# Profit Maximization for Cache-Enabled Vehicular Mobile Edge Computing Networks

Wenqi Zhou, Junjuan Xia, Fasheng Zhou, Lisheng Fan, Xianfu Lei, Arumugam Nallanathan, and George K. Karagiannidis

*Abstract*—**In this paper, we investigate a multiuser cache-enabled vehicular mobile edge computing (MEC) network, where one edge server (ES) has some caching and computing capabilities to assist the task computing from the vehicular users. The introduce of caching into the MEC network significantly affects the system performance such as the latency, energy consumption and profit at the ES, which imposes a critical challenge on the system design and optimization. To solve this challenge, we firstly design the vehicular MEC network in a non-competitive environment by maximizing the profit of the ES with a predetermined threshold of user QoE, and jointly exploit the caching and computing resources in the network. We then model the optimization problem into a binary integer programming problem, and adopt the cross entropy (CE) method to obtain the effective offloading and caching decision with a low complexity. Simulation results are finally presented to verify that the proposed scheme can achieve the near optimal performance of the conventional branch and bound (BnB) scheme, while sharply reduce the computational complexity compared to the BnB.**

*Index Terms*—**Mobile edge computing, edge caching, computation offloading, profit maximization, cross entropy**

## I. Introduction

With the rapid development of communication systems, an ever-increasing number of mobile devices and Internet data have brought a critical challenge on massive communication and computing. To solve this challenge, one promising technique is mobile edge computing (MEC), which starts from cloud computing, and evolves to deploy computational resources to the nodes nearby the users. This can help reduce the communication and computation overhead significantly. In the MEC networks, the quality of experience (QoE) of users mainly depends on the latency, energy consumption and payment, which has been extensively studied in the literature [1]–[5].

Profit is an important performance of metric for resource providers in MEC networks, which can help encourage servers providing edge computing services, and enhance the users' QoE [6]–[13]. In this direction, the authors in [6], [7] investigated how to maximize the profit of the MEC networks through the computational capability allocation among users, where the users' QoE was taken into account by considering different system factors such as the latency, energy consumption, and user payment. In addition, a three-tire cloud computing architecture was investigated in [8], where the profit was maximized for the cloud provider by optimizing the request scheduling under the constraint of the user service price and service latency. Moreover, the authors in [9], [10] investigated the cloud computing system and maximized the profit of cloud brokers, where the pricing strategy was devised by jointly taking into account the diversity and cost in the users' QoE. In further, the profit maximization problem of the cloud service provider was studied in [11], [12], where the fund allocation and server configuration were optimized with the constraint on the service time of users.

To further enhance the performance of the MEC networks, caching is introduced into the system by pre-storing popular contents close to the users to reduce the overhead of the communication and computation. It is of vital importance to jointly optimize the caching and computing services in the cache-enabled MEC networks, in order to enhance the system performance [14]–[17]. In this direction, S. Bi *et.al* studied how to devise the cache-enabled MEC networks by maximizing the QoE of users [14]. The user utility was maximized in the cache-enabled MEC networks, through studying the problem of joint service caching and task offloading under the constraint of the storage, computation and user budget [15]. In addition, the authors in [16], [17] studied the cache-enabled Internet of Vehicles (IoV) edge computing network to minimize the system latency, where the computation offloading and caching were jointly investigated. So far, there has been little work on the profit maximization in the cache-enabled MEC networks, where the joint impact of caching and computing should be fully taken into account. This motivates the work in this paper.

In this paper, we study a multiuser cache-enabled vehicular MEC network with one edge server (ES), where the ES can assist the task computation of the vehicular users through its caching and computing resources. For this system, we firstly design the network in a non-competitive environment with the goal of maximizing the profit of the ES, while satisfying the QoE of users. And then, the maximization problem is modeled into a binary integer programming problem, and we adopt cross entropy (CE) method to solve the problem of offloading and caching decisions with a low complexity. Finally, simulations are conducted to demonstrate the superiority of the proposed scheme over the traditional ones.

W. Zhou, J. Xia, F. Zhou and L. Fan are with the School of Computer Science, Guangzhou University, Guangzhou, China (e-mail: 2112006056@e.gzhu.edu.cn, {xiajunjuan,zfs,lsfan}@gzhu.edu.cn).

X. Lei is with the Provincial Key Lab of Information Coding and Transmission, Southwest Jiaotong University, Chengdu 610031, China, and also with National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China (e-mail: xflei@swjtu.edu.cn).

A. Nallanathan is with the School of Electronic Engineering and Computer Science, Queen Mary University of London, London, U.K (e-mail:a.nallanathan@qmul.ac.uk).

George K. Karagiannidis is with the Wireless Communications and Information Processing (WCIP) Group, Electrical and Computer Engineering Department, Aristotle University of Thessaloniki, 54 124 Thessaloniki, Greece (e-mail: geokarag@auth.gr).

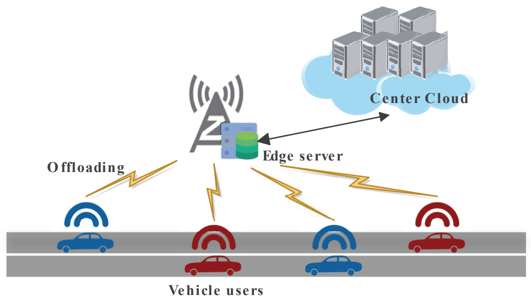

Fig. 1. System model of cache-enabled vehicular MEC networks.

## II. System Model

Fig. 1 depicts the system model of a cache-enabled vehicular MEC network, where there is one ES surrounded by $M$ vehicular users $\{u_m | 1 \leq m \leq M\}$, and the ES can download resources from the center cloud through a wired link. These $M$ users have some computational tasks that need to be executed by in-car applications. As the users are generally of limited computational capability, the ES with a powerful computational capability should help compute the tasks through wireless links from the users to the ES. Moreover, the ES is equipped with a storage space of $\Omega$, which can pre-store some applications in order to speed up the computation. Specifically, if some frequently used applications have already been pre-stored in the ES, the user does not need to offload these applications[1] but to send the contextual information (e.g., the user related parameters, which are negligible compared to the application itself.) and wait for the corresponding task computational result from the ES. Otherwise, the vehicle needs to offload the applications for the task execution or compute the task locally. In the following part, we detail the caching model as well as the computation and communication model.

### A. Caching Model

Let $\mathcal{I}$ denote the set of applications in the network, and there are $I$ applications in total. Due to the limited storage space, only a part of applications can be cached in the ES. The caching strategy should take into account some factors, such as the application popularity, caching cost and so on. For the $i$-th application, we use $x_i$ to present the associated caching decision, given by

$$x_i = \begin{cases} 1, & \text{If application } i \text{ is cached in the ES,} \\ 0, & \text{Else.} \end{cases} \tag{1}$$

The constraint in the storage space at the ES is

$$\sum_{i=1}^{I} x_i \omega_i \leq \Omega. \tag{2}$$

where $\omega_i$ is the application size in bits. Without loss of generality, we use the Zipf distribution [18] to model the application popularity, given by

$$p_i = \frac{i^{-\gamma}}{\sum_{i_1=1}^{I} {i_1}^{-\gamma}}, \tag{3}$$

[1]Vehicles can outsource the in-car applications to the ES for the task execution [17].

where $p_i$ is the popularity of the $i$-th application, and $\gamma > 0$ is an essential parameter of Zipf distribution.

### B. Computation and Communication Model

As mentioned before, if the application is not cached in the ES, then the task has to be computed locally, or computed by the ES through offloading. Let $\mathcal{I}_m$ denote the application set requested by the $m$-th user to compute their tasks, where $\mathcal{I}_m \subseteq \mathcal{I}$. When tasks needed to be computed locally, the local latency and energy consumption of user $m$ can be written as

$$L_m^{local} = \sum_{i \in \mathcal{I}_m} (1 - x_i)(1 - y_{m,i}) \frac{\omega_i \zeta}{f_m}, \tag{4}$$

$$E_m^{local} = \sum_{i \in \mathcal{I}_m} (1 - x_i)(1 - y_{m,i}) \omega_i f_m^2 \zeta \epsilon_m, \tag{5}$$

where $y_{m,i}$ is the binary offloading decision, in which $y_{m,i} = 0$ corresponds to the local computation while $y_{m,i} = 1$ indicates the full offloading of the $i$-th application from user $u_m$ to the ES for the task computing. Notation $\zeta$ is the computational workload, $f_m$ is the computational capability of user $m$, and $\epsilon_m$ is the energy consumption coefficient of user $m$. When tasks are executed through the offloading, the transmission latency and energy consumption of user $m$ are

$$L_m^{trans} = \sum_{i \in \mathcal{I}_m} (1 - x_i) y_{m,i} \frac{\omega_i}{r_m}, \tag{6}$$

$$E_m^{trans} = \sum_{i \in \mathcal{I}_m} (1 - x_i) y_{m,i} \frac{\omega_i}{r_m} P_m, \tag{7}$$

where $P_m$ is the transmit power of user $m$, and $r_m$ is the transmission data rate,

$$r_m = B_m \log_2 \left(1 + \frac{P_m |h_m|^2}{\sigma^2}\right), \tag{8}$$

in which $B_m$ is the wireless channel bandwidth of user $m$, $h_m \sim \mathcal{CN}(0, \xi)$ is the channel parameter of the link from user $m$ to the ES, and $\sigma^2$ is the variance of additive white Gaussian noise (AWGN) at the ES. After offloading, the ES can compute the tasks in parallel, where virtual machines are created to uniformly allocate the computational capability to the users. Accordingly, the latency and energy consumption of computing the tasks from user $m$ at the ES are

$$L_m^{es} = \sum_{i \in \mathcal{I}_m} (1 - x_i) y_{m,i} \frac{\omega_i \zeta}{f_{es}/M}, \tag{9}$$

$$E_m^{es} = \sum_{i \in \mathcal{I}_m} (1 - x_i) y_{m,i} (f_{es}/M)^2 \omega_i \zeta \epsilon_{es}, \tag{10}$$

where $f_{es}$ is the computational capacity of the ES, and $\epsilon_{es}$ is the associated energy consumption coefficient. From (9) and (10), we can obtain the total computational latency and energy consumption of the ES as,

$$L^{es} = \max(L_1^{es}, L_2^{es}, ..., L_M^{es}), \tag{11}$$

$$E^{es} = \sum_{m=1}^{M} E_m^{es}. \tag{12}$$

Moreover, the total latency and energy consumption of user $m$ are

$$L_m = \max(L_m^{local}, L_m^{trans} + L_m^{es}), \tag{13}$$

$$E_m = E_m^{local} + E_m^{trans}. \tag{14}$$

## III. Problem Formulation and Profit Maximization

In this section, we firstly formulate the offloading and caching problem to maximize the server profit while guaranteeing the QoE of users, and then employ the CE method to obtain the offloading and caching decision. We further discuss the computational complexity of the CE-based method.

### A. Problem Formulation

When the server helps accomplish the computational tasks from the users, it should be awarded some profit due to user payment, and the profit award from user $m$ is

$$R_m = \beta_1 \sum_{i\in\mathcal{I}_m} (1-x_i) y_{m,i}\omega_i + \beta_2 \sum_{i\in\mathcal{I}_m} x_i\omega_i, \tag{15}$$

in which $\beta_1$ and $\beta_2$ are the price coefficients (Unit: cents/Mbits) of computing and caching at the ES, respectively. The total profit award of the ES is

$$\lambda_{award} = \sum_{m=1}^{M} R_m. \tag{16}$$

Meanwhile, to assist the computation, the server has to suffer some profit loss due to computing and pre-storing the applications. Specifically, the profit loss of the server due to computing depends on the computational energy consumption and latency, given by

$$\lambda_{comp} = \eta_e E^{es} + \eta_l L^{es}, \tag{17}$$

where $\eta_e$ and $\eta_l$ are the economic factors (Unit: cents/J, cents/s) of the energy consumption and latency, which turn the energy consumption and latency into the economic sense. In addition, the ES also suffers some profit loss due to the caching, depending on the size of the application,

$$\lambda_{cache} = \eta_s \sum_{i\in\mathcal{I}}^{I} x_i\omega_i, \tag{18}$$

where $\eta_s \geq 0$ is the economic factor (cents/Mbits) for caching.

Note that the above $\beta_1$, $\beta_2$, $\eta_e$, $\eta_l$, and $\eta_s$ should be set according to the specific application scenarios and requirements. Take $\beta_1$ as an example to explain the impact of these coefficients on the system profit. If $\beta_1$ is set too high, users may be reluctant to offload tasks to the ES due to high payment, which may lead to an decreased profit at the ES. On the contrary, if $\beta_1$ is set too small, the ES may suffer economic loss due to little revenue. Hence, the setting of these economic coefficients including $\beta_1$ should jointly take into account the user payment and the ES profit.

By jointly taking into account the award and loss, we obtain the overall profit of the server (Unit: cents) as

$$\lambda_{ES} = \lambda_{award} - \lambda_{comp} - \lambda_{cache}. \tag{19}$$

On the other hand, by jointly considering the latency, energy consumption and the computational payment, we can write the QoE of user $m$ as

$$U_m = \eta_e E_m + \eta_l L_m + \eta_p R_m, \tag{20}$$

where $\eta_p \in [0,1]$ is a factor to indicate the importance of the price for users. For the entire system, we can formulate the system design by maximizing the server profit and guaranteeing the QoE of users meanwhile, given by

$$\begin{aligned} \mathbf{P}:\ & \max_{\{x_i, y_{m,i}\}} \lambda_{ES} \\ s.t.\, & C_1: U_m < U_{th}, \forall m \in [1,M], \\ & C_2: x_i \in \{0,1\}, \forall i \in \mathcal{I}, \\ & C_3: \sum_{i=1}^{I} x_i\omega_i \leq \Omega, \\ & C_4: y_{m,i} \in \{0,1\}, \forall m \in [1,M], \forall i \in \mathcal{I}_m, \end{aligned} \tag{21}$$

where $U_{th}$ is a predetermined threshold of the QoE, and the optimization variables come from the caching variables $\{x_i | i \in \mathcal{I}\}$, and the offloading variables $\{y_{m,i} | 1 \leq m \leq M, i \in \mathcal{I}_m\}$.

Note that the above optimization problem is a binary integer programming problem, which is hard to be solved by the conventional method such as convex optimization. Although it can be solved by the branch-and-bound (BnB) algorithm, its huge complexity makes it difficult for practical applications. In many practical vehicular MEC networks, vehicles have to complete the task within the timescale of millisecond, in order to give a quick response on the vehicle operation. To meet this requirement, we turn to use the CE method to solve the problem, which is a heuristic algorithm and widely used in integer nonlinear programming problems thanks to its advantages of implementation simplicity and fast convergence [19]–[21].

### B. CE-Based Offloading and Caching Scheme

In this part, we employ the CE algorithm to solve the binary caching and offloading decision in (21). Let $J = I + \sum_{m=1}^{M} I_m$, where $I_m$ is the number of the applications in $\mathcal{I}_m$, and a $J$-dimensional vector $\boldsymbol{z} = [z_1, z_2, ..., z_J]$ can be constituted, where it consists of $\{x_i | i \in \mathcal{I}\}$ and $\{y_{m,i} | m \in [1,M], i \in \mathcal{I}_m\}$. In particular, $z_j \in \{0,1\}$ for $j \in [1,J]$, and the first $I$ elements represent the caching decision, while the rest elements denote the offloading decision. The idea of using the CE algorithm to solve the problem in (21) lies in that we try to learn a distribution of caching and offloading decision, which is close to the true distribution of the optimal decision. Specifically, we use $q(\boldsymbol{z}, \boldsymbol{\mu})$ to denote the true distribution of the optimal caching and offloading decision, and $g(\boldsymbol{z}, \boldsymbol{\mu})$ represents a theoretically-tractable distribution that needs to be learned, where $\boldsymbol{\mu} = [\mu_1, \mu_2, ..., \mu_J]$ denotes the mean of the distribution. For the two distributions, the cross entropy is given by

$$H(q,g) = -\sum q(\boldsymbol{z}, \boldsymbol{\mu}) \ln g(\boldsymbol{z}, \boldsymbol{\mu}), \tag{22}$$

which represents the distance between the two distributions. From $H(q,g)$, we aim to learn $g(\boldsymbol{z}, \boldsymbol{\mu})$ by iterative training, in order to minimize the cross entropy. For the binary integer programming problem in (21), we firstly choose Bernoulli distribution as a feasible solution of $g(\boldsymbol{z},\boldsymbol{\mu})$, expressed by

$$g(\boldsymbol{z},\boldsymbol{\mu}) = \prod_{j=1}^{J} \mu_j^{z_j}(1-\mu_j)^{1-z_j}, \tag{23}$$

where $u_j \in [0,1]$ is the mean of element $z_j$, i.e., $\Pr(z_j = 1) = u_j$. We then generate some samples of caching and offloading decision for the learning, according to the distribution $g(\boldsymbol{z},\boldsymbol{\mu})$. Let $\mathcal{N}$ denote the set of feasible samples. In iteration $k \in [1,K]$, we repeatedly generate the sample $\boldsymbol{z}_n$ according to (23), where $n \in \mathcal{N}$. In detail, the element $z_j$ in $\boldsymbol{z}_n$ is parallelly generated according to the Bernoulli distribution of $\Pr(z_j = 1) = u_j$ and $\Pr(z_j = 0) = 1 - u_j$. Meanwhile, the sample $\boldsymbol{z}_n$ that does not satisfy the constraint $C_1$ and $C_3$ in (21) is removed during the process of producing samples. After we obtain $N$ samples for the set $\mathcal{N}$, the sample generation at iteration $k$ is completed. Based on $\mathcal{N}$, we further learn the $\boldsymbol{\mu}$ of $g(\boldsymbol{z},\boldsymbol{\mu})$ to minimize the cross entropy. As the decision samples in the set $\mathcal{N}$ are independent, we can obtain $q(\boldsymbol{z},\boldsymbol{\mu}) = \frac{1}{N}$, and write the minimum of cross entropy as

$$\min H(q,g) = \max \frac{1}{N}\sum_{n=1}^{N} \ln g(\boldsymbol{z}_n,\boldsymbol{\mu}). \tag{24}$$

As the optimal caching and offloading decision in (21) is to maximize $\lambda_{ES}$, the feasible samples of the decision in $\mathcal{N}$ are sorted in descending order according to the value of $\lambda_{ES}$ at each iteration. The first $N_{elite}$ decisions are selected as elite decisions to update the parameters $\boldsymbol{\mu}$, which is written as

$$\boldsymbol{\mu}^* = \arg\max_{\boldsymbol{\mu}} \frac{1}{N}\sum_{n=1}^{N_{elite}} \ln g(\boldsymbol{z}^{(n)},\boldsymbol{\mu}), \tag{25}$$

where $\boldsymbol{z}^{(n)}$ indicates the $n$-th elite decision. According to [20], we can obtain the value of $\boldsymbol{\mu}^*$ at iteration $k$, where the element $\mu_j^*$ of $\boldsymbol{\mu}^*$ is obtained by

$$\mu_j^* = \frac{1}{N_{elite}}\sum_{n=1}^{N_{elite}} z_j^{(n)}. \tag{26}$$

We update $\boldsymbol{\mu}_{(k)}$ based on $\boldsymbol{\mu}_{(k-1)}$ and $\boldsymbol{\mu}^*_{(k)}$ of iteration $k-1$ and $k$ by

$$\boldsymbol{\mu}_{(k)} = b\boldsymbol{\mu}^*_{(k)} + (1-b)\boldsymbol{\mu}_{(k-1)}, \tag{27}$$

where $b \in [0,1]$ is the learning rate, which can be set in the range of $[0.4, 0.9]$ to achieve a fine result [21]. After $K$ iterations, we can finally obtain the estimate of the caching and offloading decision. The overall CE-based learning algorithm is summarized in Algorithm 1.

### C. Complexity Analysis

In this part, we provide some analysis on the computational complexity of the CE method. For the $J$-dimensional binary integer programming problem, CE method needs to update the parameters of the $J$ elements in $K$ iterations. Therefore, the computational complexity of CE method is $\mathcal{O}(JK)$. In contrast, the computational complexity of the conventional BnB is close to that of the exhaustive method, as it is performed sequentially. Although pruning can be carried out in the process of BnB to reduce the complexity, its computational complexity is still about $\mathcal{O}(2^J)$. Obviously, the computational complexity of CE method is much lower than that of BnB, which is about $\frac{JK}{2^J}$ of the BnB. Moreover, BnB requires a large memory for storage, especially when the problem dimension increases. Therefore, CE algorithm shows more advantages with the increase of the problem dimension $J$, and it is much more readily to be implemented in a parallel way.

**Algorithm 1** CE-based joint offloading and caching scheme

```
Input: $\mathcal{N} = []$, $\boldsymbol{\mu}_{(0)} = 0.5 \times \mathbf{1}_{J\times 1}$
 1: for $k = 1 : K$ do
 2:     while $|\mathcal{N}| < N$ do
 3:         generate $\boldsymbol{z}_n$ based on $g(\boldsymbol{z},\boldsymbol{\mu})$ under constraints $C_1$
 4:         and $C_3$
 5:     end while
 6:     Calculate the function (19) of $N$ samples
 7:     Sort $\{\lambda_{ES}(\boldsymbol{z}_n)\}_{n=1}^{N}$
 8:     Select the first $N_{elite}$ samples as elites
 9:     $\lambda_{ES}^{max} = \text{Average}\{\lambda_{ES}(\boldsymbol{z}_n)\}_{n=1}^{N_{elite}}$
10:     Update $\boldsymbol{\mu}_{(k+1)}$
11: end for
Output: $\lambda_{ES}^{max}$, $\boldsymbol{z}$
```

## IV. Simulation Results and Discussions

In this part, we provide some simulation results to verify the proposed studies. If not specified, there are 500 applications in the network which are of the same size, and each user requests 10 applications for the task computing, according to the Zipf distribution with $\gamma = 2$. The size of the application is 80Mbits, and the storage space is 30Gbits. Moreover, the wireless links in the network experience Rayleigh block fading with the average channel gain of unity, and the wireless bandwidth of each user is 40MHz. The transmit power of users is 2W, and the variance of AWGN is $\sigma^2 = 1 \times 10^{-2}$W. In further, the computational workload $\zeta$ is set to 2cycle/bit, and the CPU cycle frequencies of the users and ES are set to 400MHz and 800MHz, respectively. The energy consumption coefficients of the users and ES are set to $1 \times 10^{-26}$ and $1 \times 10^{-28}$, respectively. Furthermore, the price coefficients $\beta_1$ and $\beta_2$ are set to 1cent/Mbits and 5cents/Mbits, respectively. The economic factors $\eta_e$, $\eta_l$, and $\eta_s$ are set to 100cents/J, 100cents/s, and 10cents/Mbits, respectively. Such setting of economic coefficients can make the award and loss of the ES at the same magnitude, which can help analyze the impact of the system parameters on the profits. The profit of the ES in the simulation is in $10^4$.

In order to show the performance of the proposed CE-based method, we demonstrate the convergence of the proposed CE-based scheme in a small-scale experiment, and give the profit of the BnB as a benchmark for comparison, where there are two users in the network, the total applications is 20, and the storage space is 1200Mbits. Fig. 2 shows the convergence of the proposed scheme versus the number of

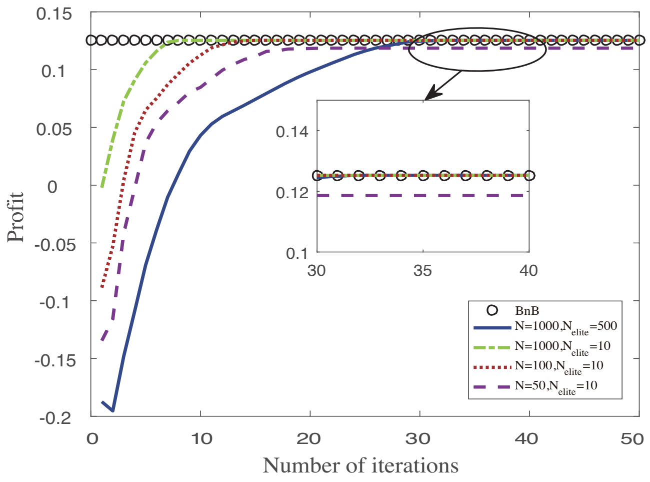

Fig. 2. Profit of the proposed CE-based scheme versus the number of iterations.

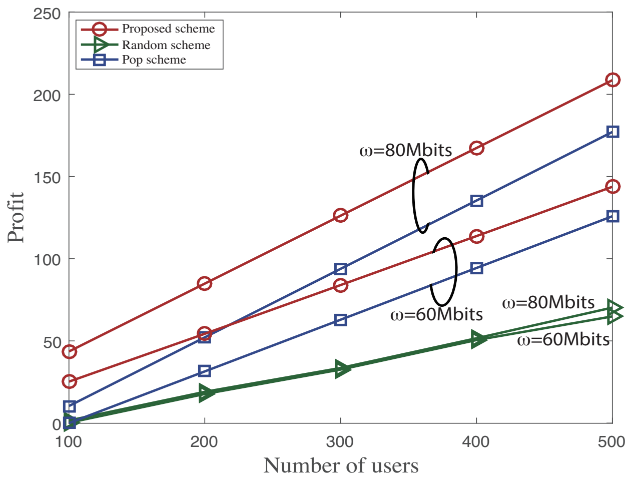

Fig. 4. Impact of the user number on the profit of the three schemes.

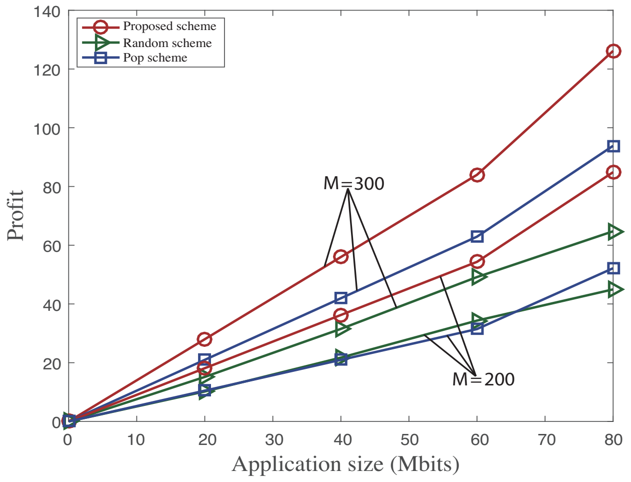

Fig. 3. Profit of the three schemes versus the application size.

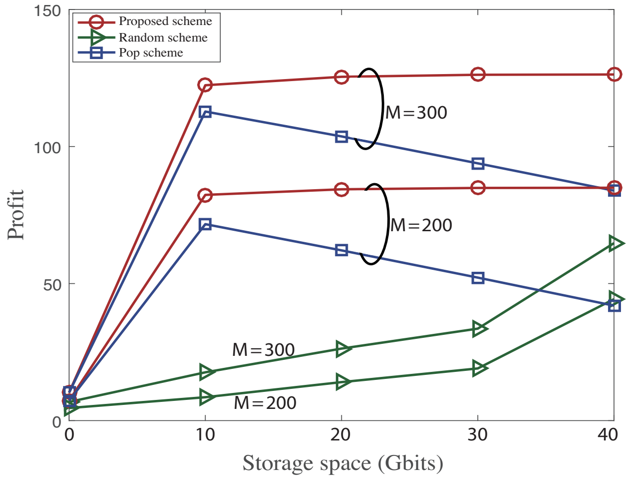

Fig. 5. Effect of the storage space on the profit of the three schemes.

iterations under different settings of $N$ and $N_{elite}$, where the number of iteration varies from 0 to 50. We can observe from this figure that for different settings of $N$ and $N_{elite}$, the proposed scheme can converge to the optimal or near optimal profit of BnB, and the convergence speed is affected by the specific values of $N$ and $N_{elite}$. In particular, the convergence rate is fast when $N_{elite}$ is small with respect to $N$, and the proposed CE-based method is much easier to converge with a larger $N$ at the cost of the increased complexity. Therefore, the proposed method can provide a flexible tradeoff between the performance and complexity by using different values of $N$ and $N_{elite}$. Moreover, the computational complexity of the BnB is about $\mathcal{O}(2^{40})$, while that of the CE based scheme is about $\mathcal{O}(40 \times 50)$, which is about $0.2\%$ of the conventional BnB algorithm. These results indicate that the CE based scheme can achieve the near-optimal performance with a significantly reduced complexity.

Fig. 3 depicts the profit of the proposed scheme versus the application size, where there are 200 or 300 users in the network and the size of the application varies from 0 to 80Mbits. For comparison, we plot the profits of the random scheme which randomly selects the offloading and caching decisions, and we also plot the profit of the popularity based caching scheme (pop scheme) which always selects the most popular applications to cache. From this figure, we can see that the profit of the three schemes increases with a larger application size, as more payment is incurred from the caching and computing. Moreover, the proposed scheme is superior to the other two schemes. In particular, when the application size is 80Mbits and the user number is 300, the profit of the proposed scheme is about $25.6\%$ and $48.7\%$ higher than that of the pop scheme and random scheme, respectively. This is because that the proposed scheme can effectively select some applications for caching and offloading, while the other two schemes fail. In further, for the three schemes, the profit with $M$=300 is higher than that with $M$=200, as more users result in a larger payment. The results in Fig. 3 attests the effectiveness of the proposed scheme.

Fig. 4 portrays the impact of the user number on the profit of three schemes, where the user number varies from 100 to 500 and the application size is 60Mbits or 80Mbits. From this figure, we can observe that the profit of the three schemes increases with a larger number of users, as more users result in a larger number of requests, and the payment to the ES increases accordingly. Moreover, for various values of $M$, the proposed scheme is shown to outperform the other two schemes. In particular, when the number of users is 500 and application size is $80$Mbits, the profit of the proposed scheme is about $66.3\%$ and $15.1\%$ better than that of the random scheme and pop scheme, respectively. This indicates that the proposed scheme can reasonably decide which applications to be cached and offloaded. On the contrary, the other two schemes can not make the decision flexibly, which results in a lower profit. In further, when the application size is 60 Mbits, the profit of the three schemes is lower than that with 80 Mbits, as a larger application size leads to more payment. The results in Fig. 4 verify the superiority of the proposed scheme in large-scale networks.

Fig. 5 illustrates the effect of the storage space on the profit of the three schemes, where there are 200 or 300 users and the storage space of the ES changes from 0 to 40Gbits. We can observe from this figure that the profit of the proposed scheme increases with the storage space when the space is in the low region. This is because that caching can help improve the computation and meet the QoE of users. However, when the space is larger than $10$Gbits, the profit of the proposed scheme remains almost unchange. In contrast, the profit of the pop scheme decreases with the space. This is because that, unlike the other two schemes, the proposed scheme always selects an appropriate caching decision to maximize the profit of ES, rather than caches more applications as the storage space increases. Therefore, the profit of the proposed scheme can outperform the other schemes. In particular, when the storage space is $40$Gbits and user number is 300, the profit of the proposed scheme improves by $33.6\%$ and $48.7\%$ over that of the pop scheme and random scheme, respectively. In further, the profit of the three schemes with $M$=300 is higher than that of the schemes with $M$=200. These results in Fig. 5 further demonstrate that the proposed scheme is effective in large-scale networks.

## V. Conclusions

In this paper, we investigated a cache-enabled vehicular MEC network, and designed the network in a non-competitive environment by maximizing the profit of the ES while satisfying the QoE of users. The optimization problem was modeled into a binary integer programming problem, and the CE method was used to obtain the effective offloading and caching decisions with a low complexity. Simulation results were finally provided to demonstrate the effectiveness of the proposed scheme. In particular, the proposed scheme can achieve almost the same performance as the conventional BnB algorithm, while sharply reduce the computational complexity compared to the BnB. In future works, we will study the cache-enabled MEC networks with multiple servers, and investigate the joint impact of caching and collaboration among servers on the system profit.

## References

[1] Q. Gu, Y. Jian, G. Wang, and R. Fan, "Mobile edge computing via wireless power transfer over multiple fading blocks: An optimal stopping approach," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 10 348–10 361, 2020.

[2] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. Wirel. Commun.*, vol. 17, no. 3, pp. 1784–1797, 2018.

[3] J. Kang, X. Li, J. Nie, Y. Liu, M. Xu, Z. Xiong, D. Niyato, and Q. Yan, "Communication-efficient and cross-chain empowered federated learning for artificial intelligence of things," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 2966–2977, 2022.

[4] F. Zhou and R. Q. Hu, "Computation efficiency maximization in wireless-powered mobile edge computing networks," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 5, pp. 3170–3184, 2020.

[5] J. Kang, Z. Xiong, X. Li, Y. Zhang, D. Niyato, C. Leung, and C. Miao, "Optimizing task assignment for reliable blockchain-empowered federated edge learning," *IEEE Trans. Veh. Technol.*, vol. 70, no. 2, pp. 1910–1923, 2021.

[6] X. Huang, B. Zhang, and C. Li, "Platform profit maximization on service provisioning in mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 13 364–13 376, 2021.

[7] Q. Wang, S. Guo, J. Liu, C. Pan, and L. Yang, "Profit maximization incentive mechanism for resource providers in mobile edge computing," *IEEE Trans. Serv. Comput.*, vol. 15, no. 1, pp. 138–149, 2022.

[8] P. Cong, G. Xu, J. Zhou, M. Chen, T. Wei, and M. Qiu, "Personality- and value-aware scheduling of user requests in cloud for profit maximization," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 1991–2004, 2022.

[9] P. Cong, Z. Zhang, J. Zhou, X. Liu, Y. Liu, and T. Wei, "Customer adaptive resource provisioning for long-term cloud profit maximization under constrained budget," *IEEE Trans. Parallel Distributed Syst.*, vol. 33, no. 6, pp. 1373–1392, 2022.

[10] J. Mei, K. Li, Z. Tong, Q. Li, and K. Li, "Profit maximization for cloud brokers in cloud computing," *IEEE Trans. Parallel Distributed Syst.*, vol. 30, no. 1, pp. 190–203, 2019.

[11] K. Li, J. Mei, and K. Li, "A fund-constrained investment scheme for profit maximization in cloud computing," *IEEE Trans. Serv. Comput.*, vol. 11, no. 6, pp. 893–907, 2018.

[12] J. Mei, K. Li, and K. Li, "Customer-satisfaction-aware optimal multi-server configuration for profit maximization in cloud computing," *IEEE Trans. Sustain. Comput.*, vol. 2, no. 1, pp. 17–29, 2017.

[13] J. Mei, K. Li, A. Ouyang, and K. Li, "A profit maximization scheme with guaranteed quality of service in cloud computing," *IEEE Trans. Computers*, vol. 64, no. 11, pp. 3064–3078, 2015.

[14] S. Bi and L. Huang, "Joint optimization of service caching placement and computation offloading in mobile edge computing systems," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 7, pp. 4947–4963, 2020.

[15] X. Pham, T. Nguyen, V. Nguyen, and E. Huh, "Joint service caching and task offloading in multi-access edge computing: A qoe-based utility optimization approach," *IEEE Commun. Lett.*, vol. 25, no. 3, pp. 965–969, 2021.

[16] Z. Ning, K. Zhang, X. Wang, L. Guo, X. Hu, J. Huang, B. Hu, and R. Y. Kwok, "Intelligent edge computing in internet of vehicles: A joint computation offloading and caching solution," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2212–2225, 2021.

[17] C. Tang, C. Zhu, H. Wu, Q. Li, and J. J. P. C. Rodrigues, "Toward response time minimization considering energy consumption in caching-assisted vehicular edge computing," *IEEE Internet Things J.*, vol. 9, no. 7, pp. 5051–5064, 2022.

[18] J. Xia, L. Fan, W. Xu, X. Lei, and X. Chen, "Secure cache-aided multi-relay networks in the presence of multiple eavesdroppers," *IEEE Trans. Commun.*, vol. 67, no. 11, pp. 7672–7685, 2019.

[19] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L'Ecuyer, "The cross-entropy method for optimization," in *Handbook of statistics*. Elsevier, 2013, vol. 31, pp. 35–59.

[20] S. Zhu, W. Xu, L. Fan, K. Wang, and G. K. Karagiannidis, "A novel cross entropy approach for offloading learning in mobile edge computing," *IEEE Wirel. Commun. Lett.*, vol. 9, no. 3, pp. 402–405, 2020.

[21] P. de Boer, D. P. Kroese, and S. Mannor, "A tutorial on the cross-entropy method," *Ann. Oper. Res.*, vol. 134, no. 1, pp. 19–67, 2005.

# Response to Reviewers' Comments: VT-2022-04335

Profit Maximization for Cache-Enabled Vehicular Mobile Edge Computing Networks

March 1, 2023