In the first stage, a Bayes method was used to classify tasks based on historical scheduling data. In the second stage, tasks were allocated on VMs dynamically. An algorithm for selecting VMs to provide task solutions in the cloud with minimal energy consumption was presented by Bindu et al. [11].

Ahmed et al. [2] analyzed the effect of network-centric parameters on the runtime application migration and on the execution of applications in MCC. By investigating the runtime application migration, the authors concluded that application characteristics and dynamic conditions of the network significantly affect the application performance metrics. In [8], Asghar and Jung surveyed the scheduling strategies in the context of edge cloud computing in various aspects such as advantages and disadvantages, QoS parameters, and fault tolerance. It also surveyed such scheduling approaches to evaluate which one is feasible under what circumstances.

Alakbarov and Alakbarov [5] presented the balanced placement of mobile users' queries taking into consideration the location and technical capabilities of cloudlets located near base stations of Wireless Metropolitan Area Networks (WMAN). In [22], Mike et al. explored the deployment of cloud services and the distribution of mobile users to cloud services in WMAN. An algorithm was proposed to solve the problem, which allows hosting cloud services in WMAN regions with high user density and assigning mobile users to hosted cloud services while balancing their workload. Alakberov [3] considered the balanced distribution of the tasks in the cloudlet network and proposed a strategy to select cloudlets based on users' requirements. Alakberov [4], proposed a strategy for the selection of high-performance cloudlets considering the types of applications.

3. ARCHITECTURE OF CLOUDLET-BASED MOBILE COMPUTING CLOUDS

Cloud-based mobile edge computing is widely used to reduce network latency and power consumption on mobile devices. Selection of the most suitable cloudlet that allows users to quickly run applications in a set of cloudlets remains a big challenge. The architecture of a hierarchically structured cloudlet-based mobile external computing system in a wireless metropolitan area network environment is shown for efficient use of cloudlet resources (Figure 1). The Resource Management Center (RMC) was used to ensure the optimal distribution of applications on the cloud servers in the Cloudlet network.

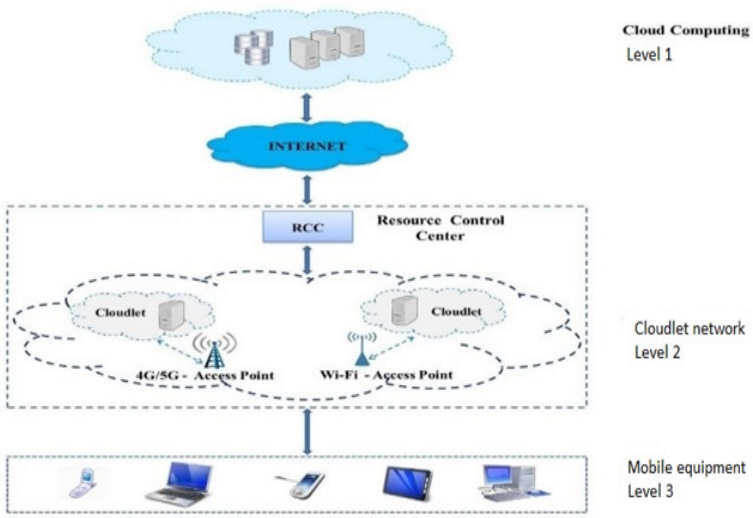

Fig. 1. An architecture of cloudlet-based mobile computing network.

On the 1st level, the servers of the cloud computing system are located, on the 2nd level, cloudlets are placed near the base stations, and on the 3rd level and mobile devices are located. The RCC, which is used in the 2nd level of the hierarchically structured network contains information about the location of cloudlets and their technical

capabilities. The task of the RCC is to store the data identifiers (ID) of all existing cloudlets and mobile users in the cloudlet network in the form of a chart and to direct the applications to the cloudlets according to the user's request. Data on applications used by users is collected in the created network. Based on the mobile user's request, applications are downloaded from remote cloud servers to the RCC of the cloudlet network. The management server in the RCC determines the cloudlet in which the mobile user's issue should be resolved based on the chart in the server. The central server also provides data exchange between cloudlets or between users and cloudlets. The control center contains information about the computer hardware used to create cloudlets. At the same time, the control center collects information about the technical capabilities of cloudlets and to which cloudlet users (stationary) are located close. Therefore, according to the incoming request, the software add-on called from the cloud servers is placed in the cloudlet that meets the user's requirements. Mobile internet users download and use their applications on the cloudlets that are close to them, which in turn frees the network from overloading. This architecture provides a solution to some of the stated problems (problem solution time, power consumption, delays in the communication channel, disconnections, etc.), although partially. Thus, cloudlets are created near the base stations of mobile operators in order for users to effectively use the services of mobile cloud technologies. The advantages of using cloudlets include fast service access, support mobility, and reduced roaming costs.

The purpose of mobile edge computing (cloudlet-based network) created in a wireless metropolitan area network (WMAN) environment rapidly processes the users' issues and reduces network delays. In a wireless metropolitan network, the presence of many mobile users and the solution to their problems on remote cloud servers cause delays in the network. Edge computing is used to solve this problem. Placing applications with a high usage frequency in cloudlets located near users who actively use these applications is a topical issue.

Thus, in the mobile edge computing environment mentioned in the article, a the planning model is proposed that ensures the reduction of processing time, network delays, and power consumption of mobile devices by optimal allocation of cloud resources in cloudlets.

4. PROBLEM STATEMENT, NOTATIONS, AND DEFINITIONS

Considering following:

- the physical distance between users and cloud resources (applications);
- usage frequency of cloud resources;
- activity of users;
- volume of resources;
- storage capacity of servers,

cloud resources should be allocated in cloudlets and users assigned to servers in such a way, that users are satisfied with the declared quality of service of cloud service providers

(CSPs). The solution to this problem can allow us to achieve the following goals at the same time:

- to minimize the time that users spend downloading and processing cloud resources, i. e., reducing user costs (because a long time means a lot of costs);
- late depletion of power supply of users' mobile devices, i. e., increasing power sustainability;
- avoiding overload of network structure more than intended, i. e., reducing network delays.

Therefore, considering the abovementioned, a mathematical model was proposed to download the cloud resources on the cloudlet, rapidly receive results, reduce network delays, and load balance of cloudlets.

Notations. The following notations are introduced:

- $\mathbb{U} = \{U_i \,|\, i = 1, \ldots, n\}$ – the set of mobile users, where n is the number of mobile users;
- $\mathbb{R} = \{R_j \,|\, j = 1, \ldots, m\}$ – the set of the most frequently requested cloud resources by users, where m is the number of resources;
- $\mathbb{v} = \{v_j \,|\, j = 1, \ldots, m\}$ – the volume of cloud resources;
- $\mathbb{F} = \|f_{i,j}\|$ – the frequency matrix of the use of cloud resources by users;
- $\mathbb{D} = \|d_{i,j}\|$ – the physical distance matrix from users to cloud resources;
- $\mathbb{S} = \{S_q \,|\, q = 1, \ldots, k\}$ – the set of servers, where k is the number of servers;
- $\mathbb{V} = \{V_q \,|\, q = 1, \ldots, k\}$ – the storage capacity of servers;

where v_j is the volume of resource R_j; V_q is the storage capacity of server S_q; $f_{i,j}$ is the usage frequency of the resource R_j by user U_i; $d_{i,j}$ is the physical distance from the user U_i to the resource R_j. Here, it is assumed that $m \gg n \gg k$.

Definition 1. (User activity.) The activity of mobile users (α_i) is defined as follows:

$$\alpha_i = \frac{\sum\limits_{j=1}^{m} f_{i,j}}{\sum_{p=1}^{n} \sum\limits_{q=1}^{m} f_{p,q}}, \quad i = 1, \ldots, n. \tag{1}$$

From Eq. (1) follows that:

$$\sum_{i=1}^{n} \alpha_i = 1. \tag{2}$$

Definition 2. (Resource usage frequency.) Analogously the relative usage frequency of cloud resources (β_j) can be calculated as follows:

$$\beta_j = \frac{\sum\limits_{i=1}^{n} f_{i,j}}{\sum_{p=1}^{n}\sum_{q=1}^{m} f_{p,q}}, \quad j = 1, \ldots, m. \tag{3}$$

From Eq. (3) we derive:

$$\sum_{j=1}^{m} \beta_j = 1. \tag{4}$$

As can be seen, there are two types of the distance between the mobile user $U = \{U_j \mid i = 1, \ldots, n\}$ and the cloud resource $R = \{R_j \mid j = 1, \ldots, m\}$: the semantic $\mathrm{F} = \|f_{i,j}\|$ and the physical $D = \|d_{i,j}\|$

Before formalizing the model, let's normalize the matrices $\mathbb{F}$ and $\mathbb{D}$. Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1.

Definition 3. (Matrix normalization.) The normalization of the matrices $\mathbb{F}$ and $\mathbb{D}$ is done on the basis of sum-based normalization:

$$\lambda_{i,j} = \frac{f_{i,j}}{\sum\limits_{p=1}^{m} f_{i,p}} \quad , \quad i = 1, \ldots, n; \quad j = 1, \ldots, m; \tag{5}$$

$$\delta_{i,j} = \frac{d_{i,j}}{\sum\limits_{p=1}^{m} d_{i,p}}, \quad i = 1, \ldots, n; \quad j = 1, \ldots, m. \tag{6}$$

From Eqs.(5) and (6) follows that:

$$\sum_{j=1}^{m} \lambda_{i,j} = 1, \quad i = 1, \ldots, n; \tag{7}$$

$$\sum_{j=1}^{m} \delta_{i,j} = 1, \quad i = 1, \ldots, n, \tag{8}$$

where $\lambda_{i,j}$ and $\delta_{i,j}$ are normalized values of the elements $f_{i,j}$ and $d_{i,j}$ of the matrices $\mathbb{F}$ and $\mathbb{D}$, respectively.

5. CONSTRAINED K-MEANS ALGORITHM

This section proposes a clustering approach for the optimal placement of resources ($\mathbb{R}$) among servers ($\mathbb{S}$) and optimal assignment of users ($\mathbb{U}$) to the cloud servers. Before clustering each resource R_j should be represented as a feature vector $\rho_j = \{\rho_{i,j} \mid i = 1, \ldots, n\}$, $j = 1, \ldots, m$. The feature $\rho_{i,j}$ is determined as follows:

$$\rho_{i,j} = \alpha_i \beta_j \lambda_{i,j} \delta_{i,j} \quad , \quad i = 1, \ldots, n; \quad j = 1, \ldots, m. \tag{9}$$

Here clusters stand for cloudlets. Therefore, the number of clusters will equal to k.

In general, there are two types of clustering: hard and soft. In this paper, we consider the hard clustering problem. The hard clustering problem is the distribution of the cloud resources $\rho = \{\rho_1, \ldots, \rho_m\}$, $(\rho_j = (\rho_{1,j}, \ldots, \rho_{n,j}) \in R^n,\ j = 1, \ldots, m)$ into a given number k of disjoint subsets $C_q \in \rho$, $q = 1, \ldots, k$ with respect to predefined criteria such that [10]:

- $C_q \neq \emptyset$ for any $q = 1, \ldots, k$; (i. e., each subset should have at least one resource allocated);
- $C_q \cap C_p = \emptyset$ for any $q, p = 1, \ldots, k$; $q \neq p$ (i. e., the subsets should not overlap);
- $\rho = \bigcup\limits_{q=1}^{k} C_q$ (i. e., each resource should definitely be allocated to a subset);
- no constraints are imposed on the subsets C_q, $q = 1, \ldots, k$.

In another word, in hard clustering, one cloud resource can belong to one cluster only. In contrast, soft clustering is a form of clustering where each cloud resource may belong to multiple clusters according to a certain degree, called membership degree, and ranging in the unit interval.

The subsets $C = \{C_q \mid q = 1, \ldots, k\}$ are called clusters. We assume that each cluster C_q can be identified by its center $O_q = (O_{1,q}, \ldots, O_{n,q}) \in R^n$, $q = 1, \ldots, k$ The k-means and k-nearest neighbor algorithms are well-known clustering algorithms. Various improvements of these algorithms are still being made [31]. In this paper, we use the k-means clustering algorithm. Then k-means clustering algorithm can be formulated as follows. Find cluster centers $O = (O_1, \ldots, O_k) \in R^n$ such that sum of the L_2-norm distance squared between each point $\rho_j = (\rho_{1,j}, \ldots, \rho_{n,j}) \in R^n$ and its nearest cluster center $O_q = (O_{1,q}, \ldots, O_{n,q}) \in R^n$ is minimized:

$$\mathcal{F}(x) = \frac{1}{m} \sum_{q=1}^{k} \sum_{j=1}^{m} \|\rho_j - O_q\|_2^2 \, x_{q,j} \to \min \tag{10}$$

subject to

$$\sum_{q=1}^{k} x_{q,j} = 1 \tag{11}$$

for any $j = 1, \ldots, m$ where $x_{q,j}$ is the decision variable of the optimization problem, which is defined as follows:

$$x_{q,j} = \begin{cases} 1 & \text{if resource } \rho_j \text{ is allocated to the cluster } C_q \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

and $\|\rho_j - O_q\|_2$ is the L_2-norm distance (i. e., Euclidean distance) between two vectors $\rho_j = (\rho_{1,j}, \ldots, \rho_{n,j}) \in R^n$ and $O_q = (O_{1,q}, \ldots, O_{n,q}) \in R^n$

$$\|\rho_j - O_q\|_2 = \sqrt{\sum_{i=1}^{n} (\rho_{i,j} - O_{i,q})^2} \tag{13}$$

and

$$O_q = \frac{\sum\limits_{j=1}^{m} x_{q,j}\rho_j}{\sum\limits_{j=1}^{m} x_{q,j}} \quad , \quad q = 1, \ldots, k. \tag{14}$$

Eq. (11) provides that each resource will be allocated only one cluster.

The optimization problem (10) – (12) can be solved by the k-means algorithm iteratively [12].

Algorithm 1. (Classic k-means algorithm.) Given a dataset $\rho = \{\rho_1, \ldots, \rho_m\}$. Compute cluster centers $O_1^{t+1}, \ldots, O_k^{t+1}$ at iteration $(t+1)$ using the following steps:

Step 1 (*Number of clusters*). Choose the number of clusters k.

Step 2 (*Initial cluster centers*). Set $t = 0$ and select k random data from the dataset as cluster centers $O^t = (O_1^t, \ldots, O_k^t)$.

Step 3 (*Cluster assignment*). For each data $\rho_j = (\rho_{1,j}, \ldots, \rho_{n,j}) \in \mathrm{R}^n$, assign ρ_j to cluster C_q such that center O_q^t is nearest to ρ_j in the L_2-norm.

Step 4 (*Cluster update*). Recompute the center O_q^{t+1} of newly formed cluster C_q using Eq. (14):

$$O_q^{t+1} = \frac{\sum\limits_{j=1}^{m} x_{q,j}^t \rho_j}{\sum\limits_{j=1}^{m} x_{q,j}^t} \quad , \quad q = 1, \ldots, k. \tag{15}$$

Step 5 (*Stopping criterion*). Stop the algorithm if one of the following criteria is satisfied:

1) Centers of newly formed clusters do not change, i. e., $O_q^{t+1} = O_q^t$, $q = 1, \ldots, k$;
or 2) maximum number of iterations $t_{\max}$ is met,
else increment t by 1 and go to Step 3.

It can be easily seen that the solution to the problem (10) – (12) may not be the solution to the problem stated in this article. Because the k-means the algorithm does not guarantee that empty clusters will not emerge, on the other hands, the sum of volumes of resources placed in clusters (i. e., cloud servers) will not be greater than the server storage capacity. Therefore, it is required to impose additional constraints on the clusters based on the problem statement. Given the above, the constrained k-means clustering the problem can be modeled as follows:

$$\mathcal{F}(x) = \frac{1}{m} \sum_{q=1}^{k} \sum_{j=1}^{m} \|\rho_j - O_q\|_2^2 \, x_{q,j} \to \min \tag{16}$$

subject to

$$\sum_{q=1}^{k} x_{q,j} = 1 \tag{17}$$

for any $j = 1, \ldots, m$

$$\sum_{j=1}^{m} v_j x_{q,j} \leq V_q \tag{18}$$

for any $q = 1, \ldots, k$.

$$1 \leq \sum_{j=1}^{m} x_{q,j} < m \tag{19}$$

for any $q = 1, \ldots, k$.

Eq. (17) provides that each cloud resource will be allocated only to one server. Eq. (18) guarantees that the sum of the volume of cloud resources to be allocated on the server cannot exceed its storage capacity. And Eq. (19) guarantees that each cluster should have at least one cloud resource allocated and all the resources cannot be allocated to one server only.

Analogously to Algorithm 1 (classic k-means algorithm), below given an iterative algorithm to solve constrained k-means clustering problem (16)-(19) [12].

Algorithm 2. (Constrained k-means algorithm.) Given a dataset $\rho = \{\rho_1, \ldots, \rho_m\}$. Compute $O_1^{t+1}, \ldots, O_k^{t+1}$ at iteration $(t+1)$ using the following steps:

Step 1 (*Number of clusters*). Choose the number of clusters k.

Step 2 (*Initial cluster centers*). Set $t = 0$ and select k random data from the dataset as cluster centers $O^t = (O_1^t, \ldots, O_k^t)$.

Step 3 (*Cluster assignment*). Fix $O^t = (O_1^t, \ldots, O_k^t)$ and find a solution $x_{q,j}^t$ of linear assignment problem (16)-(19).

Step 4 (*Cluster update*). Recompute the center O_q^{t+1} of newly formed cluster C_q as follows:

$$O_q^{t+1} = \begin{cases} \dfrac{\sum\limits_{j=1}^{m} x_{q,j}^t \rho_j}{\sum\limits_{j=1}^{m} x_{q,j}^t} & \text{if } \sum\limits_{j=1}^{m} x_{q,j}^t \geq 1 \\ O_q^t & \text{otherwise} \end{cases}, \quad q = 1, \ldots k. \tag{20}$$

Step 5 (*Stopping criterion*). Stop the algorithm if one of the following criteria:
1) Centers of newly formed clusters do not change, i. e., $O_q^{t+1} = O_q^t$, $q = 1, \ldots k$;
or 2) maximum number of iterations $t_{\max}$ is met,
else increment t by 1 and go to Step 3.

6. EXPERIMENTAL RESULTS

Let's assume that there are 7 users, 30 resources, and 3 cloudlets. The resources usage frequency matrix ($\mathbb{F}$) and their activity (α_i) are given in Table 1. The normalized usage frequency matrix is given in Table 2. The distance matrix and normalized distance matrix are given in Tables 3 and 4, respectively. Table 5 lists the volume of cloud resources and the storage capacity of servers. The user-resource matrix calculated by using Eq. (9) is given in Table 6.

$\mathbb{F} = \|\mathbf{f_{i,j}}\|\ , (i = 1, \ldots, 7; j = 1, \ldots, 30)$								
$\mathbb{R}$	β_j	$\mathbb{U} = \{U_i \mid i = 1, \ldots, 7\}$						
		U_1	U_2	U_3	U_4	U_5	U_6	U_7
R_1	$\beta_1 = 0.0222$	25	75	25	23	38	26	59
R_2	$\beta_2 = 0.0389$	90	56	54	92	68	55	59
R_3	$\beta_3 = 0.0368$	32	31	99	61	85	85	55
R_4	$\beta_4 = 0.0336$	42	64	97	83	72	19	33
R_5	$\beta_5 = 0.0340$	56	28	33	81	64	85	68
R_6	$\beta_6 = 0.0405$	80	68	64	36	68	89	89
R_7	$\beta_7 = 0.0359$	81	41	64	94	64	30	64
R_8	$\beta_8 = 0.0328$	93	35	11	17	89	84	71
R_9	$\beta_9 = 0.0276$	90	85	45	16	33	45	22
R_{10}	$\beta_{10} = 0.0324$	58	64	77	51	51	76	18
R_{11}	$\beta_{11} = 0.0306$	37	25	38	34	72	77	90
R_{12}	$\beta_{12} = 0.0399$	94	27	89	36	83	87	71
R_{13}	$\beta_{13} = 0.0377$	95	55	45	63	38	69	95
R_{14}	$\beta_{14} = 0.0421$	87	51	46	85	97	47	100
R_{15}	$\beta_{15} = 0.0282$	24	47	22	74	90	25	62
R_{16}	$\beta_{16} = 0.0269$	52	50	61	16	24	99	26
R_{17}	$\beta_{17} = 0.0386$	91	63	46	82	43	72	73
R_{18}	$\beta_{18} = 0.0328$	69	42	32	86	31	97	43
R_{19}	$\beta_{19} = 0.0241$	44	74	31	28	49	49	19
R_{20}	$\beta_{20} = 0.0277$	55	24	15	92	92	26	33
R_{21}	$\beta_{21} = 0.0360$	31	37	47	92	71	64	97
R_{22}	$\beta_{22} = 0.0353$	34	56	76	72	25	78	89
R_{23}	$\beta_{23} = 0.0412$	57	81	90	63	67	81	63
R_{24}	$\beta_{24} = 0.0340$	80	75	88	28	33	75	36
R_{25}	$\beta_{25} = 0.0390$	89	70	66	85	60	82	24
R_{26}	$\beta_{26} = 0.0231$	53	18	32	55	71	29	24
R_{27}	$\beta_{27} = 0.0297$	17	39	24	74	94	60	54
R_{28}	$\beta_{28} = 0.0325$	56	57	17	81	73	79	33
R_{29}	$\beta_{29} = 0.0325$	97	29	12	58	27	73	100
R_{30}	$\beta_{30} = 0.0334$	18	98	66	90	41	50	44
		$\alpha_{\mathbf{i}}$ $(i = 1, \ldots, 7)$						
		$\alpha_1 =$ 0.1498	$\alpha_2 =$ 0.1284	$\alpha_3 =$ 0.1240	$\alpha_4 =$ 0.1516	$\alpha_5 =$ 0.1487	$\alpha_6 =$ 0.1569	$\alpha_7 =$ 0.1406

Tab. 1. Frequency matrix ($\mathbb{F}$), users' activity ($\alpha_{\mathbf{i}}$) and resource usage frequency (β_j).

$\mathbf{\Lambda} = \|\lambda_{\mathbf{i,j}}\|\ ,\ i = 1,\ldots,7; j = 1,\ldots,30$							
$\mathbb{R} = \{R_j \mid j = 1,\ldots,30\}$	$\mathbb{U} = \{U_i \mid i = 1,\ldots,7\}$						
	U_1	U_2	U_3	U_4	U_5	U_6	U_7
R_1	0.0137	0.0479	0.0165	0.0124	0.0210	0.0136	0.0344
R_2	0.0493	0.0358	0.0357	0.0498	0.0375	0.0288	0.0344
R_3	0.0175	0.0198	0.0655	0.033	0.0469	0.0444	0.0321
R_4	0.0230	0.0409	0.0642	0.0449	0.0397	0.0099	0.0193
R_5	0.0307	0.0179	0.0218	0.0438	0.0353	0.0444	0.0397
R_6	0.0438	0.0435	0.0423	0.0195	0.0375	0.0465	0.0519
R_7	0.0443	0.0262	0.0423	0.0509	0.0353	0.0157	0.0373
R_8	0.0509	0.0224	0.0073	0.0092	0.0491	0.0439	0.0414
R_9	0.0493	0.0543	0.0298	0.0087	0.0182	0.0235	0.0128
R_{10}	0.0317	0.0409	0.0509	0.0276	0.0281	0.0397	0.0105
R_{11}	0.0203	0.016	0.0251	0.0184	0.0397	0.0403	0.0525
R_{12}	0.0515	0.0173	0.0589	0.0195	0.0458	0.0455	0.0414
R_{13}	0.0520	0.0351	0.0298	0.0341	0.0210	0.0361	0.0554
R_{14}	0.0476	0.0326	0.0304	0.0460	0.0535	0.0246	0.0583
R_{15}	0.0131	0.0300	0.0146	0.0400	0.0496	0.0131	0.0362
R_{16}	0.0285	0.0319	0.0403	0.0087	0.0132	0.0518	0.0152
R_{17}	0.0498	0.0403	0.0304	0.0444	0.0237	0.0376	0.0426
R_{18}	0.0378	0.0268	0.0212	0.0465	0.0171	0.0507	0.0251
R_{19}	0.0241	0.0473	0.0205	0.0152	0.0270	0.0256	0.0111
R_{20}	0.0301	0.0153	0.0099	0.0498	0.0507	0.0136	0.0193
R_{21}	0.0170	0.0236	0.0311	0.0498	0.0392	0.0335	0.0566
R_{22}	0.0186	0.0358	0.0503	0.0390	0.0138	0.0408	0.0519
R_{23}	0.0312	0.0518	0.0595	0.0341	0.0370	0.0423	0.0368
R_{24}	0.0438	0.0479	0.0582	0.0152	0.0182	0.0392	0.0210
R_{25}	0.0487	0.0447	0.0437	0.0460	0.0331	0.0429	0.0140
R_{26}	0.0290	0.0115	0.0212	0.0298	0.0392	0.0152	0.0140
R_{27}	0.0093	0.0249	0.0159	0.0400	0.0518	0.0314	0.0315
R_{28}	0.0307	0.0364	0.0112	0.0438	0.0403	0.0413	0.0193
R_{29}	0.0531	0.0185	0.0079	0.0314	0.0149	0.0382	0.0583
R_{30}	0.0099	0.0626	0.0437	0.0487	0.0226	0.0261	0.0257

Tab. 2. Normalized frequency matrix ($\mathbf{\Lambda}$).

$\mathbb{D} = \|\mathbf{d_{i,j}}\|\ , i = 1, \ldots, 7; j = 1, \ldots, 30$							
$\mathbb{R} = \{R_j \mid j = 1, \ldots, 30\}$	$\mathbb{U} = \{U_i \mid i = 1, \ldots, 7\}$						
	U_1	U_2	U_3	U_4	U_5	U_6	U_7
R_1	8369	5935	4814	3222	2915	9309	5134
R_2	4516	2390	6828	2280	3971	9225	8967
R_3	3796	2770	7462	8471	4634	7078	8271
R_4	7273	2676	7387	6803	2776	7622	6681
R_5	5927	6123	3590	7648	6328	6499	4479
R_6	7868	7085	8281	6274	9151	5900	5175
R_7	9184	5153	8909	9927	3062	2242	6266
R_8	4646	6613	6708	3101	7189	8077	8136
R_9	5914	1552	6853	4108	2794	9003	6629
R_{10}	7632	9909	3210	7496	4358	8778	7014
R_{11}	4863	1954	5605	7025	5103	6686	5598
R_{12}	1979	3757	4968	5799	6625	4308	7314
R_{13}	9289	4015	9909	6299	2917	6853	2579
R_{14}	3941	2583	4550	9440	6586	5031	3940
R_{15}	5181	1039	8109	4316	7825	8620	5951
R_{16}	4517	4393	5786	4138	7053	5788	9679
R_{17}	7910	9264	3816	4758	6785	6313	6239
R_{18}	9742	4114	1763	9928	9463	7586	4194
R_{19}	5022	4871	1882	7477	5414	4442	4349
R_{20}	5297	2520	6884	7023	8179	4117	6336
R_{21}	6447	9697	5579	5778	3596	8298	2225
R_{22}	4204	8628	5381	9786	8423	6686	3552
R_{23}	8400	9878	2884	9835	2095	4956	8148
R_{24}	8765	1754	3131	3100	4756	7854	4949
R_{25}	1754	1471	4637	4755	2127	2914	9850
R_{26}	3488	9875	5569	8524	3599	3897	5989
R_{27}	7648	1178	2053	8633	6983	2445	5044
R_{28}	9763	7268	8792	7895	9730	3228	6074
R_{29}	4904	4994	9297	2906	8717	9890	3196
R_{30}	2314	8070	4814	6512	3436	8387	3351

Tab. 3. Distance matrix ($\mathbb{D}$).

$\mathbb{\Delta} = \|\delta_{\mathbf{i,j}}\|\ ,\ i = 1,\ldots,7; j = 1,\ldots,30$							
$\mathbb{R} = \{R_j \mid j = 1,\ldots,30\}$	$\mathbb{U} = \{U_i \mid i = 1,\ldots,7\}$						
	U_1	U_2	U_3	U_4	U_5	U_6	U_7
R_1	0.0464	0.0392	0.0284	0.0167	0.0175	0.0485	0.0293
R_2	0.0250	0.0158	0.0403	0.0118	0.0238	0.0480	0.0511
R_3	0.0210	0.0183	0.0440	0.0438	0.0278	0.0369	0.0472
R_4	0.0403	0.0177	0.0436	0.0352	0.0167	0.0397	0.0381
R_5	0.0328	0.0404	0.0212	0.0396	0.0380	0.0338	0.0255
R_6	0.0436	0.0468	0.0489	0.0325	0.0549	0.0307	0.0295
R_7	0.0509	0.0340	0.0526	0.0514	0.0184	0.0117	0.0357
R_8	0.0257	0.0436	0.0396	0.0160	0.0432	0.0421	0.0464
R_9	0.0328	0.0102	0.0404	0.0213	0.0168	0.0469	0.0378
R_{10}	0.0328	0.0102	0.0404	0.0213	0.0168	0.0469	0.0378
R_{11}	0.0269	0.0129	0.0331	0.0364	0.0306	0.0348	0.0319
R_{12}	0.0110	0.0248	0.0293	0.0300	0.0398	0.0224	0.0417
R_{13}	0.0514	0.0265	0.0585	0.0326	0.0175	0.0357	0.0147
R_{14}	0.0218	0.0170	0.0269	0.0488	0.0395	0.0262	0.0225
R_{15}	0.0287	0.0069	0.0479	0.0223	0.0470	0.0449	0.0339
R_{16}	0.0250	0.0290	0.0341	0.0214	0.0423	0.0301	0.0552
R_{17}	0.0438	0.0611	0.0225	0.0246	0.0407	0.0329	0.0356
R_{18}	0.0540	0.0271	0.0104	0.0514	0.0568	0.0395	0.0239
R_{19}	0.0278	0.0321	0.0111	0.0387	0.0325	0.0231	0.0248
R_{20}	0.0293	0.0166	0.0406	0.0363	0.0491	0.0214	0.0361
R_{21}	0.0357	0.0640	0.0329	0.0299	0.0216	0.0432	0.0127
R_{22}	0.0233	0.0569	0.0318	0.0506	0.0506	0.0348	0.0203
R_{23}	0.0465	0.0652	0.0170	0.0509	0.0126	0.0258	0.0465
R_{24}	0.0485	0.0116	0.0185	0.0160	0.0285	0.0409	0.0282
R_{25}	0.0097	0.0097	0.0274	0.0246	0.0128	0.0152	0.0562
R_{26}	0.0193	0.0652	0.0329	0.0441	0.0216	0.0203	0.0342
R_{27}	0.0424	0.0078	0.0121	0.0447	0.0419	0.0127	0.0288
R_{28}	0.0541	0.0480	0.0519	0.0409	0.0584	0.0168	0.0346
R_{29}	0.0272	0.0330	0.0549	0.0150	0.0523	0.0515	0.0182
R_{30}	0.0128	0.0533	0.0284	0.0337	0.0206	0.0437	0.0191

Tab. 4. Normalized distance matrix (Δ).

Volume of cloud resources, $\mathbb{v} = \{v_j \mid j = 1, \ldots, 30\}$														
$v_1 =$ 5	$v_2 =$ 24	$v_3 =$ 17	$v_4 =$ 25	$v_5 =$ 8	$v_6 =$ 8	$v_7 =$ 23	$v_8 =$ 24	$v_9 =$ 31	$v_{10} =$ 20	$v_{11} =$ 25	$v_{12} =$ 17	$v_{13} =$ 7	$v_{14} =$ 8	$v_{15} =$ 26
$v_{16} =$ 9	$v_{17} =$ 20	$v_{18} =$ 8	$v_{19} =$ 21	$v_{20} =$ 29	$v_{21} =$ 15	$v_{22} =$ 14	$v_{23} =$ 15	$v_{24} =$ 10	$v_{25} =$ 5	$v_{26} =$ 19	$v_{27} =$ 26	$v_{28} =$ 23	$v_{29} =$ 23	$v_{30} =$ 34
Storage capacity of servers, $\mathbb{V} = \{V_q \mid q = 1, 2, 3\}$														
$V_1 = 118$						$V_2 = 137$						$V_3 = 169$		

Tab. 5. Volume of resources ($\mathbb{v}$) and storage capacity of servers ($\mathbb{V}$).

In order to evaluate the effectiveness of the proposed model, it was compared to the models proposed in [7]. The solutions of the models are provided in Table 7.

The Algorithm 2 was implemented with Matlab 2022a. To solve the optimization problem (16)-(19) the mixed-integer linear programming package of the Optimization Toolbox was used. The characteristics of the computer on which the experiments were carried out: Lenovo Y50, Intel Core i7, 2.5 GHz, x64; RAM: 8 GB; CPU: 4 cores, 8 logical processors; GPU: NVIDIA GeForce GTX 860M, 4 Gb; HDD: 1 TB.

The maximum number of iterations is set to 1000, $t_{\max} = 1000$. The result of the experiment ($x_{q,j}^{\text{cluster}}$) is given in the first column of Table 7. The solutions of the models proposed in [7] are also included in Table 7 to evaluate the effectiveness of the proposed model. Here, $x_{q,j}^{\text{cluster}}$ denotes the solution provided by the model (16)-(19), while $x_{q,j}^{2}$ denotes the solution corresponding to the objective function $\mathcal{F}_2$ proposed in [7].

In Table 7, $x_{q,j}^{\gamma=0.4}$, $x_{q,j}^{\gamma=0.5}$and $x_{q,j}^{\gamma=0.6}$ are the solutions corresponding to the 0.4, 0.5 and 0.6 values of the parameter γ for the objective function $\mathcal{F}_1$ proposed in [7]. To compare the model's value of the objective function provided in (16) is also calculated. The values of the objective function $\mathcal{F}(x)$ (16) for all solutions are provided in the last row of Table 7. As seen from Table 7, $x_{q,j}^{\text{cluster}}$ is the best solution. Thus, for this solution, the value of the objective function $\mathcal{F}(x)$ is the smallest, is equal to 0.001049. This value is 18.3% better than the value provided by the second-best solution, $\mathcal{F}(x_{q,j}^{\gamma=0.4}) = \mathcal{F}(x_{q,j}^{\gamma=0.6}) = 0.001284$. It confirms that the proposed model provides a sufficiently good result

7. CONCLUSION

As a result of the rapid increase in the number of mobile devices connected to the Internet network, there are large delays in the delivery of data to mobile users due to network loading. Edge computing (edge, cloudlet, etc.) technologies are widely used to eliminate network delays. The article proposes to use mobile edge computing systems (cloudlet networks) to eliminate resource scarcity, power consumption in mobile devices and delays in communication channels Reduction of network delays is demonstrated by optimally placing cloud resources in the cloudlets close to users. In the article, a clustering-based model was proposed for the optimal placement of cloud resources in cloudlets. The proposed model takes into account the activity of users, usage frequency of cloud resources, physical distance between users and cloud resources, as well as the storage capacity of cloudlets in order to optimally allocate cloud resources in cloudlets. The proposed model is formalized as an optimization problem. Then, in order to solve

$\rho = \|\rho_{i,j}\|, i = 1,\ldots,7; j = 1,\ldots,30$							
$\mathbb{R} = \{R_j \mid j = 1,\ldots,30\}$	$\mathbb{U} = \{U_i \mid i = 1,\ldots,7\}$						
	U_1	U_2	U_3	U_4	U_5	U_6	U_7
R_1	2.11E-06	5.36E-06	1.29E-06	6.99E-07	1.21E-06	2.30E-06	3.15E-06
R_2	7.18E-06	2.82E-06	6.94E-06	3.46E-06	5.17E-06	8.43E-06	9.62E-06
R_3	2.03E-06	1.71E-06	1.31E-05	8.06E-06	7.13E-06	9.44E-06	7.82E-06
R_4	4.67E-06	3.12E-06	1.17E-05	8.06E-06	3.31E-06	2.08E-06	3.47E-06
R_5	5.13E-06	3.16E-06	1.95E-06	8.95E-06	6.79E-06	8.03E-06	4.85E-06
R_6	1.16E-05	1.06E-05	1.04E-05	3.88E-06	1.24E-05	9.09E-06	8.73E-06
R_7	1.21E-05	4.11E-06	9.91E-06	1.42E-05	3.47E-06	1.03E-06	6.74E-06
R_8	6.44E-06	4.11E-06	1.17E-06	7.34E-07	1.03E-05	9.51E-06	8.87E-06
R_9	6.66E-06	1.97E-06	4.11E-06	7.69E-07	1.25E-06	4.77E-06	1.88E-06
R_{10}	6.51E-06	1.11E-05	3.88E-06	5.26E-06	3.55E-06	9.23E-06	1.91E-06
R_{11}	2.50E-06	8.09E-07	3.15E-06	3.10E-06	5.53E-06	6.73E-06	7.21E-06
R_{12}	3.38E-06	2.19E-06	8.55E-06	3.54E-06	1.08E-05	6.39E-06	9.70E-06
R_{13}	1.51E-05	4.51E-06	8.14E-06	6.35E-06	2.06E-06	7.62E-06	4.32E-06
R_{14}	6.55E-06	3.00E-06	4.26E-06	1.43E-05	1.32E-05	4.25E-06	7.76E-06
R_{15}	1.59E-06	7.46E-07	2.44E-06	3.82E-06	9.78E-06	2.60E-06	4.87E-06
R_{16}	2.87E-06	3.20E-06	4.60E-06	7.56E-07	2.24E-06	6.58E-06	3.17E-06
R_{17}	1.26E-05	1.22E-05	3.28E-06	6.38E-06	5.54E-06	7.48E-06	8.21E-06
R_{18}	1.00E-05	3.07E-06	8.96E-07	1.19E-05	4.74E-06	1.03E-05	2.77E-06
R_{19}	2.42E-06	4.70E-06	6.81E-07	2.14E-06	3.15E-06	2.24E-06	9.32E-07
R_{20}	3.66E-06	9.05E-07	1.38E-06	7.58E-06	1.02E-05	1.26E-06	2.70E-06
R_{21}	3.27E-06	6.99E-06	4.57E-06	8.12E-06	4.53E-06	8.17E-06	3.64E-06
R_{22}	2.29E-06	9.22E-06	6.98E-06	1.05E-05	3.66E-06	7.86E-06	5.22E-06
R_{23}	8.96E-06	1.78E-05	5.17E-06	1.08E-05	2.85E-06	7.06E-06	9.89E-06
R_{24}	1.08E-05	2.42E-06	4.54E-06	1.25E-06	2.63E-06	8.56E-06	2.84E-06
R_{25}	2.77E-06	2.18E-06	5.78E-06	6.70E-06	2.45E-06	3.98E-06	4.32E-06
R_{26}	1.94E-06	2.23E-06	2.00E-06	4.60E-06	2.91E-06	1.12E-06	1.56E-06
R_{27}	1.75E-06	7.38E-07	7.08E-07	8.05E-06	9.60E-06	1.86E-06	3.78E-06
R_{28}	8.07E-06	7.28E-06	2.35E-06	8.82E-06	1.14E-05	3.54E-06	3.05E-06
R_{29}	7.02E-06	2.55E-06	1.75E-06	2.32E-06	3.76E-06	1.00E-05	4.86E-06
R_{30}	6.32E-07	1.43E-05	5.13E-06	8.30E-06	2.32E-06	5.98E-06	2.30E-06

Tab. 6. User-resource matrix.

the optimization problem, the constrained k-means algorithm is developed. It is shown that the proposed algorithm provides increased power sustainability in mobile devices, saving time spent on data processing, as well as reducing network delays. The results of the experiment showed that the proposed model is superior to other models and therefore, is promising.

$x_{q,j}^{\text{cluster}}$	$x_{q,j}^{2}$	$x_{q,j}^{\gamma=0.4}$	$x_{q,j}^{\gamma=0.5}$	$x_{q,j}^{\gamma=0.6}$
$x_{3,1}=1$	$x_{1,1}=1$	$x_{3,1}=1$	$x_{1,1}=1$	$x_{3,1}=1$
$x_{2,2}=1$	$x_{1,2}=1$	$x_{2,2}=1$	$x_{2,2}=1$	$x_{2,2}=1$
$x_{2,3}=1$	$x_{1,3}=1$	$x_{1,3}=1$	$x_{1,3}=1$	$x_{1,3}=1$
$x_{3,4}=1$	$x_{1,4}=1$	$x_{3,4}=1$	$x_{1,4}=1$	$x_{3,4}=1$
$x_{2,5}=1$	$x_{1,5}=1$	$x_{1,5}=1$	$x_{1,5}=1$	$x_{1,5}=1$
$x_{2,6}=1$	$x_{1,6}=1$	$x_{1,6}=1$	$x_{1,6}=1$	$x_{1,6}=1$
$x_{2,7}=1$	$x_{1,7}=1$	$x_{1,7}=1$	$x_{1,7}=1$	$x_{1,7}=1$
$x_{1,8}=1$	$x_{1,8}=1$	$x_{1,8}=1$	$x_{1,8}=1$	$x_{1,8}=1$
$x_{3,9}=1$	$x_{1,9}=1$	$x_{1,9}=1$	$x_{1,9}=1$	$x_{1,9}=1$
$x_{3,10}=1$	$x_{1,10}=1$	$x_{1,10}=1$	$x_{1,10}=1$	$x_{1,10}=1$
$x_{2,11}=1$	$x_{1,11}=1$	$x_{1,11}=1$	$x_{3,11}=1$	$x_{1,11}=1$
$x_{2,12}=1$	$x_{1,12}=1$	$x_{1,12}=1$	$x_{1,12}=1$	$x_{1,12}=1$
$x_{1,13}=1$	$x_{2,13}=1$	$x_{1,13}=1$	$x_{1,13}=1$	$x_{1,13}=1$
$x_{2,14}=1$	$x_{1,14}=1$	$x_{1,14}=1$	$x_{1,14}=1$	$x_{1,14}=1$
$x_{3,15}=1$	$x_{1,15}=1$	$x_{1,15}=1$	$x_{1,15}=1$	$x_{1,15}=1$
$x_{3,16}=1$	$x_{1,16}=1$	$x_{1,16}=1$	$x_{1,16}=1$	$x_{1,16}=1$
$x_{2,17}=1$	$x_{1,17}=1$	$x_{1,17}=1$	$x_{1,17}=1$	$x_{1,17}=1$
$x_{2,18}=1$	$x_{1,18}=1$	$x_{3,18}=1$	$x_{1,18}=1$	$x_{3,18}=1$
$x_{3,19}=1$	$x_{1,19}=1$	$x_{1,19}=1$	$x_{1,19}=1$	$x_{1,19}=1$
$x_{3,20}=1$	$x_{1,20}=1$	$x_{1,20}=1$	$x_{1,20}=1$	$x_{1,20}=1$
$x_{2,21}=1$	$x_{3,21}=1$	$x_{3,21}=1$	$x_{3,21}=1$	$x_{3,21}=1$
$x_{2,22}=1$	$x_{1,22}=1$	$x_{1,22}=1$	$x_{1,22}=1$	$x_{1,22}=1$
$x_{2,23}=1$	$x_{1,23}=1$	$x_{1,23}=1$	$x_{1,23}=1$	$x_{1,23}=1$
$x_{3,24}=1$	$x_{3,24}=1$	$x_{1,24}=1$	$x_{1,24}=1$	$x_{1,24}=1$
$x_{3,25}=1$	$x_{1,25}=1$	$x_{1,25}=1$	$x_{2,25}=1$	$x_{1,25}=1$
$x_{3,26}=1$	$x_{3,26}=1$	$x_{3,26}=1$	$x_{3,26}=1$	$x_{3,26}=1$
$x_{3,27}=1$	$x_{3,27}=1$	$x_{3,27}=1$	$x_{1,27}=1$	$x_{3,27}=1$
$x_{3,28}=1$	$x_{1,28}=1$	$x_{1,28}=1$	$x_{1,28}=1$	$x_{1,28}=1$
$x_{1,29}=1$	$x_{2,29}=1$	$x_{1,29}=1$	$x_{2,29}=1$	$x_{1,29}=1$
$x_{3,30}=1$	$x_{2,30}=1$	$x_{1,30}=1$	$x_{2,30}=1$	$x_{1,30}=1$
$\mathcal{F}\left(x_{q,j}^{\text{cluster}}\right)$ $=$ **0001049**	$\mathcal{F}\left(x_{q,j}^{2}\right)$ $=0.001287$	$\mathcal{F}\left(x_{q,j}^{\gamma=0.4}\right)$ $=0.001284$	$\mathcal{F}\left(x_{q,j}^{\gamma=0.5}\right)$ $=0.001309$	$\mathcal{F}\left(x_{q,j}^{\gamma=0.6}\right)$ $=0.001284$

Tab. 7. Solutions of the proposed model and the models proposed in [7].

In the future, we are planning to improve the model and develop an algorithm to solve an optimization problem, conduct experiments on real data, and compare it with the state-of-the-art methods.

ACKNOWLEDGEMENT

We would like to thank Editor-in-Chief Prof. Sergej Čelikovský and the anonymous reviewers for their valuable comments that helped to improve the quality of the paper.

(Received July 10, 2022)

REFERENCES

[1] A. Ahmed and E. Ahmed: A survey on mobile edge computing. In: 2016 10th International Conference on Intelligent Systems and Control 2016, pp. 1–8.DOI:10.1109/ISCO.2016.7727082

[2] E. Ahmed, A. Akhunzada, M. Whaiduzzaman, A. Gani, S. H. Ab Hamid, and R. Buyya: Network-centric performance analysis of runtime application migration in mobile cloud computing. Simul. Modelling Practice Theory *50* (2015), 42–56.DOI:10.1016/j.simpat.2014.07.001

[3] R. Alakberov: Strategy for reducing delays and energy consumption in cloudlet-based mobile cloud computing. Int. J. Wireless Networks Broadband Technol. *10* (2021), 1, 32–44.DOI:10.4018/IJWNBT.2021010102

[4] R. G. Alakberov: Clustering method of mobile cloud computing according to technical characteristics of cloudlets. Int. J. Computer Network Inform. Security *14* (2022), 3, 75–87.DOI:10.5815/ijcnis.2022.03.06

[5] R. Alakbarov and O. Alakbarov: Procedure of effective use of cloudlets in wireless metropolitan area network environment. Int. J. Computer Networks Commun. *11* (2019), 1 93–107.DOI:10.5121/ijcnc.2019.11106

[6] M. Ala'anzy, M. Othman, Z. M. Hanapi, and M. A. Alrshah: Locust inspired algorithm for cloudlet scheduling in cloud computing environments. Sensors *21* (2021), 7308, 1–19.DOI:10.3390/s21217308

[7] R. M. Alguliyev and R. G. Alakbarov: Integer programming models for task scheduling and resource allocation in mobile cloud computing. Int. J. Computer Network Inform. Security, 2023 (in press).

[8] H. Asghar and E. S. Jung: A survey on scheduling techniques in the edge cloud: issues, challenges and future directions. arXiv.org 2022, 1–19. https://arxiv.org/abs/2202.07799

[9] P. Azad and N. J. Navimipour: An energy-aware task scheduling in the cloud computing using a hybrid cultural and ant colony optimization algorithm. Int. J. Cloud Appl. Computing *7* (2017), 4, 20–40.DOI:10.4018/IJCAC.2017100102

[10] A. M. Bagirov: Modified global k-means algorithm for minimum sum-of-squares clustering problems. Pattern Recognition *41* (2008), 10, 3192–3199.DOI:10.1016/j.patcog.2008.04.004

[11] G. H. Bindu, K. Ramani, and C. S. Bindu: Energy aware multi objective genetic algorithm for task scheduling in cloud computing. Int. J. Internet Protocol Technol. *11* (2018), 4, 242–249.DOI:10.1504/IJIPT.2018.10016310

[12] P. S. Bradley, K. P. Bennett, and A. Demiriz: Constrained k-means clustering. Technical Report MSR-TR-2000-65, Microsoft Research, Redmond 2000, pp. 1–8.

[13] X. Chen, L. Jiao, W. Z. Li, and X. M. Fu: Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans. Networking *24* (2015), 5, 2795–2808.DOI:10.1109/TNET.2015.2487344

[14] L. Chen, S. Zhou, and J. Xu: Computation peer offloading for energy-constrained mobile edge computing in small-cell networks. IEEE ACM Trans. Networking *26* (2018), 4, 1619–1632.DOI:10.1109/TNET.2018.2841758

[15] D. Dalan: An overview of edge computing. Int. J. Engrg. Res. Technol. *7* (2019), 5, 1–4.

[16] M. Hu, L. Zhuang, D. Wu, Y. P. Zhou, X. Chen, and L. Xiao: Learning driven computation offloading for asymmetrically informed edge computing. IEEE Trans. Parallel Distributed Systems *30* (2019), 8, 1802–1815. DOI:10.1109/TPDS.2019.2893925

[17] K. Liao, J. Yang, and L. Miao: Mobile edge computing offload strategy based on energy aware. In: International Conference on Network Communication and Information Security 2021, pp. 1–9.DOI:10.1117/12.2628417

[18] L. Lin, P. Li, J. Xiong, and M. Lin: Distributed and application-aware task scheduling in edge-clouds. In: 2018 14th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN) 2018, pp. 165–170.DOI:10.1109/MSN.2018.000-1

[19] R. Lin, Z. Zhou, S. Luo, Y. Xiao, and M. Zukerman: Distributed optimization for computation offloading in edge computing. IEEE Trans. Wireless Commun. *19* (2020), 12, 8179–8194.DOI:10.1109/TWC.2020.3019805

[20] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi: Resource scheduling in edge computing: a survey. IEEE Commun. Survey Tutorials *23* (2021), 4, 2131–2165.DOI:10.1109/COMST.2021.3106401

[21] P. Mach and Z. Becvar: Mobile edge computing: a survey on architecture and computation offloading. IEEE Commun. Surveys Tutorials *19* (2017), 3, 1628–1656.DOI:10.1109/COMST.2017.2682318

[22] J. Mike, J. Cao, and W. Liang: Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. IEEE Trans. Cloud Computing *5* (2017), 4, 725–737.DOI:10.1109/TCC.2015.2449834

[23] A. Mukherjee, D. Priti, D. De, and R. Buyya: IoTF2N: An energy-efficient architectural model for IoT using Femtolet-based fog network. J. Supercomputing *75* (2019), 11, 7125–7146.DOI:10.1007/s11227-019-02928-0

[24] A. Nasr, N. A. El-Bahnasawy, G. Attiya, and A. El-Sayed: Cloudlet scheduling based load balancing on virtual machines in cloud computing environment. J. Internet Technol. *20* (2019), 5, 1376–1378.

[25] M. Sachula, Y. Wang, Z. Miao, and K. Sun: Joint optimization of wireless bandwidth and computing resource in cloudlet-based mobile cloud computing environment. Peer-to-Peer Networking Appl. *11* (2018), 3, 462–472.DOI:10.1007/s12083-017-0544-x

[26] D. K. Sajnani, A. R. Mahesar, A. Lakhan, and I. A. Jamali: Latency aware and service delay with task scheduling in mobile edge computing. Commun. Network *10* (2018), 4, Article ID 87708.DOI:10.4236/cn.2018.104011

[27] Y. Shen, Z. Bao, X. Qin, and J. Shen: Adaptive task scheduling strategy in cloud: when energy consumption meets performance guarantee. World Wide Web *20* (2016), 155–173.DOI:10.1007/s11280-016-0382-4

[28] K. Shenoy, P. Bhokare, and U. Pai: Fog computing future of cloud computing. Int. J. Sci. Res. *4* (2015), 6, 55–56. DOI:10.1891/2156-5287.5.1.55

[29] G. Shreya, A. Mukherjee, S. Ghosh, and R. Buyya: Mobi-IoST: mobility-aware cloud-fog-edge-iot collaborative framework for time-critical applications. IEEE Trans. Network Science Engrg. *7* (2019), 4, 2271–2285.DOI:10.1109/TNSE.2019.2941754

[30] R. S. Somula and S. Ra: A survey on mobile cloud computing: mobile computing+cloud computing (MCC=MC+CC). Scalable Computing: Practice and Experience *19* (2018) 4, 309–337.DOI:10.12694/scpe.v19i4.1411

[31] O. Vencalek and D. Hlubinka: A depth-based modification of the k-nearest neighbour method. Kybernetika *57* (2021), 1, 15–37. DOI:10.14736/kyb-2021-1-0015

[32] X. Y. Wang, Z. L. Ning, and S. Guo: Multi-agent imitation learning for pervasive edge computing: a decentralized computation offloading algorithm. IEEE Trans. Parallel Distributed Systems *32* (2020), 2, 411–425. DOI:10.1109/TPDS.2020.3023936

[33] L. C. Yang, H. L. Zhang, X. Li, H. Ji, and V. C. M. Leung: A distributed computation offloading strategy in small-cell networks integrated with mobile edge computing. IEEE ACM Trans. Networking *26* (2018), 6, 2762–2773.DOI:10.1109/TNET.2018.2876941

[34] M. Yuyi, C. You, J. Zhang, K. Huang, and K. Letaief: A survey on mobile edge computing: the communication perspective. IEEE Commun. Surveys Tutorials *19* (2017), 4, 2322–2358.DOI:10.1109/COMST.2017.2745201

[35] F. Zhang, J. Ge, Z. Li, C. Li, C. Wong, L. Kong, B. Luo, and V. Chang: A load-aware resource allocation and task scheduling for the emerging cloudlet system. Future Generation Computer Systems *87* (2018), 438–456.DOI:10.1016/j.future.2018.01.053

[36] P. Zhang and M. Zhou: Dynamic cloud task scheduling based on a two-stage strategy. IEEE Trans. Automat. Sci. Engrg. *15* (2018), 2, 772–783.DOI:10.1109/TASE.2017.2693688

Rasim M. Alguliyev, Institute of Information Technology, AZ1141, B. Vahabzade Street, 9A, Baku. Azerbaijan.
e-mail: r.alguliev@gmail.com

Ramiz M. Aliguliyev, Institute of Information Technology, AZ1141, B. Vahabzade Street, 9A, Baku. Azerbaijan.
e-mail: r.aliguliyev@gmail.com

Rashid G. Alakbarov, Institute of Information Technology, AZ1141, B. Vahabzade Street, 9A, Baku. Azerbaijan.
e-mail: rashid.alakberov@gmail.com