QoE-Driven Computation Offloading: Performance Analysis and Adaptive Method

Quyuan Luo*, Weisong Shi†, and Pingzhi Fan*
*School of Information Science and Technology, Southwest Jiaotong University, Chengdu 611756, China
†Department of Computer Science, Wayne State University, Detroit, MI 48202, USA
E-mail: qyluo@swjtu.edu.cn

2021 13th International Conference on Wireless Communications and Signal Processing (WCSP) | 978-1-6654-0785-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/WCSP52459.2021.9613623

***Abstract*—With the proliferation of the Internet of Things (IoT) and the concomitant computation-intensive tasks, the surging demand for computation offloading can be expected. How to optimally make offloading decisions towards best quality of experience (QoE) represents a fundamental research problem. In this paper, considering the different QoE requirements and the inherent characteristic of tasks, we investigate the computation offloading problem in edge computing enabled IoT networks. Particularly, we first establish task local computing model and task offloading model by fully considering the parallel and serial processing characteristics of tasks. Based on a thorough theoretical performance analysis, a QoE-driven adaptive computation offloading (QEACO) strategy is proposed. And users can optimally and adaptively make offloading decisions towards best QoE. Finally, simulation results indicates that QEACO can significantly improve the QoE of users compared to several benchmark schemes.**

***Index Terms*—Internet of things, computation offloading, quality of experience, parallel and serial processing, performance analysis**

I. Introduction

The explosive growth of Internet of Things (IoT) and intelligent devices, e.g., mobile phones and wearable devices, brings the ever-increasing requirements for communication and computing [1]. And a large number of IoT applications have high demand for computing, such as video-assisted real-time navigation in automotive driving, intelligent personal assistants in smart home and artificial intelligence (AI)-based applications in industry [2]. However, how to execute these computation-intensive applications on IoT terminals with limited computational capability faces huge challenges. Initially, traditional cloud computing has been widely regarded as a feasible paradigm. However, the unstable transmission and long latency make it difficult to satisfy the quality of experience (QoE) of users. Therefore, edge computing as a promising solution pushes the computational resources to the edge of the network, e.g., roadside units (RSUs), access points (APs) and base stations (BSs). In this way, computation-intensive applications can be offloaded to the proximity of IoT users to reduce latency [3].

Recent years, many works have focused on computation offloading in edge computing networks for different QoE requirements. For reducing latency, the computation offloading problem combined with content caching of Internet of vehicles in [4] was formulated to minimize the total network latency under long-term energy constraints for RSUs. The authors in [5] jointly considered a multi-hop partial computation offloading and network flow scheduling framework, where the average completion time of all tasks is minimized. And the literature [6] considered to minimize the task execution latency in a device-to-device (D2D)-enabled 5G computation offloading scenario. The literature [7] considered to jointly minimize the delay and cost for vehicular edge computing. For reducing energy consumption of energy-constrained IoT devices, the authors in both [8] and [9] jointly considered the computation offloading and resource allocation problems, with the aim to minimize the weighted sum of terminal energy consumption. Also, utilizing the concept of digital twin, the authors in [10] proposed to minimize the long-term energy efficiency for industrial IoT devices. For some composite performance indexes, the authors in [11] considered that each device tries to minimize its own cost, which is a combination of task completion time and energy consumption. In [12], the computation offloading in vehicular edge computing is formulated as an utility maximization problem, where utility is defined as the offloading task number to the edge server.

Marvelous solutions have been proposed to address the computation offloading problem. Although lots of research and projects have been working on how to offload tasks, how to evaluate the offloading process is still an open question. For example, how long latency the offloading brings, how much acceleration the offloading can be obtained, how much bandwidth the offloading requires, and how much overhead/cost the offloading consumes? In particular, different type of users in IoT networks pursue different QoE performance. For example, the connected and autonomous vehicles prefer a low task processing time during offloading for safety purpose, while the unmanned aerial vehicles (UAVs) prefer an energy-efficiency computation offloading for a long battery life. In this regard, it is imperative to design an adaptive computation offloading strategy based on IoT users' QoE requirements. Moreover, in existing works, the application tasks are directly processed locally or offloaded, ignoring the tasks themselves characterized by serialized and parallelizable computing parts.

Motivated by the discussion above, we aim to propose a QoE-driven adaptive computation offloading (QEACO) strategy for edge computing enabled IoT networks, where the characteristic of tasks is fully considered and each IoT user can optimally and adaptively make its offloading decision based on its QoE requirement. Specifically, we first establish the detailed computation offloading model, where the serialized and parallelizable computing characteristic of tasks is fully considered. Then, we present a thorough performance analysis on IoT users' different QoE requirements for task processing latency, task processing energy consumption, computing acceleration, and computing cost. Thereafter, the QEACO strategy is proposed based on the QoE requirements. The contributions of this paper are summarized as follows.

- *Model*: Considering the serialized and parallelizable processing part of a task, as well as the computing acceleration, we establish the detailed task local computing model and task offloading model.
- *Analysis*: Based on a theoretical performance analysis, we propose a QEACO strategy, where each IoT user can optimally and adaptively make its offloading decision towards its best QoE.
- *Validation*: Based on real-world vehicular traces, extensive simulations demonstrates the effectiveness of our proposed QEACO over several benchmark schemes.

The rest of the paper is organized as follows. We introduce the system model in Section II. The thorough performance analysis and the proposed QEACO strategy are presented in Section III. In Section IV, performance evaluation is presented. Finally, we conclude the paper in Section V.

II. System Model

Fig. 1 illustrates a typical computation offloading scenario in IoT networks, where various tasks would be generated from IoT users. We use three items to describe an arbitrary task T, i.e., $T = \{D, \alpha, c\}$, where D stands for the data size, α ($0 \leq \alpha \leq 1$) stands for the parallelizable fraction[1], and c represents the processing density (in CPU cycles/bit) [3]. The IoT users, such as CAVs and drones, can be connected to the edge through various communication channels (such as 5G and dedicated short range communication (DSRC)). We denote the bandwidth of orthogonal communication channels by B and the number of channels is denoted by M. T can be processed locally, or offloaded to edge, which will be discussed in the following.

A. Task Local Computing Model

We denote the number of users' cores by n_1, and the processing capability (i.e., the amount of CPU frequency in cycles/s) of each core assigned for local computing as f^l, then the power consumption of each core for local computing is expressed as $p^l = \kappa(f^l)^3$, where κ is a coefficient reflecting the relationship between processing capability and power consumption.

According to the Amdahl's law [13], the local computing time of T, which consists of the serialized computing time t_s^l and the parallelizable computing time t_p^l, can be expressed as

$$t_1 = t_s^l + t_p^l = \frac{c(1-\alpha)D}{f^l} + \frac{\alpha c D}{n_1 f^l}. \tag{1}$$

[1]Parallelizable fraction refers to the ratio of the amount of data that can be processed in parallel to the total data volume.

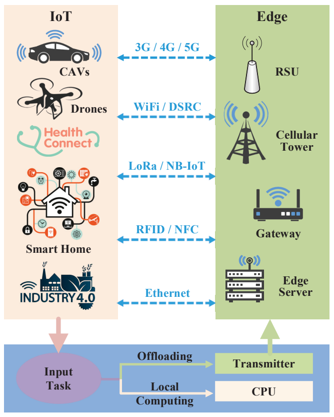


Fig. 1. Computation offloading in IoT networks.

For simplicity, we neglect the time to split and combine the task in this paper. And the energy consumption for local computing is formulated as

$$E^l = p^l t_s^l + n_1 p^l t_p^l = \kappa c D (f^l)^2. \tag{2}$$

For local computing, the cost only comes from the energy consumption, which is formulated as

$$C_l = \varrho E^l = \varrho \kappa c D (f^l)^2, \tag{3}$$

where ϱ is a weight coefficient indicating the energy consumption cost of one unit energy.

B. Task Offloading Model

We denote the path loss by $d^{-\vartheta}$, where d denotes the distance from the user to the edge server, and ϑ denotes the path loss exponent [14]. Based on Shannon's formula, when data is offloaded from the user to the edge server through a communication channel, the transmission rate can be expressed as $r = B\log_2(1 + \frac{P_{tr}|h|^2}{\omega_0 d^{\vartheta}})$, where P_{tr} is the transmission power, h is the channel fading coefficient, and ω_0 denotes the white Gaussian noise power. Assume that N denotes the average number of users with computation tasks within the coverage area of an edge server, the average number of communication channels assigned for each user is $\tilde{M} = \frac{M}{N}$. Accordingly, the transmission delay for offloading D-bits of data to the edge server can be expressed as

$$t^{up} = \frac{D}{r\tilde{M}} = \frac{DN}{rM}. \tag{4}$$

And the energy consumption for transmission is expressed as

$$E^{tr} = P_{tr} t^{up} = \frac{P_{tr} D N}{rM}. \tag{5}$$

Let $n_2 \times N$ denote the number of cores of the edge server, f^e denote the processing capability of each core ($f^e > f^l$), then the average number of cores assigned for one user is n_2 ($n_2 > n_1 \geq 1$). The power consumption of each core of the edge server to process data is $p^e = \kappa(f^e)^3$. And the computing time of T, which consists of the serialized computing time t_s^e and the parallelizable computing time t_p^e, can be expressed as

$$t^e = t_s^e + t_p^e = \frac{c(1-\alpha)D}{f^e} + \frac{\alpha c D}{n_2 f^e}. \tag{6}$$

And the energy consumption of the edge server for processing the D-bits of task T is formulated as

$$E^e = p^e t_s^e + n_2 p^e t_p^e = \kappa c D (f^e)^2. \tag{7}$$

For task offloading, the total latency is a combination of uplink transmission delay and computing time of the edge server[2], which is formulated as

$$t_2 = t^{up} + t^e = \frac{DN}{rM} + \frac{cD}{f^e}(1-\alpha+\frac{\alpha}{n_2}). \tag{8}$$

And the total cost is a combination of transmitting energy cost, communication resource cost, and edge server computation and energy costs. Accordingly, the total cost is formulated as

$$\begin{aligned} C_{off} &= \varrho E^{tr} + p_1 t^{up}\tilde{M} + p_2 t^e + \varrho E^e \\ &= \frac{\varrho P_{tr} DN}{rM} + \frac{p_1 D}{r} + \frac{p_2 cD}{f^e}(1-\alpha+\frac{\alpha}{n_2}) \\ &\quad + \varrho\kappa c D (f^e)^2, \end{aligned} \tag{9}$$

where p_1 denotes the cost for using a communication channel per unit time, p_2 denotes the cost for the edge server processing data per unit time.

C. Computing Acceleration

According to Amdahl's law [13], the obtained speedup when D_i-bits of data is computed locally is formulated as $S_1 = \frac{1}{(1-\alpha)+\frac{\alpha}{n_1}}$. Similarly, the obtained speedup when D_i-bits of data is computed by the edge server is formulated as $S_2 = \frac{1}{(1-\alpha)+\frac{\alpha}{n_2}}$. However, when data is offloaded to the edge, the actual latency not only comes from the computing latency, but also the transmission delay. In this circumstance, the actual computing acceleration is expressed as

$$A = \frac{t_1}{t_2} = \frac{\frac{c}{f^l}(1-\alpha+\frac{\alpha}{n_1})}{\frac{N}{rM} + \frac{c}{f^e}(1-\alpha+\frac{\alpha}{n_2})} \tag{10}$$

III. Performance Analysis and QEACO Strategy

A. Latency and Acceleration

For those latency-sensitive tasks, such as the perception of CAVs, latency is the first consideration. According to formulas (1) and (8), many parameters impacts the latency performance. We first reveal how data size D influences the latency and acceleration. According to formulas (1) and (8), the latency for both t_1 and t_2 are linear functions of D, with slope $k_1^d = \frac{c(1-\alpha+\frac{\alpha}{n_1})}{f^l}$ and $k_2^d = \frac{N}{rM} + \frac{c(1-\alpha+\frac{\alpha}{n_2})}{f^e}$, intercept $b_1^d = 0$ and $b_2^d = 0$, respectively, and are shown as Fig. 2. According to formula (10), the acceleration is a constant function of data size, which is smaller than 1 if the latency of offloading is larger than that of local computing, and larger than 1 if the latency of offloading is smaller than that of local computing, as shown in the red lines of Fig. 2.

Then we reveal how parallelizable fraction α influences the latency and acceleration. According to formulas (1) and (8), the latency for both t_1 and t_2 is a linear function of α, with slope $k_1^p = \frac{cD(\frac{1}{n_1}-1)}{f^l} \le 0$ and $k_2^p = \frac{cD(\frac{1}{n_2}-1)}{f^e} < 0$, intercept $b_1^p = \frac{cD}{f^l}$ and $b_2^p = \frac{cD}{f^e} + \frac{DN}{rM}$, respectively. When $\alpha = 1$, the latency is denoted as $t_1(\alpha=1) = \frac{cD}{n_1 f^l}$ and $t_2(\alpha=1) = \frac{cD}{n_2 f^e} + \frac{DN}{rM}$, respectively. Based on different values of k_1^p, k_2^p, b_1^p, b_2^p, $t_1(\alpha=1)$, and $t_2(\alpha=1)$, there are six different latency variation trends, along with the acceleration variation trends, as shown in Fig. 3.

[2]We neglect the output return process since the processing result is usually very tiny [11].

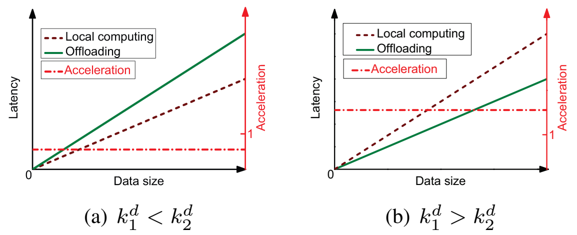


Fig. 2. Latency and acceleration with varying data size

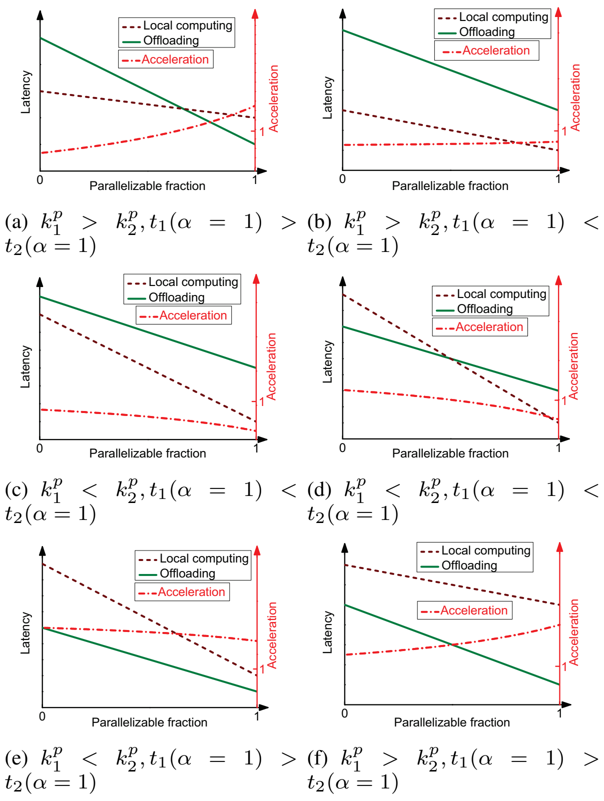


Fig. 3. Latency and acceleration with varying parallelizable fraction. For (a), (b), and (c), $b_1^p < b_2^p$; for (d), (e), and (f), $b_1^p > b_2^p$.

Fig. 4 illustrates how bandwidth influences the latency and acceleration. According to formulas (1) and (8), the latency of local computing and offloading is a constant function and an inverse proportional function of bandwidth, respectively. Especially, t_2 can be reformulated as the form of $t_2 = \frac{a}{B} + b$, where $a = \frac{DN}{M\log_2(1+\frac{P_{tr}|h|^2}{\omega_0 d^\vartheta})}$ and $b = \frac{cD(1-\alpha+\frac{\alpha}{n_2})}{f^e}$. Since $n_1 < n_2$ and $f^l < f^e$, hence $\frac{cD(1-\alpha+\frac{\alpha}{n_1})}{f^l} > \frac{cD(1-\alpha+\frac{\alpha}{n_2})}{f^e}$, which means the limit of t_2 is smaller than t_1.

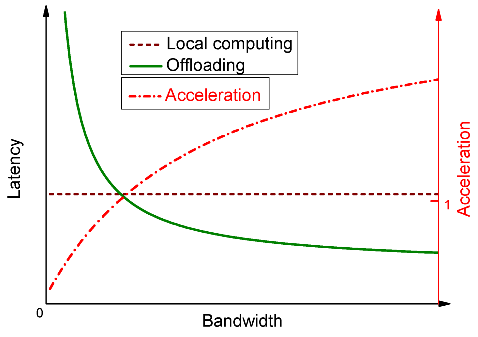
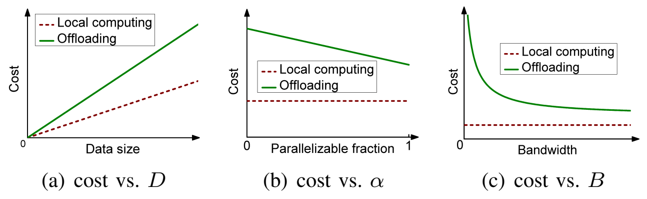


Fig. 4. Latency and acceleration with varying bandwidth.

Moreover, we can also draw a conclusion from formula (10) that the acceleration is an increasing function of f^e and n_2, which also guides the edge service providers to deploy more powerful edge servers with more cores for providing a higher acceleration for IoT users.

B. Cost

The costs for local computing and offloading are denoted as formulas (3) and (9), respectively. Since $\varrho\kappa cD(f^l)^2 < \varrho\kappa cD(f^e)^2$, it is obviously that $C_l < C_{off}$. We reveal how cost varies in terms of data size, parallelizable fraction, and bandwidth in Fig. 5. It can be seen from Fig. 5(a) that both the costs of local computing and offloading increase with increasing data size. The cost of offloading decreases with increasing parallelizable fraction while the cost of local computing keeps a constant value, as shown in Fig. 5(b). Moreover, when bandwidth increases, the cost of offloading decreases and tends to be a constant value $\frac{p_1 D}{r} + \frac{p_2 cD}{f^e}(1-\alpha+\frac{\alpha}{n_2}) + \varrho\kappa cD(f^e)^2$, while the cost of local computing is a constant function of bandwidth, as shown in Fig. 5(c).



(a) cost vs. D (b) cost vs. α (c) cost vs. B

Fig. 5. Cost with varying data size, parallelizable fraction, and bandwidth.

C. QEACO Strategy

Based on the performance analysis presented above, a user can make an optimal offloading decision to obtain it's best QoE according to the following QEACO strategy. First, a user determine its priority for QoE (such as latency, acceleration, energy, or cost). Second, based on the formulas presented in Section II, the performance of both task local computing and offloading are obtained, respectively. Third, compare the performance of the two ways, and choose the one with a better QoE performance. For example, if a user has a priority for latency, the local computing time t_1 and the total offloading latency t_2 are first obtained according to formulas (1) and (8). If t_2 is less than t_1, offloading is preferred; otherwise, local computing is preferred.

IV. Simulation Results

We consider an autonomous driving scenario where a vehicle has a latency-sensitive object identification task. The length and width of each lane are 1000 m and 4 m, respectively. And one RSU is deployed in the middle of roadside. The trajectory of the vehicle is randomly chosen from GAIA Open Dataset of DiDi Express [15]. The number of cores of the vehicle and the edge server at the RSU side are 12 and 128 respectively, the processing capacities of the vehicle and the edge server is 1.4×10^8 cycles/s and 3×10^9 cycles/s, respectively. For the uplink wireless communication, we set the average transmission rate varying from 10 Mbps to 50 Mbps. The main simulation parameters are summarized in Table I. And we compare the system performance of our proposed QEACO against the following benchmark schemes:

- *Local-Comp-Only (LCO)*: the vehicle computes its task locally;
- *Offload-Comp-Only (OCO)*: the vehicle offloads its task to edge server to process;
- *Single-Core-Comp (SCC)*: the task is processed by just one core processor of vehicle or edge server.

TABLE I
Simulation Parameters

Parameters	Values
Data size of the task D	1 Mbit
Parallelizable fraction α	$0.1 \sim 0.6$
Processing density c	10 cycles/bit
Processing capacity of vehicle f^l	1.4×10^8 cycles/s
Processing capacity of edge server f^e	3×10^9 cycles/s
Number of cores of vehicle n_1	12
Number of cores of edge server $n_2 \times N$	128
Average uplink transmission rate r	$\{20, 100\}$ Mbps

Fig. 6 illustrates the latency and acceleration performance under different parallelizable fraction. It is obvious the variation trends of LCO and OCO is the same as that in Fig. 3(d). And the latency of OCO is smaller than that of LCO until the parallelizable fraction is bigger than about 0.33, as the part circled in the red in Fig. 6. Our proposed QEACO has the best performance among the four schemes. Because the vehicle in QEACO can adaptively choose the optimal offloading choice to obtain a lower latency towards its best QoE performance. It is noteworthy that the performance of SCC keeps unchanged with varying parallelizable fraction. This is because when parallelizable fraction is small, task being processed locally in SCC would result in a large latency than being processed at the edge server since the poorer processing capacity of vehicle. Thus, offloading is always the optimal choice for different parallelizable fration. Moreover, the computing time would keep unchanged since $n_2 = 1$ for SCC thus the total latency keeps unchanged according to formulas (6) and (8).

To reveal how bandwidth influence the performance, we let the average transmission rate vary from 10 Mbps to 50

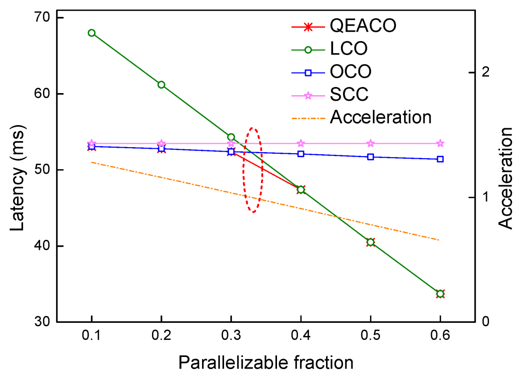


Fig. 6. Latency and acceleration under different parallelizable fraction.

Mbps in Fig. 7. It is obvious that the results of LCO and OCO in Fig. 7 is the same as the variation trends discussed in Fig. 4. The latency of OCO is decreased with the increasing average transmission rate while that of LCO keeps unchanged. This is because no transmission process is needed for LCO. Our proposed QEACO outperforms other schemes due to its adaptive offloading choice. The variation trend of SCC is nearly the same as OCO, which means the vehicle in SCC always choose offloading action to obtain a lower latency since the powerful processing capability of edge server. It is noteworthy that the latency of OCO is bigger than that of LCO until the average transmission rate is bigger than about 26 Mbps, as the part circled in the red. This indicates that wireless communication is the main factor determining the performance of OCO, and increasing the transmission rate can greatly improve the performance of OCO and QEACO.

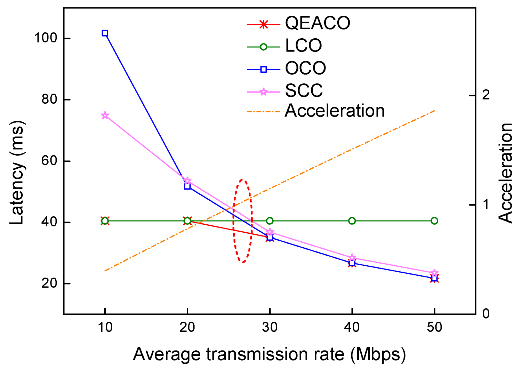


Fig. 7. Latency and acceleration under different average transmission rate.

V. Conclusion

In this paper, we investigated the computation offloading problem in edge computing enabled IoT networks. Considering the parallel and serial processing characteristics of tasks, we first establish both task local computing model and task offloading model. Based on the different preferences of users for QoE, we then present a thorough performance analysis and discuss how QoE performance vary in terms of data size, parallelizable fraction, and bandwidth. And a QEACO strategy is therefore proposed, where a user can optimally and adaptively make its offloading decision towards its best QoE. Finally, simulation results show that QEACO can outperform the benchmark schemes.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (No. 62101463 and No. 61731017), in part by the Fundamental Research Funds for the Central Universities (No. 2682021CX044), in part by the 111 project (No. 111-2-14).

References

[1] R. Lin, Z. Zhou, S. Luo, Y. Xiao, X. Wang, S. Wang, and M. Zukerman, "Distributed optimization for computation offloading in edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 12, pp. 8179–8194, 2020.

[2] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource scheduling in edge computing: A survey," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2021, doi:10.1109/COMST.2021.3106401.

[3] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1584–1607, 2019.

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

[5] Y. Sahni, J. Cao, L. Yang, and Y. Ji, "Multi-hop multi-task partial computation offloading in collaborative edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1133–1145, 2021.

[6] U. Saleem, Y. Liu, S. Jangsher, X. Tao, and Y. Li, "Latency minimization for d2d-enabled partial computation offloading in mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4472–4486, 2020.

[7] Q. Luo, C. Li, T. Luan, and W. Shi, "Minimizing the delay and cost of computation offloading for vehicular edge computing," *IEEE Transactions on Services Computing*, pp. 1–1, 2021, doi:10.1109/TSC.2021.3064579.

[8] W. Wen, Y. Cui, T. Q. S. Quek, F.-C. Zheng, and S. Jin, "Joint optimal software caching, computation offloading and communications resource allocation for mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7879–7894, 2020.

[9] M. Sheng, Y. Wang, X. Wang, and J. Li, "Energy-efficient multiuser partial computation offloading with collaboration of terminals, radio access network, and edge server," *IEEE Transactions on Communications*, vol. 68, no. 3, pp. 1524–1537, 2020.

[10] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Deep reinforcement learning for stochastic computation offloading in digital twin networks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4968–4977, 2021.

[11] S. Jošilo and G. Dán, "Computation offloading scheduling for periodic tasks in mobile edge computing," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 667–680, 2020.

[12] S. Li, S. Lin, L. Cai, W. Li, and G. Zhu, "Joint resource allocation and computation offloading with time-varying fading channel in vehicular edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 3384–3398, 2020.

[13] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the AFIPS Spring Joint Computer Conference*, 1967.

[14] Q. Luo, C. Li, T. H. Luan, W. Shi, and W. Wu, "Self-learning based computation offloading for internet of vehicles: Model and algorithm," *IEEE Transactions on Wireless Communications*, vol. 20, no. 9, pp. 5913–5925, 2021.

[15] Didi, "Urban traffic time index and trajectory data (new)," https://gaia.didichuxing.com, accessed June 5, 2021.