Optimal-Delay-Guaranteed Energy Efficient Cooperative Offloading in VEC Networks

Yao Zhu[1], Tianyu Yang[2], Yulin Hu[1*], Wanting Gao[1], and Anke Schmeink[1]
[1]ISEK Research Group, RWTH Aachen, Germany,
Email: $zhu|hu|gao|schmeink$@isek.rwth-aachen.de
[2]Communications and Information Theory Group (CommIT), TU Berlin, Germany,
Email: $tianyu.yang$@tu-berlin.de

***Abstract*—Taking into consideration of vehicle mobility and fairness, in this paper we provide a cooperative offloading algorithm maximizing the energy efficiency for a vehicular edge computing network, while guaranteeing the shortest delay of the worst case vehicle. In particular, through exploiting the geometrical feature of the unidirectional road, we formulate a mixed integer convex problem by jointly designing the offloading selection and allocating the computation resource simultaneously. The optimization is carried out by a proposed two-step optimization algorithm: We first optimize the server selection to obtain the minimized achievable delay, and subsequently optimize jointly the selection and resource allocation to maximize the energy efficiency while maintaining the optimal achievable delay. Via simulations, we present the advantage of proposed algorithms and evaluate the system performance.**

***Index Terms*—energy efficiency, offloading, vehicular edge computing, vehicular communication**

I. Introduction

Vehicular applications, e.g., autonomous driving, on-wheel infotainment services and intelligent transportation, have gained a lot of popularity [1]. Along with the rapid development of those applications, the explosive demands on processing computation-intensive and latency-critical tasks become an inevitable challenge to be addressed [2]. Therefore, the newly emerged vehicular edge computing (VEC) network shows significant advantage by offloading the computation services remotely at the edge of the networks, instead of merely relying on the on-vehicle devices. Moreover, in comparison to the mobile cloud computing (MCC), the VEC servers are deployed along with the road side units (RSUs), which provides the computation capacity while fulfilling the latency requirements due to the close proximity to the vehicles.

A crucial part of VEC is to design the offloading scheme, which decides not only whether to offload or not, but also including what and how to offload [3], [4]. The authors in [5] propose a cooperative partial offloading scheme allocating the computation resources and time slots to vehicles. In addition, a low-complexity algorithm is proposed in [6] for a multi-server offloading problem by decomposing it into sub-problems of offloading selection and resource allocation. An offloading framework is provided in [7] by leveraging the Stackelberg game theoretic approach, where both the cost and the gain of the offloading are jointly considered. Moreover, [8] minimizes the costs of vehicles and VEC servers at the same time, while taking channel states and traffic arrival model into account.

However, most of the existing works addressing the cooperative offloading problem in VEC networks either ignore task queue of the server into account or formulate non-convex problems which are generally solved by alternative searching, resulting in sub-optimal solutions and costing significant complexities [9]–[11]. Moreover, how to characterize the completion delay of tasks and energy consumption should be jointly considered in the cooperative offloading designs.

In this work, we consider a VEC network with multiple servers providing service to multiple users. We exploit the geometrical and unidirectional feature of the VEC to provide an energy efficient offloading design while guaranteeing the minimized maximum delay among the servers. In particular, the main contributions are:

- By taking the advantages of the relaxed binary and introducing intermediate variables, we formulate an elegant mixed integer convex optimization problem (which can be solved efficiently) instead of a complex non-convex problem.
- To address the energy efficiency problem, a two-step algorithm is proposed, which optimizes both the offloading decision and computation resource allocation, while considering the queue status in each VEC server. Different from existing studies regarding the related topics formulating complex non-convex problems and obtaining merely suboptimal solutions, the proposed approach (including the problem formulation and the corresponding two-step algorithm) achieves the global optimum.

The rest of the paper is organized as follows. We present the system model in Section II. Section III first introduces the intermediates to formulate the optimization problems, then provides the optimal two-step offloading scheduling design accordingly. We provide the numerical results in Section V. Finally, Section VI concludes the paper.

II. System Model

We consider a unidirectional road (or one side of a two-way road) with length L_R, where M road side units (RSUs) are located along the road side. Each RSU is equipped with

$*$Y. Hu is the corresponding author.

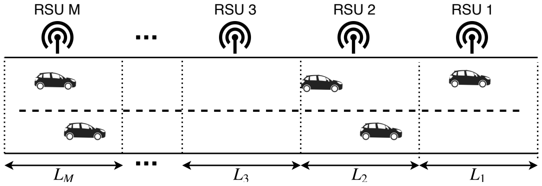

Fig. 1. An illustration of the considered VEC network.

a VEC server. As shown in Fig. 1, the road is divided into M segments according to the corresponding coverage area of each VEC server. We denote by $m \in \mathcal{M} = \{1, ..., M\}$ the index of the RSU and by $L_m \in \{L_1, ..., L_M\}$ the length of segment m, where $\mathcal{M}$ is the RSU set. We sort the index of the RSUs by descending of the position x^{s}_m, i.e., $x^{\mathrm{s}}_1 \geq x^{\mathrm{s}}_2 ... \geq x^{\mathrm{s}}_M$. Therefore, the location of the RSU m is given by $x^{\mathrm{s}}_m = L_R - \sum_{j=1}^{m} L_j + L_m/2$. Furthermore, we denote by $F_m \in \{F_1, ..., F_M\}$ the maximal CPU-frequency that the server m is able to provide for the offloaded tasks. Due to the limited computation resource, each server can only compute a single task at the same time.

Due to the limited local computation resource and the possible fact that processing the tasks of vehicles needs certain (environment) information from the RSUs, each task considered in this work is required to be offloaded to a VEC server and be executed within a latency constraint. We consider a time-slotted manner for the offloading scheduling, i.e., each scheduling decision is made at the beginning of each time slot only for the vehicles that request offloading before the time slot, where the waiting time in the queue is reflected by the individual delay constraint. In particular, at the beginning of a period of interest, N vehicles (at random locations on the road) with a constant velocity v request to offload latency-sensitive tasks to the VEC servers. Each request from vehicle $n \in \mathcal{N} = \{1, ..., N\}$ contains a task that can be described as a tuple $(d_n, c_n, T^{\max}_n)$, where $\mathcal{N}$ is the index set of vehicles, d_n is the size of input data and c_n is the required computing workloads. Moreover, $T^{\max}_n$ represents the maximum delay tolerance of the task of n-th vehicle[1]. Let x^{v}_n be the position of the vehicle n on the road. Similar to the index of the RSUs, we sort the index of vehicles by descending of the positions, i.e., $x^{\mathrm{v}}_1 \geq x^{\mathrm{v}}_2 \geq ... \geq x^{\mathrm{v}}_N$. Furthermore, we denote by $\mathbf{A} \in \mathbb{Z}_2^{N \times M}$ the decision matrix and $\mathcal{A}$ the corresponding set, in which the element $a_{n,m} \in \{0, 1\}$ is the decision variable, i.e., $a_{n,m} = 1$ or $a_{n,m} = 0$ indicates that task n is assigned to RSU m or not. Recall that the road is unidirectional, task n is only able to be offloaded to RSUs being in front of the vehicle, i.e., $a_{n,m} \leq x^{\mathrm{s}}_m - x^{\mathrm{v}}_n + 1$. In this way, $a_{n,m}$ is possible to be 1 if and only if $x^{\mathrm{s}}_m \geq x^{\mathrm{v}}$. Although all vehicles on the road segment could request offloading service at any positions, requests from the vehicles already located after the last RSU M will not be responded by the VEC network (on the road segment of interest), i.e., they will be serviced by the VEC network on the next road segment. Furthermore, each task should be assigned once and only once, i.e., $\sum_{m=1}^{M} a_{n,m} = 1, \forall n \in \mathcal{N}$. Once decisions are made, a vehicle starts to offload the task via a wireless link when it enters the coverage area of the assigned RSU.

A. Traffic model

To offload the task to the server m, the vehicle n must firstly move to the edge of the corresponding coverage area. If the vehicle n is already located in the area, the vehicle starts the offloading immediately. Hence, the driving delay is given by $t^{\mathrm{v}}_{n,m} = \max\{\frac{x^{\mathrm{s}}_m - L_m/2 - x^{\mathrm{v}}_n}{v}, 0\}$.

B. Wireless transmission model in the offloading

We assume that the channel between a RSU and a vehicle has Line-of-Sight (LoS). Hence, the channel gain can be predicted with the known distance [15]. After arriving the coverage area of an RSU, a vehicle is able to offload its task via wireless link to the RSU. The transmission delay/cost for offloading task n to the RSU m is given by $t^{\mathrm{r}}_{n,m} = \frac{d_n}{B \log_2(1 + P_n h_{n,m}/N_0)}$, where B is the bandwidth, P_n is the transmit power of vehicle n, $h_{n,m}$ is the channel gain between vehicle n and RSU m, and N_0 is the noise power, respectively. To avoid inter-vehicle interference, the offloading process is only carried out if all previous offloading processes are completed, which can be guaranteed by constraining the offloading decision $a_{n,m}$, which can be written as

$$\begin{aligned} a_{n,m} t^{v}_{n,m} + (1 - a_{n,m})(t^{v}_{n-1,m} + t^{r}_{n-1,m}) \\ \geq a_{n-1,m}(t^{v}_{n-1,m} + t^{r}_{n-1,m}). \end{aligned} \tag{1}$$

When the computation process is finished, RSUs transmit the computation results to the vehicles through a micro-cell [12]. Since the data sizes of the results are normally small, the energy cost of a back-transmission from a RSU to a vehicle is constant and more importantly negligible in comparison to the task offloading process [13], [14].

C. Computation model

The CPU frequency at each server is assumed to be limited. For the server at RSU m, at most F_m frequency can be allocated for computing the offloaded task. If frequency $f_{n,m}$ is allocated for the offloaded task with workload c_n, computing the task costs a computation delay $t^{\mathrm{c}}_{n,m}$, which is given by $t^{\mathrm{c}}_{n,m} = \frac{c_n}{f_{n,m}}$. By exploiting the dynamic frequency and voltage scaling (DVFS) technique, we assume that the server is able to adjust the frequency $f_{n,m}$ per task. However, the frequency cannot exceed the maximal frequency F_m. On the other hand, the minimal frequency $f^{\min}_m$ must at least fulfill the condition, that the execution of the task is able to be finished before the vehicle n leaving the coverage area of the server m, i.e.,

$$f^{\min}_{n,m} = \max\{\frac{c_n v}{L_m}, \frac{c_n v_n}{x^{\mathrm{s}}_m + L_m/2 - x^{\mathrm{v}}_n}\}. \tag{2}$$

[1] For the sake of simplicity, in this work, we represent the task of vehicle n as task n and the server of RSU m as the server m.

Furthermore, the server m offers no computation resource to the vehicle n, if it is not selected. As a result, the frequency constraints can be described as inequality chains, i.e., $a_{n,m}f_{n,m}^{\min} \leq f_{n,m} \leq a_{n,m}F_m$. We define the CPU-frequency matrix $\mathbf{f} \in \mathbb{R}^{N\times M}$ that contains elements $f_{n,m}, \forall n \in \mathcal{N}, m \in \mathcal{M}$.

D. Energy consumption model

Both the transmit energy of the vehicles for offloading tasks and the computation energy of the RSUs are taken into account. Specifically, the transmit energy E^{r} relating to the transmit power and duration is obtained as

$$E^{\mathrm{r}} = \sum\nolimits_n \sum\nolimits_m a_{n,m} t_{n,m}^{\mathrm{r}} P_n. \tag{3}$$

The computational energy E^{c} relating to the CPU-frequency and assigned workloads is obtained as [16]

$$E^{\mathrm{c}} = \sum\nolimits_n \sum\nolimits_m \kappa c_n f_{n,m}^2, \tag{4}$$

where $\kappa > 0$ is a hardware related parameter.

III. Energy Efficiency Maximization with an Optimal Overall Delay Guarantee

In this section, we provide our joint design on optimizing both offloading decision and CPU-frequency allocation for a considered time period and for a VEC network on a road segment of interest. The joint design has two aims, i.e., the overall delay minimization and energy efficiency maximization, which have the first and second priorities, respectively. To this end, we in the following first characterize the completion time cost of each task, subsequently following the characterization formulate the problem for minimizing the overall delay (of all tasks), and finally further introduce a energy minimization process while guaranteeing the minimized overall delay.

A. Completion time cost of a task

As a VEC network in the unidirectional road, its computation resource is unequally distributed according to the geometrical positions of those vehicles, i.e., vehicle n has less offloading choices if it is located close to the end of the road segment. To determine the completion time of the task from vehicle n, we need to track when and where task is processed while taking the vehicle location in to account. Thus, to facilitate modelling the overall delay we introduce two intermediates: $T_{n,m}^{\mathrm{RT}}$ the ready time of task n in server m and $t_{n,m}$ the process time of task n in server m, respectively.

1) Ready time (point) $T_{n,m}^{\mathrm{RT}}$: indicates the instant when the task is ready for computing, which cannot be earlier than either the offloading time of the vehicle n or the process time of previous task in server m. It reads as

$$T_{n,m}^{\mathrm{RT}} = \max\{t_{n-1,m}, a_{n,m}(t_{n,m}^{\mathrm{v}} + t_{n,m}^{\mathrm{r}})\}. \tag{5}$$

Specially, we have $T_{0,m}^{\mathrm{RT}} = 0$ and $t_{0,m} = 0$. It is worth to mention that the task can not be offloaded before the vehicle arrives the coverage area. Moreover, if the server m is not selected for task n, i.e., $a_{n,m} = 0$, $T_{n,m}^{\mathrm{RT}}$ only depends on the previous tasks.

2) Process time (point) $t_{n,m}$: implies the instant that the computation process of task n is finished in server m:

$$t_{n,m} = T_{n,m}^{\mathrm{RT}} + a_{n,m}^2 t_{n,m}^{\mathrm{c}}. \tag{6}$$

Note that the decision variable expressed as $a_{n,m}^2$ is for the purpose of maintaining convexity in the optimization problem and it does not influence the results since it is binary. Furthermore, it holds that $t_{n,m} \geq T_{n,m}^{\mathrm{RT}} \geq t_{n-1,m}$, where the equality holds if $a_{n,m} = 0$. It implies that the process time of task n in the server m is virtually identical with the previous task $n-1$ in the same server m if that server is not selected. Therefore, the actual complete time of task n is $\sum_m a_{n,m} t_{n,m}$. It is worth to mention such expression only holds for the unidirectional road.

With the intermediates $t_{n,m}$ and $T_{0,m}^{\mathrm{RT}}$, the model in the VEC network can be intuitively interpreted as follows: a vehicle n offloads $M-1$ **empty** tasks with no input data to other servers except the real selected one, while the real task with input data d_n is offloaded to the selected server. At the same time, server m computes N tasks for all vehicles, including empty tasks.

B. Overall delay minimization

The overall delay of all tasks is limited by the completion time cost of the last task. Since all N tasks are offloaded to the VEC servers, the overall delay is equivalent to the maximal process time among the servers for the last task N, i.e., $\max\limits_m\{t_{N,m}\}$. We minimize the overall delay by optimally allocating the CPU-frequency $\mathbf{f}$, as well as the offloading decision $\mathbf{A}$. The problem is formulated as

$$\mathbf{P1}: \underset{\mathbf{f},\mathbf{A}}{\text{minimize}} \quad \max_m \; t_{N,m} \tag{7a}$$

$$\text{subject to}$$

$$\sum\nolimits_{m=1}^{M} a_{n,m} = 1, \; \forall n \in \mathcal{N}, \tag{7b}$$

$$a_{n,m} \leq x_m^{\mathrm{s}} - x_n^{\mathrm{v}} + 1, \; \forall n \in \mathcal{N}, m \in \mathcal{M}, \tag{7c}$$

$$a_{n,m} f_{n,m}^{\min} \leq f_{n,m} \leq a_{n,m} F_m, \; \forall n \in \mathcal{N}, m \in \mathcal{M}, \tag{7d}$$

$$a_{n,m} t_{n,m}^{v} + (1 - a_{n,m})(t_{n-1,m}^{v} + t_{n-1,m}^{r}) \geq a_{n-1,m}(t_{n-1,m}^{v} + t_{n-1,m}^{r}), \tag{7e}$$

$$a_{n,m} \in \{0,1\}, \; \forall n \in \mathcal{N}, m \in \mathcal{M}, \tag{7f}$$

where constraint (7b) ensures that each task will be and only be offloaded once. Constraint (7c) represents the geometrical relationship between vehicle n and possible RSUs. The available CPU-frequency for task n in server m is restricted by maximal available CPU-frequency F_m, minimal required CPU-frequency $f_{n,m}^{\min}$ to ensure feedback process and decision $a_{m,n}$ in constraint (7d). Constraint (7e) ensures that there is no interference between vehicles. The integer constraint (7f) is for the binary decision variable.

1) Optimal solution of (7a)*:* Since $\mathbf{A}$ is integer variable with $2^{N\times M}$ possible offloading combinations of $\mathcal{A}$, we first decompose **P1** into $2^{N\times M}$ sub-problems for a given $\mathbf{A}$, where each sub-problem can be written as

$$\begin{aligned} \mathbf{P1.1}: \underset{\mathbf{f}}{\text{minimize}} \quad & \max_{m} \ t_{N,m} && \text{(8a)} \\ \text{subject to} \quad & \text{(7b), (7c), (7d) and (7e).} \end{aligned}$$

Then, we have the following lemma to characterize the relaxed problem:

Lemma 1. $\mathbf{P1.1}$ *is convex in* $\mathbf{f}$ *with any fixed* $\mathbf{A}$.

Proof. First, we let $\mathbf{A} = \mathbf{A}^{\circ} \in \mathcal{A}$, where $\mathbf{A}^{\circ}$ is an arbitrary fixed server selection. It is trivial to show the constraints are either affine or convex. We focus on the objective. As the object function contains a maximum function, we replace it with a variable p so that it follows

$$t_{N,m} \leq p, \ \forall m \in \mathcal{M}. \tag{9}$$

Based on (5), $T^{\mathrm{RT}}_{n,m} \geq t_{n-1,m}$ and $T^{\mathrm{RT}}_{n,m} \geq a_{n,m}(t^{\mathrm{v}}_{n,m} + t^{\mathrm{r}}_{n,m})$ hold. Hence, the second derivative of $t_{n,m}$ to $\mathbf{f}$ is

$$\frac{\partial^2 t_{n,m}}{\partial \mathbf{f}^2} = \begin{cases} 2a^2_{i,j}\frac{c_i}{f^3_{i,j}}, & i = n \text{ and } j = m \\ 0, & \text{otherwise} \end{cases} \geq \mathbf{0}, \tag{10}$$

where i is the index of column and j is the index of row, respectively. As a results, **P1.1** is convex in $\mathbf{f}$. □

Thus, $\mathbf{P1}$ is a mixed integer convex problem (MICP) and can be solved efficiently via the solvers, e.g., CVX [17] and Gurobi [18] with computational complexity of $\mathcal{O}(2^{N\times M})$. To further reduce the computational complexity, we exploit the constraint that each task can only be offloaded once, i.e., $\sum_{m=1}^{M} a_{n,m} = 1, \ \forall n \in \mathcal{N}$, to pre-define a feasible set of $\mathbf{A}$. In particular, any feasible variable $\mathbf{A}$ must consist a single 1-element and M-1 0-elements in any column n. Therefore, we denote the feasible set $\hat{\mathcal{A}}$ that contains total $N \times M$ possible combinations of $a_{n,m}$, i.e., $\hat{\mathcal{A}} = \{\mathbf{A} | \sum_{m=1}^{M} a_{n,m} = 1, \forall n \in \mathcal{N}\}$. The optimal solution of A must be the element of $\hat{\mathcal{A}}$. Instead of going through all $2^{N\times M}$ convex subproblems, we only need to compute the feasible set $\mathcal{A}$ and solve $N \times M$ convex subproblems. As a result, we reduce overall computational complexity from $\mathcal{O}(2^{N\times M})$ to $\mathcal{O}(N \times M)$.

2) Greedy searching method: Nevertheless, the complexity of the above approach is high, especially for large scale networks with. Therefore, we propose a greedy searching method addressing this issue, which introduces a significantly low complexity. In particular, we first relax the integer constraints (7f) for $\mathbf{P1}$ as $0 \leq a_{n,m} \leq 1$. Then, we have the following lemma to characterize the relaxed problem:

Lemma 2. $\mathbf{P1}$ *is jointly convex over* $\mathbf{f}$ *and* $\mathbf{A}$ *with the relaxation of* $\mathbf{A}$.

Proof. We have already proven $\frac{\partial^2 t_{n,m}}{\partial \mathbf{f}^2} \geq \mathbf{0}$ in Lemma 1. Similarly, the second derivative of $t_{n,m}$ with respect to $\mathbf{A}$ is given by

$$\frac{\partial^2 t_{n,m}}{\partial \mathbf{A}^2} = \begin{cases} 2\frac{c_i}{f_{i,j}}, & i = n \text{ and } j = m \\ 0, & \text{otherwise} \end{cases} \geq \mathbf{0}. \tag{11}$$

To show the joint convexity, we further investigate the determination of Hessian matrix $\mathbf{H}(t_{n,m})$

$$\det \mathbf{H}(t_{n,m}) = \begin{cases} 4a^2_{i,j}\frac{c_i}{f^4_{i,j}} - \left(\frac{2a_{i,j}c_i}{f^2_{i,j}}\right)^2, i = n, \ j = m \\ \qquad\qquad 0, \text{ otherwise} \end{cases} \geq \mathbf{0}. \tag{12}$$

As a result, both the second derivatives are non-negative and the Hessian matrix is positive semi-definite. Hence, $t_{n,m}$ is jointly convex in both $\mathbf{A}$ and $\mathbf{f}$.

Similarly, we can show that all the constraints are either convex or affine with relaxed $\mathbf{A}$. Therefore, $\mathbf{P1}$ is jointly convex in $\mathbf{f}$ and $\mathbf{A} \in [0,1]^{n\times m}$. □

Thus, we first solve $\mathbf{P1}$ as a relaxed convex problem without integer constraint (7f) to obtain optimal solution $\tilde{\mathbf{f}}^*$ and $\tilde{\mathbf{A}}^*$. Then, convert $\tilde{\mathbf{A}}^*$ to an integer solution by taking both (7b) and (7f) into account as follows:

$$a^*_{n,m} = \begin{cases} 1 & \text{if } m = \arg\max_{j\in\mathcal{M}} \tilde{a}^*_{n,j}, \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

Finally, we solve **P1.1** with $\mathbf{A} = \mathbf{A}^*$ to obtain the solution $\mathbf{f}^*$ and the result $t^*_{N,m}$. Comparing to the MICP, such method reduces the complexity significantly. However, it is worth to mention that it provides only a sub-optimal solution instead of guaranteeing global optimum by solving MICP.

C. Two-step energy minimization

The overall delay of the network is constrained by the optimal solution to $\mathbf{P1}$. Here we provide a further step network optimization maximizing the energy efficiency while guaranteeing the minimal overall delay. Note that the total duration of process time is restricted, i.e., maximizing the energy efficiency is equivalent to minimizing the energy consumption. This energy consumption minimization problem is formulated as

$$\begin{aligned} \mathbf{P2}: \underset{\mathbf{f},\mathbf{A}}{\text{minimize}} \quad & E = \alpha E^{\mathrm{r}} + \beta E^{\mathrm{c}} && \text{(14a)} \\ \text{subject to} \quad & t_{N,m} \leq t^{\max}_{N,m}, \ \forall m \in \mathcal{M} && \text{(14b)} \\ & \text{(7b), (7c), (7d), (7e) and (7f)} \end{aligned}$$

where α and β are non-negative weight factors indicating the importance of the energy consumption at vehicles and servers, respectively. In particular, the objective represents the absolute total energy consumption of the network when we set $\alpha = \beta = 1$. Note that $\mathbf{P2}$ has the same convex constraints as $\mathbf{P1}$, i.e., it is also convex if the convexity of its objective function can be shown. We show this convexity and further characterize $\mathbf{P2}$ in the following lemma:

Lemma 3. *Energy consumption E is jointly convex in* $\mathbf{f}$ *and* $\mathbf{A}$ *with the relaxation of* $\mathbf{A}$.

Proof. The total energy consumption E consists of transmission energy consumption E^{r} and computational energy consumption E^{c}. It is trivial to show that each component of E^{r}, namely $a_{n,m}t^{\mathrm{r}}_{n,m}P_n$, is convex in $\mathbf{A}$. Therefore, E^{r}

is also convex in $\mathbf{A}$. Similarly, we have the second derivative of component of E^{c} with respect to $\mathbf{f}$ is non-negative, i.e., $\frac{d^2 \kappa c_n f_{n,m}^2}{d\mathbf{f}^2} \geq 0$,. Furthermore, E^{r} does not depend on $\mathbf{f}$ and E^{c} does not depend on $\mathbf{A}$, i.e., they are jointly convex in both $\mathbf{f}$ and $\mathbf{A}$. As the sum of E^{r} and E^{c}, E is jointly convex in both $\mathbf{f}$ and $\mathbf{A}$. □

Therefore, $\mathbf{P2}$ can also be solved with both standard MICP method or the proposed greedy method after having $t_{N,m}^{\max}$ from $\mathbf{P1}$. Via a two-step algorithm subsequently solving $\mathbf{P1}$ and $\mathbf{P2}$, the minimal energy consumption is achieved while guaranteeing the optimal overall delay. The corresponding computational complexity level is $\mathcal{O}(2N \times M)$. The pseudo code of the algorithm is given in below.

Algorithm 1 Two-step optimization problem

```
Compute feasible set $\mathcal{A}$ according to constraint (7b).
Solve $\mathbf{P1}$ according to Lemma 2 within $\mathcal{A}$, get results $t_{N,m}^{\circ}, \forall m \in \mathcal{M}$.
Let $t_{N,m}^{\max} = t_{N,m}^{\circ}, \forall m \in \mathcal{M}$ and update the constraints (14b).
Solve $\mathbf{P2}$ according to Lemma 3 within $\mathcal{A}$, get solution $\mathbf{A}^*, \mathbf{f}^*$.
```

IV. Simulation Results and Discussion

In this section, we present simulation results to show the performance advantage of our proposed design in comparison to other benchmark schemes.

We consider $M = 5$ RSUs deployed along a 100 meter long unidirectional road. Each RSU is equipped with a VEC server with a coverage range 20m. The maximum computational resource of each server F_m is uniformly distributed in the range from 3 to 5 GHz. The speed of vehicles is set to $v = 120$km/h. The locations of vehicles are also randomly distributed within the road. In addition, task size d_n and required computation resource c_n follow uniform distributions $\mathcal{U}(100, 300)$KB and $\mathcal{U}(0.5, 1.5)$GHz, respectively. Furthermore, we consider the bandwidth as $B = 1$MHz, the transmission power of each vehicle as $P = 100$mW, and the noise power $N_0 = 10^{-10}$mW. We also set $\alpha = \beta = 1$ in the simulations. Following [16], we set $\kappa = 10^{-11}$ as the hardware parameter for the computational energy consumption. To verify our proposed schemes, we introduce the nearest offloading (NO) scheme as a benchmark, where all vehicles offload tasks to the nearest reachable VEC server with a given CPU frequency F_m.

In Fig. 2, we show the key features of proposed two-step optimization compared to the delay minimization approach in $\mathbf{P1}$. The left sub-figure and the right sub-figure present the process status of each server for each approach, respectively. The height of the rectangle is the process time of the last task in the corresponding server. Moreover, the color represents the CPU-frequency for preceding the tasks. The darker color is showed, the higher CPU-frequency is allocated at that time, where white color implies the server is idle. As $P1$ aims at minimizing the maximum of $t_{N,m}$, the bottleneck of the system is the highest rectangle (server 3 and 4 in the top sub-figure). Therefore, prolonging of the process time or changing offloading decision for the non-bottleneck tasks does not affect the overall delay, but reduces the computational energy consumption according to (4) via decreasing the CPU-frequency. The prolongation of process time and the changes in server selection are possible to be carried out in following scenarios: $i)$ The current task is already finished, but the next assigned vehicle does not finish the offloading or does not arrive, e.g., the blank space of server 4 is filled up in our two-step algorithm. $ii)$ The process time of all assigned tasks in the server is smaller than the process time in the bottleneck server, and the surplus between the completion time cost in the bottleneck server and that server is sufficient to compute another task, e.g., one of the assigned tasks in the server 2 is taken over by the server 1 so that the process time of both servers is prolonged. $iii)$ The process time of all assigned tasks in the server is smaller than the bottleneck server, but the surplus is insufficient to compute one more task, e.g., the tasks in server 5 can be prolonged until the maximal process time is equal to the process time in the server 4, but the take-over of another task is impossible.

To show the advantage of the two-step optimization, we plot the overall delay and total energy consumption versus the number of vehicles N in Fig. 3 and Fig. 4 with the same setup. The results are obtained under the NO algorithm, delay optimization and two-step optimization. Generally, increasing the number of vehicles N also increases both the process time and energy consumption regardless of the schemes. Comparing to the NO algorithm, where the vehicle simply chooses the nearest server to offload the task, solving $\mathbf{P1}$ optimizes the selection to avoid the computational traffic jam in the system, resulting a lower overall delay. Nevertheless, if there are only few vehicles on the road, the optimization does not always improve the overall delay. By keeping increasing N, the performance gap between NO algorithm and overall delay optimization grows more significantly. As our two-step optimization guarantees the same optimal delay as $\mathbf{P1}$, the delay performance is also identical between the duo. From the perspective of energy consumption, since the CPU-frequency is fixed in the NO algorithm, the offloading consumes the highest energy. On the other hand, if we focus on the delay optimization, only the bottleneck servers has to provide high CPU-frequency. For other servers, the CPU-frequency can be freely chosen as long as the process time of the servers does not surpass the bottleneck ones. Therefore, the energy consumption decreases comparing to NO algorithm. Our two-step optimization therefore prolongs the process time of all possible tasks to the limit while guaranteeing the optimal overall delay, resulting a minimum of the energy consumption.

V. Conclusion

In this paper, we studied the cooperative offloading scenario from the perspective of both delay and energy consumption. By exploiting the geometrical feature, we first formulated an elegant mixed integer convex optimization problem instead

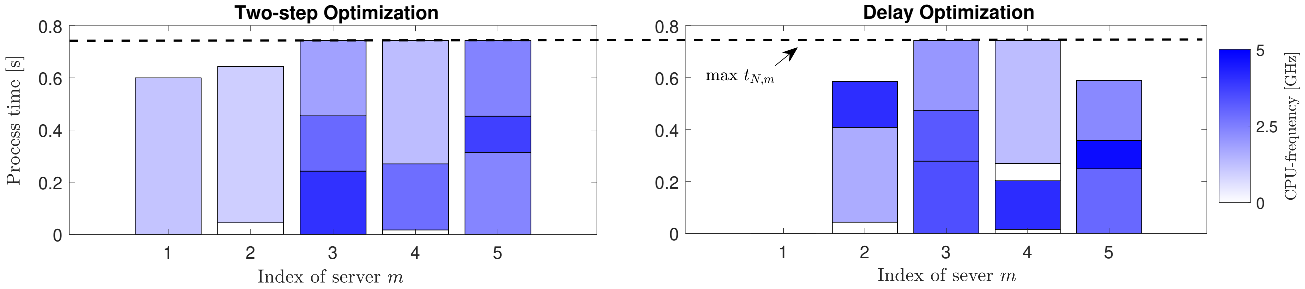

Fig. 2. The processing time of each server. The color represents the corresponding CPU-frequency for executing the tasks.

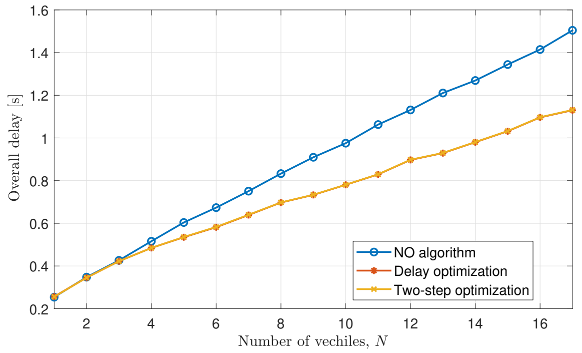

Fig. 3. Overall delay vs number of vehicles under variant algorithms.

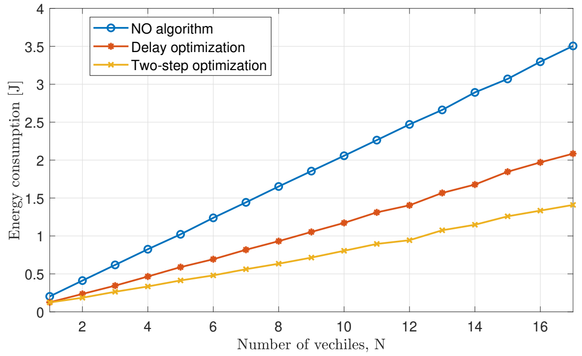

Fig. 4. Energy consumption v.s. number of vehicles N under variant algorithms.

of a complex non-convex problem to minimize the overall delay. Through the analytical insight, we proposed a two-step optimization algorithm that achieves the minimal energy consumption while guaranteeing the optimal delay constraints. Via numerical simulation, we illustrated the key features of our algorithm and compared the system performance with other benchmark schemes, to show the advantage of the proposed algorithm.

It is worth to point out that the approach introduced in this work obtaining the global optimal solution can also be applied in solving other dual-objective mobile offloading problems, e.g., maximizing average offloading rate while minimizing worst case delay.

References

[1] L. Liang, H. Peng, G. Y. Li, X. Shen, "Vehicular communications: A physical layer perspective", *IEEE Trans. Veh. Technol.*,vol. 66, no. 12, pp. 10647-10659, Dec. 2017.

[2] C. Wang, Y. Li, D. Jin *et al.*, "On the serviceability of mobile vehicular cloudlets in a large-scale urban environment," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 10, pp. 2960–2970, 2016.

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

[4] Z. Jiang, S. Zhou, X. Guo, and Z. Niu, "Task replication for deadline constrained vehicular cloud computing: Optimal policy, performance analysis, and implications on road traffic," *IEEE IoT J.*, vol. 5, no. 1, pp. 93–107, 2018.

[5] H. Xing, L. Liu, J. Xu *et al.*, "Joint Task Assignment and Resource Allocation for D2D-Enabled Mobile-Edge Computing," *IEEE Trans. Commun.*, , vol. 67, no. 6, pp. 4193-4207, June 2019.

[6] T. X. Tran and D. Pompili, "Joint Task Offloading and Resource Allocation for Multi-Server Mobile-Edge Computing Networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856-868, Jan. 2019.

[7] K. Zhang, Y. Mao, S. Leng, S. Maharjan and Y. Zhang, "Optimal delay constrained offloading for vehicular edge computing networks," in *ICC*, Paris, 2017, pp. 1-6.

[8] J. Du, F. R. Yu, X. Chu*et al.*, "Computation Offloading and Resource Allocation in Vehicular Networks Based on Dual-Side Cost Minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1079-1092, Feb. 2019.

[9] N. Kumar, S. Zeadally and J. J. P. C. Rodrigues, "Vehicular delay-tolerant networks for smart grid data management using mobile edge computing," *IEEE Commun. Mag.* , vol. 54, no. 10, pp. 60-66, Oct. 2016.

[10] Z. Zhou, P. Liu, Z. Chang *et al.*, "Energy-efficient workload offloading and power control in vehicular edge computing," *2018 IEEE WCNC*, Barcelona, 2018, pp. 191-196.

[11] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, no. 5, pp. 2795–2808, 2016.

[12] E. El Haber, T. M. Nguyen, C. Assi and W. Ajib, "Macro-Cell Assisted Task Offloading in MEC-Based Heterogeneous Networks With Wireless Backhaul," in *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1754-1767, Dec. 2019.

[13] C. Wang, F. R. Yu, C. Liang *et al.*, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7432–7445, 2017.

[14] Y. Dai, D. Xu, S. Maharjan and Y. Zhang, "Joint Load Balancing and Offloading in Vehicular Edge Computing and Networks," in *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4377-4387, June 2019.

[15] C. Haslett, "Essentials of Radio Wave Propagation", *Cambridge University Press*, New York, 2008.

[16] W. Zhang, Y. Wen, K. Guan *et al.*, "Energy optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, 2013.

[17] M. Grant, S. Boyd, "CVX: Matlab software for disciplined convex programming", version 2.0 beta. http://cvxr.com/cvx, September 2013.

[18] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual" http://www.gurobi.com, 2018.