Optimal Computation Resource Allocation in Vehicular Edge Computing

Shiyu Du[1], Qibo Sun[1(✉)], Jujuan Gu[2], and Yujiong Liu[1]

[1] Beijing University of Posts and Telecommunications, Beijing 100876, China
{dsy,qbsun,yjliu}@bupt.edu.cn
[2] The 54th Research Institute of CETC, Shijiazhuang 050000, China
gujujuan@163.com

Abstract. Vehicular edge computing is proposed as a new promising paradigm that provides cloud computation capabilities in close proximity to vehicles, which can augment the capabilities of vehicles. In this paper, we study the problem of computation resource allocation of edge servers for a vehicular edge computing system. We consider the constraint of limited computation resource of edge servers and vehicles can decide that vehicular applications are locally executed or offloaded to edge servers for execution to minimize the completion time of applications. We model the problem as a Stackelberg game and then prove the existence of Nash equilibrium of the game. Furthermore, we propose an algorithm to compute the Nash equilibrium effectively. Numerical simulation results demonstrate that our proposed algorithm can greatly reduce the average completion time for all applications and outperform the benchmark approaches.

Keywords: Local and Edge Equilibrium Computing · Computation resource allocation · Vehicular edge computing · Game theory · Nash equilibrium

1 Introduction

With the development of Internet of Things and wireless communication technology, vehicles can offer a variety of computation-intensive and delay-sensitive applications, which require massive computation resources. However, the resource-constrained vehicles are difficult to meet the computation resources requirements of applications. Vehicular edge computing (VEC) is proposed as a promising paradigm that provides cloud computation capabilities in close proximity to vehicles [1]. In VEC system, edge cloud resources are deployed on Road Side Units (RSUs). Each vehicle can offload its application to RSUs for execution [2,3].

There has been some research work on VEC. The authors in [4] proposed a contract-based computation resource allocation scheme to enhance the utilities of the mobile edge computing service providers. In [5], the authors proposed a distributed reputation management system, which can impact the resource allocation policy. In [6], the authors developed an energy-efficient computation offloading scheme, which jointly optimized computation offloading decisions. Existing

Z. Zheng et al. (Eds.): BlockSys 2019, CCIS 1156, pp. 422–427, 2020.
https://doi.org/10.1007/978-981-15-2777-7_34

research generally consider that mobile edge computing servers on one RSU have infinite computation resources. However, the edge servers on one RSU always have limited computation resources, which is unable to meet the computation resource requirements of extensive vehicles.

To overcome the above problem, in this paper, we consider the constraints of limited computation resource of edge servers and the interaction between edge cloud resource providers and vehicles. Vehicles can decide that vehicular applications are locally executed or offloaded to edge servers on one RSU for execution to minimize the completion time of applications. We model the problem as a Stackelberg game which vehicles are leaders and the RSU is the follower. Under the vehicle-specific parameter of the optimal computation resource allocation strategy, the original game between vehicles can be transformed into a weighted congestion game. Next, we prove the existence of Nash equilibrium (NE) of the Stackelberg game, propose an algorithm to compute a NE. Finally, we show the numerical simulation results to prove that average completion time of all vehicular applications is greatly reduced using our proposed algorithm.

The rest of the paper is organized as follows. We describe system model and formulate the problem in Sect. 2. We prove the existence of NE of the Stackelberg game and propose an algorithm in Sect. 3. We present the numerical simulation results in Sect. 4. Finally, we conclude the paper in Sect. 5.

2 System Model and Problem Formulation

2.1 System Model

We consider that there are N RSUs located along a unidirectional road, the set of RSUs is denoted as $\mathcal{N} = \{1, 2, \cdots, N\}$. Each RSU is equipped with an edge server. The set of computation capabilities of RSUs can be denoted as $\{F^1, F^2, \cdots, F^N\}$. Based on the limited coverage range of each RSU, the road can be divided into N segments with length $\{L_1, L_2, \cdots, L_N\}$, respectively. The vehicles running within the nth segment can only access to RSU n.

There are M vehicles arriving at the start point of the road with a constant speed v. The set of vehicles is denoted as $\mathcal{M} = \{1, 2, \cdots, M\}$. Each vehicle m, $m \in \mathcal{M}$, has a computation-intensive and delay-sensitive application, which can be described as $T_m = \{x_m, b_m\}$, x_m is the size of input data of application T_m and b_m is the computation resources required to complete application T_m.

For the execution decision, we denote d_m, $d_m \in \{0, 1\}$, as the decision variable, i.e., $d_m = 1$ if vehicle m selects to offload its application T_m to RSU n for execution, and $d_m = 0$ if T_m is locally executed. Denote $\mathbf{d} = (d_m)_{m \in \mathcal{M}}$ as a computing offloading strategy profile of vehicles and denote $O_n(\mathbf{d}) \triangleq \{m | d_m = 1\}$ as the set of vehicles that offload their applications to RSU n.

We consider two computation approaches. For the local computing approach, one vehicle locally executes its application. The time cost of local computing is t_m^{local}. For the edge computing approach, the time cost of T_m by edge computing involves four parts: vehicle travel time, data transmission time, application computation time and result transmission time. Since the size of the computation

result is often small, therefore, it can be neglected. Vehicle travel time of T_m is t_m^{travel}. Data transmission time of T_m is t_m^{send}. The computation time of T_m is t_m^{comp}. Thus, the time cost of edge computing is $t_m^{edge} = t_m^{travel} + t_m^{send} + t_m^{comp}$.

2.2 Problem Formulation

The objective is to identify the decision that minimizes the completion time of each application by executing locally or offloading to RSUs for execution. The time cost that application T_m is executed can be given as $C_m(\mathbf{d}, \mathbf{p}) = I_{0,d_m} t_m^{local} + I_{1,d_m} t_m^{edge}$, where $\mathbf{p}$ is the computation resource allocation of RSU and $I_{r,d_m} = 1$ if $d_m = r$, and $d_m \neq r$ otherwise. Finally, the time cost C of the system can be given as follows

$$C(\mathbf{d}, \mathbf{p}) = \sum_{m \in \mathcal{M}} C_m(\mathbf{d}, \mathbf{p}) \tag{1}$$

We can model the above optimization problem as a multiple-leader common-follower Stackelberg game problem and given as the Vehicular Edge Computation Offloading Game (VECOG). Vehicles are leaders and one RSU is the follower. Given a strategy profile $\mathbf{d}$, the objective of one RSU is to minimize the time cost of the system and denoted as $\min_{\mathbf{p} \geq \mathbf{0}} C(\mathbf{d}, \mathbf{p})$. The objective of each vehicle can be given as $\min_{d_m} C_m(d_m, d_{-m}, \mathcal{P}_n(d_m, d_{-m}))$, where $\mathcal{P}$ is the announced computation resource allocation policy of one RSU and d_{-m} is the strategies of all other vehicles except vehicle m. Furthermore, $\mathcal{P}^*$ is an optimal policy. In this paper, we address where there is a combination for the dynamic game that neither vehicles nor the RSU have an incentive to deviate, i.e., a subgame perfect equilibrium.

3 LEEC Algorithm

We observe that the optimal computation resource allocation policy $\mathcal{P}^*$ that RSU n allocates to $\mathbf{d}$ is a vehicle-specific parameter. Thus, vehicles has different weights of computation resource allocation, the interaction of vehicles can be modeled as a vehicular-specific weighted congestion game, which can be expressed as a tuple $\Gamma(\mathcal{P}) =< \mathcal{M}, (d_m)_{m \in \mathcal{M}}, (C_m)_{m \in \mathcal{M}} >$.

Theorem 1. *Under the optimal computation resource allocation policy $\mathcal{P}^*$ of RSU n, the strategic interaction of the vehicles can be modeled as a resource-dependent congestion game, which the resource-dependent weights are w_m^n. Thus, the time cost that application T_m is executed can be given as $\widetilde{\mathcal{C}}_m(\boldsymbol{d}) = I_{0,d_m} t_m^{local} + I_{1,d_m}(w_m^n w(\boldsymbol{d}) + t_m^{travel} + t_m^{send})$, where $w(\boldsymbol{d}) = \sum_{i \in O_n(\boldsymbol{d})} w_i^n$.*

Proof. Depend on the optimal computation resource allocation policy, the computation time by edge computing is $\widetilde{\mathcal{C}}_m^{edge}(\mathbf{d}) = \sqrt{\frac{b_m}{F^n}} \sum_{i \in O_n(\mathbf{d})} \sqrt{\frac{b_i}{F^n}}$. We define

weight $w_m^n \triangleq \sqrt{\frac{b_m}{F^n}}$ for each tuple, and the time cost by edge computing in strategy profile $\mathbf{d}$ depends on the total weight $w(\mathbf{d})$. Thus, the strategic interaction of the vehicles can be modeled as a resource-dependent congestion game.

The objective of each vehicle is to minimize its time cost. We define the result strategic game $\Gamma(\mathcal{P}^*) =< \mathcal{M}, (d_m)_{m\in\mathcal{M}}, (\widetilde{C}_m)_{m\in\mathcal{M}} >$ as the Optimal Resource Allocation Computation Offloading Game (ORACOG). And if the $ORACOG$ has a pure strategy NE, then the $VECOG$ has an SPE. Furthermore, If there exists an exact potential function for the game, then the game has a pure NE.

Theorem 2. *The optimal resource allocation computation offloading game has an exact potential function defined as* $\phi(\boldsymbol{d}) = \sum_{m\in\mathcal{M}}(\phi_m^{local}(\boldsymbol{d}) + \phi_m^{edge}(\boldsymbol{d}))$, *where* $\phi_m^{edge}(\boldsymbol{d}) = I_{1,d_m}(w_m^n w_{\leq m}(\boldsymbol{d}) + t_m^{travel} + t_m^{send})$, $\phi_m^{local}(\boldsymbol{d}) = I_{0,d_m} t_m^{local}$.

Proof. First, we define two shorthand notations $w_{\leq m}(\mathbf{d}) = \sum_{j\in O_n(\mathbf{d}), j\leq m} w_j^n$ and $w_{>m}(\mathbf{d}) = \sum_{j\in O_n(\mathbf{d}), j>m} w_j^n$. Furthermore, we define $\phi_m = \phi_m^{local}(\mathbf{d}) + \phi_m^{edge}(\mathbf{d})$. Next, we consider strategy profiles $\mathbf{d} = (d_i, d_{-i})$ and $\mathbf{d}^\dagger = (d_i^\dagger, d_{-i})$, vehicle i changes its strategy between local computing and edge computing, i.e., vehicle i offloads T_i to one RSU in $\mathbf{d}$ and T_i is locally executed in $\mathbf{d}^\dagger$. Then, the difference of time cost between $\mathbf{d}$ and $\mathbf{d}^\dagger$ is $\widetilde{\mathcal{C}}_i(\mathbf{d}) - \widetilde{\mathcal{C}}_i(\mathbf{d}^\dagger) = w_i^n w(\mathbf{d}) + t_m^{travel} + t_m^{send} - t_i^{local}$, which is equal to the difference of potential function. Similarity, if vehicle i locally executes T_i in $\mathbf{d}$ and offloads T_i to one RSU in $\mathbf{d}^\dagger$, we can also show that $\phi(\mathbf{d}) - \phi(\mathbf{d}^\dagger) = \widetilde{\mathcal{C}}_i(\mathbf{d}) - \widetilde{\mathcal{C}}_i(\mathbf{d}^\dagger)$.

Thus, there exists an exact potential function for the game, and the $ORACOG$ has a pure NE. Then the $VECOG$ has an SPE. Next, we compute the NE of the $ORACOG$. We denote the proposed algorithms as Local and Edge Equilibrium Computing (LEEC) algorithm. We start the algorithm from the strategy profile that all vehicles decide to locally execute their applications. There are two phases to execute alternatingly in the algorithm. In the first phase, vehicle can offload to one RSU to decrease the completion time of application. In the second phase, vehicles can update their execution decision to find the best replies. Using the algorithm, we can compute the NE of the $ORACOG$ effectively.

4 Performance Evaluations

4.1 Simulation Setup

In this section, based on existing research works [4,7,8], we set various parameters for the numerical simulations. We consider that there is a 100-m road with one RSU, and computation capability of the RSU is assigned from 5 GHz to 10 GHz with a step value of 1 GHz. The number of vehicles is denoted from 5 to 30 with a step value of 5, and they run at the constant speed 120 km/hr. The computation capability of vehicle is assigned from the interval $[0.2, 0.4]$ GHz. The computation resource that each vehicle required is assigned from the interval $[1, 5]$ GHz. The size of input data of the vehicular application is assigned from the interval $[100, 300]$ KB.

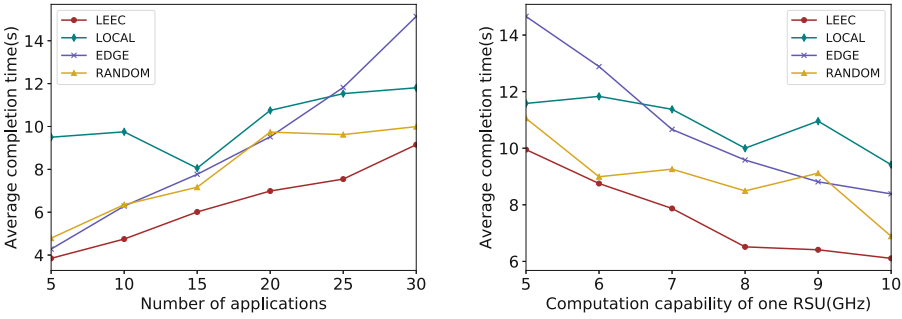
To verify the performance of our proposed algorithm, we introduce three benchmark approaches. For *Local computing* (*LOCAL*) *algorithm*, all vehicles choose to execute their vehicular applications locally. For *Edge computing* (*EDGE*) *algorithm*, all vehicles choose to offload their applications to one RSU for execution. For *Random computing* (*RANDOM*) *algorithm*, vehicles randomly choose to locally execute their applications or offload the applications to one RSU for execution.

4.2 Comparison of Average Completion Time

(1) Impact of Number of Applications: Figure 1(a) shows the impact of the number of vehicular applications. We can observe that the average completion time of all applications is reduced by 60%, 43% and 25% comparing to LOCAL, EDGE and RANDOM. This is because our proposed algorithm considers the local and edge equilibrium computing. Furthermore, as the number of applications grows, the competition for computation resource of the RSU for the vehicles becomes greater. Thus, each vehicle can be allocated less computation resource when the vehicles offload their applications to the same RSU for execution, as a result, the average completion time of the applications is greatly increased.

(2) Impact of Computation Capability of One RSU: Figure 1(b) shows the impact of computation capability of one RSU on the average completion time of all applications. It can be observed that the average completion time of all applications significantly decreases when computation capability of the RSU increases using our proposed algorithm, this is because that there are more computation capability for the vehicles to compute their applications.

From Fig. 1(a) and (b), we can clearly observe that the average completion time of all applications using our proposed algorithm is less than benchmark approaches. Furthermore, the trend of numerical simulation result is much more stable using our proposed algorithm.



(a) Impact of number of applications (b) Impact of computation capability

Fig. 1. Impact of the parameters.

5 Conclusion

In this paper, we study a computation resource allocation problem in a VEC system, which is under the constraint of limited computation resources of edge servers on one RSU. We formulate the computation resource allocation problem as an optimization problem and model the problem as a Stackelberg game which vehicles are leaders and the RSU is the follower. To solve the optimization problem, we propose an effective LEEC algorithm. Numerical simulation results demonstrate that compared with benchmark approaches, our proposed algorithm can greatly reduce the average completion time of all applications.

Acknowledgements. This research is supported by the National Natural Science Foundation of China under Grant 61571066.

References

1. Wang, S., Zhang, X., Zhang, Y., Wang, L., Yang, J., Wang, W.: A survey on mobile edge networks: convergence of computing, caching and communications. IEEE Access **5**, 6757–6779 (2017)
2. Dai, Y., Xu, D., Maharjan, S., Zhang, Y.: Joint load balancing and offloading in vehicular edge computing and networks. IEEE Internet Things J. **6**(3), 4377–4387 (2019)
3. Xiao, L., Zhuang, W., Zhou, S., Chen, C.: Learning while offloading: task offloading in vehicular edge computing network. Learning-based VANET Communication and Security Techniques. WN, pp. 49–77. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-01731-6_3
4. Zhang, K., Mao, Y., Leng, S., Vinel, A., Zhang, Y.: Delay constrained offloading for mobile edge computing in cloud-enabled vehicular networks. In: 2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM), Halmstad, pp. 288–294 (2016)
5. Huang, X., Yu, R., Kang, J., Zhang, Y.: Distributed reputation management for secure and efficient vehicular edge computing and networks. IEEE Access **5**, 25408–25420 (2017)
6. Guo, F., Zhang, H., Ji, H., Li, X., Leung, V.C.M.: An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing. IEEE/ACM Trans. Netw. **26**(6), 2651–2664 (2018)
7. Jošilo, S., Dán, G.: Wireless and computing resource allocation for selfish computation offloading in edge computing. In: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, Paris, France, pp. 2467–2475 (2019)
8. Zhang, K., Mao, Y., Leng, S., Maharjan, S., Zhang, Y.: Optimal delay constrained offloading for vehicular edge computing networks. In: 2017 IEEE International Conference on Communications (ICC), Paris, pp. 1–6 (2017)