System Completion Time Minimization with Edge Server Onboard Unmanned Vehicle

Wen Peng[1], Hongyue Wu[1], Shizhan Chen[1(✉)], Lei Dong[1], Zhuofeng Zhao[2], and Zhiyong Feng[1]

[1] College of Intelligence and Computing, Tianjin University, Tianjin, China
{peng_wen,hongyue.wu,shizhan,2118218002,zyfeng}@tju.edu.cn
[2] Beijing Key Laboratory On Integration and Analysis of Large-Scale Stream Data, North China University of Technology, Beijing, China
edzhao@ncut.edu.cn

Abstract. With the advantages of flexibility and powerful computing resources, edge servers mounted on unmanned vehicles (V-edge) have attracted significant interest in mobile edge computing (MEC). In this paper, we design an offloading scheme for vehicle-mounted edge rescue systems with the consideration of road limitations. In these systems, edge servers can receive data and process them while on the move. The objective is to minimize the completion time of tasks within the system under both time-varying communication and computation resource constraints. The formulated problem is decomposed into two subproblems, i.e., task completion time minimization within communities and V-edge travel time minimization between communities. For task completion time minimization, we propose an SQP-based iterative algorithm, which can generate feasible stopping points by using quadratic programming. V-edge travel time minimization problem can be converted to a TSP problem and thus can be solved by some existing methods. Finally, experiments are conducted to verify the usability of the proposed approach.

Keywords: Edge computing · Computation offloading · Mobility server · Unmanned vehicle

1 Introduction

In recent years, natural disasters such as hurricanes and wildfires are part of our daily life which have significant devastating effects, including but not limited to network failures [2]. Hence, numerous emergency management officials will undertake rescue operations, with rescuers carrying rescue devices for exploration, identification, and detection, which will generate intensive computation.

However, due to the limitations of computing power, storage capacity and battery capacity, the intelligent devices cannot meet the needs of these various computationally intensive applications. Therefore, how to solve these computationally intensive tasks quickly is a huge challenge. Cloud computing is

H. Gao et al. (Eds.): CollaborateCom 2022, LNICST 460, pp. 269–289, 2022.
https://doi.org/10.1007/978-3-031-24383-7_15

a worthy solution to consider. However, service requests cannot be responded promptly due to the long transmission latency. To address this challenge, a novel paradigm, Edge Computing (EC), has been proposed. Compute-intensive tasks can be offloaded to edge servers with shorter latency compared to offloading to the cloud, which can also free up the device's resources earlier.

Traditional EC-based base stations (BSs) are powerful enough to handle many complex service requests in most cases with low latency and high QoS requirements of users. However, edge servers are prone to failures due to adverse weathers, earthquake disasters, etc. [11], which will make it impossible to respond to users' requests until they are repaired. Especially, the rescue equipment cannot rely on infrastructure of BSs for communication in the case of server failure after disaster. There are many literature has raised the issue of failure recovery and user migration, such as [4,11]. But these proposed approaches either require waiting for failure recovery or cannot migrate all service demands. On the other hand, a more flexible MEC mechanism is quite necessary in rural environments and military exercises, due to high deployment costs resulting in limited BSs and APs infrastructure.

Thus, an mobile on-demand offload services are needed to ease their burden. UAVs and unmanned vehicles are integrated with EC to provide computing power and storage resource capacity. Therefore, EC can be classified into two types according to the state of edge servers: static edge servers connected to BSs and mobile edge servers mounted on vehicles or unmanned aerial vehicles (UAVs) [10].

Compared to static edge servers, mobile edge servers are more flexible and can be applied to a richer set of scenarios. In many studies of mobile edge servers, UAVs have shown their capabilities [5,8,14], but their battery endurance, computational power, and storage resources are worse compared to mobile edge servers mounted on vehicles (V-edge [10]). There is no doubt that V-edge is a better choice with a large volume of data and few road obstacles.

Accordingly, we designed a vehicle-mounted edge server rescue system to provide computational offload services. Considering the importance of each task, we need to complete all tasks in the shortest possible time. However, V-edge travels around each device to provide the service is relatively wasteful of resources. To this end, we divide the requesting devices in close proximity together according to their geographical location to form a requesting community, so that V-edge can handle the tasks of the devices in that community at the same time. Moreover, when the data size of task is large, V-edge cannot finish processing the tasks while it is moving, so that it needs to stop at a point to continue processing. However, differences in stopping positions can make a gap in the distance between the V-edge and devices, affecting the transmission time and subsequently the time to complete tasks. In order to complete all tasks as quickly as possible, we need to choose the best point to stop for V-edge. Then, complete all community tasks one by one. The main contributions of the paper are summarized as follows:

- We propose a vehicle-mounted edge server rescue system for emergency rescue, with the goal of minimizing the system completion time.
- We jointly optimize the communication and computation of the mobile and divide the original minimization problem into community task completion time minimization subproblem and V-edge travel time minimization problem between communities. For the first subproblem, quadratic programming and positive definite Hessian approximation techniques are used to formulate and solve the feasible solution by iteration. Therefore, an iterative approach based on the sequential quadratic programming (SQP) algorithm is proposed to address the complexity of joint optimization.
- The effectiveness of the algorithm is evaluated with extensive simulation results. Numerical results show that the algorithm is effective in ensuring that the system completion time is minimized.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 demonstrates the system model. Section 4 formulates the optimization problem into a TSP problem. Section 5 shows the methods to solve the optimization problem. The simulation results are described in Sect. 6, followed immediately by the conclusion in Sect. 7.

2 Related Work

In recent years, computation offloading has become an integral part of service computing and has been a popular research topic.

We hereby review the studies of computation offloading from two aspects in the following: (1) offloading to a server at a fixed location and (2) offloading to mobile devices with some computing power or edge servers.

2.1 Static Edge Computation Offloading

Many studies have investigated the offloading strategies of EC systems in static scenarios. In general, these works are made for various optimization targets [17,18,21], such as latency minimization, energy consumption minimization, and resource allocation. For instance, In [18], the authors investigated the collaboration between cloud and edge server to minimize request's latency. In [22], the authors jointly considered computational offloading and resource management to minimize network-wide weighted energy consumption. Considering the mobility of users, zhao et al. [21] proposed a mobility-aware cross-edge computation offloading framework to reduce the latency.

The above works can meet the tasks in general scenarios, but in rural areas or areas outside the range of servers, these fixed-location servers are out of reach and cannot provide service support, making it difficult for users' requests to be fulfilled. In particular, during the disaster scenarios, servers are prone to failure which will prevent it from processing user requests.

2.2 Mobile Edge Computation Offloading

Existing work has proposed that users can offload their tasks to neighboring devices or to a mobile edge server mounted on unmanned aerial vehicles (UAVs) or vehicles. For instance, In [3], chen et al. proposed a task offloading scheme merely relying on vehicle-to-vehicle (V2V) communication which can fully exploring the idle resources of vehicles and complete tasks quickly. In [14], ning et al. designed a 5G-enabled UAV-to-community offloading system and jointly considered trajectory designing and task scheduling, with the objective of maximizing the system throughput. Liu et al. [10] designed a novel vehicle-mounted edge mechanism, and jointly optimize path planning and resource allocation to maximize the returned data with deadline constrains.

In the literature, we can find that servers mounted on UAVs or unmanned vehicles can be applicable to many scenarios and make up for the shortcomings of traditional base stations. Therefore, we optimize the system in terms of the completion time of the minimized system.

3 System Model

3.1 System Model

Fig. 1 shows a real emergency rescue scenario, where many rescuers carry intelligent devices (SD in Fig. 1) for detection, identification, collection, and classification which will generate a large number of computational tasks. Each device can communicate with the server individually. Suppose that there are K devices performing rescue tasks, indexed by $\mathcal{K}$=$\{1, 2, ..., K\}$. Let $p_i = (x_i, y_i)$ denote the location of device i. Affected by the disaster, large communication facilities such as base stations may experience power outages or failures. Therefore, devices cannot offload computation-intensive tasks through edge computing base stations, etc., and need V-edge to provide various services. Thus, a V-edge assisted offloading system is designed. By optimizing the location in community and path of V-edge, computational support can be provided to the devices in the shortest possible time. For simplicity, a quasi-static network scenario [10] is considered in which the locations of devices do not change during the system time period T. We assume that the devices always perform their tasks in system time T, i.e., the number of devices is constant. Therefore, we investigate the computational offloading problem between V-edge and smart devices with the aim of minimizing the time to complete all tasks(system completion time), including the time to complete tasks within the community and the time for V-edge to reach the community.

Additionally, we assume that V-edge is full-duplex communication, which means that task uploading and result downloading can be executed simultaneously and V-edge can interact with multiple devices at the same time by using processor sharing [19]. For convenience, we assume that V-edge needs to receive all the task data before it can process the task.

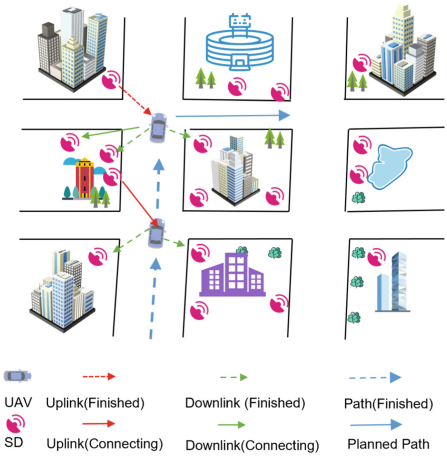


Fig. 1. Simulation result for the network.

3.2 Computing Task Offloading Model

With the objective of minimizing the system completion time, we designed a V-edge-based task offloading model. The task generated by the device i can be given by a 4-tuple $A_i(I_i, f_i, O_i, p_i)$, where I_i is the input data size (in bit), f_i denotes the CPU frequency of device (in CPU cycles per bit), O_i denotes the data size of computing result (in bits), which is depended on tasks and is smaller than I_i, p_i is the location of device i. At the beginning, V-edge can only receive demands information instead of demands data from rescue devices relying on a bootstrapping program [15]. Only within the communication range can V-edge start computation task offloading.

In V-edge, we assume that the computational resources are sufficient and by using processor sharing, V-edge can provide parallel computation for multiple computational tasks. Its maximum CPU frequency is designed as F (in CPU cycles per second).

Here are the steps for V-edge offloading process. (i) V-edge received corresponding profile of tasks A_i according to the bootstrapping program. (ii) devices upload their tasks to V-edge within the community via the uplink. (iii) V-edge processes the tasks. (iv) devices obtain the results via the downlink. (v) V-edge travels to the next community.

3.3 Communication Model

To offload the computation tasks for V-edge execution, corresponding input bits of the task need to be delivered to V-edge via uplink. Similarly, computation results need to be delivered by V-edge via downlink. We assume the wireless channels between devices and V-edge are i.i.d line-of-sight (LOS) channel [1].

Given device i located at $p_i = (x_i, y_i)$ as well as V-edge located at (x_0, y_0), the distance between device i and V-edge can be given by

$$d_{i,0} = [(x_i - x_0)^2 + (y_i - y_0)^2]^{\frac{1}{2}} \tag{1}$$

Therefore, the path loss between device i and V-edge when V-edge is moving at time t ($L_{m,i}(t)$) or when V-edge stays in a certain position ($L_{p,i}$) can be expressed as

$$L_{m,i}(t) = \eta\mu^2((l_i - vt)^2 + h_i^2), \tag{2}$$

$$L_{p,i} = \eta\mu^2((L_i + x - l_i)^2 + h_i^2) \tag{3}$$

where we denote the attenuation factors by η, and the correlation coefficient by μ which is equal to $\frac{4\pi g}{c}$, and the maximum speed of V-edge by v. Let L_i denote the distance between the position where device i can connect to V-edge with the position where all the devices within the community can connect to V-edge, as shown in the red segment in Fig. 2. And let x indicate the location of V-edge where V-edge can connect to all tasks at the same time. Let l_i the communication radius of device i on the road (as shown in Fig. 2), and h_i the vertical distance from device i to the road segment (as shown in Fig. 2).

When V-edge is moving, the transmission rates of uplink ($R^u_{m,i}(t)$) and downlink ($R^d_{m,i}(t)$) between device i and V-edge at time t can be formulated by

$$\begin{cases} R^u_{m,i}(t) = B\log_2(1 + \dfrac{p_{TX}}{L_{m,i}(t)\sigma^2 B}) \\ R^d_{m,i}(t) = B\log_2(1 + \dfrac{p_{RX}}{L_{m,i}(t)\sigma^2 B}) \end{cases} \tag{4}$$

where B represent the uplink and downlink bandwidth between device i and V-edge, and $p_{TX}(W), p_{RX}(W)$ is the transmission power and the receiving power, and σ^2 is the white Gaussian noise variance [7].

When V-edge stays in a certain position, the transmission rates of uplink ($R^u_{p,i}$) and downlink ($R^d_{p,i}$) between device i and V-edge can be formulated by

$$\begin{cases} R^u_{p,i} = B\log_2(1 + \dfrac{p_{TX}}{L_{p,i}\sigma^2 B}) \\ R^d_{p,i} = B\log_2(1 + \dfrac{p_{RX}}{L_{p,i}\sigma^2 B}) \end{cases} \tag{5}$$

3.4 Computational Model

Within the same community, the transmission time and computation time of each task are different due to the distinctions in data volume and their geographical locations. In order to complete all tasks better and faster, we optimally allocate the computational resources of V-edge. Let T^u, T^p, T^d denote The upload, computation, and download latency respectively. Since V-edge can

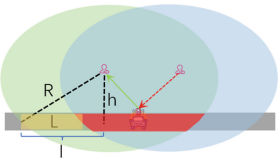


Fig. 2. Example of h,l,h of each device in a community

perform data transmission while moving, the communication distance between devices and V-edge will change over time, so do the communication rate, which affects uplink communication time and downlink communication time. When V-edge is moving, the uplink and downlink communication time $(T_m^u(i), T_m^d(i))$ can be calculated by

$$I_{m,i} = \int_0^{T_{m,i}^u} R_{m,i}^u(t)dt, \quad \forall i \in K \tag{6}$$

$$O_{m,i} = \int_0^{T_{m,i}^d} R_{m,i}^d(t)dt, \quad \forall i \in K \tag{7}$$

When V-edge interacts with devices at a point, the uplink and downlink communication time $(T_{p,i}^u, T_{p,i}^d)$ can be calculated by

$$T_{p,i}^u = \frac{I_i^r}{R_{p,i}^u}, \quad \forall i \in K \tag{8}$$

$$T_{m,i}^d = \frac{O_i^r}{R_{p,i}^d}, \quad \forall i \in K \tag{9}$$

where I_i^r and O_i^r are the residual input data or output data.

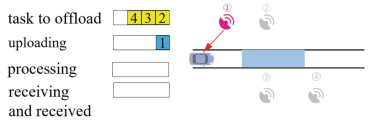
For task of device i, the processing time can be given by:

$$T_i^p = \frac{f_i I_i}{F_i}, \quad \forall i \in K \tag{10}$$

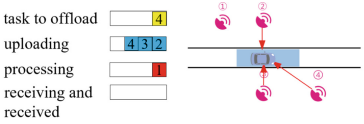
where F_i is the computational resources allocated to the task by V-edge.

3.5 Mobile Model

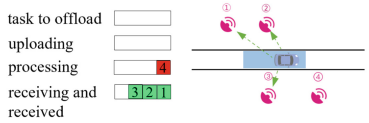
We assume that V-edge is driving on a straight road cleared in advance, regardless of turning. V-edge can only drive on roads. There is no doubt that the optimal location for V-edge is within the road to perform its task within the community. Within each community, the V-edge moves to the optimal position at the maximum speed v until the V-edge drives out of the communication range of the device just to return the result of the task. After driving away from the community, the V-edge moves at the maximum speed, expecting to reach the next community faster.



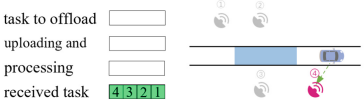
(a) First time period, task 1 is uploading and task 2,3,4 are waiting to be performed.



(b) Second time period, task 1 is processing and task 2,3,4 are uploading



(c) Third time period, task 1,2,3 are executed and downloading, task 4 is processing.



(d) Fourth time period, task 4 is downloading to the fourth device.

Fig. 3. An example of computing task offloading. V-edge finishes four tasks from four devices during different time period.

4 Problem Formulation

In this section, we analyzed the V-edge assisted offloading and formulated the problem of minimizing the system completion time.

4.1 Problem Overview

The V-edge assisted offloading process is given by an example illustrated in Fig. 3 where four subfigures display the offloading process in one community. In the Fig. 3(a), task 1 is uploading to the V-edge for execution because V-edge reaches its communication range, and the other devices wait for V-edge to enter their communication range. In the Fig. 3(b), V-edge moves to the optimal stopping position in the common communication segment (the red area in Fig. 2), and V-edge is processing the task 1 and receiving the data from task 2, task 3 and task 4 via uplink respectively. In the third subfigure, Fig. 3(c), V-edge just finished transmitting result to device 1, and is transmitting result to device 2 and device 3 via downlink respectively. At the same time, V-edge is processing the task 4. In the Fig. 3(d), task 4 is accomplished and the result is transmitted to device 4 via downlink.

With the objective of minimizing system completion time on the premise of completing all tasks, we need to optimize community completion time and travelling time of V-edge. To minimizing the community completion time, we need to divide the devices into different communities, then find the common communication road segment, and find an optimal stopping position for V-edge, which makes V-edge complete all tasks in the shortest time. Finally, the shortest path for V-edge to travel between communities is optimized.

4.2 Constraint Analysis

V-edge receives input data from the device via uplink and returns the result via downlink respectively. We assume that the V-edge moves into the community to perform the task, which may contain two parts, one that is performed on the move and one that is performed when it stops. Meanwhile, the input data may be transferred in two parts and the output results are the same. Thus, we can obtain the following equation constraints:

$$I_i = \int_0^{T^u_{m,i}} R^u_{m,i}(t)dt + R^u_{p,i}T^u_{p,i}, \forall i \in K \tag{11}$$

$$O_i = \int_0^{T^d_{m,i}} R^d_{m,i}(t)dt + R^d_{p,i}T^d_{p,i}, \forall i \in K \tag{12}$$

where both the uplink and downlink transmission rates vary with time because of changes of communication distance.

4.3 Problem Formulation

With the objective of minimizing the system completion time, intuitively, the time to complete tasks of each community and the travel time to access all communities need to be minimized. In one community, we optimized the community task completion time by optimizing the stopping points of the V-edge. Specifically, in order to ensure that each task is executed, V-edge needs to wait until all tasks are executed away from the community communication range.

To formulate the objective function, we designed some variables:

Let the binary variable α_i indicate whether the device i can complete result downloading during time τ which indicates the time that V-edge is away from the communication range of device i from the docking point, i.e.,

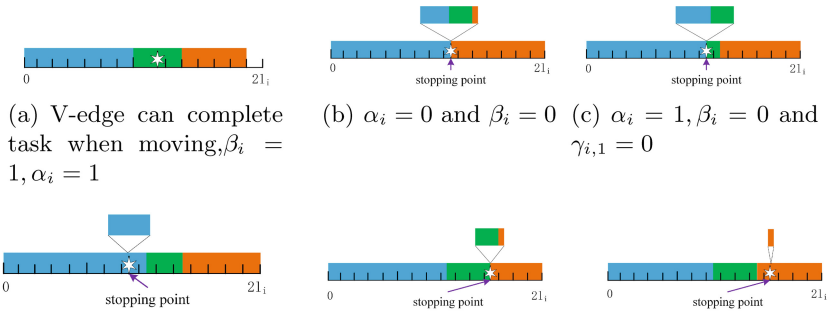
$$\alpha_i = \begin{cases} 1, \text{device } i \text{ can download result during } \tau \\ 0, \text{otherwise.} \end{cases} \tag{13}$$

Let the binary variable β_i represent whether V-edge can receiving the uploading data of device i before stopping if needed, i.e.,

$$\beta_i = \begin{cases} 1, \text{V-edge can receive the uploading data before} \\ \quad \text{stopping,} \\ 0, \text{otherwise.} \end{cases} \tag{14}$$

Let $\gamma_{i,1}$ denote whether the V-edge has completed task processing and device i has finished downloading the result during time τ, i.e.,

$$\gamma_{i,1} = \begin{cases} 1, \text{V-edge can complete processing and device } i \\ \quad \text{can receive the downloading data during } \tau, \\ 0, \text{otherwise.} \end{cases} \tag{15}$$



(a) V-edge can complete task when moving,$\beta_i = 1, \alpha_i = 1$

(b) $\alpha_i = 0$ and $\beta_i = 0$

(c) $\alpha_i = 1, \beta_i = 0$ and $\gamma_{i,1} = 0$

(d) $\alpha_i = 1, \beta_i = 0$ and $\gamma_{i,1} = 1$

(e) $\alpha_i = 0, \beta_i = 1$ and $\gamma_{i,2} = 0$

(f) $\alpha_i = 0, \beta_i = 1$ and $\gamma_{i,2} = 1$

Fig. 4. Illustration of the uploading (blue), computing (green), downloading (orange) process in difference cases that $\alpha_i, \beta_i, \gamma_{i,1}, \gamma_{i,2}$ take different values. (Color figure online)

Let the binary variable $\gamma_{i,2}$ denote whether device i can complete the task uploading and V-edge can complete task processing when the V-edge moves to the stopping point, i.e.,

$$\gamma_{i,2} = \begin{cases} 1, \text{device } i \text{ can complete task uploading and V-edge} \\ \quad \text{can complete task processing before stopping,} \\ 0, \text{otherwise.} \end{cases} \tag{16}$$

Due to the difference in the size of tasks and and geographical differences, some tasks can be completed when the V-edge is moving (as shown in Fig. 4(a)), while others require the V-edge to stop at the stopping point to process the task. Figure 4(b) shows that device i cannot upload all the data to V-edge while V-edge is heading to the stopping point, and device i cannot download all the result of task from V-edge during time period τ, which means that V-edge needs to receive the uploading data, calculate the result and deliver part of the result to the device at the stopping point. Figure 4(c) shows that device i cannot uploading all the data to V-edge while V-edge is moving to the stopping point, and device i can download all the result of task during time period τ. Figure 4(d) shows that device i cannot uploading all the data to V-edge while V-edge is moving to the stopping point, and V-edge can calculate the task and device i can download all the result during τ. Figure 4(e) shows that device i can uploading all the data to V-edge while V-edge is moving to the stopping point, V-edge need to stop at the stopping point to calculate residual data and deliver part of result to device i. Figure 4(f) indicates that V-edge can complete the reception and calculation of the uploaded data while driving to the stopping point but still need to stop at the stopping point to deliver partial results to the device i, but V-edge needs to stop waiting for device i to download part of the results.

Therefore, the upload latency of task i can be classified to four cases, as shown in Fig. 4, which can be represented by:

$$T_i^u = \begin{cases} \text{solved by : } I_i = \int_0^{T_i^u} R_{m,i}^u(t)dt, & \text{if } \beta_i = 1 \\ \frac{L_i+x}{v} + \frac{I_i - \int_0^{\frac{L_i+x}{v}} R_{m,i}^u(t)dt}{R_{p,i}^u}, & \text{if } \beta_i = 0, \alpha_i = 1, \gamma_{i,1} = 0 \\ \frac{2l_i - v(T_i^d + T_i^p)}{v} + \frac{I_i - \int_0^{\frac{2l_i - v(T_i^d+T_i^p)}{v}} R_{m,i}^u(t)dt}{R_{p,i}^u}, & \text{if } \beta_i = 0, \alpha_i = 1, \gamma_{i,1} = 1 \\ \frac{L_i+x}{v} + \frac{I_i - \int_0^{\frac{L_i+x}{v}} R_{m,i}^u(t)dt}{R_{p,i}^u}, & \text{if } \beta_i = 0, \alpha_i = 0 \end{cases} \tag{17}$$

where T_i^d is the download latency of task i and T_i^p is the computation latency of task i.

Meanwhile, the download latency of task i can be classified to four cases, as shown in Fig. 4, which can be represented by:

$$T_i^d = \begin{cases} \text{solved by:} O_i = \int_0^{T_i^d} R_{m,i}^d(t)dt, & \text{if } \alpha_i = 1 \\ \frac{2l_i - L_i - x}{v} + \frac{O_i - \int_0^{\frac{2l_i - L_i - x}{v}} R_{m,i}^d(t)dt}{R_{p,i}^d}, & \text{if } \alpha_i = 0, \beta_i = 1, \gamma_{i,2} = 0 \\ \frac{2l_i - v(T_i^u + T_i^p)}{v} + \frac{O_i - \int_0^{\frac{2l_i - v(T_i^u+T_i^p)}{v}} R_{m,i}^d(t)dt}{R_{p,i}^d}, & \text{if } \alpha_i = 0, \beta_i = 1, \gamma_{i,2} = 1 \\ \frac{2l_i - L_i - x}{v} + \frac{O_i - \int_0^{\frac{2l_i - L_i - x}{v}} R_{m,i}^d(t)dt}{R_{p,i}^d}, & \text{if } \alpha_i = 0, \beta_i = 0 \end{cases} \tag{18}$$

Let binary variable ϕ_i represent whether the V-edge can complete the device task of device i while moving, i.e.,

$$\phi_i = \begin{cases} 1, T_i^u + T_i^p + T_i^d \geq 2l_i/v, \\ 0, \text{otherwise.} \end{cases} \tag{19}$$

Thus, the stopping time for V-edge to complete the task of device i can be given by

$$Q_i(x) = \phi_i(T_i^u + T_i^p + T_i^d - 2l_i/v) \tag{20}$$

Therefore, the stopping time minimization for V-edge to perform tasks in the j^{th} community is formulated:

$$\begin{aligned} S(x) = \max_i \quad & Q_i(x), \quad i = 1, 2, ..., n_j \\ \text{s.t} \quad & x \geq 0 \\ & x \leq x_{max} \\ & \text{Constraints (11), (12)} \end{aligned} \tag{21}$$

where x_{max} is the upper limit of x in the community and n_j is demands number in community j.

Minimizing the stopping time of the V-edge within community j can be formulated as **P1**

$$\begin{aligned} \mathbf{P1}: \zeta_j = \quad & \min_{x} \max_{i} \; Q_i(x), i = 1, 2, ..., n_j \\ \text{s.t} \quad & x \geq 0 \\ & x \leq x_{max} \\ & \text{Constraints (11), (12)} \end{aligned} \tag{22}$$

In the community, the community completion time consists of two parts, the V-edge stopping time and the V-edge travel time in the community. The community task completion time minimization in community j can be formulated as **P2**

$$\begin{aligned} \mathbf{P2}: T = \quad & \zeta_j + \Gamma_j / v \\ \text{s.t} \quad & x \geq 0 \\ & x \leq x_{max} \\ & \text{Constraints (11), (12)} \end{aligned} \tag{23}$$

where Γ_j is the length of communication road segment in the community.

After obtaining the task completion times for all communities, the problem is transformed into a TSP problem which can be solved by many measures, such as [12]. And We design a graph G = (S, E), where S is the community set, and E is a set of undirected edges, which means the travelling time from communities to communities.

We associate with each edge a non-negative cost T_{ij} to denote the time cost between community i and community j. Similarly, each vertex (community) is associated with a non-negative cost T_i to denote the time cost within the community i. Let binary variable e_{ij} represent whether the V-edge is travelled T_{ij}, where $e_{ij} = 1$ indicates V-edge travelled from Community i to Community j; otherwise $e_{ij} = 0$. Similarly, let binary variable b_i represent whether the V-edge has completed the tasks in community i, where $b_i = 1$ indicates V-edge has completed the tasks, otherwise $b_i = 0$. Therefore, the problem can be formulated as follows based on the analyzed constraints:

$$\mathbf{P}: \min \sum_{i,j \in S} e_{ij} T_{ij} + b_i * T_i$$

Due to the maneuverability of the V-edge, the major challenge in solving the problem P is the nonlinear transmission under Eqs. (6), (7) as well as constraints (11), (12). At the same time, the Minimax problem is a typical non-fine optimization problem, i.e., a non-smooth optimization problem. It is a challenging to obtain the optimal solution by the algorithms with polynomial time complexity.

5 Proposed Optimization Method

In this section, we propose a Community Partition and an algorithm for jointly optimizing communication, computation and mobility to solve V-edge Location

Optimization subproblem, which reduces the optimization objective to a simple TSP problem (**P**) which can be solved by ant colony optimization algorithm [12].

5.1 Community Partition

Many existing works have investigated clustering methods, such as semi-dynamic user-specific clustering [9]. Most of the approaches are based on a random common center point where users within the community are close enough to the center point. However, due to road constraints, the V-edge can only be driven on roads, so that the communication center must be in the communication road segment. We model the community division problem as a coverage problem.

Algorithm 1. Community Partition Algorithm

Input: devices set K, crossroads set P

1: Initialization:$Q \leftarrow \emptyset, P \leftarrow \{p_1, ..., p_n\}$, $p_i = [(x_a, y_a), (x_b, y_b)]$, each device communication path Λ
2: divide all devices to the proper road $\{\Psi\}$, mark $\|\Psi\| = n$, and build road device sets $\mathbb{K}$
3: **for** each $p_i \in \Psi$ **do**
4: **for** $k_{p_i} \subset \mathbb{K}$ **do**
5: Calculate boundary device set $k_b \subseteq k_{p_i}$
6: Set new sets $q_j \leftarrow k_b$ and communication path Π_{q_j}, and $\|q\| \leq \|k_b\|$
7: **for** each $k \in k_{p_i} \backslash k_b$ **do**
8: **if** device k communication path $\Pi_{q_j} \cap \Lambda_k \neq \emptyset$ **then**
9: $q_j \leftarrow k$
10: $\Pi_{q_j} \leftarrow \Pi_{q_j} \cap \Lambda_k$
11: **else**
12: set new set q $\leftarrow$ k, and $\Pi_q \leftarrow \Lambda_k$
13: **end if**
14: **end for**
15: $Q \leftarrow Q \cup q$
16: **end for**
17: **end for**
18: **return** Q, Π

As shown in Algorithm 1, we first divide the devices into different roads according to the road constraint, Specifically, for isolated devices, i.e., there are no other devices within the communication range, they are directly classified as the one with the closest road. For other devices, they are jointly divided according to the number of neighboring devices and the road distance (Line 2). Then calculate the boundary devices for each divided road and give them a higher priority to create communities, mainly to prevent omissions (Line 5–6). Obviously, the number of communities is less than the number of devices (Line 6). For other devices, they are divided according to the distance from the existing

community and the length of the effective communication road segment (Line8-10), and the devices that exceed the communication range are reclassified into a new community (Line 12). Finally, merge into communities collection (Line 15).

5.2 V-edge Location Optimization Method

In this subsection, we consider the global optimization of sub problem **P1**. In order to minimize the task completion time within the community, we need to determine an optimal stopping location for V-edge to process all task requests within the community simultaneously.

Since function *S(x)* (21) is generally non-integrable, it is necessary to transform the function *S(x)* into a smooth function, so we introduce an auxiliary variable t to transform the min-max problem **P1** (22) into a minimization problem **P3**. For simplicity, we rearranged the inequality constraints associated with the independent variables in **P1**. Specifically, let *g(x)* denote the inequality constraint function.

$$\begin{aligned} \mathbf{P3}: & \min_{x} \quad t \\ s.t \quad & Q_i(x) - t \leq 0, i = 1, 2, ..., n_j \\ & g_1(x) = -x \leq 0 \\ & g_2(x) = x - x_{max} \leq 0 \end{aligned} \tag{24}$$

where n_j is the number of demands in community j.

Based on the theory of successive convex approximations and SQP algorithm, we proposed a novel task completion time minimization strategy that jointly optimizes computation and communication and considers server mobility (jointCCM). Firstly, we present a convex programming subproblem at each iteration point to approximate the original model, and then find the feasible directions to search for locally optimal solutions within the neighborhood of each iteration point.

For **P3**, let x_k denote the feasible point of the k^{th} iteration, obtained from the previous iteration k-1. We can yield the next feasible point by:

$$x_{k+1} = x_k + \theta_k d_k \tag{25}$$

where θ_k is the step length parameter that is determined by an appropriate line search procedure so that a sufficient decrease in a merit function is obtained, and d_k represents the feasible search direction within a certain neighborhood of x_k. Initially, x_1 is an initial feasible point in a deterministic scope, and in order to obtain the optimal solution, we keep finding the search direction and appropriate step size for the next step and iterate. To solve this problem, we will present the algorithm design of jointCCM in detail.

A. Convex Approximation Transformation. The Lagrangian function associated with Problem is described by

$$L(x, \lambda, \rho) = \sum_{i \in I} \lambda_i Q_i(x) + \sum_{j \in J} \rho_j g_j(x). \tag{26}$$

where the variables λ and ρ are the non-negative Lagrange multiplier vector, and $I = \{1, 2, ..., n_j\}$, $J = \{1, 2\}$.

Based on (26) and SQP algorithm, given the current iteration point x_k and the Lagrange multipliers vector λ, ρ, we can develop a convex quadratic programming(QP) subproblem **P4** to obtain a search direction for x_{k+1} as follows:

$$\begin{aligned} \mathbf{P4}: \min_{(d,t)\in R^{n+1}} \quad & t_k + \frac{1}{2} d_k^T W_k d_k \\ s.t \quad & Q_i(x_k) + \nabla Q_i(x_k)^T d_k - S(x_k) \leq t, i \in I, \\ & g_j(x_k) + \nabla g_j(x_k)^T d_k \leq \eta_k t, j \in J \end{aligned} \tag{27}$$

where η_k is the non-negative auxiliary variable, the matrix W_k is a positive definite approximation of the Hessian matrix of the Lagrangian function which will be covered in detail in the next section.

The above problem(**P4**) can be solved by any QP algorithm such as active-set method [6].

In summary, based on the quadratic programming subproblem, i.e., **P4**, we can obtain the iterative search direction for Eq. (25).

B. Positive-Definite Hessian Approximation. At each iteration, subproblem **P4** requires a positive definite Hessian approximation, we adopt Powell's modification of Broyden-Fletcher-Goldfarb-Shanno formula (BFGS) [16] to perform the calculation:

$$W_{k+1} = W_k + \frac{p_k p_k^T}{p_k^T B_k} - \frac{W_k B_k B_k^T W_k^T}{B_k^T W_k B_k} \tag{28}$$

where $B_k = x_{k+1} - x_k$, and p_k is the linear combination of $W_k B_k$ and y_k, as follows:

$$p_k = \epsilon_k y_k + (1 - \epsilon_k) W_k B_k, \epsilon \in [0, 1] \tag{29}$$

where $y_k = \nabla_x L(x_{k+1}, \lambda_k, \rho_k) - \nabla_x L(x_k, \lambda_k, \rho_k)$, and ϵ is designed to keep that the positive definitions of Hessian approximation, and is can be expressed as follows:

$$\epsilon_k = \begin{cases} 1, & \text{if } B_k^T y_k \geq 0.2 B_k^T W_k B_k \\ \dfrac{0.8 B_k^T W_k B_k}{B_k^T W_k B_k - y_k^T B_k}, & \text{if } B_k^T y_k < 0.2 B_k^T W_k B_k. \end{cases} \tag{30}$$

where the factor 0.2 was chosen empirically. Initially, the Hessian approximation is set up as a unit matrix.

Under appropriate conditions, theoretical analysis shows that the algorithm can converge globally [20].

We describe the V-edge position optimization algorithm in detail. First, we obtain the community data by Algorithm 1 (Line 1). For each community, we allocate the computation resources of V-edge for devices (Line 4), then we generate the stopping time function $Q_i(x)$ (Line 5). Initialize the position of V-edge

Algorithm 2. JointCCM Algorithm Based On Sequential Quadratic Programming

Input: community sets Q
Output: the time for serve each community T
1: Initialize Q ← Algorithm 1, $T \leftarrow 0$, $j \leftarrow 0$,
2: **for** $q \in Q$ **do**
3: $x_0 \in \mathbb{R}$, $H_0 \leftarrow$ E, $k \leftarrow 0$
4: allocate F to each user in the community
5: set the function $Q_i(x)$ according to Equation (20)
6: solve QP subproblem: compute (d_k, t_k) by the quadratic programming problem (27) at x_k with corresponding KKT multiplier vectors λ_k, ρ_k. If $d_k = 0$, then $T_j = t_k + \Gamma_j/v$, j ← j+1; continue next community.
7: update $x_{k+1} = x_k + \alpha_k d_k$, H_{k+1}
8: k ← k+1. Go back to Step 5.
9: **end for**
10: **return** T

x_0 (Line 3), simplify the original problem to a quadratic programming problem (18) at iteration point x_k, then solve the programming problem to get the direction d_k and the function value, if $d_k = 0$, then stop and set value for T_j (Line 6), then get the next iteration point x_{k+1} (Line 7). Correct Hessian matrix H according to BFGS and continue to the next iteration until the optimal solution is found (Line 7–8).

After obtain all the time of completing community demands, we can transform the problem into a classical TSP problem. For simplicity, we can use existing algorithms to solve the problem, which is to optimize a path to minimize the V-edge travel time. For example, ant colony optimization algorithm, dynamic programming algorithm, branch and bound method, etc.

6 Simulation Results

In this section, extensive simulation experiments are performed to verify the effectiveness of our solution and compare it with the optimal solution, and analyze the performance under different parameters.

6.1 Evaluation Setup

We simulated a 1000*1000 m^2 non-functional square area with a random distribution of users. For convenience, the initial location of V-edge is in the center of the area. The simulation uses the communication model in [7]. The basic parameters of the simulation are summarized in Table 1. We randomly generated the computational task size of the device and conducted several experiments to eliminate the effect of randomness. To observe the effect of task size on the experiment, we divided the data size into three levels: (0, 100) MB, (100, 500) MB

Table 1. Simulation parameters

Definition	Parameters	Values
Bandwidth	W	40 MHz
Attenuation factors	η	1
White Gaussian noise power	σ^2	$5 * 10^{-15}$
Max velocity of V-edge	v_{max}	20 m/s
Transmitting power	P_{TX}	1.8 W
Receiving power	P_{RX}	2 W
Computation intensity	f	unif(200, 1000) cycles/bit
V-edge CPU frequency	F	5 GHz
The carrier frequency	g	2.4 GHz
The speed of light	c	$3 * 10^8$ m/s

and (500, 1000) MB. All algorithms in this paper are performed on a computer with Intel Core i7-10700 2.90 GHz CPU and 16G RAM.

In this evaluation, we will consider a key performance metric, namely the system completion time.

Inspired by [10,14], our proposed algorithm is compared with the following six methods:

- Optimal Static (OS): V-edge provides services for uploading, computing and downloading operations after stopping at the optimal point.
- Shortest Path (SPS): The stopping location of V-edge is determined based on the shortest sum of distances from devices in the community to a point.
- Random (Random): V-edge randomly selects a point from the community's common communication segment to serve.
- Sine Cosine Algorithm (SCA): It is a novel population-based optimization algorithm which will create multiple initial random candidate solutions and requires them to fluctuate outwards or towards the best solution using a mathematical model based on sine and cosine functions [13].
- Shortest Distance Point for each device (SDP): V-edge travels to the closest point to each device to provide service and does not consider mobility communication.
- Shortest Distance Point with Communication for each device (SDPC): V-edge travels to the closest point to each device to provide service and considers communication during the movement.

In next section, the algorithm is analyzed for different system parameters using the benchmark algorithm described above. We performed 40 independent evaluation experiments in this section.

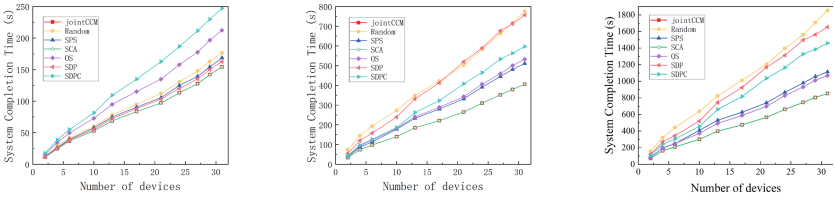


Fig. 5. System completion time under different numbers of devices. (a) $I_k \sim$ unif(1, 100) MB. (b) $I_k \sim$ unif(100, 500) MB. (c) $I_k \sim$ unif(500, 1000) MB.

6.2 Numerical Results

The system completion time of seven algorithms under different number of devices is plotted in Fig. 5. We can observe that system completion time increases when K grows. It is because more service demands need to be satisfied with growing number of devices.

System completion time under proposed jointCCM algorithm, as shown in Figs. 5(a), 5(b), and 5(c), is the lowest, compared to other algorithms. Simultaneously, we can notice that the system performance is more superior under jointCCM algorithm as the task size grows. In Fig. 5(a), the jointCCM algorithm performs 14%, 9%, 37%, 5% and 60% better than the compared Random, SPS, OS, SDP and SDPC methods respectively when number of devices is thirty-one. And in Fig. 5(b), these ratios vary to 91%, 25%, 31%, 86% and 47% respectively. And in Fig. 5(c), these ratios vary to 117%, 30%, 25%, 93% and 71% respectively. From the ratios in these three plots, we can notice that the ratios of Random and SPS increase when task size grows. This is because the stopping point of jointCCM algorithm is closer to devices so that the communication latency is shorter, compared to Random and SPS. The communication latency gap is greater when the task size increases. However, the ratios of OS algorithm in three plot decrease when task size grows, this is because a smaller portion of tasks are completed when V-edge is moving with the same time. Similarly, it can be found that as the amount of tasks increases, the jointCCM algorithm is more effective compared to SDP and SDPC. It also shows the importance of optimizing the stopping point of V-edge. In summary, jointCCM algorithm outperforms the compared methods in terms of system completion time. The algorithm performs better when the task size increases.

To investigate the effect of V-edge's computational resources, Fig. 6 shows the system completion time at different CPU frequencies. We set the maximum CPU frequency to 20 GHz in Fig. 6. As we can see, the system completion time decreases with the increase of computational resources. This is because when the CPU frequency increases, more computational resources are allocated to each task, thus reducing the computing latency. And, we can observe that the effect of computational resources on the system completion time is limited, i.e., the increasing trend becomes much slower when the CPU frequency exceeds

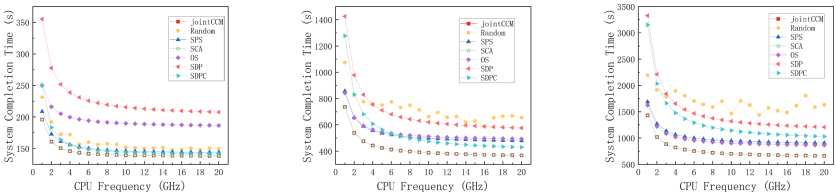


Fig. 6. System completion time under different numbers of CPU frequency. (a) $I_k \sim$ unif(1, 100) MB. (b) $I_k \sim$ unif(100, 500) MB. (c) $I_k \sim$ unif(500, 1000) MB.

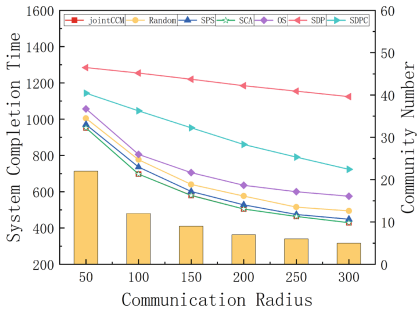


Fig. 7. System completion time and community number under different communication radius.

10 GHz. Therefore, system performance at CPU frequencies below 10 GHz is mainly affected by the lack of computational resources, which leads to an increase in task completion time. Meanwhile, it can be found that the task size has less impact on the decreasing trend from Figs. 6(a), 6(b), and 6(c) when the CPU frequency exceeds 10 GHz, and has some impact on the decreasing trend when the CPU frequency is small.

In Fig. 7, we considered the effect of the communication radius of devices on the system completion time. We set the number of devices at 30 and compared seven algorithms. The left axis of the graph is the system completion time, which is represented using a line graph, and the right axis is the number of communities, which is represented using a bar graph. We can observe that when the communication radius increases, the number of communities decreases. Meanwhile, due to the reduction of the number of communities, V-edge can complete more task demands within a community at the same time and reduce its traveling time to other communities.

In summary, it can be observed that the jointCCM algorithm based on SQP algorithm can achieve the best results. Although the SCA algorithm can also achieve this result, its convergence time is longer than that of jointCCM. In a computational offloading environment, it is preferred to choose the algorithm with faster execution time.

7 Conclusion

In this paper, we propose a vehicle-mounted edge server rescue system for emergency rescue and focus on the computational offloading problem based on mobile edge servers. Specifically, we have transforming the system completion time minimization problem into community partition, community completion time minimization subproblem and travelling time of V-edge subproblem. To solve the minimization subproblem of community completion, we have proposed jointCCM method based SQP algorithm. Numerical results have showed that the algorithm performs well.

Due to the limited coverage area of V-edge, we will consider the cooperation between V-edge and UAVs. The V-edge can carry multiple UAVs, and the V-edge can provide service to the devices on the planned path and provide energy to the UAVs, while the UAVs provide service to the devices further away or act as relays to deliver requests. The above issues will be investigated in our future work.

Acknowledgement. This work is supported by the National Natural Science Key Foundation of China grant No. 62032016 and No. 61832014, and the National Natural Science Foundation of China grant No. 62102281.

References

1. Al-Hourani, A., Kandeepan, S., Lardner, S.: Optimal lap altitude for maximum coverage. IEEE Wirel. Commun. Let. **3**(6), 569–572 (2014)
2. Ashraf, M.W., Idrus, S.M., Iqbal, F., Butt, R.A., Faheem, M.: Disaster-resilient optical network survivability: a comprehensive survey. In: Photonics, vol. 5, p. 35. Multidisciplinary Digital Publishing Institute (2018)
3. Chen, C., et al.: Delay-optimized v2v-based computation offloading in urban vehicular edge computing and networks. IEEE Access **8**, 18863–18873 (2020)
4. Du, W., He, Q., Ji, Y., Cai, C., Zhao, X.: Optimal user migration upon server failures in edge computing environment. In: 2021 IEEE International Conference on Web Services (ICWS), pp. 272–281. IEEE (2021)
5. Faraci, G., Grasso, C., Schembra, G.: Fog in the clouds: UAVs to provide edge computing to IoT devices. ACM Trans. Internet Technol. (TOIT) **20**(3), 1–26 (2020)
6. Gill, P.E., Murray, W., Saunders, M.A., Wright, M.H.: Procedures for optimization problems with a mixture of bounds and general linear constraints. ACM Trans. Math. Softw. (TOMS) **10**(3), 282–298 (1984)
7. Hou, X., Ren, Z., Wang, J., Zheng, S., Zhang, H.: Latency and reliability oriented collaborative optimization for multi-UAV aided mobile edge computing system. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 150–156. IEEE (2020)
8. Jeong, S., Simeone, O., Kang, J.: Mobile edge computing via a UAV-mounted cloudlet: optimization of bit allocation and path planning. IEEE Trans. Veh. Technol. **67**(3), 2049–2063 (2017)

9. Liu, D., Han, S., Yang, C., Zhang, Q.: Semi-dynamic user-specific clustering for downlink cloud radio access network. IEEE Trans. Veh. Technol. **65**(4), 2063–2077 (2015)
10. Liu, Y., Li, Y., Niu, Y., Jin, D.: Joint optimization of path planning and resource allocation in mobile edge computing. IEEE Trans. Mob. Comput. **19**(9), 2129–2144 (2019)
11. Lu, W., Shen, Y., Wang, T., Zhang, M., Jagadish, H.V., Du, X.: Fast failure recovery in vertex-centric distributed graph processing systems. IEEE Trans. Knowl. Data Eng. **31**(4), 733–746 (2018)
12. Meng, L., Lin, Y., Qing, S., Wenjing, F.: Research on generalized traveling salesman problem based on modified ant colony optimization. In: 2019 Chinese Control And Decision Conference (CCDC), pp. 4570–4574. IEEE (2019)
13. Mirjalili, S.: SCA: a sine cosine algorithm for solving optimization problems. Knowl. Based Syst. **96**, 120–133 (2016)
14. Ning, Z., et al.: 5g-enabled UAV-to-community offloading: Joint trajectory design and task scheduling. IEEE J. Sel. Areas Commun. **39**, 3306–3320 (2021)
15. Niu, Y., Liu, Y., Li, Y., Chen, X., Zhong, Z., Han, Z.: Device-to-device communications enabled energy efficient multicast scheduling in mmwave small cells. IEEE Trans. Commun. **66**(3), 1093–1109 (2017)
16. Powell, M.J.: The convergence of variable metric methods for nonlinearly constrained optimization calculations. In: Nonlinear Programming, vol. 3, pp. 27–63. Elsevier (1978)
17. Ren, J., Yu, G., Cai, Y., He, Y.: Latency optimization for resource allocation in mobile-edge computation offloading. IEEE Trans. Wirel. Commun. **17**(8), 5506–5519 (2018)
18. Ren, J., Yu, G., He, Y., Li, G.Y.: Collaborative cloud and edge computing for latency minimization. IEEE Trans. Veh. Technol. **68**(5), 5031–5044 (2019)
19. Sun, Y., Zhou, S., Xu, J.: EMM: energy-aware mobility management for mobile edge computing in ultra dense networks. IEEE J. Sel. Areas Commun. **35**(11), 2637–2646 (2017)
20. Wang, L., Luo, Z.: A simple SQP algorithm for constrained finite minimax problems. Sci. World J. **2014**, 159754 (2014)
21. Zhao, H., Deng, S., Zhang, C., Du, W., He, Q., Yin, J.: A mobility-aware cross-edge computation offloading framework for partitionable applications. In: 2019 IEEE International Conference on Web Services (ICWS), pp. 193–200. IEEE (2019)
22. Zhou, T., Qin, D., Nie, X., Li, X., Li, C.: Energy-efficient computation offloading and resource management in ultradense heterogeneous networks. IEEE Trans. Veh. Technol. **70**, 13101–13114 (2021)