Cluster Computing (2022) 25:2061–2083
https://doi.org/10.1007/s10586-021-03333-0

Cost-efficient mobility offloading and task scheduling for microservices IoVT applications in container-based fog cloud network

Abdullah Lakhan[1] · Muhammad Suleman Memon[2] · Qurat-ul-ain Mastoi[3] · Mohamed Elhoseny[4,5] · Mazin Abed Mohammed[6] · Mumtaz Qabulio[7] · Mohamed Abdel-Basset[8]

Received: 2 January 2021 / Revised: 23 April 2021 / Accepted: 2 June 2021 / Published online: 20 June 2021

Abstract

These days, the usage of the internet of Vehicle Things (IVoT) applications such as E-Business, E-Train, E-Ambulance has been growing progressively. These applications require mobility-aware delay-sensitive services to execute their tasks. With this motivation, the study has the following contribution. Initially, the study devises a novel cooperative vehicular fog cloud network (VFCN) based on container microservices which offers cost-efficient and mobility-aware services with rich resources for processing. This study devises the cost-efficient task offloading and scheduling (CEMOTS) algorithm framework, which consists of the mobility aware task offloading phase (MTOP) method, which determines the optimal offloading time to minimize the communication cost of applications. Furthermore, CEMOTS offers Cooperative Task Offloading Scheduling (CTOS), including task sequencing and scheduling. The goal is to reduce the application costs of communication cost and computational costs under a given deadline constraint. Performance evaluation shows the CTOS and MTOP outperform existing task offloading and scheduling methods in the VCFN in terms of costs and the deadline for IoT applications.

Keywords Task offloading · Internet of things · Task scheduling · VFCN · Mobility · Task latency · System costs

✉ Mazin Abed Mohammed
mazinalshujeary@uoanbar.edu.iq

Abdullah Lakhan
Abdullahrazalakhan@gmail.com

Muhammad Suleman Memon
msuleman@usindh.edu.pk

Qurat-ul-ain Mastoi
quratulain.mastoi@siswa.um.edu.my

Mohamed Elhoseny
melhoseny@ieee.org

Mumtaz Qabulio
Mumtaz.qabulio@usindh.edu.pk

Mohamed Abdel-Basset
mohamedbasset@zu.edu.eg

[1] College of Computer Science and Artificial Intelligence, Wenzhou University, Wenzhou 325035, China

[2] Department of Information Technology, Dadu Campus, University of Sindh, Jamshoro, Sindh, Pakistan

[3] Faculty of Computer Science and Information Technology, University of Malay a, 50603 Kuala Lumpur, Malaysia

[4] Department of Computer Science, College of Computer Information Technology, American University in the Emirates, 503000 Dubai International Academic City, UAE

[5] Faculty of Computers and Information, Mansoura University, Mansoura, Egypt

[6] College of Computer Science and Information Technology, University of Anbar, Ramadi 31001, Iraq

[7] Department of Software Engineering, Faculty of Engineering, University of Sindh, Jamshoro, Pakistan

[8] Faculty of Computers and Informatics, Zagazig University, Zagazig, Sharqiyah 44519, Egypt

1 Introduction

The Internet of Vehicle Things (IoT) is an emerging technology to boost vehicle applications in the practice [1]. The applications are Healthcare, Package Delivery, E-Transport, Advertisement, Path Planning, Self-Driving, Automobiles, 3D-Game, Augmented Reality, and Public Surveillance [2]. However, smart devices have resource-constrained issues (e.g., limited battery, limited CPU speed, and bandwidth utilization) and cannot execute the compute-intensive IoT applications locally. Vehicular Fog-Cloud Network (VFCN) is a promising solution, which offers ubiquitous wireless connection and dynamic cloud services at road infrastructure for IoT applications [3, 4]. VFCN also takes care of data security and services omnipresent for applications [5].

Since lengthy space between the devices and centralized public cloud and intermittent wireless network values incurs significant latency, it degrades the application's overall performance during mobility. The fog cloud is introduced, which is placed intermediately between edge devices and public cloud [6]. Recently, the latest Base-Stations (BTS) exploited 5G Dynamic Spectrum Access (DSA) [7] is a good solution to the mobility aware application and offers seamless connections in the vehicular network [8–10]. The DSA based BTS and VFCN offered flawlessly omnipresent communication/calculation services to the vehicular-based IoT applications [11, 12]. Due to the mobility and interactive features of applications, often cloud services are required to integrate at the roadside unit of the network [13]. It is called Vehicle to Infrastructure (V2I). All communication and computation services have a different pricing model in the VFCN. Moreover, the existing VFCN system [5–12] offering cloud services based on heavyweight virtual machines (VMs) for IoT applications. VM-based cloud services often incurred colossal overhead and long process delay problems. Applications, as mentioned earlier, consist of lightweight fine-grained tasks and require thin services.

In the proposed work, we formulate the mobility aware offloading and cost-optimal task scheduling in vehicular networks. The objective is to minimize system costs which are the combination of the communication cost and the computation cost. Individually IoT application consists of fine-grained tasks [13, 14]. The individual task has the following attributes: workload size, required CPU instruction during execution as well as deadline. We assume that the VFCN system consists of heterogeneous clouds, such as fog servers and centric clouds. Every cloud has a distinct computing speed and resource price. We design a new VFCN system based on container microservices instead of a virtual machine [14]. Each microservice performs a single business goal (e.g., login system, Health-Monitoring, and Temperature Calculator) and works similar to function as a service [15]. The users will pay the costs only for running microservices instead of whole applications, unlike existing virtual machine vehicular cloud networks [10–15]. On the other hand, containers are emerging and pursuing small service boot time, which is useful in a heterogeneous environment as compared to virtual machines [16].

Thus, mobility aware offloading and task scheduling in the heterogeneous clouds with consideration of system costs are investigated in the paper, which is different from existing works. We are analyzing the following challenges: (i) How/When to choose an optimal offloading time for task offloading? (ii) How to prioritize each task in some sequence? (iii) How to schedule geographically distributed into the heterogeneous vehicular cloud environment to satisfy latency and cost requirements? The paper makes the following contributions:

- To reduce service boot-up time, service cost, and overhead among services, we proposed a novel container microservices-based Vehicular Fog Cloud Network system. Which offers cloud services based on function as services model (i.e., on-demand) by the different fog servers. Moreover, the centric cloud provides services based on an on-reserved and on-spot model in a way, the system costs of the application are minimized.
- We derive the task offloading problem as a multi-criteria decision problem. Whereas offloading time, communication, task execution time, and the deadline is considered. We devise the mobility aware task offloading phase (MTOP) method, which chooses an optimal time for offloading in a way communication cost to be reduced. MTOP also adopts a dynamic environment and makes the optimal decision for IoT applications.
- Sequencing is the method of scheduling tasks on cloud servers in such a way to reduce the execution time, cost, and resource practice through sequencing methods. The individual task has the following attributes: data size, execution time, deadline attributes. Therefore we propose the task sequencing rules method based on different approaches. The goal is to arrange tasks in a specific order total costs of the application and task deadlines must be satisfied.
- We consider heterogeneous vehicular fog cloud networks for IoT applications; therefore, for each task to choose an optimal cloud is a challenging task. We propose a resource matching and cost-efficient task scheduling algorithm, CTOS, which determines the optimal task sequence, resource matching, and

scheduling in a specific the costs of the applications to be minimized.

The rest of the paper is organized in the following way. Section 2 demonstrates the related work of the considered problem. Section 3 reveals the problem description and formulation. Section 4 shows the proposed algorithm of the study. Section 5 assesses the performance evaluation part; Section 6 is about the conclusion and future related to the study.

2 Related work

Task offloading and scheduling methods are useful to the IoT application in practice. Many efforts were made on the task offloading issues in the literature studies. The aim was to improve devices' energy and application performance on the devices. Vehicular network-based offloading during mobility is a critical challenge that the literature studies have widely studied. Resource allocation and task scheduling are methods widely exploited to solve the IoT applications' assignment problem in the vehicular network.

Due to devices' mobility, the offloading time and scheduling in heterogeneous clouds is a challenging task. Existing studies [17–21] focused direct offloading schemes without considering mobility features of the devices. Moreover, the virtual machine-based cloud services integrated inside frameworks, which always incurred high overhead and long service boot uptime. The random task offloading schemes were investigated in these works [15, 21–28], where mobility and homogeneous mobile edge clouds are considered. Joint task offloading and scheduling problems for IoT applications investigated in these papers [29–34]. However, random and direct task offloading schemes and the homogeneous edge cloud environment for task scheduling could not satisfy the current IoT applications' challenge [35–39]. The static offloading and mobility aware services problem investigated in [40–42]. The goal is to offload all workloads of transport applications to the available servers for the processing.

Mobility aware task offloading in the vehicular network for IoT applications investigated in these studies [43–47]. These studies proposed dynamic offloading methods to the roaming IoT applications at the road infrastructure. In this way, the ubiquitous wireless connection and omnipresent service requirements of the applications are fulfilled.

The resource allocation and task scheduling problem for applications investigated in these works [48–57]. Heterogeneous fog clouds and centric cloud were considered for the task assignment in the vehicular network. Task scheduling and resource matching were key methods during resource allocation to minimize system energy, cost and improve resource utilization in the vehicular network.

However, the studies from [17–32, 17–32] investigated the offloading and scheduling problem individually and optimize the device's energy, response time, and delay of applications. Furthermore, these studies suggested many resource allocation algorithms and offloading schemes for vehicle applications. However, many things to be improved such as the sometimes offloading cost is greater than the computation cost. Whereas, communication delay exceeds computation delay. Therefore, a cooperative offloading and scheduling aware system can achieve the balance between communication cost/delay and computation cost/delay if offloading and scheduling optimize jointly in the vehicular problem. In another scenario, the vehicular applications are real-time and delay-sensitive, therefore these applications required delay-sensitive services. However, due to mobility high-paid services are required to execute the offloaded tasks within their deadlines. Therefore, the proposed existing missed these aspects and leave the space how to trade-off between cost and delay of applications in the vehicular fog-cloud system.

The proposed work optimizes the joint problem where offloading and schedules are optimized together to achieve the tradeoff between delay and cost. To ensure the mobility features of vehicular applications, the study devised the dynamic offloading and scheduling algorithms framework which achieved the optimal results jointly in the vehicular fog-cloud network.

3 Problem description

Vehicular Fog Cloud Network encourages mobility-aware applications with seamless wireless connections and ubiquitous cloud services. Generally, users of applications will have to pay communication costs and computation to perform their activities. Therefore, the VFCN must offer economical services to the user applications during mobility with minimum end-to-end delay. Whereas existing VFCN offers cloud services based on heavyweight virtual machines that are very costly and not mobility friendly. We propose a novel VFCN system based on container microservices, which we explain in the subsequent subsection.

3.1 Proposed VFCN ecosystem

The proposed Vehicular Fog Cloud Network (VFCN) system is a modular architecture. There are two main modules in the system, such as Mobility Aware Cost-Efficient Offloading Decision Engine (MCODE) module and

the Master node module, as shown in Fig. 1. The MCODE comprises different profiling technologies (i.e., Resource, program, and network profiling) and mobile technology for generating tasks data from different sensors. Furthermore, based on collected data about tasks, the MCODE module makes the offloading decision where/when to offload application tasks for execution. For instance, each task has different requirements; for instance, time-critical tasks must execute immediately, and less time-sensitive execute later. Thus, we prioritized each task via the Task Sequence phase in the Master Node module. Based on cost and latency sequence order of tasks, The Task The scheduler assigns these tasks onto the heterogeneous resources in the VFCN. Each computing node is deployed and integrated based on the container instead of virtual machines to reduce the setup-time of tasks on the appropriate cloud. Worker Manager is a sub-component in each cloud that manages a lookup table of all assigned tasks in the system. The Windows Docker X86 platform installed container images as an autonomous container for each task. The kernel allows all containers to intercommunicate on the same operating system (Window Docker). The main advantages of containers in the VFCN are that a user can only pay for utilizing functions rather than whole applications. Table 1 describes the mathematical notations of the problem formulation for the considered problem.

3.2 Application characteristics

In this problem, we consider the mobility-aware Internet of Things (IoT) applications in the VFCN. Each IoT application classifies into two types of tasks, such as delay-sensitive tasks and delay-tolerant tasks. The set of delay-sensitive tasks (e.g., critical heartbeat operations) needs an immediate response and a hard deadline. Whereas delay-tolerant tasks (e.g., patient's weekly intervention) have soft deadlines which require no quick response for execution. A Poisson distribution process follows the arrival of tasks to the system. It is the same process for all IoT applications.

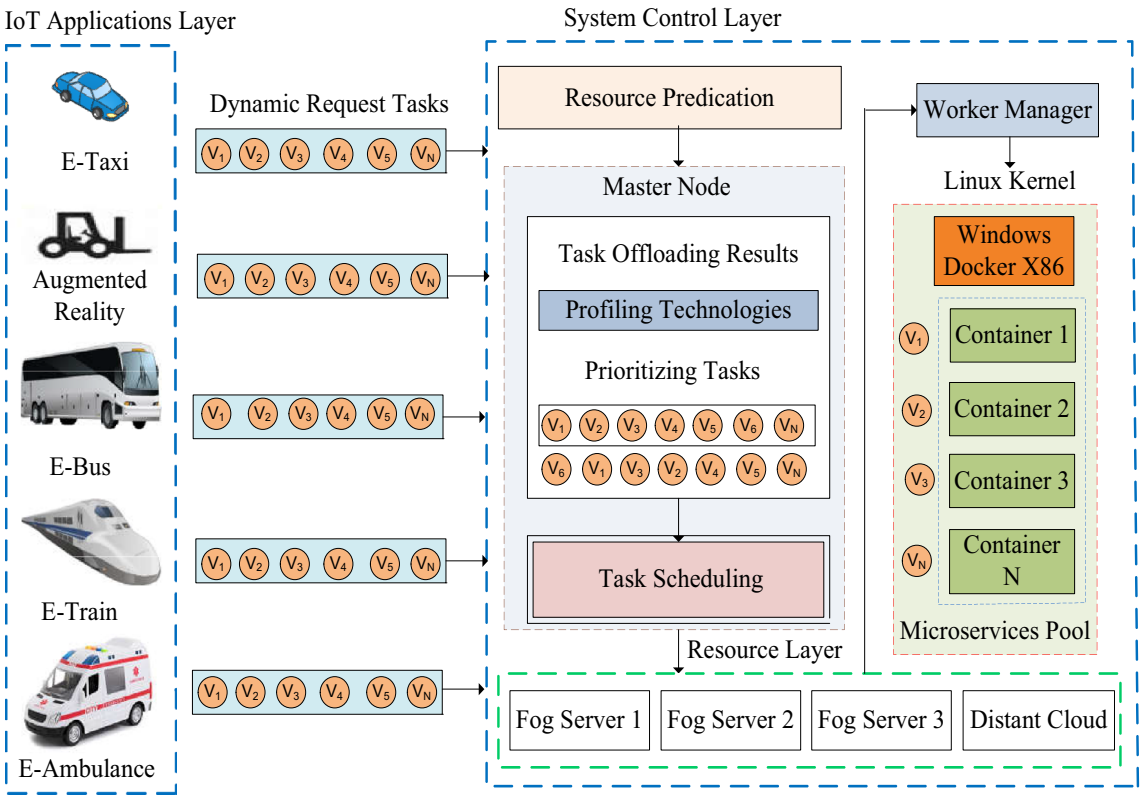

Fig. 1 Vehicular fog cloud network

Table 1 Mathematical-notation

Notation	Description
K	Number of computing nodes
k	The Kth computing node of K
rk	The resources of Kth computing ndoe
ζ_k	Speed of Kth computing node
A	A set of applications
a	The ath application of set A
a	The ath application of A
N	The Number of tasks
a, i	The ith specific task of application a
$S_{a,i}$	Data size of task i of application a
$W_{a,i}$	Workload of particular task i
$d_{a,i}$	Deadline of task i of application a
B_a	Available communication bandwidth for application a
$F_{a,i}$	Finish time of task i of application a
l_a^x	Location of application a
τ_a	Offloading time of application a
θ_c	Communication cost based on τ_a

3.3 VFCN resources

The Vehicular Fog Cloud Network (VFCN) is a cooperative assisted network that combines distinct fog nodes and cloud nodes. Each fog node is implemented at the roadside unit of base stations with different speed and resource capabilities. Generally, all fog nodes are heterogenous in their features. Simultaneously, the cloud is centralized computing that can access from anywhere via the Internet. To avoid load balancing, resource constraints, and mobility issues, VFCN exploited cooperative resources to execute the mobility-aware application in VFCN. Each computing node is composed of containers and services charges based on their usage instead of on-reserved, on-demand, and spot-instant. We implement different used-based pricing models instead of an hour, monthly, and yearly based resource-provisioning–the main reason to minimize the system costs of application during offloading and execution in the VFCN.

3.4 Pricing model of fog node

We implement the fog server based on a serverless model [58] where each service consider as a function as service (FaaS) [59]. This serverless model is useful to create microservices instead of monolithic services. Therefore, the pricing model is different from existing monolithic services-based fog servers. In microservices-based fog, the server only charges for used functions instead of entire applications. It is an on-demand FaaS based service; the users only charge for functions. We can only execute delay-sensitive tasks on the fog servers because these tasks are required immediate response because of their hard deadlines immediately.

3.5 Pricing model of cloud node

We prefer delay-tolerant tasks of application offload to the centric cloud because they have soft deadlines. The pricing models of centric clouds are on-reserved instances, and spot instance [60]. The prices of the centric cloud cheap as compared to the on-demand fog servers. Therefore, it is necessary, and we choose an optimal offloading time and pricing model applications for execution.

3.6 Problem formulation

We assume a VFCN system with a set of A applications and K adjacent fog clouds and core cloud computing. Each application $a \in \{1, 2, \ldots, A\}$ has a N number of tasks. The computational task $i \in \{1, 2, \ldots, N\}$ of application a is defined as a tuple $\{S_{a,i}, W_{a,i}, d_{a,i}, F_{a,i}\}$. Where $S_{a,i}$ shows data size, $W_{a,i}$ denotes the required cycles to process the workload of a task, $d_{a,i}$ denotes deadline, and $F_{a,i}$ denotes finish time of a task. K number of computing nodes including fog and cloud servers are considered in the vehicular fog-cloud network. Each node has its unique speed and resources, i.e., ϵ_k.

3.6.1 Communication cost scenario and calculation

The system cost consists of communication cost and computation cost to run each application in the network. However, communication cost calculated in the following way.

$$\phi_t = \begin{cases} \frac{W_{ai}}{B_a}, & O_i = 1 \quad a \leftrightarrow k, \\ \frac{W_{ai}}{B_a}, & O_i = 2 \quad k1 \leftrightarrow k2, \\ \frac{W_{ai}}{B_a}, & O_i = 3 \quad k2 \leftrightarrow k3. \end{cases} \tag{1}$$

The Eq. (1) shows communication cost is different from the application to initial fog node, e.g., O_i=1 then due to mobility the cost incurred an extra cost, i.e., O_i=2. However, due to the load balancing situation, one node can send workload to another node during mobility, e.g., O_i=3. Therefore, it is necessary, the communication cost per offloading may vary and handle smartly in the system to avoid users pay their minimum cost. Due to the mobility, we assume the initial location of a device application is

denoted by l_a^0 when generated computation tasks entered in the VFCN system. In the considered problem, we are exploiting random offloading for the device application a that may offload computation tasks to the VFCN system for execution via the Vehicle-to-Infrastructure (V2I) communication network. Each cloud in the VFCN does not overlap with each other. Thus, after offloading all the system tasks, the process should be done sequentially, not parallel. Due to the mobility, each device application should choose the best possible offloading time τ_a, which minimizes system costs of applications. When any device comes under the coverage area, then the distance between devices and VFCN becomes shorter. In this way, the communication cost and transmission latency of all tasks are minimized. At any offloading time, τ_a, we determine the transmission in the following way.

$$R_a = B_a log_2 \left\{ 1 + \frac{P_{tx} h_{ng} (d_{ag})^{-} r}{\overline{w}_o} \right\}. \tag{2}$$

In Eq. (2), the variable B_a is the bandwidth between application a and available node k in VFCN, P_{tx} illustrates transmission power, h_{ng} demonstrates channel gain, $\overline{w}_o$ shows the level of inference and noise, d_{ag} denoted the distance between device application a and any cloud in the VFCN. Furthermore, the initial location of device application a and moving speed of device application v_a at offloading time τ_a express as follows.

$$\begin{aligned} d_{ag} = \sqrt{l_g^2 + (D_g/2 - l_a^0 - v_a - \tau_a)^2}, if\ l_a^o \leq D_g/2, \\ \sqrt{l_g^2 + (D_g/2 - l_a^0 - v_a - \tau_a)^2}. if\ l_a^o > D_g/2. \end{aligned} \tag{3}$$

In Eq. (3) the variable l_g is the hop height of the VFCN, and D_g is the optimal coverage area for the offloading. When a roaming device application with speed v_a, then shows $l_a^0 \leq D_g/2$ shows the shorter distance between application and cloud. The communication cost and transmission latency of a task link with B_a bandwidth cost and data uploading time τ_t. We denoted the price of bandwidth utilization B_a for application a by a Φ_{com}.

$$\tau_u = \sum_{i \in a}^{N} \alpha_{a,i} \frac{S_{a,i}}{R_a} \times \phi_t. \tag{4}$$

The Eq. (4) determines the offloading required to transfer workload of application based on Eq. (1). The function F_a^{com} expresses the communication cost for application during offloading in the following way:

$$F_a^{com} = \tau_a \theta_c. \tag{5}$$

Equation (5) determines the communication cost of application a. The data size of a task $S_{a,i}$ directly affect uploading time and bandwidth price. Thus, in Eq. (5) the optimal offloading time F_a of application a is an important action with respect to system costs. Let $\alpha_{a,i}$=1 denotes a task i of application a upload tasks data to the any kth node, where $\alpha_{a,i}$=0 shows that tasks are not offloaded for processing. Equation (5) determines the offloading time of the application a.

3.6.2 Computation cost at different nodes

Each node in VFCN is distinct by their computing speeds which is depicted as $\zeta_j\{1, 2, \ldots, K\}$. Each cloud is consisted of C number of containers. Each container $c \in \{1, 2, \ldots, C\}$ works independently (i.e., function as a service) inside kth computing node. We employed a variable $x_{a,i,k,c} \in \{0, 1\}$ to show a task i of the application a is offloaded and assigned to the container c inside cloud k or not. We denote task completed maximum latency by a following way $T^e_{\{a,i\}}$, i.e., $T^e_{\{a,i\}} = \sum_{k=1}^{K} \sum_{c=1}^{C} x_{a,i,k,c} \cdot \frac{W_{a,i}}{\zeta_j}$. Whereas, Φ_{comp} denotes the computing cost of each node when it offers resources for execution. Thus, the average execution time of a task in the following way.

$$T^e_{a,i} = \sum_{j \in k}^{K} \sum_{c \in k}^{C} \sum_{i \in a}^{N} x_{a,i,k,c} \times \frac{W_{a,i}}{\zeta_k}. \tag{6}$$

Equation (6) calculates the computation time of application a as the required number of CPU instructions to execute the entire offloaded workload of applications. Whereas the study calculates the computation cost of application a in the following way.

$$F_a^{comp} = \sum_{i=1}^{N} T^e_{a,i} \times \Phi_{comp}. \tag{7}$$

Equation (7) determines the computation cost of application a.

3.6.3 Total lateness of applications

The total completed maximum latency consists of data offloading time, processing time, and data feedback time. The data downloading is fixed and denoted by t_B. Thus, the completed maximum latency of all tasks of application a is:

$$T_{a,i} = \max \sum_{i \in a}^{N} T^e_{a,i} + \tau_t + t_B. \tag{8}$$

Equation (8) determined the total delay of application a. We measure the total lateness of all applications in the following way.

$$T_{total} = \sum_{a=1}^{A} T_{a,i}. \tag{9}$$

Equation (9) determined the total lateness of all applications in the system during offloading, execution, and downloading the data from the system.

3.6.4 System computation cost

The total costs (i.e., communication cost and computation cost) of application a is formulated:

$$cost = F_a^{com} + F_a^{comp}. \tag{10}$$

$$T_{cost} = \sum_{a=1}^{A} cost. \tag{11}$$

Equation (11) determined the total system computation cost of all applications.

3.6.5 Conflicting objective pareto efficiency

Pareto quality, also known as Pareto optimality, is a condition in which no one person or preference criterion can be better off without making at least one other person or preference criterion worse off or losing something. Therefore, to achieve Pareto optimality, the co-operating offloading and scheduling problem solve as a joint optimization problem. In the study, we considered the trade-offs between two conflicting objectives, such as optimal decisions that need to be made. For instance, multi-objective optimization concerns involving two and three targets are the minimization of costs when optimizing the total delay of applications. The goal is to reduce costs of all applications T_{cost} while satisfying completed maximum latency T_{total} during the applications. Thus, task offloading decision and assignment $x_{a,i,k,c}$, bandwidth B_a, offloading time τ_a, and computing execution $T^e_{\{a,i,k,c\}}$ are mutually optimized. Thus, the joint optimization problem as **P1**.

$$\begin{aligned} &\min .T_{cost}, T_{total}\{x_{a,i,k,c}, B_a, \tau_a, T^e_{\{a,i,k,c\}}\} \\ &Subject \quad to \quad \sum_{a=1}^{A} T_{a,i} \le d_{a,i}, \quad \forall\{v = 1, \ldots, N\}. \end{aligned} \tag{12}$$

Equation (12) determined the conflicting objectives minimization for all applications.

$$\sum_{a=1}^{A} \tau_a, l_a^0, T_{j,0} = 0, \quad \forall\{v = 1, \ldots, N\}. \tag{13}$$

Initially, all values of all applications become zero as define in equation (13).

$$\sum_{a=1}^{A} \sum_{i=1}^{N} W_{a,i} \le kr, \quad \forall\{k = 1, \ldots, K\}. \tag{14}$$

The requested workload of applications must be less than the resources of all computing nodes as defined in Eq. (14).

$$\sum_{a=1}^{A} \sum_{i=1}^{N} x_{a,i,k,c} = 1, \quad \forall\{k = 1, \ldots, K\}. \tag{15}$$

Equation (15) ensures that, one task is assigned to only computing node at a time.

$$\sum_{k=1}^{k} x_{a,i,k,c} = 1, \quad \forall\{i = 1, \ldots, N\} a \in A. \tag{16}$$

Equation (16) ensures that, one node can execute one task at a time.

$$B_a \le B_{max}, \tag{17}$$

The requested bandwidth during offloading and downloading must be less its capacity as defined in Eq. (17).

$$x_{a,i,k,c} \in \{0, 1\}. \tag{18}$$

Equation (18) shows the binary assignment of tasks either they are scheduled or not in the VFCN.

4 Proposed CEMOTS algorithm framework

The multi-objective problems are always hard to solve when dynamic and mobility-aware services are required to run vehicle applications. A multi-objective problem of optimization requires many competing goals and has a set of optimal solutions from Pareto. Multi-objective evolutionary algorithms (MOEAs) can approximate the Pareto optimal set in a single run by evolving a population of solutions. The existing multi-objective algorithms ϵ-constraints method, Multiple-objective Branch-and-Bound, Normal Boundary Intersection (NBI), Modified Normal Boundary Intersection (NBIm), Normal Constraint (NC), Successive Pareto Optimization (SPO), Directed Search Domain (DSD), NSGA-II, PGEN (Pareto surface generation for convex multi-objective instances), IOSO (Indirect Optimization based on Self-Organization) SMS-EMOA (S-metric selection evolutionary multi-objective algorithm), and many have long convergence and computing speed during the process. Therefore, it is hard to achieve our co-operative VFCN for vehicular applications.

To solve the multi-objective problem, the study proposed Cost-Efficient Mobility Offloading and Task Scheduling (CEMOTS) algorithm framework as shown in Algorithm 1 to solve the considered mobility aware task offloading and scheduling problem in multiple steps.

(i) Mobility Aware Task Offloading Phase (MTOP) phase, (ii) Task Sequencing Phase, (iii) Task Cloud-Matching Phase, and (iv) Cost-Efficient Task Scheduling Phase (CTOS). The primary goal of CEMOTS is to optimize both objectives of all applications together. Each phase we will explain in subsections.

Algorithm 1: CEMOTS Algorithmic Framework

Input: $a \in A, i \in \{1, 2, \ldots, N\}$, $d, a, i \in \{1, 2, \ldots, N\}$
Output: T_{total}, T_{cost}

```
begin
    foreach (a  A) do
        foreach (i ∈ a in N) do
            Call Mobility Aware Task Offloading Phase;
            if (T_{a,i} ≤ d_{a,i} ∈ d, a, i) then
                Calculate objective functions for each
                application based on equation (12);
            Call Task Sequences;
            foreach (k  M) do
                if (T_{a,i} ≤ d_{a,i}) then
                    Call-Resource-Matching Phase;
                    T_total = i ∈ a ← k;
                    T_cost = i ∈ a ← k;
                    Call Co-Operative Task Offloading
                    Scheduling Phase;
            if (T_total ≤ T*_total or T_cost ≤ T*_cost) then
                Call Solution Search Phase;
                Swap T_total ← T*_total;
                Swap T_cost ← T*_cost;
    return T*_total, T*_cost;
```

4.1 Mobility aware task offloading phase (MTOP)

The study proposes a mobility-aware task offloading phase (MTOP) method with two primary goals. The first goal is to choose an optimal offloading time for all applications to minimize communication delay and communication cost of applications. The second is to offload the tasks from one computing node to another node during the mobility of applications in VFCN. The MTOP only reduces communication delay and communication cost; however, the computation delay and computation are minimized via scheduling methods.

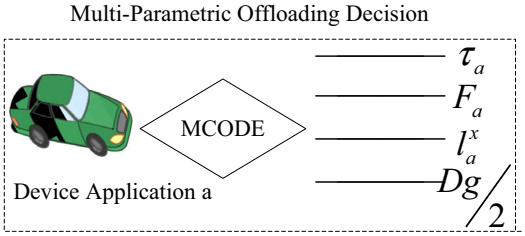

Fig. 2 Multi-parametric offloading decision for all applications

4.1.1 Mobility scenario

Figure 2 demonstrates the VFCN offers mobility aware cloud services to the roaming device application a via interconnected base stations (BTSs). These BTSs are connected via a high fiber cable, and handoff connections are managed automatically by the system. We are optimizing the objective functions T_{total} and T_{cost} that can be observed due to the initial mobility location l_a^0 of application a and moving speed v_a have performance influence in the considered optimization problem. Each application generates compute-intensive tasks, and those are arriving randomly to the system. However, due to mobility, all devices roaming among multiple access points. Whereas, a device application a is roaming proximity closer to the VFCN, and then $l_a^0 \leq D_g/2$ shows a device can choose an optimal offloading time τ_a for offloading tasks to the VFCN via V2I communication networks. We prioritize all tasks in advance, therefore $B_a \leq B_{\max}$ and $\sum_{i \in a}^{N} \{\alpha_{a,i} W_{a,i}\}$ computation size of all tasks is determined. It reduces the communication cost of an application during computation offloading.

At the optimal offloading time, all tasks are offloaded to any computing node in the VFCN. Since, the offloading decisions $\{\alpha_{a,i}\}$=1 are done. The offloading time τ_a is described as a function with computing capabilities $\sum_{k=1}^{K} \zeta_k$ computing node in the VFCN as follows:

$$\tau_a = \sum_{i \in a}^{N} \sum_{k=1}^{K} \alpha_{a,i} . \left(\frac{S_{a,i}}{R_a} + T_{a,i}^e + t_B \right) \leq d_{a,i}. \tag{19}$$

Equation (19) determines the offloading time of application a. Whenever computation tasks have randomly arrived, if the device application a is leaving the coverage area of VFCN, then $l_a^x > D_g/2$ shows the distance between device application a and clouds are increasing. Whenever the distance is larger, the offloaded tasks incurred more data transmission costs as well as time. Therefore, the offloading is optimal when $\tau_a = 1$ and channel data rate R_1 of a is determined in the following way.

$$R_a^1 = B_a log_2 \left\{ 1 + \frac{P_{tx} h_{ng} (d_{ng}(\tau_a = 1))^{-} r}{\overline{w}_o} \right\}, \tag{20}$$

Equation (20) determined available channel rates for offloading.

$$d_{ag} = (\tau_a = 1) \sqrt{l_g^2 + (l_a^1 - D_g/2)^2}. \tag{21}$$

Equation (21) determines the distance between applications and available computing nodes during mobility in the

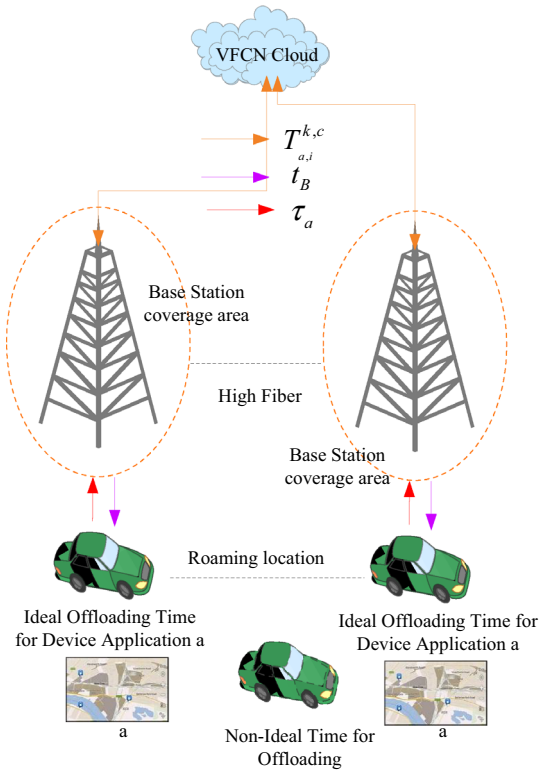

Fig. 3 Task offloading scenario during roaming of application a

VFCN. The proposed Mobility Aware Task Offloading Phase (MTOP) algorithm determines the ideal offloading time and the decision to offload tasks to minimize system costs and satisfy completed task maximum latency. Algorithm 2 shows all optimal time-selection offloading processes which minimize system costs of applications (Fig. 3).

Figure 3 denotes mobility scenario of application during offloading and roaming in VFCN.

4.1.2 Selection of ideal offloading time for IoT applications

Let's say that in the mobility case, τa offloading time is any actual number, and x is a single one. We use x to close a number with a $p(x)$ defined probability density function. If the probability density is high around a point x, this means that the random variable τa is likely to be close to the optimum time of offload. On the other hand, if $p(x) = 0$, τa, is not an optimal time to offload. We assume that Ix is some small interval around point x as shown in Fig. 3 to transform the probability density function into a probability. The likelihood of τa is that interval which depends on both the density $p(x)$ and the length of the interval to be calculated in the following way. Then we assume that p is continuous random.

$$P_r(\tau_a \in I_x) \sim p(x) \times I_x. \tag{22}$$

The Eq. (22) defines the offloading density during mobility. Here, we do not have true equality since the density of p will differ over Ix. But, as the interval I_x shrinks around the point x, the approximation gets better and better, p can get closer and closer to a constant within that tiny interval. As I_x shrinks down to x, the chance of $pr(\tau a \in Ix)$ hits zero.

In general, to determine the probability that τ_a is in any subset A of the real numbers, we add up the values of $p(x)$ in the subset. By add-up, we mean integrate the function $p(x)$ over the set A. The probability that τ_a is in A define in the following way.

$$P_r(x \in A) = \int_A p(x)dx \tag{23}$$

The Eq. (23) calculates the likelihood of random time-interval offload values. Supposing that if I is the $I = [a, b]$ interval with $a \leq b$, then the likelihood is $a \leq \tau_a \leq b$.

$$P_r(x \in I) = \int_I p(x)dx = \int_a^b p(x)dx. \tag{24}$$

The Eq. (24) specifies the best interval likelihood of the offload function. It must satisfy two conditions for a function $p(x)$ to be a probability density function. It must be non-negative because the integral equation is always non-negative (24) and must be integrated into one. Thus, the τ_a likelihood becomes 1 and is calculated as follows.

$$p(x) \geq 0 \quad \forall x, \int p(x)dx = 1. \tag{25}$$

Since the Eq. (25) is the correct way to describe a function for probability density. We don't care for p, however, either it is too reliable or discontinuous. We might decide which ad follows.

$$P_r(\tau_a \in (x, x + dx)) = p(x)dx. \tag{26}$$

dx is a small number in the Eq. (26), so that $(x, x + dx)$ is an infinitesimally short interval of I_x about x. In the

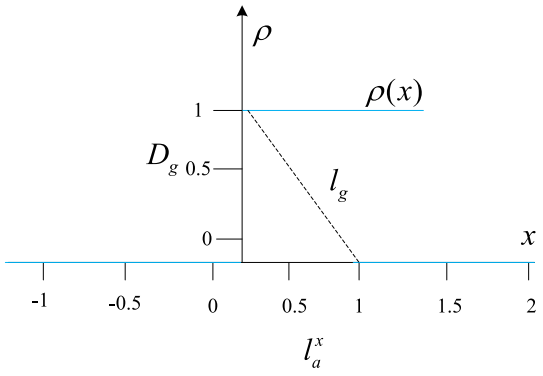

Fig. 4 probability density function for offloading time

equation equation, the case approximation becomes valid (26) at least if p is continuous.

We assume that a number in the [0, 1] interval can be given by a uniform distribution in the [0, 1] interval to τa. The function of the probability density of τ_a is defined by

$$p_x = \begin{cases} 1, & if\ x\epsilon[0,1] \\ 0, & \text{otherwise,} \end{cases} \tag{27}$$

The Eq. (27) defines the scenario of Fig. 4 about probability density function. The function $p(x)$ is a valid probability density function when it is non negative and integrates to 1. If I interval contained in [0, 1], we can say $I = [0,1]$ with $0 \le a \le b \le 1$ in the interval

$$\begin{aligned} P_r(x \in I) &= \int_I p(x)dx \\ &= \int_I 1dx \\ &= \int_a^b 1dx = b - a = I. \end{aligned} \tag{28}$$

For any I interval, $P_r(x \in I)$ is equal to the I intersection length with the [0, 1] interval as defined in the equation (28).

Algorithm 2: MTOP Scheme

Input: $a \in A$,τ_a,v_a,l_a^0,$k \in K$,$i \in \{1,2,\ldots,N\}$, $\{W_{a,i}, T_{a,i}\}$

```
begin
    foreach (i ∈ a in N) do
        Calculate distance between request and computing node based on equation (21);
        if (l_a^0 ≤ D_g/2) then
            Choose an optimal offloading time based on equation (19);
            if (τ_a ≤ d_{a,i}) & l_a^0 then
                Offload tasks VFCN system;
            Calculate system costs and total lateness based on equation (12);
            T_cost ← i^N ∈ a;
            T_total ← i^N ∈ a;
        T*_cost = T_cost + 1;
        T*_total = T_total + 1;
        else
            Re-Offloading again in optimal time;
    Return T*_cost, T*_total;
```

4.2 Task sequencing phase

The distributed tasks to be processed by the heterogeneous clouds in the VFCN. The tasks have different attributes in terms of workload, required CPU, and critical deadline. We priories this way denotes all tasks by their attributes such as deadlines and maximum lateness of all tasks, i.e., T_{di}=$\sum_{i\in a}^{N}\{0, F_{a,i} - d_{a,i}\}$. The computing task latency of a task T_{di} would be determined by actual finish time F_i minus due deadline $d_{a,i}$. The finish time $F_{a,i}$ of the task i is decided by the execution time $T_{a,i}^e$ and the available time slot assigned to a task by the cloud k. However, because of the heterogeneity of the clouds, F_i cannot be estimated before scheduling. The notation $\overline{T_{a,i}^e} = x_{k,c,a,i}\frac{\sum_{j=1}^{K} W_{a,i}}{\sum_{j=1}^{K} \zeta_j}$ is employed to estimate the average execution time of task on all clouds. The initialization of each cloud is assumed to be 0. The processing execution of a task i mathematically in the following way.

$$\begin{aligned} T_{d_i} &= \{0, F_{a,i} - d_{a,i}\} \\ \overline{F_{a,i}} &= \sum_{i\in a}^{N} x_{k,c,a,i}.T_{a,i}^e \frac{\sum_{j=1}^{K} W_{a,i}}{\sum_{j=1}^{K} \zeta_j} \end{aligned} \tag{29}$$

To satisfy the requirement of tasks with different requirements, we prioritize all tasks by the following methods.

(1) Earliest due date (EDD): Their deadlines sort all tasks. The smallest deadline task sort first (e.g., priority). Whenever the two tasks' deadline is the same, the task with the smaller size is ranked by a higher priority.
(2) Shortest Processing Time First (SPF): All tasks need to be sorted by lateness or tardiness requirements. Whenever a task incurs with the smallest lateness, that should be scheduled first by the scheduler. If the two tasks' lateness is the same, then the smallest total workload is arranged first.
(3) Smallest Slack Time (SSTF): The smallest slack time task is arranged first to the scheduler.

Figure 5 denotes task sequencing scenario application a. However, it is the same for all applications in VFCN during offloading before execution.

4.3 Task fog-cloud-matching phase

In this section, we find the resources for each task to heterogeneous computing node networks. We exploit two players game theory rule [35] to the task resource matching problem. Since tasks and resources are assumed to be players of the game. Each player has different choices to choose only the ideal node for execution. We consider the N tasks of application $a \in A$ and K resources as the players. We let the tasks rank their choices and annoying to match the tasks and clouds so that every match is stable and reduces the objective function of the applications. Let assume, we suppose the subsequent instance with the tasks N=$(v1, v2, v3, v4, v5, v6, \ldots, N)$ and clouds K=$(k1, k2, k3, \ldots, K)$.

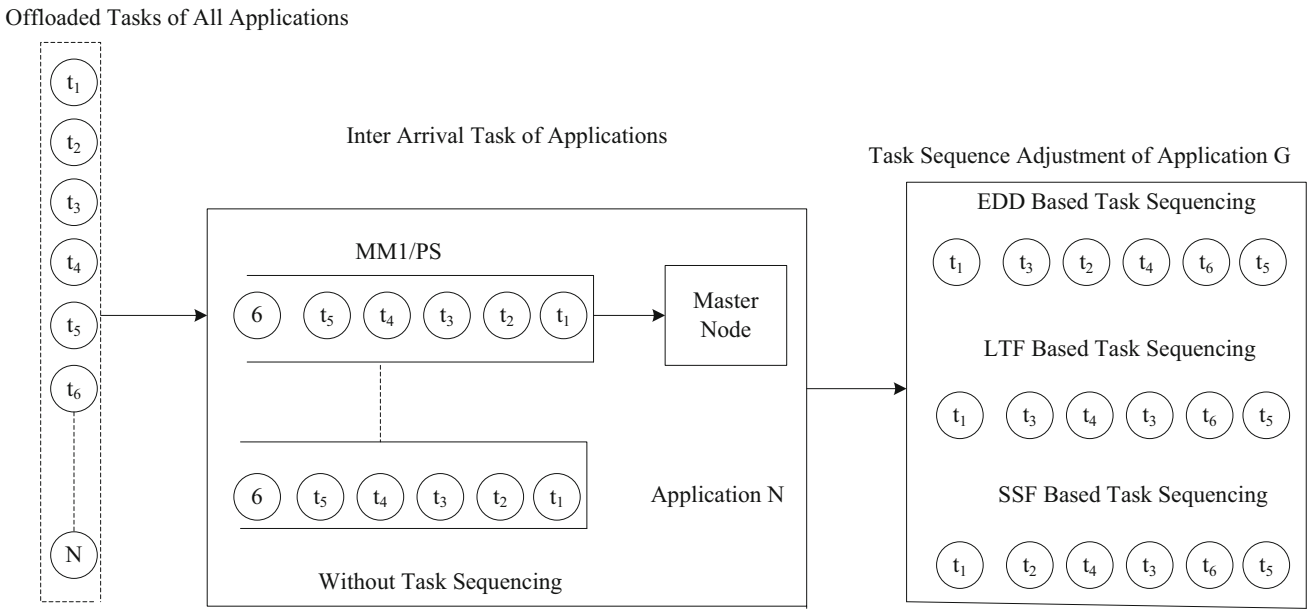

Fig. 5 Initial task sequencing of applications

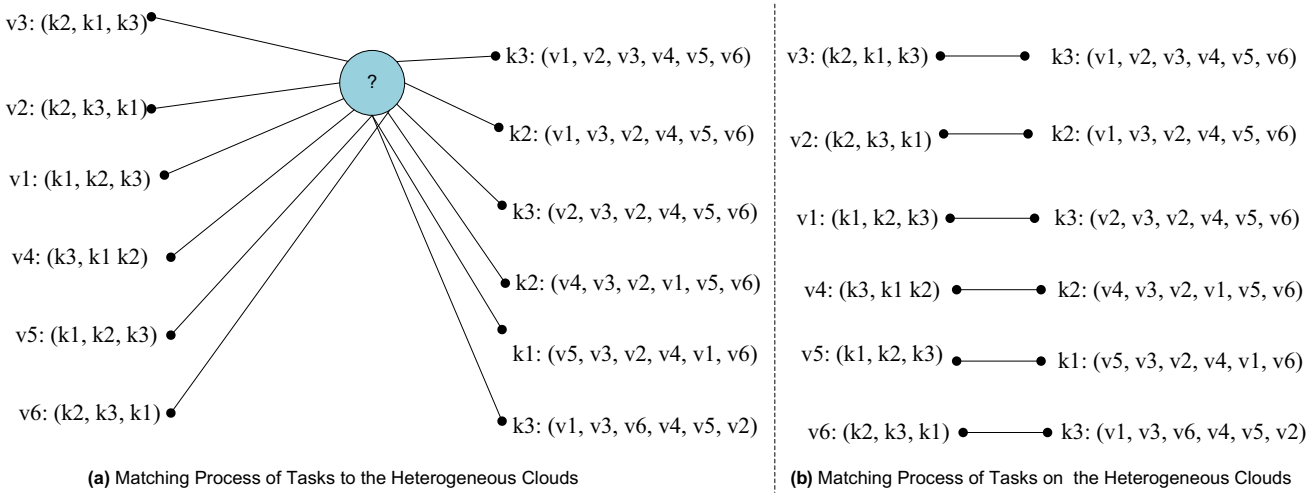

Fig. 6 Optimal searching and selection of resources

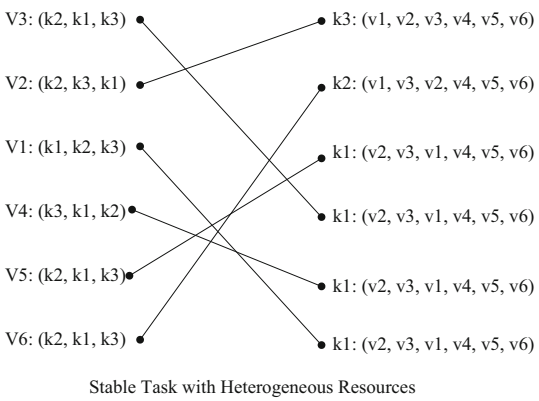

Fig. 7 A stable matching is shown

Figure 6a shows that we do not know which cloud is suitable that satisfies the requirements of tasks in advance. So that cloud $k1$ would prefer to be matched with $v2$. One likely matching is illustrated in Fig. 6b. All clouds' prices are different; therefore, it is critical to choose the appropriate cloud for the tasks to minimize application cost under deadline requirements. In this condition, $v1$ and $v2$ are getting their first selection and $v3$ their second preference. However, $k2$ prefers $v3$ so that matching is unstable.

Figure 7 the stable results of resource matching the requested tasks in the VFCN. We want to write some formal definitions of the resources matching problem by using game theory rules.

Definition 4.1 Definition of matching resources:

We assumed two disjoint sets $N \in a \in A$ and K show tasks and resources, respectively. We match each element of N and K as a reference list:

$$f : N \leftarrow M^N, \quad g : M \leftarrow N^M. \tag{30}$$

In Eq. (30), the variables f and g are the matching functions where matching B is a bijection between N and M. If $v_i \in N$ and $k \in M$ are matched by B showed as follow in Eq. (31).

$$B(v_i) = k. \tag{31}$$

Definition 4.2 Definition of a matching resources:

A pair (v_i, k) is called block a matching B if $B(v_i \in N) \neq k \in M$ but v_i prefers k to $B(v_i)$ and k refers k to v_i to $M^{-1}(k)$.

Definition 4.3 Definition of a optimal matching:

An analogous B with no blocking pair is said to be stable.

Theorem 4.1 *All possible Resource Matching algorithm executions yield the same stable matching, and in this stable matching, every task in any stable matching has the best cloud possible.*

Proof Suppose an arbitrary execution of the α algorithm yields B and another execution yields B β gives B‘ such that $\ni v_i \in N$ such that v_i prefers $v_i^{'} = B'(v_i) tok = B(v_i)$. Without loss of generality, this implies that during α k‘ must have rejected s. Suppose, again, without loss of generality, and this was the first occasion that rejection occurred during α and assumed that this rejection occurred because of $k' = B(v_i')$. We imply that v_i‘ has no stable match higher in v_i preference list than k‘ (as we have assumed that this is the first rejection). Thus v_i‘ prefers k‘ to $B'(v_i')$ so that (v_i', k') blocks B'. Each task is, therefore, match in B with his favourite stable reviewer, and since α was arbitrary, it follows that all possible executions give the same matching. □

Theorem 4.2 *Theorem of clouds sub optimality: In a task optimal stable matching, each cloud has the worst possible matching.*

Proof Assume that the result is not true. Let Bo be a suitor-optimal matching and assume that there is a stable matching B' such that $\ni k$ such that k prefers $v_i = B_o^{-1}(k)$ to $v_i = B'^{-1}(k)$. This implies that (k, v_i) blocks B' unless s prefers$B'(v_i)$ to $M_o(v_i)$ which contradicts the fact the s has no stable match that they prefer in B_o. □

Figure 8 shows the entire matching process of Algorithm 3 for all applications on resources.

Algorithm 3: Resource Matching Algorithm

```
Input: a ∈ A, v_i ∈ N, k ∈ K;
begin
    foreach (a ∈ A) do
        foreach (v_i ∈ N) do
            foreach (v_i ∈ N) do
                Allocate every v_i ∈ N to k ∈ K for all unmatched;
                Choose some unmatched v_i ∈ N, let k be the optimal node;
                if (k is unmatched) then
                    set B(v_i) = k;
                else if (k is matched) then
                    if (v_i ← k to B^-1(k)) then
                        Add matched tasks with resources to the list;
                        Add K(k) = s ← PList[v_i, k];
            Otherwise v_i remains unmatched and remove k from PList[v_i, k];
    Repeat step 2 until all v_i ∈ N are matched.
End Main
```

4.4 Dynamic task scheduling phase

Dynamic Scheduling is a strategy in which the hardware rearranges the execution of instructions to eliminate stalls while keeping data flow and exception activity intact. Dynamic scheduling has the following advantages: It can manage situations where dependencies are uncertain at compile time. Scheduling is the process to ensure the quality of service (QoS) of applications during execution on the resources. The main goal of scheduling is to execute all tasks with minimum lateness and minimum cost on the available resources without degrading their performances. This study proposes a Co-operative offloading and task scheduling (CTOS) algorithm that iteratively schedules and obtains near-optimal solutions under polynomial time. The study shows the process of scheduling Algorithm 4 in the following way.

- Algorithm 4 takes following parameters as inputs $a \in A$, $v_i \in N$, $k \in K$, $PList[v_i, k]$, Candidate-Solutions [D, C].
- The $PList[v_i, k]$ is matched list of tasks of all applications on different resources based on their QoS requirements.
- All tasks are scheduled with minimum lateness and cost under their deadline requirements.
- , However, due to mobility, Algorithm 4 calls Mobile Aware Solution Searching to replace existing objective solutions of all applications with a new solution.

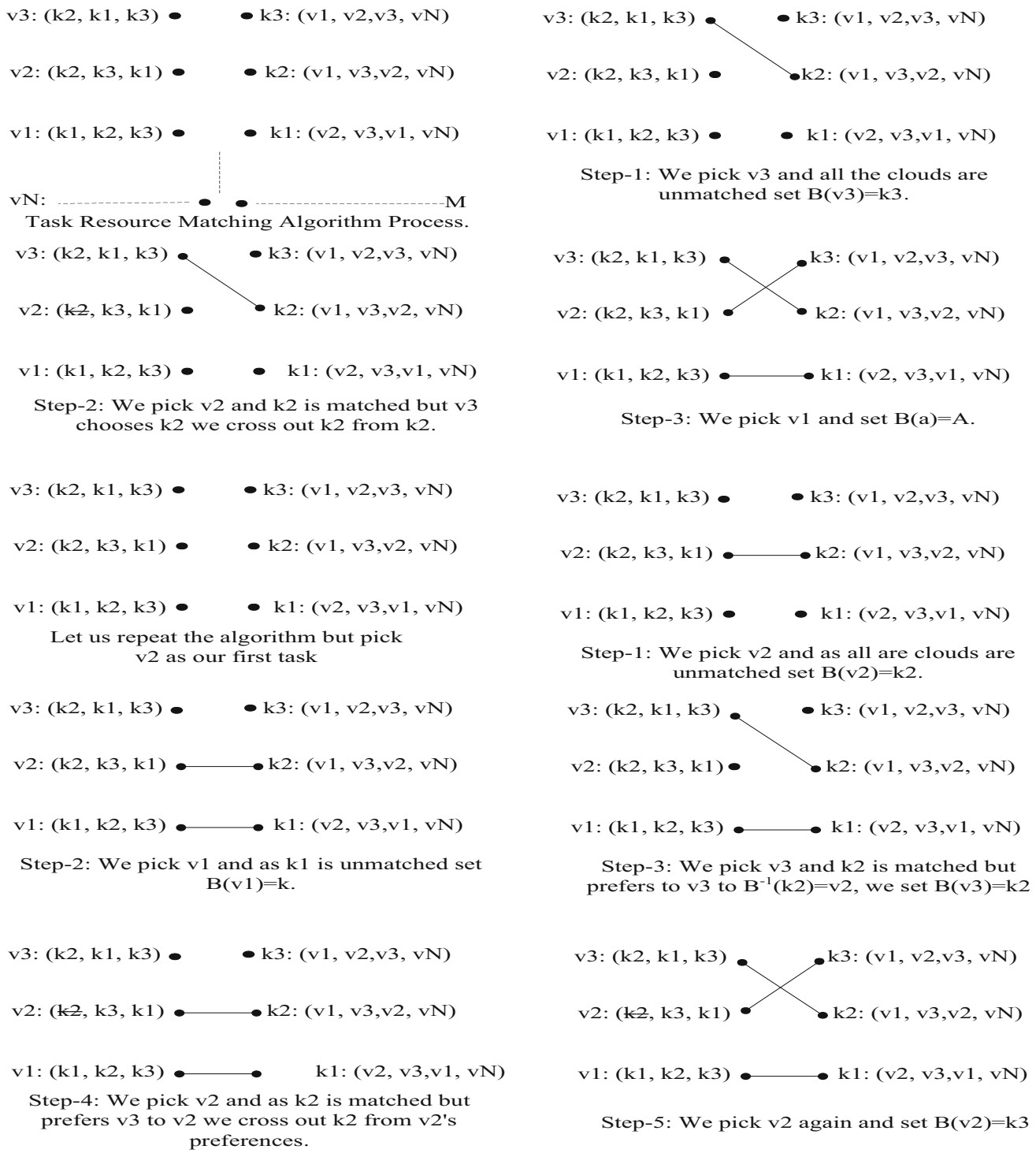

Fig. 8 Decision matching process algorithm

- Algorithm 5 returns the candidate solutions of objective functions with new values during mobility. The Candidate-Solutions [*D*, *C*] consists of two variable array-list, i.e., all solutions with optimal lateness (*D*), and all solutions with optimal costs, i.e., *C*. Algorithm 5 is a global searching-based algorithm that obtained a near-optimal solution with minimum iteration during the mobility of applications.

Algorithm 4: CTOS

Input: $a \in A$, $v_i \in N$, $k \in K$, $PList[v_i, k] \in a \in A$, Candidate-Solutions $[D, C]$;

1 **begin**
2 $a \leftarrow T_{cost}$ Initial cost of applications;
3 $a \leftarrow T_{total}$ Initial delay of applicaitons;
4 **foreach** *(Candidate-Solution*$[D, C]$ *as* $D \leftarrow T^*_{total}$, $C \leftarrow T^*_{cost}$ *)* **do**
5 **foreach** $(PList[v_i, k] \in a \leftarrow A)$ **do**
6 **if** *Calculate total cost of all applications based on equation (11);*
7 *Calculate total lateness of all applications based on equation (9);*
8 $T_{cost} \leftarrow v_i^{N \in a \in A} \leftarrow k$;
9 $T_{total} \leftarrow v_i^{N \in a \in A} \leftarrow k$;
10 **then** $v_i \leftarrow T^e_{i,k} \leq$ & $W_i \leq r_k \in K$
11 E
12 nd Inner Condition;
13 Call Algorithm 5;
14 **if** $(a \leftarrow T_{cost} \leq a \leftarrow T^*_{cost}$ & $a \leftarrow T_{total} \leq a \leftarrow T^*_{total})$ **then**
15 Swap $a \leftarrow T_{cost} \leftarrow a \leftarrow T^*_{cost}$;
16 Swap $a \leftarrow T_{total} \leftarrow a \leftarrow T^*_{total}$;
17 End IF
18 **return** $a \leftarrow T^*_{cost}$, $a \leftarrow T^*_{total}$;

4.5 Mobile aware solution searching

The Task Scheduling Process initially schedules all tasks based on the $PList[vi \in, k \in K]$ preference list. Owing to the complex and unpredictable existence of various computing resource paradigms and the intermittent shift of in-network material, the current goal is no longer to expand the best solution remaining. We, therefore, use a changed local search and neighborhood structure for task scheduling of the considered problem. Breadth-first search (BFS) [61], Hill Climbing search [62], and Beam search [63] are combined with existing studies to iteratively get the best solution between different state solutions. There are several existing local search algorithms. In the search field, there are several solutions; compared to the current one, we choose an ideal neighbor-sensitive solution. We use the 5 algorithm to illustrate the process of selecting an optimal solution between candidate solutions. Initializations of various variables are displayed on lines 1-7. We leverage the t temperature in the optimization problem to switch rapidly from the current solution to another solution. If the current t is greater than zero and still has minimal iteration in hand, we set the condition. The algorithm tests if there are delta differentials in the current solution and the new solution (this means the new solution optimal than the existing one). It will replace the current solution with an optimal solution for the candidate. Inline 12-19, processing of these conditions is carried out. If all possible candidate solutions in the search area of the neighborhood are not more optimal than the existing one, the state of line 21 is completed, the system is needed with the current solutions. The current optimal solution of the problem's goal is returned to the 5 algorithm.

Algorithm 5: Mobility Aware Solution Searching

Input : $PList[v_i \in, k \in K]$;
Output: Candidate-Solutions $[D, C]$;
1 $C \leftarrow T_{cost} \leftarrow= PList[v_i \in, k \in K]$ Initial Solution;
2 $D \leftarrow T_{total} \leftarrow= PList[v_i \in, k \in K]$ Initial Solution;
3 $C^* \leftarrow T^*_{cost} \leftarrow= PList[v_i \in, k \in K]$ Optimal Solution;
4 $D^* \leftarrow T^*_{total} \leftarrow= PList[v_i \in, k \in K]$ Optimal Solution;
5 α cooling parameter;
6 $t \leftarrow$ tempreture=1000∼500 ;
7 $iter \leftarrow 0$;
8 $max \leftarrow 10$ Maximum iterations;
9 **begin**
10 **while** $(t > 0)$ **do**
11 **while** $(iter \leq max)$ **do**
12 $T^*_{total} \leftarrow$ randomly select neighbour solution $T_{total} \in N$(Candidate-Solutions[D,C]);
13 $T^*_{cost} \leftarrow$ randomly select neighbour solution $T_{cost} \in N$(Candidate-Solutions[D,C]);
14 $\Delta \leftarrow \frac{T^*_{cost} - T_{cost}}{t}$;
15 $\Delta \leftarrow \frac{T^*_{total} - T_{total}}{t}$;
16 **if** $(\Delta \leq 0)$ **then**
17 $T_{total} \leftarrow T^*_{total}$;
18 $T_{cost} \leftarrow T^*_{cost}$;
19 **if** $(T_{total} \leq T^*_{total}$ & $T_{cost} \leq T^*_{cost})$ **then**
20 Swap current solution with new one;
21 $T_{total} \leftarrow T^*_{total}$;
22 $T_{cost} \leftarrow T^*_{cost}$;
23 **else if** $(rand(0, 1) \leq e^{\frac{\Delta}{t}})$ **then**
24 $T_{total} \leftarrow T^*_{total}$;
25 $T_{cost} \leftarrow T^*_{cost}$;
26 **else**
27 Stay with current solution;
28 $D \leftarrow T_{total} \leftarrow T^*_{total}$;
29 $C \leftarrow T_{cost} \leftarrow T^*_{cost}$;
30 Candidate-Solutions$[D, C]$;
31 $iter \leftarrow iter + 1$;
32 $t \leftarrow t \times (1 - \alpha)$;
33 End Conditions;
34 **return** Candidate-Solutions$[D, C]$;
35 End Loop;

There are many worst and optimal solutions in the searching list, as shown in Fig. 9. However, Algorithm 5 only replace the existing solutions with the new optimal solution in the VFCN for all applications. Figure 9 the minus shows the worst solutions, and minus values show

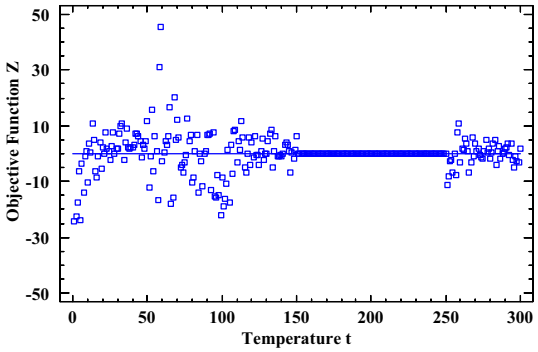

Fig. 9 Random generated objective solutions

the optimal solution with different convergence and exploration levels during mobility applications.

4.6 CEMOTS time complexity and overhead

The overhead and time complexity of the CEMOTS components via asymptotic notations $O(V, E)$ for the set of applications. The time complexity of the CEMOTS framework is equal to $O(V+E)+O(n\times m)+(n^3)$. Whereas, V shows nodes of all tasks and E shows the dependency between task i and task j due to precedence constraint requirements. The resource matching decision phase uses a modified CTOS method and MTOP method, the time complexity is $O(n\times m)$. Whereas, n the number of iterations for each task to match to m. The iteration for resource matching will repeat until and unless all tasks are assigned to all paradigms m. MTOP exploits $O(N^3)$ time complexity while applying search space for the best solution. The n is a counter variable that tells about t is slowly minimized during task scheduling onto heterogeneous paradigms. The number of matching is denoted by n. Thus, overhead and complexity of CEMOTS is equal to $O(V+E)+O(n\times m)+(n^3)$.

5 Performance evaluation

In this paper, the proposed VFCN system consists of modular architecture. The related simulation parameters of architecture are defined in Table 2. We designed a VFCN system by exploiting existing frameworks [30, 32, 33]. Furthermore, an IoT application emulator, i.e., Amazon GenyMotion [34] is deployed that running as virtual images on AWS product as a service (PaaS) virtual machines and on the desktop machine. We also implemented a cloud-based android emulator running as virtual images on a desktop machine as a software (SaaS). We have constructed a virtual cloud, i.e., fog and cloud, that will be scaled up and down as on-demand while emulator and configurations are done in the cloud. We designed all proposed schemes in the JAVA language in advance application programming interface (API) and tested on Intel (R) Core (TM) i5-3475 CPU @ 3.30GHz, 10G Memory machine. We implemented the 64bit X86 Amazon Machine Image (AMI) mobile cloud environment with Android 7.0 nougat for mobile cloud applications. We installed mobile applications APK's (i.e., Android Packages) on 64bit X86 AMI. The heterogeneous cloud configuration is illustrated in Table 3 with their characteristics and specifications. The application workloads are explained in Table 4.

Table 2 Simulation parameters

Simulation parameters	Values
Windows OS	Linux Amazon GenyMotion
Arrival process by poisson	5 min
Centos 7 runtime	X86-64-bit AMI
Languages	JAVA, XML, Python
Android Phone	Google Nexus 4, 7, and S
Experiment repetition	160 times
Simulation duration	12 h
Simulation monitoring	Every 1 h
Evaluation method	ANOVA single and multi-factor
Amazon on demand service	EC2 t3
Android operating system	GenyMotion
Mobility	Namadic
Application interface	Desktop, Cloud or Mobile APP:
t_B	50 s
B_{max}	10 Megabyte
v_a	80–120 km/h
$S_{a,i}$	1000–1500 MB
P_{tx}	0.1
h_{ng}	0.005
d_{ng}	200
$\overline{w}_o$	15
l_g	100 m
a	500–20000 devices

Table 3 Resource specifications of study

Cloud	CORE	MIPS/CORE	RAM(GB)	Storage(GB)	Cost-PH
k_1	i3	1000	800	1000	0.5¥
k_2	i5	3000	1000	2000	0.7¥
k_3	i7	5000	1200	4000	0.9 ¥
k_4	i9	10000	3200	10000	0.05¥

Table 4 Workload analysis of IoT applications

Workload	$W_{a,i}$(MB)	N
Healthcare	15.8	825
Augmented reality	24.8	600
E-Transport	38.8	640
3D-Game	29.8	750

5.1 Baseline offloading, system and algorithm approaches

- Random offloading scheme: The devices offload computation tasks to the VFCNs as decision denoted via $\alpha_{a,i} \in \{0, 1\}$. This offloading process will choose random device offloading time τ_a between 0 and 1 and follows an equal probability distribution. The offloading τ_a, $x_{k,c,a,i}$. B_a are optimized as generally.
- Direct offloading scheme: The devices send computation tasks when offloading time τ_a=1 is optimal. Whereas, remaining values τ_a, $x_{k,c,a,i}$. B_a are optimized as generally.
- TDMA (Time Division Multiple Access) based offloading scheme: The devices offload computation tasks in different $\tau_a \in \{0, 1\}$ time-slots. However, this process supports cooperation between base stations and among different fog clouds or core clouds.
- Vehicular Fog Cloud Network (OFCN): This is the proposed system that mobility-based services to vehicle applications based on container technology.
- CEMOTS is an algorithm framework that consists of MTOP and CTO's methods to run all applications under their deadlines.

For mobility scenario, we employ the mobility aware device movement trace mechanism offered by the Every-Where-lab [38]. The trace method traces the user movement in the given scenario (road network of southeast university). The entire road boundary is 17 × 17 km. Dataset provides 100,000 end-users in the one-point way to invoke cloud services during mobility, whereas location coordinates of devices are traced every 25 seconds. The proposed scheme MTOP offloads application tasks in the optimal time, minimizing system costs and executing them with minimum latency.

Figure 10 shows the offloading performances of different offloading schemes in different locations. The existing direct and random offloading schemes are static and work perfectly when the environment is not mobility-based. However, the proposed MTOP offloading scheme incurs lower system costs and latency whenever moving devices are approaching closer to the access point. Figure 11

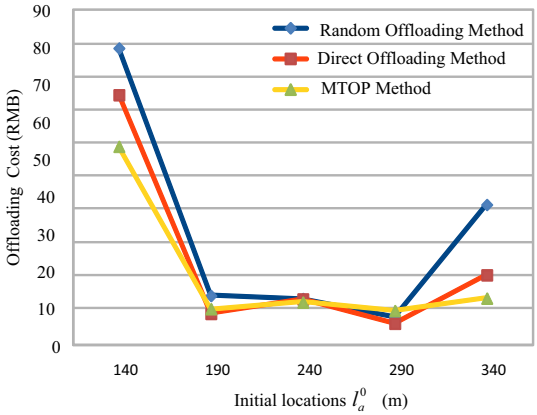

Fig. 10 Initial locations of tasks offloading

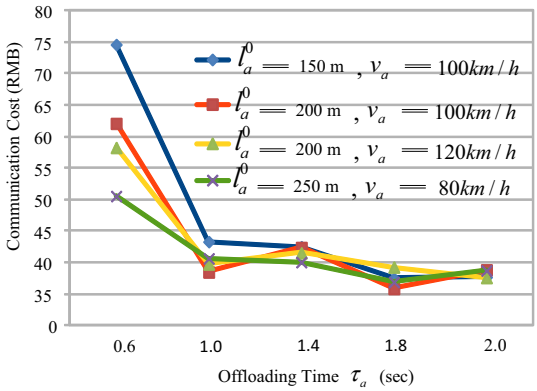

Fig. 11 Offloading time for all tasks

illustrates when to offload tasks to the cloud system for execution; different offloading time has different system costs and latency delay. Since direct offloading scheme and random offloading synchronously offload tasks to the system in different offloading times. However, these methods are seriously infected with heavy-weight overhead and system costs.

Thus, MTOP always chooses the optimal and best offloading time, which incurs minimum costs. The offloading decision is a decision that exploits different offloading schemes to make a decision when/where to offload tasks for execution. Figure 12 shows MTOP offloading is more effective than the existing offloading schemes; the reason behind this is that MTOP offloading decision always chooses an optimal offloading time which gains minimum communication time/cost and computation time/cost during offloading and scheduling. Hence, it is proved that MTOP works efficiently with optimal offloading, which is formulated as a convex optimization problem compared to static offloading schemes for IoT applications.

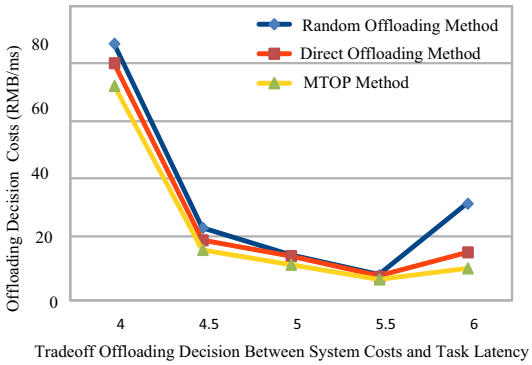

Fig. 12 Tradeoff between costs and latency offloading decision

5.2 Existing VFCN system architecture comparison

Figure 13 illustrates that, VFCN container-based system incurs lower boot-time cost compared with existing virtual machines based ThinkAir [36] and CloneCloud [37] system architecture for IoT application execution. Even though in the VFCN system, users only pay for a usage function, not for the whole application, Fig. 14 shows the CPU utilization and users system costs could be minimized by exploiting container-based VFCN based system. The average system costs and task latency could be improved and minimized by exploiting container-based microservices in the VFCN environment, as shown in Fig. 15.

The Fig. 13 shows the computation could be removed if the system suggested lightweight services based resources as compared virtual machine lightweight resources as suggested in the existing studies

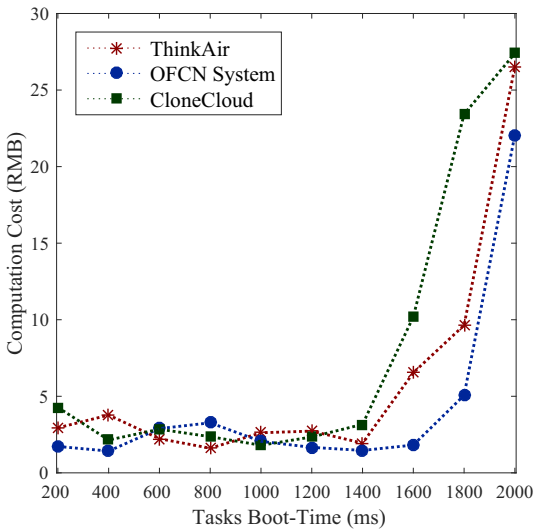

Fig. 13 Boot-time of contaienr container in VFCN

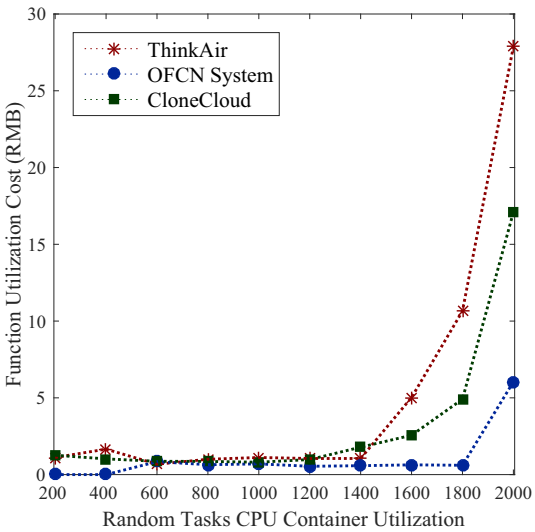

Fig. 14 CPU utilization of container in VFCN

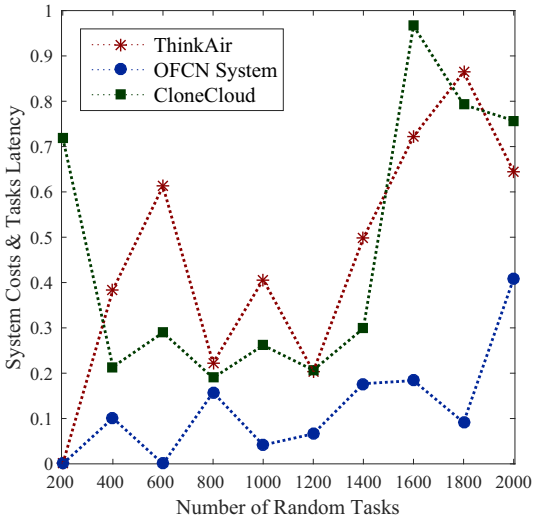

Fig. 15 Tradeoff cost and task latency

The devised system not only improved the system cost and lateness of application, but improved the overall utilization of the resources as shown in Fig. 14.

5.3 Performance metrics

In the paper, the part calibration recommends the experimental method that randomly generates various application tasks shown in Table 4. There are four distinct numbers of applications taken into account for research in the experiment. The paper sets the deadline for tasks based on the following equation for the various types of deadline constraint:

$$d_{a,i} = F_{a,i} + \gamma \times F_{a,i} \tag{32}$$

The d a, i deadline of the assignment is achieved by the earliest completion date and a definite proportion of the early completion time. We used γ to prove the parameter to manage the tightness of the deadline of the mission, with a range of values between .2 and 1, i.e. $\gamma \in \{.2, .4, .6, .8, 1\}$, as shown in Fig. 16a, so that each task can get five times the deadline size, i.e. $D1$, $D2$, $D3$, $D4$, $D5$. Furthermore, to verify algorithms' performance in the dissimilar deadline of each task, the paper calculates according to the following formula (32). The paper uses RPD (Relative Percentage Deviation) statistical analysis to measure the recitals of the VFCN, MTOP and CTOS. The component calibration parameter assesses the power consumption consumed by dissimilar parameters and system plus algorithm permutation. As shown in the following equation, the RPD estimate may be (33):

$$RPD(\%) = \frac{T^*_{cost} - T_{cost}}{T^*_{cost}} \times 100\% \tag{33}$$

T_{cost} demonstrates all applications' objective cost function during offloading and scheduling in the study.

5.4 Task sequencing and co-operative scheduling methods

The task sequencing rules suggested (i.e., EDD, SPF, and SSTF) are the components calibrated for scheduling task organization. The mean plot of proposed task sequence rules with 95.0% Tukey HSD spaces as shown in Fig. 16b, the RPD importance of the EDD is significantly lower than the SPF and SSTF as shown in Fig. 16b. The task is mapped or scheduled by the EDD rule's task sequence for the lower delay heterogeneous computing. In the VFCN for scheduling, we have therefore selected EDD for the task sequencing portion. We leverage the EDD sequence method in our paper to measure the priority of the assignment. The highest priority is to set, among other items, certain tasks that have the lowest deadline. Existing task scheduling methods such as Round Robing (Baseline 1) [39], Min-Max (Baseline 2) [40] and HEFT (Baseline 3) [41] have been introduced and compared during the experiment with the proposed CTOS scheduling scheme. Figures 17 and 18 suggest that, relative to the baseline methods, average device costs and average task latency are likely to be reduced by using CTOS.

Cost-PH value in Table 3 shows the function utilization cost per hour of resources in the VFCN for vehicle applications.

Figure 17 shows that, the conflicting objective achieved the applications quality of service requirements based on their given deadline during offloading and scheduling in the network. The proposed CTOS dynamic offloading and scheduling improved the mobility and dynamic results in the adaptive environment.

6 Conclusion

This paper studies mobility aware task offloading and scheduling problems in the vehicular networks for IoT applications. We devise a mobility-aware task offloading phase (MTOP) scheme, which determines optimal time-selection, where/when to offload tasks for execution to minimize the system's total costs. Cooperative Task Offloading Scheduling (CTOS) scheme is proposed, prioritizing all tasks into some sequence. Then schedules them according to their requirements. Simulation results show that the proposed schemes outperform existing baseline approaches in total system costs and task latency in vehicular networks. All studies in related work showed their efforts in mobility aware offloading and scheduling. However, in the result discussion part, the proposed showed the strength in terms of system cost and lateness of applications.

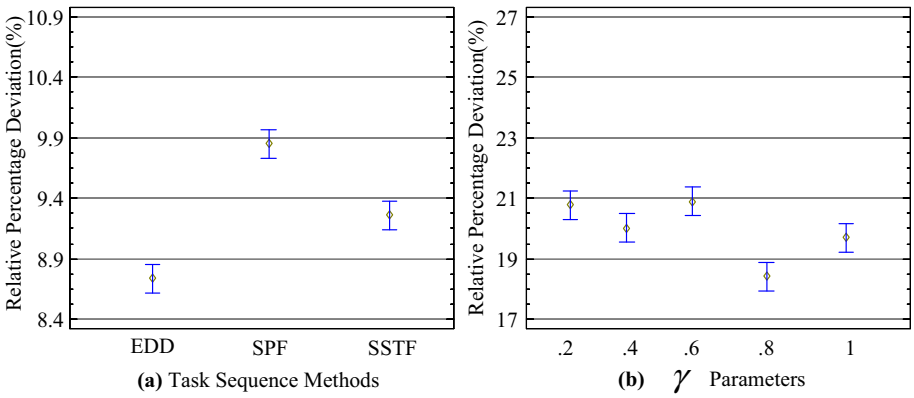

Fig. 16 The parameter and task sequence methods γ 95% confidence interval Tukey HSD mean interval chart

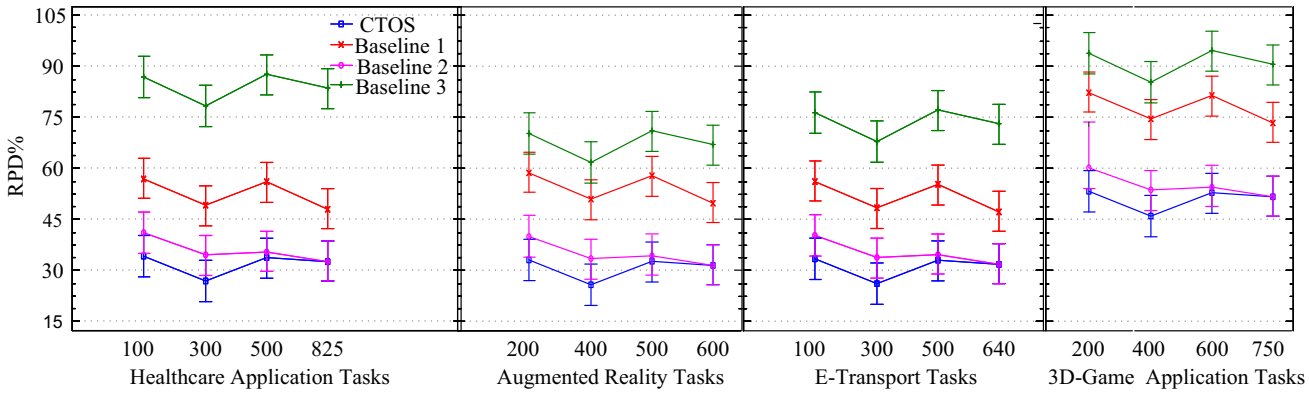

Fig. 17 System costs and latency and 95% confidence interval Tukey HSD mean interval chart

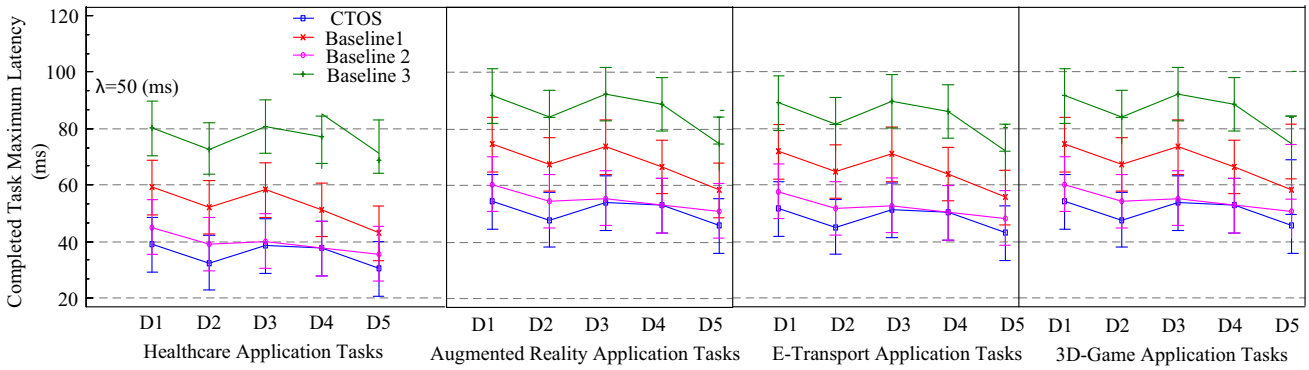

Fig. 18 Strict task completed maximum latency

We will extend our work for different kinds of applications such as the Internet of Medical Things with blockchain-enable fog cloud network in the future work. We will suggest a novel serverless system to reduces the costs of applications in terms of memory and execution. The study will suggest the reinforcement aware offloading and scheduling algorithm handle all dynamic and mobility aware failure in the study.

Author contributions All authors contributed equally to the final dissemination of the research investigation as a full article. All authors have read and agreed to the published version of the manuscript.

Funding This study was not funded.

Data availability Data Availability Statements: IoVT Applications in Container-Based Fog Cloud Network used the for the manuscript mobility real-dataset of the one organization which is available on the following link.: User Movement Simulations Project. Available. [Online]: http://everywarelab.di.unimi.it/lbs-datasim [60]. The rest of the data such as inputs and algorithm is private data and available on local machines which can not be shared publically for now.

Declarations

Conflict of interest There is no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors

References

1. Stergiou, C.L., Psannis, K.E., Gupta, B.B.: Iot-based big data secure management in the fog over a 6g wireless network. In: IEEE Internet of Things Journal (2020)
2. Gupta, B., Quamara, M.: An overview of internet of things (IoT): architectural aspects, challenges, and protocols. Concurr. Comput. **32**(21), e4946 (2020)
3. La, A., Mastoi, Q.-U.-A., Elhoseny, M., Memon, M.S., Mohammed, M.A.: Deep neural network-based application partitioning and scheduling for hospitals and medical enterprises using IoT assisted mobile fog cloud. Enterpr. Inform. Syst, pp. 1–23 (2021)
4. AlZubi, S., Shehab, M., Al-Ayyoub, M., Jararweh, Y., Gupta, B.: Parallel implementation for 3d medical volume fuzzy segmentation. Pattern Recogn. Lett. **130**, 312–318 (2020)
5. Esposito, C., Ficco, M., Gupta, B.B.: Blockchain-based authentication and authorization for smart city applications. Inform. Process. Manage. **58**(2), 102468 (2021)

6. Wang, H., Li, Z., Li, Y., Gupta, B., Choi, C.: Visual saliency guided complex image retrieval. Pattern Recognit. Lett. **130**, 64–72 (2020)
7. Adat, V., Gupta, B.: Security in internet of things: issues, challenges, taxonomy, and architecture. Telecommun. Syst. **67**(3), 423–441 (2018)
8. Podder, A.K., Al-Bukhari, A., Islam, S., Mia, S., Mohammed, M.A., Kumar, N.M., Cengiz, K., Abdulkareem, K.H.: IoT based smart agrotech system for verification of urban farming parameters. Microprocess. Microsyst. **82**, 104025 (2021)
9. Lakhan, A., Li, X.: Transient fault aware application partitioning computational offloading algorithm in microservices based mobile cloudlet networks. Computing **102**(1), 105–139 (2020)
10. Guo, H., Liu, J., Zhang, J., Sun, W., Kato, N.: Mobile-edge computation offloading for ultradense IoT networks. IEEE Internet Things J. **5**(6), 4977–4988 (2018)
11. Dong, P., Zheng, T., Yu, S., Zhang, H., Yan, X.: Enhancing vehicular communication using 5g-enabled smart collaborative networking. IEEE Wireless Commun. **24**(6), 72–79 (2017)
12. Masini, B.M., Bazzi, A., Natalizio, E.: Radio access for future 5g vehicular networks. In: 2017 IEEE 86th Vehicular Technology Conference (VTC-Fall), pp. 1–7 (2017)
13. Liu, J., Wan, J., Zeng, B., Wang, Q., Song, H., Qiu, M.: A scalable and quick-response software defined vehicular network assisted by mobile edge computing. IEEE Commun. Mag. **55**(7), 94–100 (2017)
14. Bhimani, J., Yang, Z., Mi, N., Yang, J., Xu, Q., Awasthi, M., Pandurangan, R., Balakrishnan, V.: Docker container scheduler for i/o intensive applications running on NUME SSDS. IEEE Trans. Multi-Scale Comput. Syst. **4**(3), 313–326 (2018)
15. Lakhan, A., Ahmad, M., Bilal, M., Jolfaei, A., Mehmood, R.M.: Mobility aware blockchain enabled offloading and scheduling in vehicular fog cloud computing. In: IEEE Transactions on Intelligent Transportation Systems (2021)
16. Ahuja, S.P., Wheeler, N.: Architecture of fog-enabled and cloud-enhanced internet of things applications. Int. J. Cloud Appl. Comput. (IJCAC) **10**(1), 1–10 (2020)
17. Bansal, R., Singh, V.K.: Proposed technique for efficient cloud computing model in effective digital training towards sustainable livelihoods for unemployed youths. Int. J. Cloud Appl. Comput. (IJCAC) **10**(4), 13–27 (2020)
18. Guo, H., Zhang, J., Liu, J.: Fiwi-enhanced vehicular edge computing networks: collaborative task offloading. IEEE Veh. Technol. Mag. **14**(1), 45–53 (2019)
19. Bu, S., Yu, F.R., Cai, Y., Liu, X.P.: When the smart grid meets energy-efficient communications: green wireless cellular networks powered by the smart grid. IEEE Trans. Wireless Commun. **11**(8), 3014–3024 (2012)
20. Bulla, C.M., Birje, M.N.: A multi-agent-based data collection and aggregation model for fog-enabled cloud monitoring. Int. J. Cloud Appl. Comput. (IJCAC) **11**(1), 73–92 (2021)
21. Hallappanavar, V.L., Birje, M.N.: A reliable trust computing mechanism in fog computing. Int. J. Cloud Appl. Comput. (IJCAC) **11**(1), 1–20 (2021)
22. Ahammad, I., Khan, M.A.R., Salehin, Z.U., Uddin, M., Soheli, S.J.: Improvement of QOS in an IoT ecosystem by integrating fog computing and SDN. Int. J. Cloud Appl. Comput. (IJCAC) **11**(2), 48–66 (2021)
23. Hossain, K., Rahman, M., Roy, S.: Iot data compression and optimization techniques in cloud storage: current prospects and future directions. Int. J. Cloud Appl. Comput. (IJCAC) **9**(2), 43–59 (2019)
24. Mutlag, A.A., Abd-Ghani, M.K., Arunkumar, N.A., Mohammed, M.A., Mohd, O.: Enabling technologies for fog computing in healthcare IoT systems. Future Generat. Comput. Syste. **90**, 62–78 (2019)
25. Khalaf, B.A., Mostafa, S.A., Mustapha, A., Mohammed, M.A., Abduallah, W.M.: Comprehensive review of artificial intelligence and statistical approaches in distributed denial of service attack and defense methods. IEEE Access **51**, 51691–51713 (2019)
26. Abdulkareem, K.H., Mohammed, M.A., Gunasekaran, S.S., Al-Mhiqani, M.N., Mutlag, A.A., Mostafa, S.A., Ibrahim, N.S., Ali, N.S., Ibrahim, D.A.: A review of fog computing and machine learning: concepts, applications, challenges, and open issues. IEEE Access **7**, 153123–153140 (2019)
27. Lahoura, V., Singh, H., Aggarwal, A., Sharma, B., Mohammed, M.A., Damaševičius, R., Kadry, S., Cengiz, K.: Cloud computing-based framework for breast cancer diagnosis using extreme learning machine. Diagnostics **11**(2), 241 (2021)
28. Abdulkareem, K.H., Mohammed, M.A., Salim, A., Arif, M., Geman, O., Gupta, D., Khanna, A.: Realizing an effective Covid-19 diagnosis system based on machine learning and IoT in smart hospital environment. In: IEEE Internet of Things Journal (2021)
29. Hussain, M., Wei, L.F., Lakhan, A., Wali, S., Ali, S., Hussain, A.: Energy and performance-efficient task scheduling in heterogeneous virtualized cloud computing. Sustain. Comput. **30**, 100517 (2021)
30. Mutlag, A.A., Khanapi Abd-Ghani, M., Mohammed, M.A., Maashi, M.S., Mohd, O., Mostafa, S.A., Abdulkareem, K.H., Marques, G., de la Torre Díez, I.: MAFC: multi-agent fog computing model for healthcare critical tasks management. Sensors **20**(7), 1853 (2020)
31. Memon, M.S., Lakhan, A., Mohammed, M.A., Qabulio, M., Al-Turjman, F., Abdulkareem, K.H.: Machine learning-data mining integrated approach for premature ventricular contraction prediction. Neural Comput. Applicat. **25**, 1–17 (2021)
32. Mahesar, A.R., Lakhan, A., Sajnani, D.K., Jamali, I.A.: Hybrid delay optimization and workload assignment in mobile edge cloud networks. Open Access Library J. **5**(9), 1–12 (2018)
33. Mostafa, S.A., Gunasekaran, S.S., Mustapha, A., Mohammed, M.A ., Abduallah,. W.M.: Modelling an adjustable autonomous multi-agent internet of things system for elderly smart home. In: International Conference on Applied Human Factors and Ergonomics. Springer, pp. 301–311 (2019)
34. Lakhan, A., Li, X.: Mobility and fault aware adaptive task offloading in heterogeneous mobile cloud environments. EAI Endorsed Trans Mobile Commun. Appl. **16**(5), 1–29 (2019)
35. Tomlin, C.J., Lygeros, J., Sastry, S.S.: A game theoretic approach to controller design for hybrid systems. Proc. IEEE **88**(7), 949–970 (2000)
36. Kosta, S., Aucinas, A., Hui, P., Mortier, R., Zhang, X.: Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In: Proceedings IEEE Infocom. IEEE, pp. 945–953 (2012)
37. Chun, B.-G , Ihm, S. Maniatis, P., Naik M., Patti, A. :Clonecloud: elastic execution between mobile device and cloud. In: Proceedings of the sixth conference on Computer systems. ACM, pp. 301–314 (2011)
38. Sun, X., Ansari, N.: Latency aware workload offloading in the cloudlet network. IEEE Commun. Lett. **21**(7), 1481–1484 (2017)
39. Rasmussen, R.V., Trick, M.A.: Round robin scheduling-a survey. Eur. J. Operat. Res. **188**(3), 617–636 (2008)
40. Etminani, K., Naghibzadeh, M.: A min–min max–min selective algorihtm for grid task scheduling. In: 3rd IEEE/IFIP International Conference in Central Asia on Internet. IEEE, pp. 1–7 (2007)
41. Lin , C., Lu, S.: Heft scheduling scientific workflows elastically for cloud computing. In: 2011 IEEE 4th International Conference on Cloud Computing. IEEE, pp. 746–747 (2011)
42. heng, Z., Tang, Y., Wu, H.: Joint task offloading and flexible functional split in 5g radio access network. In: 2019 International Conference on Information Networking (ICOIN)

43. Refaat, T.K., Kantarci, B., Mouftah, H.T.: Virtual machine migration and management for vehicular clouds. Veh. Commun. **4**, 47–56 (2016)
44. Chen, M., Hao, Y., Qiu, M., Song, J., Wu, D., Humar, I.: Mobility-aware caching and computation offloading in 5g ultra-dense cellular networks. Sensors **16**(7), 974 (2016)
45. Boukerche, A., Robson, E.: Vehicular cloud computing: architectures, applications, and mobility. Comput. Netw. **135**, 171–189 (2018)
46. Mustafa, A.M., Abubakr,O.M., Ahmadien, O., Ahmedin, A., Mokhtar, B.: Mobility prediction for efficient resources management in vehicular cloud computing. In *2017 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. IEEE, pp. 53–59 (2017)
47. Ning, Z., Xia, F., Ullah, N., Kong, X., Hu, X.: Vehicular social networks: enabling smart mobility. IEEE Commun. Mag. **55**(5), 16–55 (2017)
48. Yang, C., Liu, Y., Chen, X., Zhong, W., Xie, S.: Efficient mobility-aware task offloading for vehicular edge computing networks. IEEE Access **7**, 26652–26664 (2019)
49. Qiao, G., Leng, S., Zhang, K., He, Y.: Collaborative task offloading in vehicular edge multi-access networks. IEEE Commun. Mag. **56**(8), 48–54 (2018)
50. Jiang, Z., Zhou, S., Guo, X., Niu, Z.: Task replication for deadline-constrained vehicular cloud computing: Optimal policy, performance analysis, and implications on road traffic. IEEE Internet Things J. **5**(1), 93–107 (2017)
51. Adhikary, T., Das, A.K., Razzaque, M.A., Almogren, A., Alrubaian, M., Hassan, M.M.: Quality of service aware reliable task scheduling in vehicular cloud computing. Mobile Netw. Appl. **21**(3), 482–493 (2016)
52. Shojafar, M., Cordeschi, N., Baccarelli, E.: Energy-efficient adaptive resource management for real-time vehicular cloud services. IEEE Trans. Cloud Comput. **7**(1), 196–209 (2016)
53. Nabi, M., Benkoczi, R., Abdelhamid, S., Hassanein, H.S.: Resource assignment in vehicular clouds. In: 2017 IEEE International Conference on Communications (ICC). IEEE, pp. 1–6 (2017)
54. Zhou, Z., Liu, P., Feng, J., Zhang, Y., Mumtaz, S., Rodriguez, J.: Computation resource allocation and task assignment optimization in vehicular fog computing: a contract-matching approach. IEEE Trans. Veh. Technol. **68**(4), 3113–3125 (2019)
55. Rui, L., Zhang, P., Huang, H., Qiu, X.: A location-dependent task assignment mechanism in vehicular crowdsensing. Int. J. Distribut. Sensor Netw. **12**(9), 1550147716669627 (2016)
56. Zhu, C., Pastor, G., Xiao, Y., Li, Y., Ylae-Jaeaeski, A.: Fog following me: Latency and quality balanced task allocation in vehicular fog computing. In: *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, pp. 1–9 (2018)
57. Zhang, K., Mao, Y., Leng, S., Maharjan, S., Zhang, Y.: Optimal delay constrained offloading for vehicular edge computing networks. In: *2017 IEEE International Conference on Communications (ICC)*. IEEE, pp. 1–6 (2017)
58. Baldini, I., Castro, P., Chang, K., Cheng, P., Sink, V., Sakian, P., N. Mitchell, V. Muthusamy, R. Rabbah, A. Slominski, and P. Suter, "Serverless computing: Current trends and open problems," arXiv preprint arXiv:1706.03178, 2017. [Online]. Available: https://academic.microsoft.com/paper/2950821735
59. Król, M., Psaras, I.: Nfaas: named function as a service. In: Proceedings of the 4th ACM Conference on Information-Centric Networking, pp. 134–144. (2017) [Online]. Available: https://academic.microsoft.com/paper/2755939422
60. Ma, D., Huang, J.: The pricing model of cloud computing services. In: Proceedings of the 14th Annual International Conference on Electronic Commerce. ACM, pp. 263–269 (2012)
61. García, L.L., Arellano, A.G., Cruz-Santos, W.: A parallel path-following phase unwrapping algorithm based on a top-down breadth-first search approach. Optic. Lasers Eng. **124**, 105–827 (2020)
62. Quwaider, M., Shatnawi, Y.: Neural network model as internet of things congestion control using pid controller and immune-hill-climbing algorithm. Simulat. Modell. Pract. Theory **101**, 102022 (2020)
63. Araya, I., Moyano, M., Sanchez, C.: A beam search algorithm for the biobjective container loading problem. Eur. J. Operat. Res. **286**, 417–431 (2020)

Abdullah Lakhan has finished his Ph.D. in Computer Science and Technology from Southeast University China. Currently, he works at the College of Computer Science and Artificial Intelligence, Wenzhou University, Wenzhou 325035, China. He has published more than 30 high-quality conferences and journal papers in INFOCOM, ICCBB, ICCC and ICWS, IEEE Transactions, Sensors, Computing, Electronics, and many journals. Currently, he is the author of IEEE Transaction on Intelligent Transport System, IEEE ACCESS, Sensors, Computing, IEEE Transactions on Industrial Informatics, and many. His research area directions are the following: Computation and Data Offloading, Application Partitioning in RPC system, Scheduling in Fog-Cloud Computing (E-Transport, IMoT, and IIoT applications), Federated Learning in the ubiquitous environment, Cyber Security, Blockchain, Anomaly detection, and Information Security.

Muhammad Suleman Memon works as Assistant Professor in Department of Information Technology Dadu Campus, University of Sindh, Jamshoro, Sindh, Pakistan. He is currently doing his Ph.D. in the field of Artificial Intelligence from Quaid-e-Awam University of Engineering Science & Technology, Nawabshah, Sindh, Pakistan.

Qurat-ul-ain Mastoi is currently pursuing the Ph.D. degree with the Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia. Her research project is classification of premature ventricular contraction cardiac arrhythmia for different cardiovascular disease.

Mohamed Elhoseny is an Assistant Professor at the Department of Computer Science, American University in the Emirates, Dubai, UAE. Dr. Elhoseny is an ACM Distinguished Speaker and IEEE Senior Member. He received his PhD in Computers and Information from Mansoura University/ University of North Texas through a joint scientific program. Collectively, Dr. Elhoseny authored/co-authored over 150 published articles in prestigious journals, book chapters, and conference papers. Besides, Dr. Elhoseny authored/edited 17 international books published by recognized publishers such as Springer and Elsevier. His research interests include Smart Cities, Network Security, Artificial Intelligence, Internet of Things, and Intelligent Systems. Dr. Elhoseny is the founder and the Editor-in- Chief of IJSSTA journal published by IGI Global. Also, he is an Associate Editor at IEEE Journal of Biomedical and Health Informatics, IEEE Access, Scientific Reports, IEEE Future Directions, Remote Sensing, and International Journal of E-services and Mobile Applications. Moreover, he served as the cochair, the publication chair, the program chair, and a track chair for several international conferences published by recognized publishers such as IEEE and Springer. Dr. Elhoseny is the Editor-in-Chief of the Studies in Distributed Intelligence Springer Book Series, the Editor-in-Chief of The Sensors Communication for Urban Intelligence CRC Press-Taylor & Francis Book Series, and the Editor-in-Chief of The Distributed Sensing and Intelligent Systems CRC Press-Taylor& Francis Book Series. He was granted several awards by diverse funding bodies such as the Egypt State Encouragement Award in 2018, the Young Researcher Award in Artificial Intelligence from the Federation of Arab Scientific Research Councils in 2019, Obada International Prize for young distinguished scientists 2020, Mansoura University Young Researcher Award 2019, the SRGE best young researcher award in 2017, and the best Ph.D. thesis in Mansoura University in 2015. Dr. Elhoseny has 14 years of teaching experience and supervised two PhD students, two master students, and six graduation projects. Besides, he is a TPC Member or Reviewer in 100+ International Conferences and Workshops. Furthermore, he has been reviewing papers for 100+ International Journals including IEEE Communications Magazine, IEEE Transactions on Intelligent Transportation Systems, IEEE Internet of Things, Future Generation Computer Systems, IEEE Communication Letters, Elsevier Computer Communications, Computer Networks, Sustainable Cities and Society, Wireless Personal Communications, and Expert Systems with Applications. Dr. Elhoseny has been invited as a guest in many media programs to comment on technologies and related issues.

Mazin Abed Mohammed obtained his BSc. in Computer Science from the University of Anbar, Iraq. He obtained his MSc. in Information Technology from the College of Graduate Studies, Universiti Tenaga Nasional (UNITEN), Malaysia. He obtained his PhD. in Information and Communication Technology from the Faculty of Information & Communication Technology, Universiti Teknikal Malaysia Melaka, Melaka, Malaysia. He has produced more than 60 articles in journals, book chapters, conferences, and tutorials such as Future Generation Computer Systems (Elsevier), Neural Computing and Applications(Springer), IEEE Access, Journal of computational science (Elsevier), Computers & Electrical Engineering (Elsevier), The Journal of Supercomputing(Springer), Journal of medical systems(Springer), International journal of medical informatics (Elsevier), Soft Computing(Springer) , Concurrency and Computation: Practice and Experience (Wiley), and Sensors (MDPI). Also, he is Editor in Chief of Journal of Fusion: Practice and Applications, Associate Editor of International Journal of Smart Sensor Technologies and Applications IGI Global: International Publisher of Information Science and Technology Research. He is a reviewer of more than 40 reputed Journals including SCI-Indexed Journals of IEEE, ACM, Elsevier, Springer, and Wiley. Finally, His specialization and research interest include the areas of Artificial Intelligence, Biomedical Computing and Processing, Medical Image and Data Processing, Machine Learning, Deep Learning, Optimization Methods and Software Medical IoT.

Mumtaz Qabulio works as Assistant Professor in Department of Information Software Engineering, Faculty of Engineering, University of Sindh, Jamshoro, Sindh, Pakistan. She has earned her Ph.D. in the field of Wireless Sensor Networks from Institute of Mathematics & Computer Science, University of Sindh, Jamshoro, Sindh, Pakistan.

Mohamed Abdel-Basset (Senior Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in operations research from the Faculty of Computers and Informatics, Zagazig University, Egypt. He is currently an Associate Professor with the Faculty of Computers and Informatics, Zagazig University. He has published more than 200 papers in international journals and conference proceedings. His current research interests are optimization, operations research, data mining, computational intelligence, applied statistics, decision support systems, robust optimization, engineering optimization, multi-objective optimization, swarm intelligence, evolutionary algorithms, and artificial neural networks. He is working on the application of multiobjective and robust metaheuristic optimization techniques. He is also an/a editor/reviewer in different international journals and conferences.