This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2922316, IEEE Transactions on Mobile Computing

Joint Optimization of Path Planning and Resource Allocation in Mobile Edge Computing

Yu Liu, Yong Li, *Senior Member, IEEE*, Yong Niu, *Member, IEEE*, and Depeng Jin, *Member, IEEE*

***Abstract*—With the rapid development of mobile applications, mobile edge computing (MEC), which provides various cloud resources (e.g., computation and storage resources) closer to mobile and IoT devices for computation offloading, has been broadly studied in both academia and industry. However, due to the limited coverage of static edge servers, the traditional MEC technology performs badly in nowadays environment. To adapt the diverse demands, in this paper, we propose a novel mobile edge mechanism with a vehicle-mounted edge (V-edge) deployed. Aiming at maximizing completed tasks of V-edge with sensitive deadline, the problem of joint path planning and resource allocation is formulated into a mixed integer nonlinear program (MINLP). By utilizing the piecewise linear approximation and linear relaxation, we transform the MINLP into a mixed integer linear program (MILP). To obtain the near-optimal solution, we further develop a gap-adjusted branch & bound algorithm, also called GA-B&B algorithm. Moreover, we propose a low-complexity L-step lookahead branch scheme (referred to as L-step scheme) for efficient scheduling in large-scale scenarios. Extensive evaluations demonstrate the superior performance of the proposed scheme compared with the traditional static edge mechanism. Furthermore, the proposed L-step scheme achieves close performance to the near-optimal solution, and significantly improves the task completion percentage of state-of-the-art schemes by over 10%.**

***Index Terms*—Mobile edge computing, vehicles, computation task offloading, path planning, piecewise linear approximation.**

I. Introduction

The increasing popularity in smartphones and IoT devices is driving the development of mobile applications including face/speech recognition, image/video processing, real-time online gaming, etc., most of which are computation-intensive as well as latency-critical. Thus, these resource-intensive requirements pose significant challenges to mobile and IoT devices with limited processing capability. To address challenges above, mobile edge computing (MEC) has been proposed and widely studied [1]–[3]. With the concept of pushing mobile computing, cache and other network functions to the network edges, MEC is committed as a promising technology to relieve the prominent contradiction between service requirement and resource shortage. Especially, computation offloading is the most popular user-oriented use case in MEC. By offloading computation tasks to the edge server instead of local execution, the system performance (e.g., computation and latency requirements) will be greatly enhanced.

Many works about MEC have been published [4]–[19]. According to the state of edge servers, we can classify them into two types of mechanisms. In particular, the first kind of works focuses on the static edge connected with base stations (BS) or access points (AP) [4]–[9], which are deployed at fixed locations, while the other kind of works studies the mobile edge mounted on vehicles or unmanned aerial vehicles (UAVs) [10]–[19].

In recent years, the optimization research for the static scenario has become mature [4]–[9]. Due to the static deployment of edge servers, these works mainly considered the resource allocation with the optimization of system delay, energy consumption, as well as the quality of service requirements. Specially, the resource allocation with deadline sensitive tasks attracted much attention for its utility and importance. For instance, Yin *et al.* [4] proposed a multi-resource allocation algorithm to accommodate deadline sensitive tasks, where service level agreements were considered. Munoz *et al.* [5] developed the joint optimization of the radio and computational resource usage, in which the tradeoff between energy and latency was exploited with task completion deadline satisfied. Moreover, Fan *et al.* [6] presented a deadline-aware task scheduling mechanism for edge computing, where the collaboration between edge nodes and rented cloud resources was exploited by edge service providers. An ant colony optimization based heuristic algorithm was also proposed in [6] to maximize the profits of edge service provider while meeting the tasks' deadline. On the other hand, there are also some related works on joint optimization in the resource allocation. For example, Mao *et al.* [7] designed a sub-optimal algorithm for single-user MEC systems to jointly optimize the task offloading scheduling and device energy consumption. Zhang *et al.* [8] built a distributed potential game to jointly optimize the computation offloading strategy policy and computation resource scheduling with the existence of Nash equilibrium proved. Besides, Liu *et al.* [9] utilized the queuing theory to give a thorough study on the system delay, energy consumption, and monetary cost of offloading process in multi-user MEC systems. What's more, these three system objectives are jointly optimized by finding the optimal offloading probability and transmission power for each user.

Y. Liu, Y. Li, and D. Jin are with Beijing National Research Center for Information Science and Technology (BNRist), Department of Electronic Engineering, Tsinghua University, Beijing 100084, China (e-mails: liyong07@tsinghua.edu.cn, liuyu2419@126.com).

Y. Niu is with the State Key Laboratory of Rail Traffic Control and Safety, Beijing Engineering Research Center of High-speed Railway Broadband Mobile Communications, and the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, 100044, China (e-mail: niuy11@163.com).

This work was supported in part by The National Key Research and Development Program of China under grant 2017YFE0112300, the National Nature Science Foundation of China under 61861136003, 61621091 and 61673237, Beijing National Research Center for Information Science and Technology under 20031887521, and research fund of Tsinghua University - Tencent Joint Laboratory for Internet Innovation Technology.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2922316, IEEE Transactions on Mobile Computing

Nevertheless, the limited coverage of the static edge yields new serious issues in MEC. As for realistic urban environments, due to path loss and complicated radio environments, the static edge cannot support too many offloading tasks in a large region. Consequently, it is not able to cope with regular huge computation demands generated by urban intelligent transportation systems and sensor information collection. Moreover, it is envisioned that densely deploying static edges for realistic urban environments would be impractical in monetary cost. On the other hand, a more flexible MEC mechanism is quite necessary for areas with limited available infrastructure of BSs and APs, such as in rural environments, disaster relief as well as military applications. In these scenarios, users may not be able to move towards the vicinity of static edge, instead requiring the edge to move towards users [10]. Therefore, the mobile edge mechanism attracted attention and developed recently [10]–[19].

As a matter of fact, the similar idea of mobile mechanism has been applied in the communication field by Project Loon [20]. The project built the balloon network in the stratosphere, and high speed internet was accessible via the balloon relaying and transmission. Up to now, researchers have demonstrated connection speeds of up to 10 Mbps between balloons over 100 km apart. Inspired by the mobile mechanism, the mobile edge mechanism with UAVs and vehicles come into our sight. For example, Sathiaseelan *et al.* [11] firstly presented a preliminary idea of Cloudrone, where UAVs are deployed in the sky to provide various services for ground users. Later, a mobile edge infrastructure adopting UAVs was proposed in [12], and the authors evaluated the performance of the UAV-mounted edge in terms of coverage to users. In addition, an UAV-mounted edge was introduced in [13], and successive convex approximation strategies were introduced to optimize the bit allocation and moving trajectory of the mobile edge. Natalizio *et al.* [14] also introduced a Sport Event Filming problem, where UAVs are deployed for filming a sport event. By utilizing two families of algorithms including the nearest neighbor and the ball movement interception, the satisfaction of event viewers is maximized with the UAV travelling distance minimized. Meanwhile, vehicular edge (a.k.a vehicular fog) also received considerable attention. Wang *et al.* [15] proposed a vehicular edge computing (VEC) caching scheme, where parked vehicles were considered as edges, and an auction game based caching algorithm was presented to minimize the average latency to mobile users. Similarly, the concept of vehicular fog computing (VFC) was proposed in [16] by utilizing vehicles as the infrastructures for communication and computation. By contributing their computing resources, vehicles act like fog nodes in the context of fog computing, and different scenarios were also discussed to show the capacities of VFC. Furthermore, Cao *et al.* [17] leveraged connected vehicles as the edge computing platform to enhance the quality of experience (QoE), and designed a QoE based node selection strategy. Both [10] and [18] focused on the scheduling of a vehicle with powerful computing resources, applied in military and battlefield environments. Recently, Zhou *et al.* [19] also exploited moving intelligence in vehicular fog networks, where computing resources of vehicles provide diverse fog computing services and applications for vehicles and pedestrians.

According to the related works above, the flexible mobile edge mechanism can be broadly applied in the intelligent transportation systems, sensor systems, large-scale events, and emergency scenarios, where the static edge resource may be not available. On the other hand, intelligent connected vehicles equipped with strong computation power, are expected to be the promising paradigm of future vehicles [19]. Therefore, the mobile edge mechanism is promising to integrate with vehicular networks in the future, fully exploiting the computation resources in urban environments.

Unfortunately, in the area of mobile edge mechanism, few existing works consider the deadline sensitive tasks, which are important to realistic environments. Besides, most works ignore the downloading of computation results at edges for relatively small data size, while the downloading result size is actually significant for quite a few computation-intensive applications. Additionally, most works in VEC and VFC are based on parked vehicles, which are not mobile in fact. Overall, taking both the path planning and resource allocation for mobile edges into consideration is much more difficult than the only resource allocation in the static edge scenario.

Motivated by the limitations of existing literature as well as the promising gains of mobile edge mechanism, we propose a novel vehicle-mounted edge mechanism in this paper. Especially, we consider several hotspots with multiple independent tasks demanding for computation support, and a mobile V-edge for computation offloading. Moreover, both the deadline sensitive tasks and downloading process are considered for practical scenarios. Furthermore, we study the total returned data maximization problem, and jointly optimize the mobile edge's path and resource allocation. The contributions of the paper are three-fold, which are summarized as follows.

- We propose the novel vehicle-mounted edge mechanism, and formulate the total returned data maximization problem into a mixed integer nonlinear program (MINLP), with V-edge's path and resource allocation optimized. Furthermore, the MINLP problem is transformed into a mixed integer linear program (MILP), by utilizing the piecewise linear approximation and linear relaxation. We further propose the GA-B&B algorithm to obtain the near-optimal solution.
- We design a low-complexity L-step lookahead branch scheme (L-step scheme), which consists of a reward model and a computation offloading scheduling to solve the problem. The reward model is combined with a path planning criterion to plan V-edge's path. Besides, transmission bandwidths are fully exploited in the offloading process with V-edge's proximity to hotspots.
- Extensive evaluations under various system parameters are carried out to evaluate the proposed vehicle-mounted edge mechanism and validate the performance of our proposed algorithms. The results demonstrate that our proposed mobile edge mechanism with the near-optimal GA-B&B algorithm achieves better performance compared with the static edge mechanism, and about 20% of task completion percentage is improved in some cases. Evaluations also show that our proposed L-step scheme

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2922316, IEEE Transactions on Mobile Computing

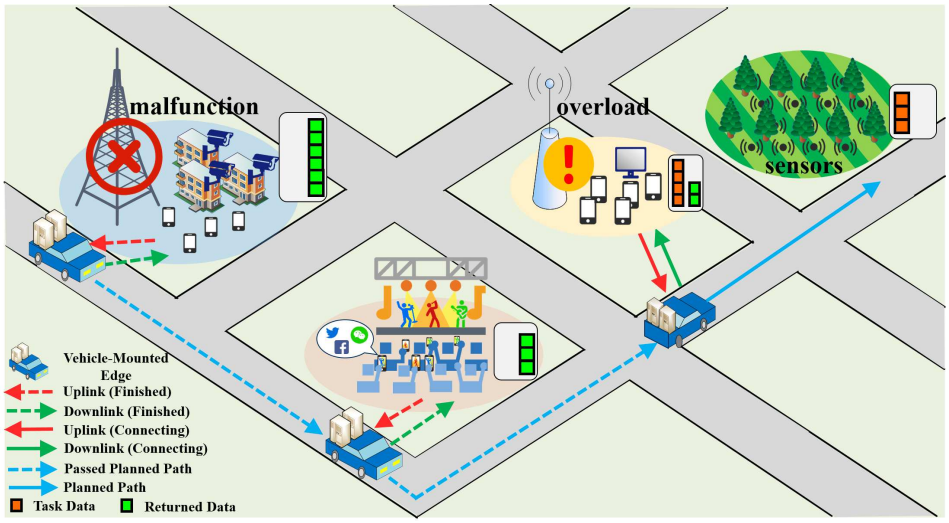


Fig. 1. Illustration of the system scenario with a V-edge in mobile edge computing. Due to the malfunction or the overload of nearby base stations, some hotspots offload tasks to V-edge. With path planning, V-edge also executes tasks for hotspots in concerts as well as sensor information collection.

gains close performance to the near-optimal solution with much lower complexity, and it outperforms the other existing scheduling schemes with over 10% of task completion percentage increased.

We organize the rest of this paper as follows. Section II introduces the system model and gives a problem overview with an example. Section III formulates the joint path planning and resource allocation optimization problem into a MINLP problem, and transforms the problem into a MILP problem. Section IV and V develop the near-optimal GA-B&B algorithm and the low-complexity L-step scheme. Section VI evaluates the performance under various system parameters. In light of our results, this paper is concluded in Section VII.

II. System Overview

A. System Model

As shown in Fig. 1, we consider a realistic urban environment with a V-edge and K hotspots, indexed by $\mathcal{K} = \{1, 2, ..., K\}$. Let $\boldsymbol{p_k} = (x_k, y_k)$ denote the location of hotspot k. On the one hand, emergencies such as malfunction and the overload of nearby base stations lead to the shortage of computation resource. On the other hand, large-scale events like concerts, as well as massive data processing in sensor regions, also cause task congestions at hotspots. Therefore, V-edge is demanded by each hotspot for computation support with a hard completion deadline, which means that the hotspot will search for other kinds of computation support or compute locally if V-edge fails to finish its task before deadline, and the task exceeding the deadline becomes invalid for V-edge, respectively. To enable tractable analysis and obtain useful insights, we employ a quasi-static network scenario [21], where all hotspots offload their task requests at the beginning of the computation offloading period, and the number of hotspots remains unchanged during the period while it may change across different periods. Based on this, we study the offloading process between the moving edge and hotspots with the goal of maximizing the total returned data by V-edge.

To determine the offloading process as well as V-edge's trajectory, the total system time period T is partitioned into N non-overlapping time slots of equal length with $\tilde{t}$ seconds, i.e., $T = N\tilde{t}$. Moreover, the length $\tilde{t}$ is chosen to be sufficiently small for V-edge's position to be approximately constant in each time slot. The position of V-edge in the n-th time slot is denoted as $\boldsymbol{p_n^e} = (x_n^e, y_n^e)$, for $n \in \mathcal{N} = \{1, 2, ..., N\}$. According to the realistic environment, the initial position of V-edge are predetermined by $\boldsymbol{p_1^e} = (x_1^e, y_1^e)$, and its maximum speed is defined as $v_{\max}$. Thus, we can obtain the trajectory of V-edge by $\{\boldsymbol{p_n^e}\}_{n\in\mathcal{N}}$ (shown as the blue line in Fig. 1). In addition, the velocity vector in the n-th time slot can be acquired as,

$$\boldsymbol{v_n^e} = \frac{\boldsymbol{p_{n+1}^e} - \boldsymbol{p_n^e}}{\tilde{t}}. \tag{1}$$

B. Computation Task Offloading Model

With the downloading of computation results considered, we adopt the four-field task model in [13] to describe the computation task at hotspots. The task at hotspot k can be represented by the notation $A_k(I_k, \gamma_k, \tau_k^d, O_k)$, which contains the task input-data size I_k (in bits), the computation intensity γ_k (in CPU cycles per bit), the completion deadline τ_k^d (in second), and the output/input ratio O_k. Especially, the maximum completion deadline determines the number of time slots, i.e., $N = \max\{\lfloor \tau_k^d/\tilde{t} \rfloor\}_{k\in\mathcal{K}}$. Besides, O_k means the number of output bits produced by per input bit, which is relevant with the task property. Accordingly, the output result size of task A_k is $I_k \cdot O_k$. For example, the ratio is low for compression tasks, while high for processing tasks of VR/AR applications.

As for computation resources of V-edge, it is equipped with a server of maximum CPU frequency F^e (in CPU cycles per second), and can provide parallel computation for multiple tasks from multiple users simultaneously using processor sharing [22]. In addition, there is a bootstrapping program run in the system [23], from which V-edge knows

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2922316, IEEE Transactions on Mobile Computing

the locations of hotspots and their corresponding task profiles A_k. Especially, hotspots first transmit task requests with task profiles A_k to nearby base stations, then base stations forward the information to V-edge. V-edge starts computation task offloading after receiving requests.

To enable the task offloading for hotspot k, the offloading process is divided into three steps; (i) hotspot k transmits input data to V-edge via the uplink (shown as the red line in Fig. 1); (ii) V-edge executes computation for the input data (shown as orange squares in Fig. II); (iii) V-edge returns output data (shown as green squares in Fig. II) to the hotspot k via the downlink (shown as the green line in Fig. 1).

C. Communication Model

Considering the path loss and randomness effects in practice, communication channels between V-edge and hotspots are typically modeled as i.i.d frequency-flat block fading channels [24], [25]. Denote the small-scale fading channel power gain between V-edge and hotspot k in the n-th time slot as $h_{k,n}$, which is assumed to have a bounded mean value, i.e., $\mathbb{E}[h_{k,n}] \triangleq \overline{h_k} < \infty$ [24]. Thus, the channel power gain between V-edge and hotspot k in the n-th time slot can be expressed as follows,

$$H_{k,n} = h_{k,n} k_0 \left\| \boldsymbol{p_k} - \boldsymbol{p_n^e} \right\|^{-\theta}, \tag{2}$$

where we denote the path loss exponent by θ, and k_0 is a constant coefficient and proportional to $(\frac{\lambda}{4\pi})^2$.

Therefore, the maximum achievable transmission rate $R_{k,n}^u$ (uplink) or $R_{k,n}^d$ (downlink) between V-edge and hotspot k in the n-th time slot can be obtained as,

$$R_{k,n}^m = W \log_2 \left(1 + \frac{P_t^m H_{k,n}}{\sigma^2} \right), \tag{3}$$

where we denote the bandwidth by W (Hz), the noise power by σ^2, the transmission power by P_t^m (W), and $m = u$ for uplink while $m = d$ for downlink.

Based on (2) and (3), $R_{k,n}^m$ can also be expressed as,

$$R_{k,n}^m = W \log_2 \left(1 + \frac{P_t^m h_{k,n} k_0 \left\| \boldsymbol{p_k} - \boldsymbol{p_n^e} \right\|^{-\theta}}{\sigma^2} \right). \tag{4}$$

Additionally, V-edge is assumed to be half-duplex, i.e., V-edge has at most one transmission link with one hotspot simultaneously, either the uplink or downlink.

D. Problem Overview

In this subsection, we discuss the key idea in the vehicle-mounted edge mechanism by an example shown in Fig. 2. The four subfigures exhibit the network topology and mechanism operation in four time slots successively. For simplicity, we just consider one hotspot with computation demand. Besides, its task can be divided into four subtasks for execution, shown as the initialization part in Fig. 2(a).

Fig. 2(a) shows the topology and operation in the first time slot. Due to the channel state limitation, the hotspot offloads three subtasks (shown as orange blocks 1, 2, 3) to V-edge via the uplink, and the computation process starts upon subtask arrival. Then in the second time slot, V-edge processes the first subtask (shown as yellow block 1), and receives the last subtask from the hotspot, as shown in Fig. 2(b). In the third time slot, V-edge further computes two subtasks, and returns part computation results (shown as green blocks 1, 2) to the hotspot via the downlink, as shown in Fig. 2(c). Finally, in Fig. 2(d), the fourth subtask is finished processing and remaining computation results are downloaded by the hotspot. It should be mentioned that V-edge keeps moving in the whole procedure, and the whole task is completed before its deadline.

As can be observed, to maximize the total returned data to hotspots (shown as green blocks in Fig. 2), the problem involves the path planning and resource allocation for V-edge. Besides, in the vehicle-mounted edge mechanism, the computation and transmission can happen concurrently. Owing to the flexibility of the mobile edge, the channel rate between the edge and hotspots can be significantly improved. Therefore, path planning greatly affects the system performance. Moreover, due to the high correlation among offloading tasks, hotspot positions, and V-edge status, the computation offloading scheduling also plays an important role in the multi-hotspot scenario.

III. Problem Formulation & Analysis

A. Problem Formulation

According to the three steps of offloading process introduced in Section II-B, in the n-th time slot, for $n \in \mathcal{N}$, we denote $D_{k,n}^u$ as the size of input data via the uplink from the hotspot k to V-edge, $D_{k,n}^c$ as the size of data processed for the hotspot k by V-edge, and $D_{k,n}^d$ as the size of output data via the downlink from V-edge to the hotspot k. Moreover, two decision variables $a_{k,n}^u$ and $a_{k,n}^d$ are defined to indicate whether the transmission link between V-edge and the hotspot k is scheduled in the n-th time slot, where $a_{k,n}^u$ is for the uplink, and $a_{k,n}^d$ for the downlink. For instance, the uplink is scheduled in the n-th time slot when $a_{k,n}^u$ is equal to 1.

Since our objective function is the total valid returned data by V-edge which also means the total amount of downlink data before the corresponding deadline, we should calculate the downlink data from V-edge to each hotspot in each time slot, i.e., $D_{k,n}^d$. Hence, by summing all $D_{k,n}^d$ together, the objective function can be expressed as,

$$\sum_{k=1}^{K} \sum_{n=1}^{N} D_{k,n}^d. \tag{5}$$

We now consider the system constraints. Due to the non-negative data offloading, we have the basic constraint,

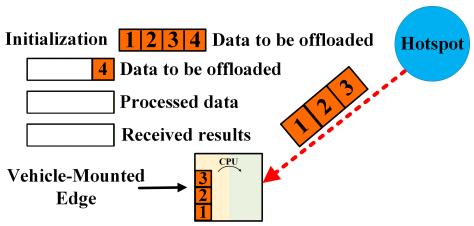
$$D_{k,n}^u, D_{k,n}^c, D_{k,n}^d \geq 0, \quad \forall k \in \mathcal{K}, \ \forall n \in \mathcal{N}. \tag{6}$$

However, the task becomes invalid when exceeding the deadline. Thus, for each hotspot, neither transmission links nor computation exists after the deadline time slot, which can be obtained as,

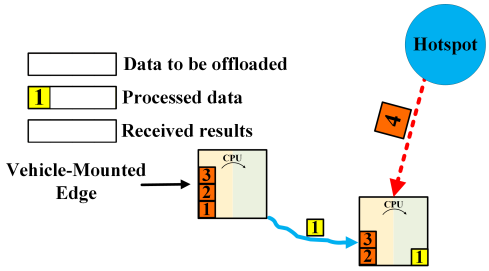
$$D_{k,n}^u, D_{k,n}^c, D_{k,n}^d = 0, \quad \forall k \in \mathcal{K}, \ \forall n = \lfloor \tau_k^d / \tilde{t} \rfloor, ..., N. \tag{7}$$

Besides, both the uplink and downlink data in each time slot is limited by the channel conditions, which means that

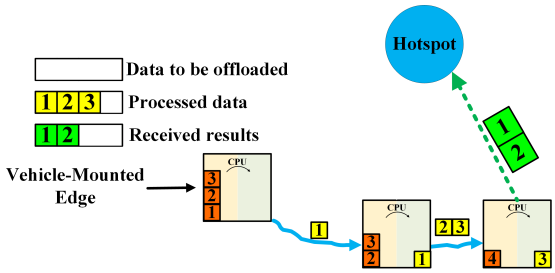
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2922316, IEEE Transactions on Mobile Computing



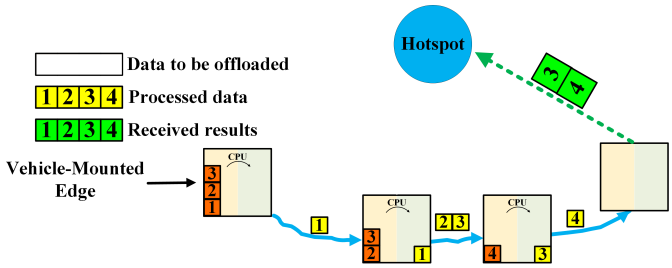
(a) Network topology and mechanism operation in the first time slot. V-edge receives first three subtasks via the uplink.



(b) Network topology and mechanism operation in the second time slot. V-edge receives the last subtask via the uplink, and computes the first task.



(c) Network topology and mechanism operation in the third time slot. V-edge computes two subtasks, and returns the results of first two subtasks to the hotspot via the downlink.



(d) Network topology and mechanism operation in the fourth time slot. V-edge computes remaining subtasks, and returns corresponding results to the hotspot via the downlink.

Fig. 2. An example of the vehicle-mounted edge mechanism with one hotspot in four time slots. V-edge finishes the three steps of uplink, computation and downlink for each subtask from the hotspot during the move. The orange, yellow and green blocks show the uplink, computation and downlink data, respectively.

the size of transmission data should not exceed the maximum achievable transmission rate. Such requirement yields the following,

$$D^u_{k,n} \le R^u_{k,n}\tilde{t}, \quad \forall k \in \mathcal{K}, \ \forall n \in \mathcal{N}. \tag{8}$$

$$D^d_{k,n} \le R^d_{k,n}\tilde{t}, \quad \forall k \in \mathcal{K}, \ \forall n \in \mathcal{N}. \tag{9}$$

In addition, the amount of transmission data is also constrained by the decision variables. For example, when the decision variable $a^u_{k,n} = 1$, the uplink data is non-negative, which is limited by channel conditions in constraint (8). Otherwise, the uplink transmission from hotspot k to V-edge is unavailable, i.e., $D^u_{k,n} = 0$. By employing the "big-M" method in [26], we can write the constraint as follows,

$$D^u_{k,n} \le M a^u_{k,n}, \quad \forall k \in \mathcal{K}, \ \forall n \in \mathcal{N}, \tag{10}$$

$$D^d_{k,n} \le M a^d_{k,n}, \quad \forall k \in \mathcal{K}, \ \forall n \in \mathcal{N}, \tag{11}$$

where M is a number safely bigger than any of the numbers that may appear on any side of the inequalities. Note that when $a^u_{k,n} = 1$, the parameter M is sufficiently large so that the associated constraint becomes redundant. Otherwise, it is enforced that $D^u_{k,n}$ can only be equal to zero with constraint (6) considered. The "big-M" has the same effect to $a^d_{k,n}$.

Since the data is offloaded through three steps including the uplink, computation, and downlink, there are numerical relationships among these variables, which can be formulated as,

$$\sum_{n=1}^{N} D^u_{k,n} \le I_k, \quad \forall k \in \mathcal{K}, \tag{12}$$

$$\sum_{i=1}^{n} D^c_{k,i} \le \sum_{i=1}^{n} D^u_{k,i}, \quad \forall k \in \mathcal{K}, \ \forall n \in \mathcal{N}, \tag{13}$$

$$\sum_{i=1}^{n} D^d_{k,i} \le O_k \sum_{i=1}^{n} D^c_{k,i}, \quad \forall k \in \mathcal{K}, \ \forall n \in \mathcal{N}, \tag{14}$$

where (12) ensures that the total uplink data by hotspot k do not exceed its task input-data size. For the hotspot $k \in \mathcal{K}$ and time slot $n \in \mathcal{N}$, the size of data processed in first n time slots is required not larger than the size of data received by V-edge in these n time slots, as shown in (13). While (14) imposes that the size of data returned to the hotspot is no larger than the size of available data computed in these n time slots.

Considering the finite computation resources at V-edge, the total allocated CPU cycles for hotspots should be limited. Then we construct the constraint as follows,

$$\sum_{k=1}^{K} D^c_{k,n}\gamma_k \le F^e\tilde{t}, \quad \forall n \in \mathcal{N}. \tag{15}$$

According to the half-duplex assumption, V-edge is scheduled with no more than one transmission link in the time slot, which is acquired as,

$$\sum_{k=1}^{K} \left(a^u_{k,n} + a^d_{k,n}\right) \le 1, \quad \forall n \in \mathcal{N}. \tag{16}$$

As previously mentioned, we have the constraint on the initial position a well as the maximum speed of V-edge, which can be written as,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2922316, IEEE Transactions on Mobile Computing

$$\boldsymbol{p_1^e} = (x_1^e, y_1^e)\,. \tag{17}$$

$$\left\|\boldsymbol{v_n^e}\right\| = \frac{\left\|\boldsymbol{p_{n+1}^e} - \boldsymbol{p_n^e}\right\|}{\tilde{t}} \le v_{\max}, \quad \forall n \in \{1, 2, ..., N-1\}. \tag{18}$$

Therefore, the design problem is formulated as follows,

$$\textbf{P1:} \quad \max_{\substack{\{D_{k,n}^u\},\{D_{k,n}^c\},\{D_{k,n}^d\},\\ \{\boldsymbol{p_n^e}\},\{a_{k,n}^u\},\{a_{k,n}^d\}}} \sum_{k=1}^{K}\sum_{n=1}^{N} D_{k,n}^d. \tag{19}$$

s.t. Constraints (6)–(18).

The first major challenge to solve problem **P1** is the nonlinear term in constraint (8), caused by the nonlinear expression of transmission rate and edge mobility. Besides, constraint (18) is a quadratic constraint due to the norm expression of position change. Accordingly, problem **P1** is a MINLP, where $a_{k,i}^u$ and $a_{k,i}^d$ are binary variables, while other variables are continuous. Since problem **P1** is more complex than the NP-complete 0-1 Knapsack problem [27], it is computationally unacceptable to utilize an exhaustive search for solving problem **P1**. Therefore, we introduce the piecewise linear approximation in the next subsection for approximately solving the MINLP.

B. Problem Transformation

Piecewise linear approximation is a widely used method to approximate the nonlinearities with piecewise linear functions, where the variable space is partitioned into a finite number of non-overlapping regions with linear expressions [28]. Moreover, the approach has the potential to approximate the nonlinear functions arbitrarily close by introducing enough regions.

As for the problem **P1**, the original nonlinear function form of $R_{k,n}^m$ is already at hand with (4), which can be further written as,

$$R_{k,n}^m = W \log_2 \left(1 + \frac{P_t^m h_{k,n} k_0 \left(\Delta x_{k,n}^2 + \Delta y_{k,n}^2\right)^{-\frac{\theta}{2}}}{\sigma^2}\right), \tag{20}$$

where $\Delta x_{k,n} = x_k - x_n^e$ is the X-axis difference between the hotspot k and V-edge in the n-th time slot, and $\Delta y_{k,n} = y_k - y_n^e$ is the Y-axis difference, respectively.The value of nonlinear term $R_{k,n}^m$ is determined by $\Delta x_{k,n}$ and $\Delta y_{k,n}$, noted as function $R_{k,n}^m(\Delta x_{k,n}, \Delta y_{k,n})$, and $m = u$ for uplink while $m = d$ for downlink.

To approximate the nonlinear function $R_{k,n}^m(\Delta x_{k,n}, \Delta y_{k,n})$ by a piecewise linear function $\hat{R}_{k,n}^m(\Delta x_{k,n}, \Delta y_{k,n})$, we first define a 2-D grid over the space of $\{\Delta x_{k,n}\}_{k\in\mathcal{K},n\in\mathcal{N}}$ and $\{\Delta y_{k,n}\}_{k\in\mathcal{K},n\in\mathcal{N}}$. Especially, we let $\Delta x_{\min} = \Delta x_1 < \Delta x_2 < ... < \Delta x_{M_x} = \Delta x_{\max}$ be a uniform partition of $\{\Delta x_{k,n}\}_{k\in\mathcal{K},n\in\mathcal{N}}$, and $\Delta y_{\min} = \Delta y_1 < \Delta y_2 < ... < \Delta y_{M_y} = \Delta y_{\max}$ be a uniform partition of $\{\Delta y_{k,n}\}_{k\in\mathcal{K},n\in\mathcal{N}}$, in which M_x and M_y determine the grid width. The corresponding $\hat{R}_{k,n}^m(\Delta x_i, \Delta y_j)_{i\in\{1,2,...,M_x\},j\in\{1,2,...,M_y\}}$ can obtained by (20). In this way, the nonlinearity is approximated by the 2-D grid, as illustrated in Fig. 3.

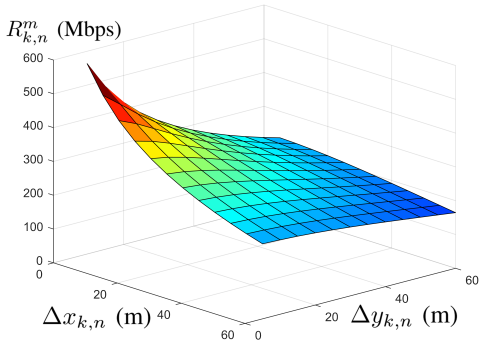


Fig. 3. Illustration of the piecewise linear approximation using the 2-D grid. The nonlinear function $R_{k,n}^m(\Delta x_{k,n}, \Delta y_{k,n}$, i.e., the continued surface is approximated by the piecewise function $\hat{R}_{k,n}^m(\Delta x_{k,n}, \Delta y_{k,n})$, i.e., the discrete grids.

With the piecewise linear approximation, the problem **P1** is transformed. However, the traditional modeling on the grid implies the introduction of auxiliary binary variables as well as combinatorial conditions, which may lead to intractable models. Hence, a special ordered set of variables of type II (SOS2) is introduced to make it easier to find global optimal solutions to the transformed problem containing piecewise linear approximations [29]. Especially, an ordered set of variables $\lambda = (\lambda_1, \lambda_2, ..., \lambda_m)$ is said to be SOS2 if at most two of the variables in the set are nonzero and the nonzero variables are adjacent [30]. Besides, the SOS2 variables can be declared in most mixed integer program solvers [31], [32], which makes the piecewise linear approximation more convenient.

Based on the approximation process and introduced SOS2 sets, the nonlinear constraint (8) is replaced by,

$$D_{k,n}^u \le \hat{R}_{k,n}^u(\Delta x_{k,n}, \Delta y_{k,n})\tilde{t}, \quad \forall k \in \mathcal{K},\ \forall n \in \mathcal{N} \tag{21}$$

$$D_{k,n}^d \le \hat{R}_{k,n}^d(\Delta x_{k,n}, \Delta y_{k,n})\tilde{t}, \quad \forall k \in \mathcal{K},\ \forall n \in \mathcal{N} \tag{22}$$

$$\Delta x_{k,n} = \sum_{m=1}^{M_x} \lambda_{k,n,m}^x \Delta x_m, \quad \forall k \in \mathcal{K},\ \forall n \in \mathcal{N} \tag{23}$$

$$\Delta y_{k,n} = \sum_{m=1}^{M_y} \lambda_{k,n,m}^y \Delta y_m, \quad \forall k \in \mathcal{K},\ \forall n \in \mathcal{N} \tag{24}$$

$$\sum_{m=1}^{M_x} \lambda_{k,n,m}^x = 1,\ \sum_{m=1}^{M_y} \lambda_{k,n,m}^y = 1, \quad \forall k \in \mathcal{K},\ \forall n \in \mathcal{N} \tag{25}$$

$$\lambda_{k,n,m}^x \ge 0, \quad \forall k \in \mathcal{K},\ \forall n \in \mathcal{N},\ \forall m \in \{1, 2, ..., M_x\} \tag{26}$$

$$\lambda_{k,n,m}^y \ge 0, \quad \forall k \in \mathcal{K},\ \forall n \in \mathcal{N},\ \forall m \in \{1, 2, ..., M_y\} \tag{27}$$

$$(\lambda_{k,n,1}^x, \lambda_{k,n,2}^x, ..., \lambda_{k,n,M_x}^x) \text{ is SOS2}, \quad \forall k \in \mathcal{K},\ \forall n \in \mathcal{N} \tag{28}$$

$$(\lambda_{k,n,1}^y, \lambda_{k,n,2}^y, ..., \lambda_{k,n,M_y}^y) \text{ is SOS2}, \quad \forall k \in \mathcal{K},\ \forall n \in \mathcal{N} \tag{29}$$

where constraint (21) is the piecewise linear approximation to constraint (8), and the introduced SOS2 variables $(\lambda_{k,n,1}^x, \lambda_{k,n,2}^x, ..., \lambda_{k,n,M_x}^x)_{k\in\mathcal{K},n\in\mathcal{N}}$ and $(\lambda_{k,n,1}^y, \lambda_{k,n,2}^y, ..., \lambda_{k,n,M_y}^y)_{k\in\mathcal{K},n\in\mathcal{N}}$ are utilized to adequately approximate the nonlinear function with the linear interpolation on the 2-D grid, as shown in constraints

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2922316, IEEE Transactions on Mobile Computing

(23)-(29).

On the other hand, quadratic constraint (18) limits V-edge's moving range between adjacent time slots into a circular region with the radius of $v_{\max}\tilde{t}$. For the sake of simplicity of formulation, we adopt the linear relaxation, and approximate the circular region by a square region, which can be formulated as,

$$\left|x^e_{n+1} - x^e_n\right| \le v_{\max}\tilde{t}, \quad \forall n \in \{1, 2, ..., N-1\}. \tag{30}$$

$$\left|y^e_{n+1} - y^e_n\right| \le v_{\max}\tilde{t}, \quad \forall n \in \{1, 2, ..., N-1\}. \tag{31}$$

Based on the piecewise linear approximation and linear relaxation, the transformed problem to **P1** can be defined as,

$$\textbf{P2:} \quad \max_{\substack{\{D^u_{k,n}\},\{D^c_{k,n}\},\{D^d_{k,n}\},\\ \{\boldsymbol{p^e_n}\},\{a^u_{k,n}\},\{a^d_{k,n}\}}} \sum_{k=1}^{K}\sum_{n=1}^{N} D^d_{k,n}. \tag{32}$$

s.t. Constraints (6)–(7), (10)–(17), (21)–(31).

According to the piecewise linear approximation to the nonlinear constraints as well as the relaxation of quadratic constraints, problem **P2** is a MILP. The number of constraints is $\mathcal{O}(KNM_{\max})$, where $M_{\max}$ is the bigger one between M_x and M_y. The number of decision variables is $\mathcal{O}(KN)$, and the number of ordered variables for SOS2 is $\mathcal{O}(KNM_{\max})$. Although MILP is generally still NP-hard, plenty of research has been made on efficient algorithms and solvers for MILP [28], [30]–[32]. Thus, we solve problem **P2** with the branch & bound algorithm in Section IV.

C. Problem Formulation in Static Edge Mechanism

To compare the performance of our proposed vehicle-mounted edge mechanism with the traditional static edge mechanism, as well as understand the difficulties of solving problem **P2**, we first formulate the same problem in static edge scenario.

The main difference of mobile and static edge mechanisms is the state of the edge. Due to the edge mobility in the mobile mechanism, time-varying channel (4) and nonlinear constraint (8) are introduced in problem **P1**. However, the static edge mechanism can remove these constraints for no mobility. The transmission rate between the static edge and hotspot k is fixed, which can be obtained as,

$$R^{sm}_k = W\log_2\left(1 + \frac{P^m_t\overline{h_k}k_0\left\|\boldsymbol{p_k} - \boldsymbol{p^e_1}\right\|^{-\theta}}{\sigma^2}\right), \tag{33}$$

where $\boldsymbol{p^e_1}$ is the position of the static edge as well as the initial position of V-edge in the mobile edge mechanism, and $m = u$ for uplink while $m = d$ for downlink.

Thus, the constraint (8) in the static edge mechanism can be written as,

$$D^u_{k,n} \le R^{su}_k\tilde{t}, \quad \forall k \in \mathcal{K},\ \forall n \in \mathcal{N}. \tag{34}$$

$$D^d_{k,n} \le R^{sd}_k\tilde{t}, \quad \forall k \in \mathcal{K},\ \forall n \in \mathcal{N}. \tag{35}$$

As a result, the same problem in the static edge scenario is formulated as,

$$\textbf{P3:} \quad \max_{\substack{\{D^u_{k,n}\},\{D^c_{k,n}\},\{D^d_{k,n}\},\\ \{a^u_{k,n}\},\{a^d_{k,n}\}}} \sum_{k=1}^{K}\sum_{n=1}^{N} D^d_{k,n}. \tag{36}$$

s.t. Constraints (6)–(7), (10)–(16), (34)–(35).

Without the nonlinear constraint or quadratic constraint in the mobile edge mechanism, problem **P3** is a MILP. The number of both constraints and decision variables is $\mathcal{O}(KN)$. Obviously, problem **P3** is much easier than **P2**, and we solve it with the optimization tools [31], [32] in Section VI.

IV. Gap-adjusted Branch & Bound Algorithm

In this section, we consider global optimization for problem **P2**, within the limits imposed by the grid points and piecewise linear approximation. To achieve this, we propose a gap-adjusted branch & bound algorithm.

The first proposal of the branch & bound algorithm dates back to the work of [33] for discrete programming. Nowadays, this approach has become the most commonly used tool for solving NP-hard optimization problems, especially the MILP problems. The two fundamental concepts of the branch & bound algorithm are relaxation and constraint enforcement. More specifically, for a maximization problem, the relaxation is utilized to obtain the upper bound of the optimal solution, while constraint enforcement is utilized to exclude solutions that are feasible to the relaxation but not to the original MILP problem [30].

As for the problem **P2**, both integer constraints (10)–(11), (16) and SOS2 constraints (28)–(29) are relaxed to obtain a linear program (LP) problem. The upper bound of the optimal solution for problem **P2** can be obtained by solving the relaxed LP problem. Then we partition the set of feasible solutions into subsets, i.e., the original MILP problem **P2** is branched into several sub-MILP problems. By enforcing the integer and SOS2 constraints on sub-MILP problems, we can obtain the lower bound of the problem **P2**, as well as bound the value of the best feasible solution in subsets and discard subsets worse than the lower bound. Moreover, we can apply the same idea to those sub-MILP problems, solving the corresponding LP relaxations. The optimality is demonstrated until the difference between the current upper and lower bounds, also known as the gap, is zero.

Therefore, a tradeoff between the convergent gap and execution time can be observed. The branch & bound algorithm has the potential to approximate the optimal solution arbitrarily close with a small enough gap. On the other hand, the introduction of small gap implies the additional search time in the algorithm execution, and thus we should adjust the gap according to the expected time cost with the approximation error under control. Here we consider the relative gap, defined as the gap divided by the upper bound, and propose the GA-B&B algorithm.

The pseudo-code for the GA-B&B algorithm is presented in Algorithm 1. We denote ε as the initial value for relative gap $rGap$, α (greater than 1) as the adjustment parameter to the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2922316, IEEE Transactions on Mobile Computing

Algorithm 1: GA-B&B Algorithm

1 **Input:** $\mathcal{K}, \boldsymbol{I}, \boldsymbol{\tau^d}, \boldsymbol{\gamma}, \boldsymbol{O}, v_{\max}, \boldsymbol{p_1^e}, M_x, M_y, \tilde{t}, \varepsilon, \alpha, t^*$;
2 **Initialization:** $rGap = \varepsilon, F = 0$;
3 Set the 2-D grid by M_x and M_y with the piecewise linear approximation;
4 Set constraints (10)–(17), (21)–(31) as Con.
5 Set the objective function (32) as Obj;
6 **while** $F == 0$ **do**
7 $\quad optimize(Obj, Con, rGap, t^*)$;
8 $\quad$ **if** solution obtained **then**
9 $\quad\quad F = 1$;
10 $\quad$ **else**
11 $\quad\quad rGap = rGap \cdot \alpha$;
12 **Return:** $\boldsymbol{D^u}, \boldsymbol{D^c}, \boldsymbol{D^d}, \boldsymbol{p}$.

Algorithm 2: The Framework of L-step Scheme

1 **Input:** $\mathcal{K}, \boldsymbol{I}, \boldsymbol{\tau^d}, \boldsymbol{\gamma}, \boldsymbol{O}, v_{\max}, \boldsymbol{p_1^e}, \beta, L, J$;
2 **Initialization:** $n = 1, l = 1, \mathcal{H} = \mathcal{K}$;
3 **while** $n \leq N$ && $|\mathcal{H}| > 0$ **do**
4 $\quad [k^*] =$ **Alg4** $(n, \boldsymbol{p_n^e}, \boldsymbol{I}, l)$;
5 $\quad [n^*, \{\boldsymbol{p_i^e}\}_{i\in\mathcal{N}^*}, \{D^u_{k^*,i}, D^c_{k^*,i}, D^d_{k^*,i}\}_{i\in\mathcal{N}^*}] =$ **Alg3** $(n, \boldsymbol{p_n^e}, \boldsymbol{I}, k^*)$;
6 $\quad I_{k^*} = 0$, $\mathcal{H} = \mathcal{H} - \{k^*\}$;
7 $\quad$ **for** each hotspot $i \in \mathcal{H}$ **do**
8 $\quad\quad$ **if** $n^* > \lfloor \tau_i^d / \tilde{t} \rfloor$ **then**
9 $\quad\quad\quad \mathcal{H} = \mathcal{H} - \{i\}$;
10 $\quad$ Update $\boldsymbol{D^u}, \boldsymbol{D^c}, \boldsymbol{D^d}, \boldsymbol{p^e}$;
11 $\quad n = n^*$;
12 **Return:** $\boldsymbol{D^u}, \boldsymbol{D^c}, \boldsymbol{D^d}, \boldsymbol{p}$.

relative gap, and t^* as the expected time cost. The branch and bound process is denoted by the function $optimize$. For simplicity, the system model variables are expressed in vector form, as in line 1. The 2-D grid for the piecewise linear approximation is defined in line 3. Then, we define the constraints as well as objective function in lines 4-5. Based on the branch & bound algorithm [28], [30], the formulated problem **P2** is solved with the relative gap and expected time cost conditions satisfied, as in line 7. Furthermore, the solving process terminates if the optimal solution is found or execution time exceeds t^*. If the MILP problem is successfully solved in expected time, the algorithm returns the path planning and resource allocation results, as in lines 8-9 and line 12. Otherwise, the algorithm relaxes the convergent condition by multiplying the gap with α, and iteratively searches the optimal solution, as in lines 10-11.

According to [34], the computational complexity of the branch and bound algorithm for the 0-1 Knapsack problem is $\mathcal{O}(2^N)$. Therefore, due to time cost of global search and optimality proof in the branch & bound algorithm, the GA-B&B algorithm is not applicable for large-scale problems in practice. Besides, the algorithm complexity is analyzed with numerical simulations in Section VI.

V. L-step Lookahead Branch Scheme

To solve the formulated MILP for large-scale scenarios, we propose an L-step scheme in this section, which has a low computational complexity compared with GA-B&B algorithm. Considering the joint optimization of path planning and resource allocation, V-edge serves hotspots one by one. Especially, we propose a resource allocation algorithm to schedule the offloading process between V-edge and the serving hotspot, including uplink/downlink transmission and computation. Furthermore, based on a reward model with the path planning criterion, we propose an L-step recursion algorithm to determine the next hotspot for serving.

A. The Framework of L-step Scheme

To illustrate the L-step scheme further, we present the framework in Algorithm 2. In line 2, the L-step scheme defines $\mathcal{H}$ as the set of hotspots demanding for computation support, and initializes $\mathcal{H}$ with $\mathcal{K}$. According to hotspots and their tasks, the L-step scheme iteratively arranges V-edge for each hotspot's task offloading, until all hotspots are scheduled or system time exceeds total time slots, as shown in line 3. Since the optimization objective is to maximize the total returned data by V-edge, the achieved returned data of hotspot selection can be modelled as the reward. Thus, line 4 first calls the L-step recursion algorithm to select the hotspot k^* with the maximum reward, which is presented in Algorithm 4 in detail. Then, line 5 calls the resource allocation algorithm to allocate corresponding resources for hotspot k^*, which is described in Algorithm 3. Especially, n^* is the system time slot when V-edge finishes tasks for hotspot k^*. $\{\boldsymbol{p_i^e}\}_{i\in\mathcal{N}^*}$ and $\{D^u_{k^*,i}, D^c_{k^*,i}, D^d_{k^*,i}\}_{i\in\mathcal{N}^*}$ are the planned path and resource allocation during the task offloading for hotspot k^*. Line 6 removes the finished hotspot k^* from $\mathcal{H}$, while lines 7-9 remove the hotspots with the completion deadline ahead of the system time. The resource allocation and path planning schemes are updated in line 10, and returned in line 12. The computational complexity of the L-step scheme will be analyzed in Section V-C.

B. Resource Allocation Algorithm

The advantage of the vehicle-mounted edge mechanism relies on the mobility of V-edge. According to the formula (4), the channel condition becomes better when V-edge approaches to the hotpot. Thus, if the hotspot k is selected as the serving hotspot, V-edge moves straight to hotspot k, and Fig. 4 gives the illustration of the resource allocation algorithm.

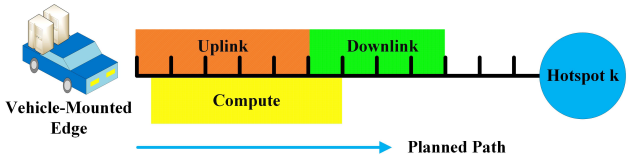


Fig. 4. Illustration of the resource allocation algorithm in L-step scheme. The uplink transmission, task execution, and downlink transmission are finished when V-edge approximates the serving hotspot.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2922316, IEEE Transactions on Mobile Computing

The resource allocation algorithm partitions the time cost from V-edge's current position to hotspot k into time slots. During the move to hotspot k, the maximum achievable transmission rate at each time slot is available. Thus, the algorithm first schedules the uplink tasks from hotspot k, and then arranges the downlink. In the transmission, V-edge computes the input data simultaneously with resource constraints satisfied. Besides, the computation may end later than the uplink due to limited computation resources, and V-edge returns available computation results first. Compared with the uplink, the downlink rate is larger due to V-edge's proximity to hotspot k, and the downlink results size is smaller than input data size generally. Accordingly, the downlink time cost is less than the uplink, as shown in Fig. 4.

Algorithm 3 presents the pseudo-code of the resource allocation algorithm. For the serving hotspot k, the algorithm calculates the left time slots for its task in line 3. In line 5, based on the illustration of Fig. 4, the algorithm schedules the task in n_k^{sch} time slots with resource constraints satisfied. However, the task I_k sometimes is too large to be executed in n_k^{left} time slots, then the algorithm reduces the task input data size I_k by multiplying a constant β (less than 1), and reschedules the offloading process, as in lines 6-7. The algorithm updates the system time n^*, planned path $\{\boldsymbol{p_i^e}\}_{i\in\mathcal{N}^*}$ in lines 9-10. The returned data from V-edge to hotspot k is modelled as the reward r_k in line 11. Finally, the algorithm returns the scheduling scheme for hotspot k in line 12.

Algorithm 3: Resource Allocation Algorithm

1 **Input:** $n, \boldsymbol{p_n^e}, \boldsymbol{I}, k$;
2 **Initialization:** $F = 0$;
3 $n_k^{\text{left}} = \lfloor \tau_k^d / \dot{t} \rfloor - n$;
4 **while** $F == 0$ **do**
5 Schedule the offloading process of taks I_k with $\{D_{k,i}^u, D_{k,i}^c, D_{k,i}^d\}_{i\in\{n+1,n+2,\ldots,n+n_k^{\text{sch}}\}}$ subject to (8)–(14), $n_k^{\text{sch}} \le n_k^{\text{left}}$;
6 **if** no feasible solution **then**
7 $I_k = I_k \cdot \beta$;
8 **else**
9 $n^* = n + n_k^{\text{sch}}, \mathcal{N}^* = \{n+1, n+2, ..., n^*\}$;
10 V-edge moves straight to hotspot k at v_{max}, with $\{\boldsymbol{p_i^e}\}_{i\in\mathcal{N}^*}$ obtained;
11 $r_k = I_k \cdot O_k$, $F = 1$;
12 **Return:** $r_k, n^*, \{\boldsymbol{p_i^e}\}_{i\in\mathcal{N}^*}, \{D_{k,i}^u, D_{k,i}^c, D_{k,i}^d\}_{i\in\mathcal{N}^*}$.

According to the illustration of Fig. 4, the computational complexity of the resource allocation is $\mathcal{O}(N)$. On the other hand, the while loop has at most $\log_\beta \frac{1}{I_{\max}}$ iterations by reducing the maximum task input data size to 1. Thus, the worst case computational complexity of the resource algorithm is $\mathcal{O}(N \log_\beta \frac{1}{I_{\max}})$, which can be implemented in practice.

C. L-step Recursion Algorithm

As mentioned before, L-step recursion algorithm builds a reward model with the path planning criterion for V-edge, to determine the serving hotspot in the next step.

Firstly, we introduce the definition of the path planning criterion here. Obviously, the serving hotspot selection is related to the task input-data size, left time for task execution, and the distance between the hotspot and V-edge. More specifically, due to the optimization objective of returned data, intuitively, V-edge should first serve the hotspot with large task size. On the other hand, to avoid missing tasks, it is better for V-edge to first serves the hotspot whose task is going invalid. Besides, distance plays an important role. Since the short distance brings the large channel rate, the bandwidth is fully exploited by selecting the near hotspot for serving. Consequently, the path planning criterion is characterized as the hotspot selection priority, which is denoted by pr, and can be expressed as,

$$pr = \frac{I}{n^{\text{left}} d}, \tag{37}$$

where I is the task input-data size of the hotspot, n^{left} is the left time slots to the completion deadline, and d is the distance between the hotspot and the current position of V-edge.

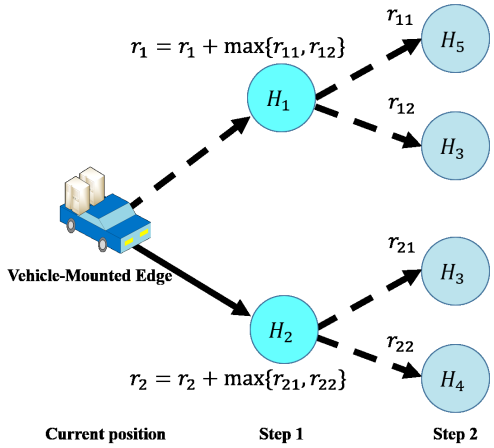


Fig. 5. An example of the L-step recursion algorithm in L-step scheme (L =2, J =2).

Then, Fig. 5 presents an example of the L-step recursion algorithm, where L is equal to 2. Based on the path planning criterion, the L-step recursion algorithm first selects J hotspots with large selection priority, and the selection is divided into J branches. In our example, J is equal to 2, and H_1 and H_2 are selected as possible serving hotspots. Motivated by the lookahead thought, the algorithm calculates the returned data when V-edge selects H_1 as the next serving hotspot, and denotes it as the reward r_1, which is also the reward in step 1. What's more, the algorithm tries to look further by calculating rewards of possible serving hotspots after H_1. In Fig. 5, the algorithm calculates rewards r_{11} and r_{12}, respectively, and defines the larger one as the reward in step 2. It is obvious the calculation process is similar to the recursion thought. As a result, the possible reward of choosing H_1 as the next serving hotspot is the sum of rewards in step 1 and step 2. Similarly, the algorithm calculates the reward of another branch (hotspot H_2 and possible hotspots after). Finally, V-edge selects to serve the hotspot with the maximum reward (H_2 in our example). In practice, the L-step lookahead

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2922316, IEEE Transactions on Mobile Computing

operation for reward calculation is achieved by the recursion, which is invoked twice in this case. Especially, the L-step recursion algorithm calls itself after calculating r_1 at H_1 and r_2 at H_2, respectively.

Owing to the benefit of the path planning criterion, the L-step recursion algorithm is able to find the hotspot with the highest reward in J branches, while the brute force search has to calculate rewards for all N hotspots, and selected the highest one in N branches. Obviously, the brute force search needs more recursions as well as computation cost. Thus, the original N branches are pruned to J ($< N$) branches in the L-step recursion algorithm with the path planning criterion.

Algorithm 4: L-step Recursion Algorithm

1 **Input:** $n, \boldsymbol{p_n^e}, \boldsymbol{I}, l$;
2 **Initialization:** $\{r_i\}_{i\in\mathcal{K}} = 0, \mathcal{J} = \emptyset$;
3 **for** each hotspot $k \in \mathcal{K}$ **do**
4 $\quad n_k^{\text{left}} = \lfloor \tau_k^d / \tilde{t} \rfloor - n$, $d_{k,n} = \left\| \boldsymbol{p_k} - \boldsymbol{p_n^e} \right\|$;
5 $\quad pr_k = \frac{I_k}{n_k^{\text{left}} d_{k,n}}$;
6 **if** $l == L$ **then**
7 $\quad$ **while** $|\mathcal{J}| < J$ **do**
8 $\quad\quad$ Find the hotspot $k \in \mathcal{K}$ with the maximum pr_k;
9 $\quad\quad [r_k] =$ **Alg3** $(n, \boldsymbol{p_n^e}, \boldsymbol{I}, k)$;
10 $\quad\quad I_k = 0$, $\mathcal{J} = \mathcal{J} \cup \{k\}$;
11 **else**
12 $\quad$ **while** $|\mathcal{J}| < J$ **do**
13 $\quad\quad$ Find the hotspot $k \in \mathcal{K}$ with the maximum pr_k;
14 $\quad\quad [r_k, n^*, \{\boldsymbol{p_i^e}\}_{i\in\mathcal{N}^*}] =$ **Alg3** $(n, \boldsymbol{p_n^e}, \boldsymbol{I}, k)$;
15 $\quad\quad I_k = 0$, $\mathcal{J} = \mathcal{J} \cup \{k\}$;
16 $\quad\quad [r_0] =$ **Alg4** $\left(n^*, \boldsymbol{p_{n^*}^e}, \boldsymbol{I}, l+1\right)$;
17 $\quad\quad r_k = r_k + r_0$;
18 Find the hotspot $k^* \in \mathcal{K}$ with the maximum r_{k^*};
19 **Return:** k^*, r_{k^*}.

In Algorithm 4, we give the pseudo-code of the L-step recursion algorithm. Regarding to the illustration above, the recursion applies to the algorithm implementation. Line 1 inputs the current time slot n and the current lookahead step l, while line 2 initializes the reward of each hotspot as zero. In lines 3-5, the algorithm calculates the selection priority of each hotspot. When the current step is the final step, the algorithm finds J branches of possible serving hotspots, and calls the resource allocation algorithm to get the corresponding reward, as in lines 6-10. Otherwise, in lines 11-14, the algorithm first calls the resource allocation algorithm, Algorithm 3, to get the reward, and obtains the system time slot n^* as well as V-edge position $\boldsymbol{p_{n^*}^e}$ after serving the hotspot. To look further hotspot rewards after the current step, based on the obtained results before, the algorithm calls itself with the current step increased, and obtains the sum reward r_0 in steps $l+1 \sim L$, as shown in line 16. The reward for hotspot k in steps $l \sim L$ is updated in line 17. Finally, line 18 finds the hotspot with maximum reward in L steps, which is returned in line 19. Accordingly, hotspot k^* is determined as the next serving hotspot by V-edge.

From the illustration of Fig. 5 and the pseudo-code, Algorithm 4 is traversing a complete J-branch tree with L layers, where $(J^1 + J^2 ... + J^{L-1}) = \mathcal{O}(J^{L-1})$ times recursions are invoked. Besides, Algorithm 3 is executed at each node in the tree. Since the tree has $(1 + J^1 + J^2 ... + J^L) = \mathcal{O}(J^L)$ nodes and computational complexity of Algorithm 3 is $\mathcal{O}(N \log_\beta \frac{1}{I_{\max}})$, computational complexity of Algorithm 4 is $\mathcal{O}(J^L N \log_\beta \frac{1}{I_{\max}})$. Considering the framework, the computational complexity of L-step scheme is $\mathcal{O}(J^L N^2 \log_\beta \frac{1}{I_{\max}})$. Generally, J and L are assigned with small values no more than 5 (as shown in Section VI later), which is enough for good performance, thus the L-step scheme is feasible in practice.

VI. Performance Evaluation

In this section, we evaluate the performance of our proposed vehicle-mounted edge mechanism and the near-optimal GA-B&B algorithm. Moreover, we compare the proposed L-step scheme with the near-optimal solution, and analyze its performance under various system parameters.

A. Evaluation Setup

We simulate a 1000×1000 m^2 square area where several hotspots are uniformly distributed. For generality and convenience, the initial position of V-edge locates in the center of the area. The computation model settings in [22] and the channel model suggested in [24], [25] are adopted in the simulation, where the small-scale fading channel power gains are exponentially distributed with unit mean, i.e., $h_{k,n} \sim$ Exp(1), $k \in \mathcal{K}$. Furthermore, the simulation parameters, are summarized in Table I. We perform 50 independent experiments for the evaluation, and plot the mean of results in figures. The GA-B&B algorithm is achieved with the optimization tools YALMIP [31] and Gurobi [32], on a computer with Intel Core i7-6700K 4 GHz CPU and 32 GB RAM.

TABLE I
Simulation Parameters

Param	Value	Param	Value
W	40 MHz	I_k	unif(1, 50) MB
σ^2	2×10^{-13} W/Hz	O_k	0.5
θ	3	τ_k^d	unif(25, 120) s
P_t^m	1 W	γ_k	unif(500, 1000) cycles/bit
$\tilde{t}$	5 s	F^e	5 GHz
$v_{\max}$	20 m/s	N	10
t_{up}	400 s	ε	1×10^{-4}
$\Delta x_{\min}$	-1000	$\Delta y_{\min}$	-1000
$\Delta x_{\max}$	1000	$\Delta y_{\max}$	1000
M_x	10	M_y	10
L	5	J	2
α	10	β	0.9

In this evaluation, we consider two main performance metrics as follows.

- **Completion percentage**: Valid returned data of V-edge divided by the total amount of demanded output data of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2922316, IEEE Transactions on Mobile Computing

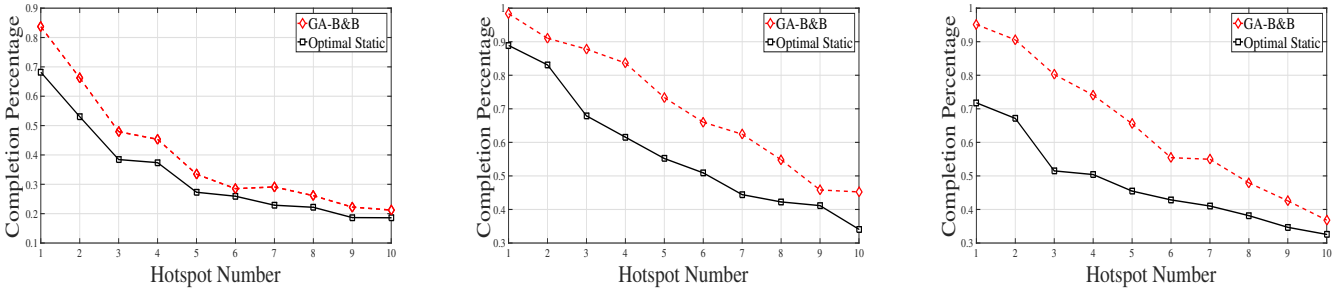


Fig. 6. Completion percentage comparison of the mobile edge and static edge mechanisms under different numbers of hotspots. (a) τ_k^d ∼unif(25, 60) s, I_k ∼unif(1, 50) MB. (b) τ_k^d ∼unif(25, 60) s, I_k ∼unif(1, 20) MB. (c) τ_k^d ∼unif(25, 120) s, I_k ∼unif(1, 50) MB.

tasks at hotspots, which is denoted by *CP*, and can be written as,

$$CP = \frac{\sum_{k=1}^{K}\sum_{n=1}^{N} D_{k,n}^{d}}{\sum_{k=1}^{K} I_k \cdot O_k}. \tag{38}$$

- **Mobility Efficiency**: Valid returned data of V-edge divided by its trajectory length (in unit of MB/m), which is denoted by *ME*, and can be expressed as,

$$ME = \frac{\sum_{k=1}^{K}\sum_{n=1}^{N} D_{k,n}^{d}}{\sum_{n=1}^{N-1} \left\| \boldsymbol{p}_{\boldsymbol{n+1}}^{\boldsymbol{e}} - \boldsymbol{p}_{\boldsymbol{n}}^{\boldsymbol{e}} \right\|}. \tag{39}$$

In Section VI-B, we compare the mobile and static edge mechanisms, where the near-optimal performance of the GA-B&B algorithm (noted as **GA-B&B**) is compared with the benchmark schemes:

- **Optimal Static**: A static edge server is deployed at the area center, and optimal solutions of problem **P3** are obtained with above optimization tools.

In Section VI-C, we compare the performance of L-step with the near-optimal solution and other four benchmark scheme in the mobile edge mechanism:

- **Maximum Task First (MTF)**: According to the optimization objective, V-edge always serves the hotspot with the maximum task size. Besides, the resource algorithm in L-step scheme is utilized for computation offloading. MTF is suitable for the tasks whose importance is determined by task size.
- **Shortest Remaining Time (SRT)**: Borrowing the idea of shortest remaining time first in task scheduling [35], V-edge serves the hotspot in the priority of task remaining time, and allocate resources with Algorithm 3. SRT is suitable for the tasks whose importance is determined by task deadline.
- **Shortest Distance First (SDF)**: To exploit the bandwidth, V-edge serves the nearest hotspot with the same resource allocation process in L-step scheme. SDF is suitable for the tasks whose importance is determined by the distance.
- **Random**: V-edge randomly serves the hotspot without task deadline considered.

In Section VI-D, the L-step scheme is analyzed under different system parameters with above benchmark algorithms. We perform 1000 independent experiments for the evaluation in this part.

B. Mobile/Static Edge Mechanism Comparison

To analyze the performance of different mechanisms, we compare the completion percentage of mobile and static edge mechanisms under different hotspot group sizes in Fig. 6. In particular, to test the robustness of the proposed GA-B&B algorithm as well as prove the superiority of the mobile edge mechanism, we adjust the distribution of the task size and the completion deadline in subplots of Fig. 6.

In particular, all subplots of Fig. 6 show the declining impact of increasing hotspot group size to the completion percentage. A larger hotspot group size results in a larger amount of tasks. However, due to finite computation resources and the same expected deadline for tasks, the edge is not able to compute much more tasks compared with small hotspot group size case. Even so, our proposed vehicle-mounted edge mechanism outperforms the static edge mechanism in different system environments.

Taking Fig. 6(a) as an example, when there are five hotspots demanding for computation support, the vehicle-mounted edge mechanism achieves a completion percentage of 33%, which improves the static edge mechanism performance by 7%. The expected valid computed data can be estimated by $K \times \mathbb{E}(\boldsymbol{I}) \times CP = 5 \times 25.5 \times 33\% = 42.08$ MB. As for the same group size in Fig. 6(c), by extending the expected completion deadline, the achieved completion percentage significantly increases to 65% and the improved value is 20%. The expected value is 82.875 MB, respectively. From the results, we can obtain the importance of completion deadline to the system performance.

Moreover, it can be observed that the difference between two mechanisms is reducing with the increase of hotspot group. A possible explanation to the phenomenon is that, the hotspot increase weakens the advantage of flexible position change of V-edge. Here we consider an extreme case, in which the hotspot group is large enough to cover the area. Then

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2922316, IEEE Transactions on Mobile Computing

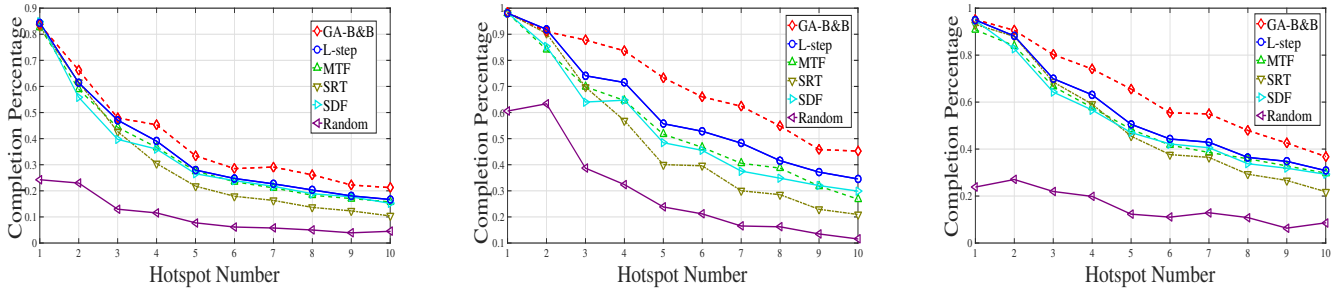


Fig. 7. Completion percentage comparison of near-optimal solution and L-step scheme under different numbers of hotspots. (a) τ_k^d ∼unif(25, 60) s, I_k ∼unif(1, 50) MB. (b) τ_k^d ∼unif(25, 60) s, I_k ∼unif(1, 20) MB. (c) τ_k^d ∼unif(25, 120) s, I_k ∼unif(1, 50) MB.

TABLE II
EXECUTION TIME COMPARISON OF ALL SCHEMES UNDER DIFFERENT # HOTSPOTS (IN SECOND)

Scheme \ # Hotspots	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
GA-B&B	14.8061	29.9471	118.4274	187.5683	188.2830	225.2451	290.2677	400.1509	353.5429	430.5672
Optimal Static	1.4923	3.2954	13.5766	27.0712	14.4897	15.8373	34.5675	32.1401	23.4707	30.3613
L-step	0.0034	0.0038	0.0038	0.0046	0.0032	0.0036	0.0034	0.0035	0.0034	0.0038
MTF	0.0020	0.0025	0.0023	0.0022	0.0018	0.0019	0.0021	0.0021	0.0021	0.0020
SRT	0.0011	0.0016	0.0016	0.0017	0.0018	0.0019	0.0021	0.0022	0.0021	0.0023
SDF	0.0011	0.0015	0.0016	0.0018	0.0018	0.0019	0.0020	0.0020	0.0020	0.0021
Random	0.0015	0.0020	0.0019	0.0019	0.0018	0.0018	0.0019	0.0019	0.0020	0.0019

compared with the fixed edge, V-edge move will not affect the average distance to hotspots, i.e., the average channel rate to hotspots. Consequently, the transmission data as well as the completion percentage of two mechanisms are almost same. It can be envisioned that two lines in Fig. 6 will coincide when the hotspot is large enough. But for the practical environment, the vehicle-mounted edge mechanism always achieve better performance.

C. Comparison with Near-optimal Scheme

To evaluate the performance of the low-complexity L-step scheme, we plot the completion percentage comparison of L-step scheme and the near-optimal solution as well as other benchmark schemes in Fig. 7. Specifically, performances in different system environments are shown in subplots.

According to the results, the L-step scheme always achieves the highest completion percentage compared with other four benchmark schemes. For example in Fig. 7(b), when the hotspot group size is ten, the L-step scheme improves the completion percentage by about 5% compared with the best bench scheme, which demonstrates the advantages of the path planning criterion as well as the L-step lookahead techniques. From the results in Fig. 7(a) and Fig. 7(c), we can observe that the gap between the L-step scheme and near-optimal solution is negligible under heavy traffic (large task sizes). In terms of ten hotspot scenario, the gaps in both figures are no more than 6%.

Furthermore, we show the average execution time of all schemes under different numbers of hotspots in Table II, where the simulation parameters are the same as Table I. As we can observe, due to the global search, the near-optimal scheme, GA-B&B scheme takes much longer time than other schemes. Moreover, the execution time of GA-B&B scheme increases quickly with the increase of hotspot number. For instance, the average execution time of GA-B&B scheme for ten hotspot scenario is about 430 seconds, while the maximum task completion deadline is 120 seconds, which implies that GA-B&B scheme is not suitable for online scheduling in the mobile edge mechanism. On the other hand, the L-step scheme is able to finish in less than 0.01 seconds, which indicates it is more computational efficiency. Furthermore, we have also analyzed the execution time of L-step scheme under large-scale problems with 50 hotspots, and the time cost is about 0.0054s, which again demonstrates the practical feasibility of the scheme.

Considering the robust performance in different environments, the short execution time as well as the negligible gaps with the near-optimal solution, the L-step scheme is a promising solution for large-scale problems in the mobile edge mechanism.

D. Performance Analysis of L-step Scheme

The completion percentage of five schemes under different task sizes is plotted in Fig. 8. Other simulation parameters are the same as Table I. We can observe that the completion percentage of schemes decreases globally with the increase task size, which agrees with intuitions. However, almost no change is found in the valid return data by V-edge when the maximum task size exceeds 40 MB. For example, when the maximum task size equals 40 MB, the completion percentage for 10 hotspots is 38%, i.e., the expected valid computed data is about $K \times \mathbb{E}(\boldsymbol{I}) \times \boldsymbol{CP} = 10 \times 20.5 \times 38\% = 78$ MB. As for the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2922316, IEEE Transactions on Mobile Computing

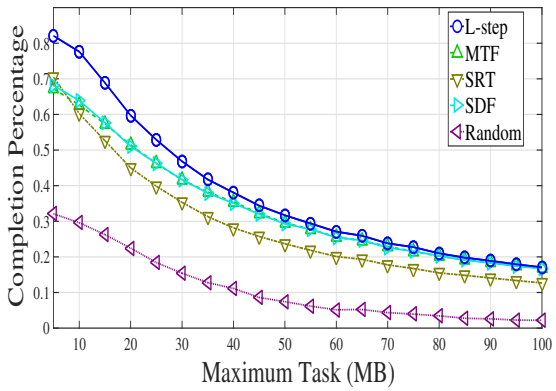


Fig. 8. Completion percentage of five schemes under different task sizes.

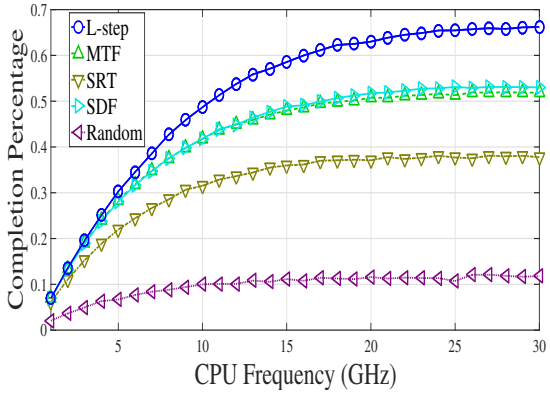


Fig. 10. Completion percentage of five schemes under different edge computation resource.

55 MB case, the completion percentage is about 29%. Thus, the corresponding expected valid computed data is 81 MB. This phenomenon is mainly caused by the finite computation resource at the edge, which yields the close returned data with all resources utilized. On the other hand, although the L-step scheme always achieves the best performance, the completion percentages of L-step scheme, MTF and SDF schemes are almost the same when the task size exceeds 70 MB. This is mainly because V-edge can only serve one or two hotspots with such tasks before the completion deadline, thus the L-step and other two schemes select the same serving hotspots, which leads to the same performance.

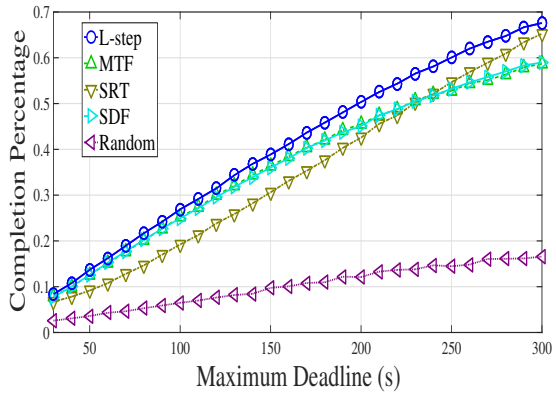


Fig. 9. Completion percentage of five schemes under different task deadline.

To investigate the impact of the completion deadline, Fig. 9 exhibits the completion percentage of five schemes under different completion deadlines. From the results, we can observe that the completion percentage increases linearly with the extension of completion deadline. As mentioned before, the completion deadline determines the serving time of V-edge. It can be envisioned that V-edge will complete all tasks if the deadline is late enough. Hence, under the late completion deadline, the SRT which aims not to miss tasks, can achieve better performance than MTF and SDF, and close to the L-step scheme. Furthermore, the decreased gap between SRT and L-step scheme at the deadline of 300 seconds proves the above supposition. Based on Fig. 8 and Fig. 9, we can obtain the robustness of the L-step scheme, especially the effectiveness of the path planning criterion.

In Fig. 10, we reveal the relationship between the completion percentage and the computation resources of V-edge. Actually, with multiple cores equipped, the sum CPU frequency of the MEC servers ranges from 1 GHz to even 100 GHz in existing literature [22], [36]. As for Fig. 10, we consider the maximum CPU frequency of 30 GHz, which can be implemented easily in practice. As the results show, the completion percentage increases with the increase of computation resources, since more tasks can be computed with larger CPU frequency. Again, the L-step scheme outperforms other schemes. However, the increasing trend becomes much slower when the CPU frequency exceeds 24 GHz. Especially, as the CPU frequency increases from 5 GHz to 24 GHz, the completion percentage increases over twofold from 30% to 65%, while the completion percentage seems unchanged after 24 GHz. Therefore, the system performance with the CPU frequency less than 24 GHz is mainly affected by the computation resources, and the computation resource lack leads to the low completion percentage. As for the case of CPU frequency larger than 24 GHz, the gap between different schemes indicate the gain of the path planning criterion as well as lookahead thoughts. For instance, compared with SDF, the L-step scheme increases the completion percentage by 13% at the CPU frequency of 25 GHz.

Considering the impact of the CPU frequency above, we depict the completion percentage comparison of five schemes under different hotspot groups with two CPU frequency settings in Fig. 11. Especially, the simulated hotspot group extends to 50, while Fig. 11(a) shows the 5 GHz scenario and Fig. 11(b) shows the 25 GHz case, respectively. Although the L-step scheme achieves the highest completion percentage in both subplots, the gap between the L-scheme and the best benchmark scheme in Fig. 11(a) is quite small, which is mainly limited by the finite computation resources. The larger gap in Fig. 11(b) indicates that the L-step scheme fully exploits the utilization of computation resources. Accordingly, with the CPU frequency of 25 GHz, the L-step scheme increases the completion percentage of MTF by an average of 8%.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2922316, IEEE Transactions on Mobile Computing

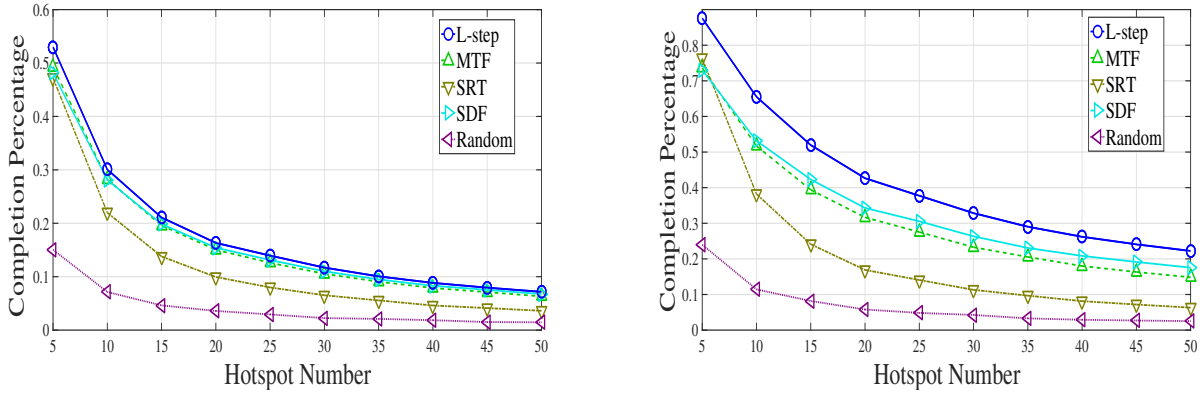


Fig. 11. Completion percentage comparison of five schemes under different numbers of hotspots. (a) $F^e = 5$ GHz. (b) $F^e = 25$ GHz.

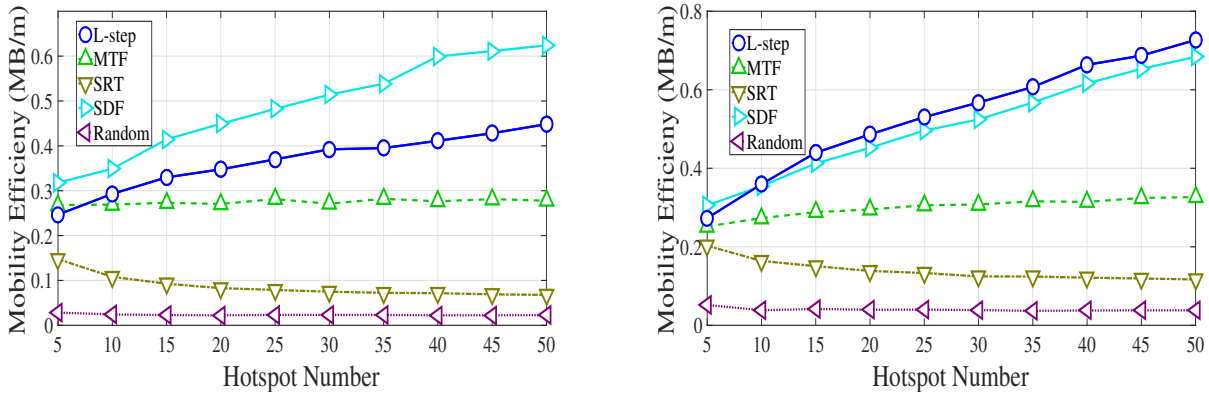


Fig. 12. Mobility efficiency comparison of five schemes under different numbers of hotspots. (a) $F^e = 5$ GHz. (b) $F^e = 25$ GHz.

On the other hand, when the hotspot group size exceeds 30, the performance of the L-step scheme is mainly affected by the transmission bandwidth and completion deadline. For example, the expected valid computed data for 30 hotspots is about $K \times \mathbb{E}(\boldsymbol{I}) \times CP = 30 \times 20.5 \times 33\% = 202.95$ MB, and 227.85 MB for 50 hotspots, respectively. The finite bandwidth and fixed completion deadline limit uplink data, and furthermore decrease valid computed data of V-edge.

In order to analyze the effectiveness and feasibility of the L-step scheme, we compare the mobility efficiency of five schemes under different hotspot group sizes with two CPU frequency settings in Fig. 12. The mobility efficiency means the valid computed data in unit V-edge's trajectory. Since the path planning criterion of MTF, SRT and Random is not related with the distance, the hotspot increase does not change the average length of the planned path, thus the mobility efficiency of these three schemes fluctuates little with the hotspot group size. Moreover, it is worth noting that, MTF outperforms SRT and Random due to the consideration of task size priority. On the other hand, the mobility efficiency of the L-step scheme and SDF shows the positive correlation with the hotspot group size. Due to the limited computation resources in Fig. 12(a), MTF even achieves higher mobility efficiency than the L-step scheme, i.e., V-edge in MTF computes less valid data than the L-step scheme but with much shorter trajectory. With enough computation resources in Fig. 12(b), the L-step scheme achieves the highest mobility efficiency. For example, the mobility efficiency of the L-step scheme under 30 hotspots with 5 GHz is 0.39, while the value increases to 0.57 with 25 GHz, which again proves the rationality of our path planning criterion and lookahead thoughts. Taking into consideration the high CPU frequency in practice, the L-step scheme is able to achieve high mobility efficiency.

In sum of results above, the proposed vehicle-mounted edge mechanism outperforms the traditional static edge mechanism. Furthermore, our proposed L-step scheme significantly improves the system performance compared with other benchmark schemes, and results under different environments demonstrate the robustness and effectiveness.

VII. Conclusions

In this paper, we proposed a novel vehicle-mounted edge mechanism in MEC, and considered the computation offloading problem in the multiple hotspots scenario. Based on the piecewise linear approximation and linear relaxation techniques, we presented a GA-B&B algorithm to obtain the near-optimal solution. Moreover, we designed a low-complexity

L-step scheme for large-scale scenarios with less time cost. In L-step scheme, we proposed a reward model with a path planning criterion to decide V-edge's trajectory, and proposed a transmission scheduling algorithm for resource allocation. Finally, performance evaluations demonstrate that the novel mobile edge mechanism outperforms the traditional static edge mechanism. Furthermore, comparison between $L-$step scheme and the near-optimal solution demonstrates L-step scheme achieves close performance in practice.

Considering the diverse system requirement on energy consumption and latency requirement, we will extend the single mobile edge to multiple mobile edges, and jointly optimize these demands. Since this paper focuses on hotspots, we will also work on the general case that mobile users offload tasks to mobile edges. Besides, we will take the transportation cost of V-edge into consideration, and evaluate the system performance from an economic perspective. Furthermore, the applications of the proposed mobile edge mechanism in vehicular networks are needed to investigate.

REFERENCES

[1] "Mobile-edge computing—-Introductory technical white paper", White Paper, ETSI, Sophia Antipolis, France, Sept. 2014. [Online]. Available: https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_-_introductory_technical_white_paper_v1%2018-09-14.pdf, Accessed on: Apr. 10, 2018.

[2] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, fourth quarter, 2017.

[3] P. Mach, and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, third quarter, 2017.

[4] B. Yin, Y. Cheng, L. X. Cai, and X. Cao, "Online SLA-aware multi-resource allocation for deadline sensitive jobs in edge-clouds," in *Proc. IEEE GLOBECOM*, Singapore, Singapore, Dec. 4–8, 2017, pp. 1–6.

[5] O. Munoz, A. P. Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.

[6] J. Fan, X. Wei, T. Wang, T. Lan, and S.Subramaniam, "Deadline-aware task scheduling in a tiered IoT infrastructure," in *Proc. IEEE GLOBECOM*, Singapore, Singapore, Dec. 4–8, 2017, pp. 1–7.

[7] Y. Mao, J. Zhang, and K. B. Letaief, "Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems," in *Proc. IEEE WCNC*, San Francisco, CA, USA, Mar. 19–22, 2017, pp. 1–6.

[8] J. Zhang, W. Xia, F. Yan, and L. Shen, "Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing," *IEEE Access*, vol. 6, pp.19324–19337, Mar. 2018.

[9] L. Liu, Z. Chang, X. Guo, S. Mao, and T.Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 283–294, Feb. 2018.

[10] S. A. Zanlongo, A. C. Wilson, L. Bobadilla, and T. Sookoor, "Scheduling and path planning for computational ferrying," in *Proc. IEEE Mil. Commun. Conf.*, Baltimore, MD, USA, Nov. 1–3, 2016, pp. 636-641.

[11] A. Sathiaseelan, A. Lertsinsrubtavee, A. Jagan, P. Baskaran, and J. Crowcroft, "Cloudrone: Micro clouds in the sky," in *ACM DroNet'16 Workshop*, Singapore, Singapore, Jun. 26–30, 2016, pp. 41–44.

[12] M. Narang, S. Xiang, W. Liu, J. Gutierrez, and L. Chiaraviglio, "UAV-assisted edge infrastructure for challenged networks," in *Proc. IEEE INFOCOM Workshop*, Atlanta, GA, USA, May 1–4, 2017, pp. 60–65.

[13] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 2049–2063, Mar. 2018.

[14] E. Natalizio, R. Surace, V. Loscrí, F. Guerriero, and T. Melodia, "Two families of algorithms to film sport events with flying robots," in *IEEE MASS*, Hangzhou, China, Oct. 14-16, 2013, pp. 319–323.

[15] S. Wang, Z. Zhang, R. Yu, and Y. Zhang, "Low-latency caching with auction game in vehicular edge computing," in *IEEE ICCC*, Qingdao, China, Oct. 22–24, 2017, pp. 1–6.

[16] X. Hou, *et al.*, "Vehicular fog computing: A viewpoint of vehicle as the infrastructures," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3860–3873, June 2016.

[17] Y. Cao, and Y. Chen, "QoE-based node selection strategy for edge computing enabled internet-of-vehicles (EC-IoV)," in *IEEE VCIP*, St. Petersburg, FL, USA, Dec 10–13, 2017, pp. 1–4.

[18] A. Monfared, M. Ammar, E. Zegura, D. Doria, and D. Bruno, "Computational ferrying: Challenges in deploying a mobile high performance computer," in *Proc. IEEE WoWMoM*, Boston, MA, USA, June 14–17, 2015, pp. 1–6.

[19] S. Zhou, Y. Sun, Z. Jiang, and Z. Niu, "Exploiting moving intelligence: Delay-optimized computation offloading in vehicular fog networks," *IEEE Commun. Mag.*, to be published.

[20] Google Inc., Mountain View, CA, USA, "Project Loon," [Online]. Available: https://x.company/loon/, Accessed on: Apr. 15, 2018.

[21] Q. V. Pham, T. LeAnh, N. H. Tran, and C. S. Hong, "Decentralized computation offloading and resource allocation in heterogeneous networks with mobile edge computing," *arXiv preprint arXiv:1803.00683*, 2018.

[22] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2637–2646, Oct. 2017.

[23] Y. Niu, *et al.* "Device-to-device communications enabled energy efficient multicast scheduling in mmWave small cells," *IEEE Trans. Commun.*, vol. 66, no. 3, pp. 1093–1109, Mar. 2018.

[24] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol 16, no. 9, pp. 5994–6009, Sept. 2017.

[25] C. Niu, Y. Li, R. Q. Hu, and F. Ye, "Fast and efficient radio resource allocation in dynamic ultra-dense heterogeneous networks," *IEEE Access*, vol. 5, pp. 1911–1924, Feb. 2017.

[26] D. Bertsimas and J. N. Tsitsiklis, "The simplex method," in *Introduction to Linear Optimization*, Belmont, MA, USA: Athena Scientific, 1997, ch. 3, sec. 5, pp. 117–119.

[27] D. Pisinger, "Where are the hard knapsack problems?" *Comput. Oper. Res.*, vol. 32, no. 9, pp. 2271–2284, Sept. 2005.

[28] B. Geißler, A. Martin, A. Morsi, and L. Schewe, "Using piecewise linear functions for solving MINLPs," in *Mixed Integer Nonlinear Programming*, New York, NY, USA: Springer, 2012, ch. 5, sec. 1, pp. 287–314.

[29] E. Beale and J. Tomlin, "Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables," in *Proc. 5th Int. Conf. Oper. Res.*, Venice, Italy, 1970, pp. 447–454.

[30] P. Belotti, *et al.*, "Mixed-integer nonlinear optimization," *Acta Numerica*, vol. 22, no. 22, pp. 1–131, Apr. 2013.

[31] J. Löfberg, "YALMIP: A toolbox for modelling and optimization in MATLAB," in *Proc. IEEE CASD*, Taipei, Taiwan, Sept. 2–4, 2004, pp. 284–289.

[32] Gurobi (2018). *Gurobi Optimizer Reference Manual, Version 8.0*. [Online]. Available: http://www.gurobi.com/documentation/8.0/refman.pdf, Accessed on: Apr. 10, 2018.

[33] A. H. Land and A. G. Doig, "An automatic method of solving discrete programming problems," *Econometrica*, vol. 28, no. 3, pp. 497–520, Jul. 1960.

[34] A. Shaheen and A. Sleit, "Comparing between different approaches to solve the 0/1 Knapsack problem," in *Int. J. Netw. Security*, vol. 16, no. 7, pp. 1-10, Jul. 2016.

[35] W. Stallings, "Scheduling algorithms," in *Operating systems: Internals and design principles*, Upper Saddle River, NJ, USA: Pearson, 2012, ch. 9, sec. 2, pp. 413–414.

[36] M. Li, R. Yu, P. Si, and Y. Zhang, "Green machine-to-machine communications with mobile edge computing and wireless network virtualization," *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 148–154, May, 2018.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2922316, IEEE Transactions on Mobile Computing

Yu Liu received the B.E. degree in electronic engineering from Tsinghua University, Beijing, China, in 2018. He is currently pursuing the Ph.D. degree in electronic engineering with Tsinghua University, Beijing, China. His research interests include wireless networks, edge computing, and optimization.

Yong Li (M'09-SM'16) received the B.S. degree in electronics and information engineering from Huazhong University of Science and Technology, Wuhan, China, in 2007 and the Ph.D. degree in electronic engineering from Tsinghua University, Beijing, China, in 2012. He is currently a Faculty Member of the Department of Electronic Engineering, Tsinghua University.

Dr. Li has served as General Chair, TPC Chair, TPC Member for several international workshops and conferences, and he is on the editorial board of two IEEE journals. His papers have total citations more than 4100. Among them, ten are ESI Highly Cited Papers in Computer Science, and four receive conference Best Paper (run-up) Awards. He received IEEE 2016 ComSoc Asia-Pacific Outstanding Young Researchers and Young Talent Program of China Association for Science and Technology.

Yong Niu (M'2017) received the B.S. degree in electrical engineering from Beijing Jiaotong University, China, in 2011, and his Ph.D. degree in electronic engineering from Tsinghua University, Beijing, China, in 2016. He was the recipient of a 2018 International Union of Radio Science (URSI) Young Scientist Award.

He is currently an Associate Professor with the State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University. During November 2014 to April 2015, he visited University of Florida, FL, USA as a Visiting Scholar. His research interests are in the areas of networking and communications, including millimeter wave communications, device-to-device communication, medium access control, and software-defined networks. He received the Ph.D. National Scholarship of China in 2015, Outstanding Ph. D Graduates and Outstanding Doctoral thesis of Tsinghua University in 2016, and Outstanding Ph. D Graduates of Beijing in 2016. He has served as Technical Program Committee (TPC) member for IWCMC 2017, VTC2018-Spring, IWCMC 2018, INFOCOM 2018, and ICC 2018, and also session chair for IWCMC 2017. He also received the Outstanding Doctorate Dissertation award from Chinese Institute of Electronics in 2017.

Depeng Jin received the B.S. and Ph.D. degrees from Tsinghua University, Beijing, China, in 1995 and 1999 respectively both in electronics engineering.

He is an associate professor at Tsinghua University and vice chair of Department of Electronic Engineering. Dr. Jin was awarded National Scientific and Technological Innovation Prize (Second Class) in 2002. His research fields include telecommunications, high-speed networks, ASIC design and future Internet architecture.