A Novel Probabilistic-Performance-Aware and Evolutionary Game-Theoretic Approach to Task Offloading in the Hybrid Cloud-Edge Environment

Ying Lei[1], Wanbo Zheng[2(✉)], Yong Ma[3], Yunni Xia[1(✉)], and Qing Xia[4]

[1] Software Theory and Technology Chongqing Key Lab, Chongqing University, Chongqing, China
xiayunni@hotmail.com
[2] School of Mathematics, Kunming University of Science and Technology, Yunnan 650500, China
zwanbo2001@163.com
[3] School of Computer and Information Engineering, Jiangxi Normal University, Nanchang, China
[4] Chongqing Key Laboratory of Smart Electronics Reliability Technology, Chongqing, China

Abstract. The mobile edge computing (MEC) paradigm provides a promising solution to solve the resource-insufficiency problem in mobile terminals by offloading computation-intensive and delay-sensitive tasks to nearby edge nodes. However, pure edge resources can be limited and insufficient for computational-intensive applications raised by multiple users, which calls for a hybrid architecture with a centralized cloud server and multiple edge nodes and smart resource management strategies in such hybrid environment. The problem is however challenging due to the distributed nature and intrinsic dynamicness of the environment. Existing researches in this direction mainly see that edge servers are with constant performance and consider the offloading decision-making as a static optimization problem. In this paper, instead, we consider that geographically distributed edge servers are with time-varying performance and introduce a dynamic offloading strategy based on a probabilistic evolutionary game-theoretic framework. To validate our proposed framework, we conduct experimental case studies based on a real-world dataset of cloud edge resource locations and show that our proposed approach outperforms traditional ones in terms of multiple metrics.

Keywords: Task offloading · Mobile edge computing · Evolutionary game theory · Probabilistic QoS

1 Introduction

Due to limited battery power, storage capacity and computational power, mobile devices face challenges in executing delay-sensitive and resource-hungry complex

H. Gao et al. (Eds.): CollaborateCom 2020, LNICST 349, pp. 255–270, 2021.
https://doi.org/10.1007/978-3-030-67537-0_16

applications such as augmented reality and online gaming. Mobile Edge Computing (MEC) is widely believed as a remedy to alleviate this problem [10,11]. In MEC, the mobile edge is enhanced with analysis and storage capabilities, possibly by a dense deployment of computational servers or by strengthening the already-deployed edge entities such as small cell base stations. Therefore, mobile devices are allowed to offload their computationally expensive tasks to the edge servers while meeting some specific quality of service (QoS) requirements [8]. This process, referred to as computation offloading, is feasible due to the fact that edge servers are usually deployed near mobile users, specifically in comparison to the remote cloud servers.

Its challenge lies in that offloading decision can't be made by using conventional centralized resource allocation schemes, due to the fact that such mechanism requires the availability of global knowledge of status of all involving nodes. As a result, distributed and autonomous approaches, where the individual mobile devices and mobile edge servers make autonomous offloading decisions, are in high need.

For the purpose described above, we develop a novel dynamic offloading approach by using a probabilistic-performance-aware game-theoretic model. Different from most existing methods in this direction, our proposed method considers that geographically distributed edge servers are with time-varying performance [19] instead of the constant one. To validate the proposed framework, we conduct experimental case studies based on a real-world dataset of cloud/edge resource locations. Experimental results clearly suggest that our proposed approach outperforms traditional ones in terms of multiple metrics.

2 Related Work

As an important theme of mobile edge computing (MEC) [10], mobile task offloading has attracted a lot of research interests recently. Existing studies can be classified into two categories in terms of scheduling fashion: online solutions and offline solutions [4,14,30]. For online solutions, the offloading decision for every mobile task will be made at once when it arrives. While the offline ones usually split the continuous-time into discrete time slots, requests which arrive in the same time slot will be scheduled together at the end of that time slot.

Most of the current studies are offline ones, for example, Chen *et al.* [8] formulate the task offloading problem at a certain time slot as a mixed integer non-linear program in software defined ultra dense network (SD-UDN), then transform this optimization problem into two sub-problems and develop a Lagrange-multiplier-based algorithm to solve them. Hosseinzadeh *et al.* [18] proposes an ANN-PSO algorithm which combines machine learning and metaheuristic algorithm, the behavior model in LTS is divided into two stages to evaluate the process of service selection and composition, in order to avoid premature convergence and guarantee the requested QoS factors. Alameddine *et al.* [4] define the dynamic task offloading and scheduling (DTOs) problem as a Mixed Integer Program (MIP), they also propose a novel thoughtful decomposition based on the technique of the Logic-Based Benders Decomposition to reduce

the searching space resulted by massive requests. Alfakih *et al.* [5] proposed an offloading decision-based state-action-reward-state-action method (OD-SARSA) to sovle the question of developing an efficient resource management model for the selected MEC server in a multi-edge network by reducing system cost, including energy consumption and computing time delay. Zaw *et al.* [29] formulate the resource allocation problem in multi-edge network as a Generalized Nash Equilibrium Problem (GNEP) and prove the existence and uniqueness of the game, two distributed algorithms are proposed to update Lagrange Multipliers on client and MBS respectively. Fantacci *et al.*[15] establish a matching game with externalities between the application requested by the IIoT devices and the ECSs to raise the VRCs placement problem in a hybrid EC-Cloud network structure for an IIoT scenario. Xia *et al.* [26] regard the cloud-edge network as a graph, and propose an optimal approach named EDD-IP based on the Integer Programming technique to solve the problem of edge data distribution (EDD) in the edge computing environment. For a large-scale EDD problems, EDD-A algorithm they proposed could find the approximate solutions effectively.

There are also some studies focus on the online solution of this problem, for example, Liu *et al.* [22] and Du *et al.* [14] assume the task arrivals are i.i.d over time and their average arrival rates are pre-given. Chen *et al.* [7] and Xu *et al.* [28] consider task arrival follows Poisson process, and random arrival rate is used capture the time change of task arrival mode at different times. While Zhao *et al.* [30] assume the unloading demands of mobile users follow the binomial distribution of given probability. Based on these assumptions, a local edge double-layer queuing model is established, and the Lyapunov optimization technique is used to maintain the stability of the system. Xia *et al.* [24] considers the arbitrary arrival time of requests and propose a best-fit-decreasing-based method to yield offloading decision in real-time. However, their approach is a centralized one. Xia *et al.* [27] introduce data migration and data caching technology into task offloading in the edge environment, model the collaborative edge data caching problem (CEDC) as a constrained optimization problem, and propose an online algorithm based on Lyapunov optimization to achieve the goal of saving bandwidth cost, reducing network latency and minimizing access costs. Aslanpour *et al.* [6] comprehensively discusses the performance and metrics for cloud, fog and edge computing from multiple perspectives, so as to help researchers identify appropriate metrics for evaluating performance by analyzing specific scenarios.

3 Modeling and Formulation

3.1 System Model

In this paper, we consider a hybrid environment composed of a macro centralized cloud server is deployed, and K micro edge nodes. It is assume that the hybrid environment accomodates N users and each user is allowed to access both the centralized cloud server or one of the edge ones. The coverage of cloud servers is usually large enough to cover all users in a certain geographical area. On the

contrary, edge nodes cover their near-by area, and are weaker than cloud one in terms of computing power, storage resources and other resource allocation. According [13], a large area is usually divided into J small service regions with such that the number of edge servers in each region is nearly equal. Users in each region form a population, and they have the stationary proportion to adopt the same strategy under the assumption of bounded-rationality.

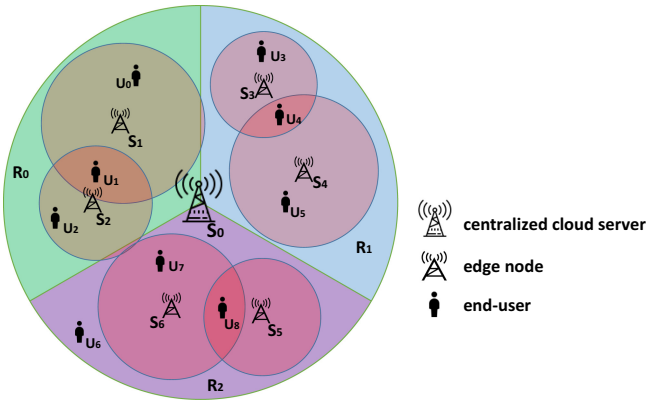

Fig. 1. Edge computing environment example

Figure 1 shows a good example of the hybrid environment described above. In this example, there exists a centralized cloud server (S_0), 6 edge ones ($S_1 \sim S_6$) and 9 users ($U_0 \sim U_8$). Coverage areas of different edge nodes may overlap and thus a user may have multiple candidates for task offloading [14]. For example, U_1 can be allocated to S_0, S_1 or S_2. In contrast, U_6 can only offload its task to S_0. We partition the area into three regions of the same size ($R_0 \sim R_2$), and two edge nodes are deployed in each region.

The notation used in this paper is shown in Table 1.

3.2 Computation Model

The tasks offloading time for U_i can be estimated as $T_i = \Delta_i^{ul} + \Delta_i^{dl} + \Delta_i^{bh} + \Delta_i^{exe}$ is composed of four parts [3], where Δ_i^{ul} indicates the uplink communication time, Δ_i^{dl} the downlink time, Δ_i^{bh} the backhaul link time and Δ_i^{exe} the backhaul link time. According to [9,20], Δ_i^{bh} can be infinitesimal, and the downlink time Δ_i^{dl} can usually be a constant ξ. Δ_i^{ul} and Δ_i^{exe} are calculated according to the network conditions and device performance. Therefore, the T_i can be expressed as:

$$\begin{aligned} T_i = t(i,k) &= \Delta_i^{ul} + \Delta_i^{dl} + \Delta_i^{bh} + \Delta_i^{exe} \\ &= \frac{b_i}{w_k \eta_i^k} + \xi + \frac{d_i}{f_k} \end{aligned} \tag{1}$$

Table 1. Key Notations

Notation	Description
S_k	The k^{th} server
lat_k, lng_k	Latitude and longitude of S_k
R_k	Coverage radius of S_k
W_k	Maximum bandwidth of S_k
F_k	Maximum computing power of S_k
C_f^k	Unit price for computing of S_k
C_t^k	Unit price for communication of S_k
V_k	Rent of S_k
num_k	The number of users who select cloud service on S_k
w_k	Real-time bandwidth of the cloud service on S_k
f_k	Real-time computing power of the cloud service on S_k
U_i	The i^{th} end-user
lat_i, lng_i	Latitude and longitude of U_i
b_i	Amount of data of U_i's task
d_i	Number of calculation instructions of U_i's task
t_i	The maximum tolerable delay of U_i
c_i	The maximum tolerable cost of U_i
a_i	The population to which U_i belongs
$dist(i,k)$	The distance between U_i and S_k
P_j	The jth population

where $\eta_i^k = \lambda / dist(i,k)$ is decided by the distance between U_i and S_k. As the distance increases, the bit error rate increases and the average transmission speed decreases [20]. w_k and f_k indicate the averaged computing power and averaged bandwidth of S_k, respectively. f_k can be calculated based on its corresponding historical distribution [19] according to:

$$f_k = \int_{-\infty}^{\infty} Pr(t) t \mathrm{d}t. \tag{2}$$

Acoording (3), monetary cost of U_i comprises three parts, namely the communication cost, the computation cost, and the fees for renting server. C_t^k and C_f^k indicate the price per unit transmission rate and the charge per unit computation resources of S_k, respectively. The rent V_k^{rent} is related to the current load of the server. If the server is used more efficiently and provides services for more users, the lower the rent shared by each user.

$$\begin{aligned} C_i = c(i,k) &= C_k^{commu} + C_k^{compu} + V_k^{rent} \\ &= C_t^k \frac{b_i}{w_k \eta_i^k} + C_f^k \frac{d_i}{f_k} + \frac{V_k}{num_k} \end{aligned} \tag{3}$$

We use US_i to represent U_i's user satisfaction, which can be expressed as the probability that both offloading time and monetary cost constraints are met [19]:

$$US_i = Pr(T_i \leq t_i) \cdot Pr(C_i \leq c_i) \tag{4}$$

US_{P_j} represents the satisfaction of the entire population, where num_j denotes the number of users in P_j:

$$US_{P_j} = \sum_{i=1}^{num_j} US_i \tag{5}$$

US is the satisfaction of the whole area and it is regarded as the ultimate optimization objective:

$$US = \sum_{j=1}^{J} US_{P_j} \tag{6}$$

4 Evolutionary Game Formulation

In this section, we first give the evolutionary game formulation for the problem. Due to the limitation of computation resource and bandwidth resource in each edge node, end-users in different regions have to compete for resources in clouds. Based on the replicator dynamics [25], we consider an evolutionary stable strategy (ESS) for the problem of hybrid-edge-cloud-based task offloading.

4.1 Game Formulation

A normal game includes three factors: the player set, the strategy set, and the payoff function of every player when choosing a strategy. In the context of an evolutionary game, the population is utilized to represent the group of players with the same properties [17]. We introduce the formulation of the evolutionary game as below:

Players: End-users are denoted as players in the game.

Population: Players are grouped into different populations by geographical locations. We denote the set of population as $\{\mathbb{N}_0, \mathbb{N}_1, \ldots, \mathbb{N}_J\}$. Players in each population are all located in the same geographical region.

Strategy: Each player's strategy refers to the selection of servers. In this game, there are $1 + K$ services for players to select. Accordingly, the server selection strategy can be denoted as $\mathbb{K} = \{0, 1, 2, \ldots, K\}$, which refers to the selection of the centralized cloud server and K edge nodes. $\mathbb{S}_j$ which notes the server selection set of P_j is the subset of $\mathbb{K}$.

Population Share: num_j^k denotes the number of end-users who selecting strategy S_k for population $\mathbb{N}_j$. Thus, $x_j^k = num_j^k / num_j$ is population share of S_k in population $\mathbb{N}_j$, where $x_j^k \in [0, 1]$.

Population State: The population shares of all servers constitute the population state denoted by a vector $\mathbf{x}_j = \left[x_j^0, x_j^1, x_j^2, \ldots, x_j^K\right]$. We have $\sum_{k=0}^{K} x_j^k = 1$. We use a matrix $\mathbb{X}$ to denote the population state space which contains all J population states.

$$\mathbb{X} = \begin{pmatrix} x_0^0 & x_0^1 & \ldots & x_0^K \\ x_1^0 & x_1^1 & \ldots & x_1^K \\ \vdots & \vdots & \ddots & \vdots \\ x_{J-1}^0 & x_{J-1}^1 & \ldots & x_{J-1}^K \end{pmatrix} \tag{7}$$

Payoff: The payoff of a player is decided by its net utility function. The net utility function is based on (4).

$$\pi_j^k(i) = Pr(T_i \leq t_i) \cdot Pr(C_i \leq c_i) \tag{8}$$

4.2 Evolutionary Stable Strategy (ESS)

In a traditional game theory, all players can achieve a stable state where no player can further obtain extra benefit by unilaterally changing its strategy. Such a state is called Nash equilibrium (NE). In this work, we call the game Γ as the game of service selection, where N is the player set, $\mathbb{S} = \{s_1, s_2, \ldots, s_N\}$ is the strategy set of all players, and $\Pi = \{\pi_1, \pi_2, \ldots, \pi_N\}$ is the set of payoff function. Let $\mathbb{S}_{-i} = \{s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_N\}$ be a strategy profile of all players except player i, and $\pi(s_i, s_{-i})$ is set to be the payoff function of player i when this player selects the strategy s_i while others select s_{-i}. Then, the corresponding Nash equilibrium can be defined as below.

Definition 1. *A Nash equilibrium (NE) of a strategic game* $\Gamma =< \mathbb{N}, \mathbb{S}, \Pi >$ *is a profile* $s^* \in \mathbb{S}$ *of actions with the property that for every player* $i \in N$ *we have:*

$$\pi(s_i, s_{-i}^*) \leq \pi(s_i^*, s_{-i}^*), \forall s_i \in \mathbb{S} \tag{9}$$

The Nash equilibrium has a property of self-reinforcement that each player has no motivation to deviate from this equilibrium. The general solution to obtain the Nash equilibrium is on the assumption of complete rationality among all players. However, with a small perturbation, all players may change their strategies to reach another Nash equilibrium. In an evolutionary game theory [17], an equilibrium strategy is adopted among players with bounded rationality, which can resist small disturbances. This equilibrium strategy is called ESS and is defined as below.

Definition 2. *A strategy profile* $\mathbb{S}^* = \{s_1^*, s_2^*, \ldots, s_N^*\}$ *is an ESS iff* $\forall s_i \notin \mathbb{S}^*, s_{-i} \neq s_{-i}^*$*:*

$$\begin{aligned} &1. \quad \pi(s_i, s_{-i}^*) < \pi(s_i^*, s_{-i}^*) \\ &2. \quad \pi(s_i, s_{-i}^*) = \pi(s_i^*, s_{-i}^*), \pi(s_i, s_{-i}) < \pi(s_i^*.s_{-i}) \end{aligned} \tag{10}$$

Compared with Nash equilibrium, the condition (1) of Definition 2 ensures that ESS is a Nash equilibrium (NE). The condition (2) of Definition 2 ensures the stability of the game process. During the process of strategy evolution, players using mutation strategy will decrease until all players in the population asymptotically approach to the ESS. In our problem, end-users adapt their strategies among a finite set of strategies to get a better payoff. In each time, each end-user can have his/her own strategy set and the information of average payoff in the same population. Each end-user can repeatedly evolve his/her strategy over time for the cloud service selection. After sufficient repetitive stages, all end-users' strategy profile approaches to an ESS. The process of this strategy replication can be modeled by replicator dynamics, which is described in the next section.

4.3 Replicator Dynamics

In the dynamic evolutionary game [12], replicator dynamics provides a method to acquire the population information of others and converges towards an equilibrium selection. It is also significant to investigate the speed of convergence of strategy adaptation to reach evolutionary equilibrium (EE) that the population will not change its selection [23]. The basic idea is that in a population of players with bounded rationality, strategy with better results than average level will be gradually adopted by more players, and the proportion of players' strategies will change consequently. It is given as below:

$$\dot{x}_j^k(t) = \delta x_j^k(t)[\pi_j^k(t) - \bar{\pi}_j(t)] \tag{11}$$

where δ is used to control the convergence speed of strategy adaption for players in the same population. $\pi_j^k(t)$ is the current payoff of the individuals choosing strategy k in P_j, $\bar{\pi}_j(t)$ is the average payoff of $\mathbb{N}_j$. The growth rate $\dot{x}_j^k(t)$ is relevant to the difference between the payoff $\pi_j^k(t)$ for that selection strategy and the population's average payoff $\bar{\pi}_j^k(t)$ as well as the current size of population share $\pi_j^k(t)$. The average payoff $\bar{\pi}_j(t)$ of the population can be derived as:

$$\bar{\pi}_j(t) = \sum_{k=0}^{K} x_j^k(t)\pi_j^k(t) \tag{12}$$

Based on the replicator dynamics of strategy selection in P_j, the number of end-users that choose the strategy s has a positive growth trend in the population if their payoff is above the average payoff in the same population. Through setting

Algorithm 1. Evolutionary Game Algorithm

Input: N : number of end-users; K : number of servers; J : number of populations; ε : error factor.
Output: $\mathbb{X}_j^*$: The ESS of P_j

Initialize: $b_i, d_i, t_i, c_i, W_k, F_k, V_k, R_k, C_f^k, C_t^k, \lambda, \xi, \delta$
$t \leftarrow 0$
$\mathbb{X}_j^* \leftarrow 0, \forall j \in J$
Each player determines its available server set as the strategy set $\mathbb{S}_i$ based on its location information
Each player randomly selects server k from its strategy set $\mathbb{S}_i$
Each servers acquires n_k and allocates w_k and f_k.
All servers calculate the resources they have allocated and send the information to the players.
Player calculates revenue π_j^k based on information returned by the servers
The server collects the revenue π_j^k returned by the players and calculates the $\bar{\pi}_j$
while $|\pi_j^k - \bar{\pi}_j| \geq \varepsilon$ **do**
    Each population computes $\dot{x}_j^k$,and update $x_j^k = x_j^k + \dot{x}_j^k$
    update $\mathbb{X}_j$
    Player calculates revenue π_j^k based on information of $\mathbb{X}_j$.
    The server collects the revenue π_j^k returned by the players and calculates the $\bar{\pi}_j$
    Plays change their strategies with probability δ, when their payoff is less than the average payoff in the same population.
    Update $t = t + \tau$
end while

$\sum_k \dot{x}_j^k(t) = 0$, we can get the fixed point of the replicator dynamics, in which the population state will not change and no player is willing to change its strategy since all players in the same population have the same payoff.

The algorithm based on replicator dynamics is described as:

5 Experimental Evaluation

To validate our proposed method, we carry out simulative experiments based on the Edge User Allocation (EUA) dataset [16] for positions of cloud servers and end-users, and a dataset of [32] for the performance of cloud servers. Figure 2 shows that the area contains 200 end-users, 1 centralized cloud server and 12 edge servers. The servers are marked red. End-users of the same population share the same color.

As for the performance data for the centralized cloud, we test a typical third-party commercial cloud service, *i.e.*, Tencent cloud [2]. Figures 3 and 4 show its measured thoughput and computing performance (in terms of MIPS, *i.e.*, Million Instructions Per Second) of the centralized cloud, and these data are partitioned into 24 consecutive windows [1].

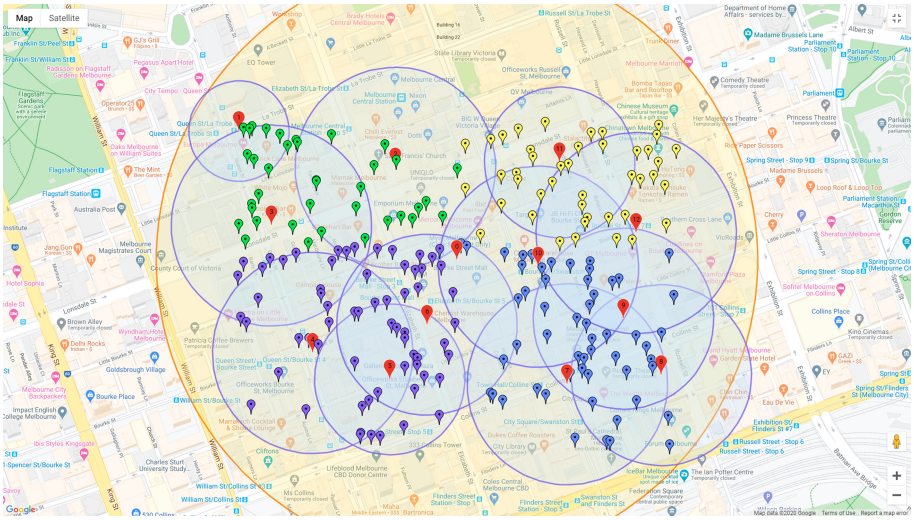

Fig. 2. Geographical distribution of experimental data

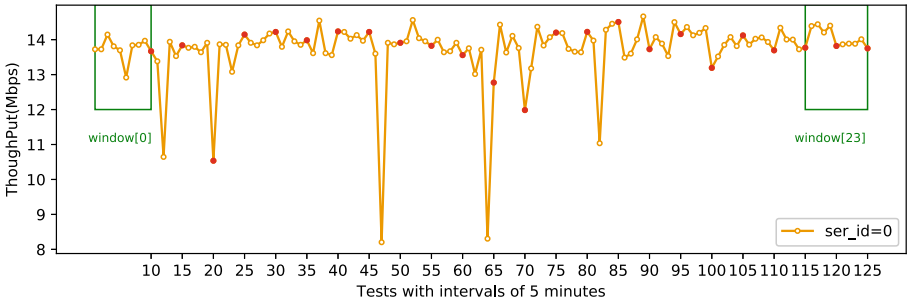

Fig. 3. Throughput for the centralized cloud

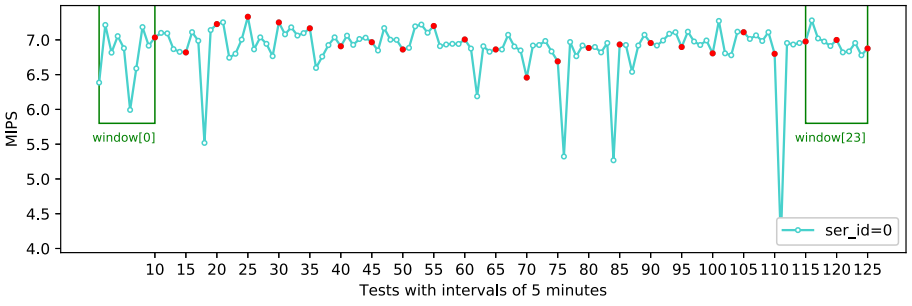

Fig. 4. MIPS for the centralized cloud

We compare our proposed method with three existing ones: Greedy [31], Genetic Algorithm (GA) [21], and Nash-based Game [9].

Greedy. Greedy algorithm refers to the algorithm that takes the best or optimal choice in each step when solving a problem, hoping to lead to the best or optimal algorithm. In this environment, each player chooses the server that can bring the highest US to offload based on the current resource allocation, regardless of whether other player's strategy changes will affect his next strategy selection.

Genetic Algorithm (GA). GA is a method to find the optimal solution by simulating the natural evolution process. We encode the players' strategy selection, get new populations through crossover and mutation, and then select through Roulette Wheel Selection.

Nash-Based Game. Game theory is another distributed algorithm based on Nash equilibrium, called Nash-based, which treats tasks equally without considering dynamically allocating computation resources to different types.

As can be seen from Figs. 5, 6, 7 and 8: 1) our method beats its peers at all 24 windows in terms of total user satisfaction, *i.e.*, US; 2) our method and Nash-based Game outperform others in terms monetary cost and our method shows more stable advantage, fewer iteration rounds, and offloading time than Nash-based Game through all windows.

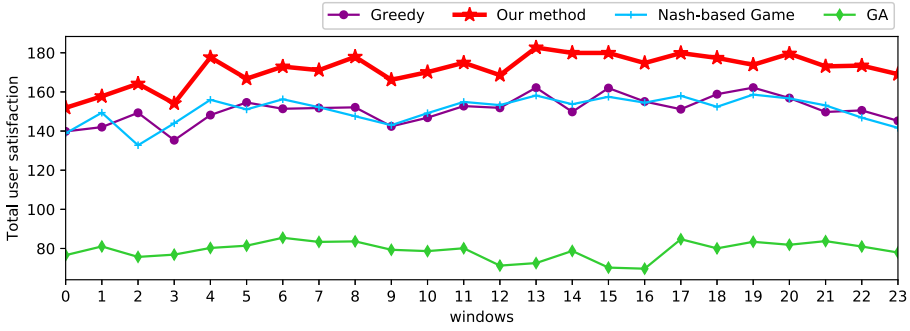

Fig. 5. Comparison of total user satisfaction at different windows

In this part, we compare the convergence of two game theory algorithms for the population after mutation. First of all, the population reached EE 1. After five iterations, a small number of users in the population will mutate randomly, showing that the size of user tasks changes, and the tolerable delay and cost will also change. After the mutation, the population readjust the offloading strategy to reach EE 2. The number of iterations to reach the new EE is taken as a measure of the population's convergence under a certain game theory strategy.

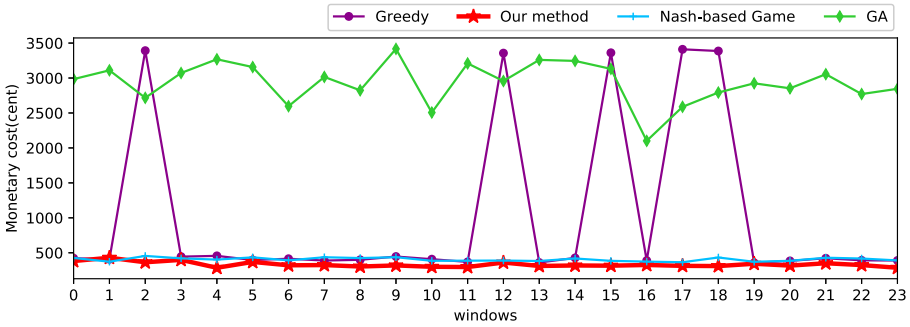

Fig. 6. Comparison of monetary cost at different windows

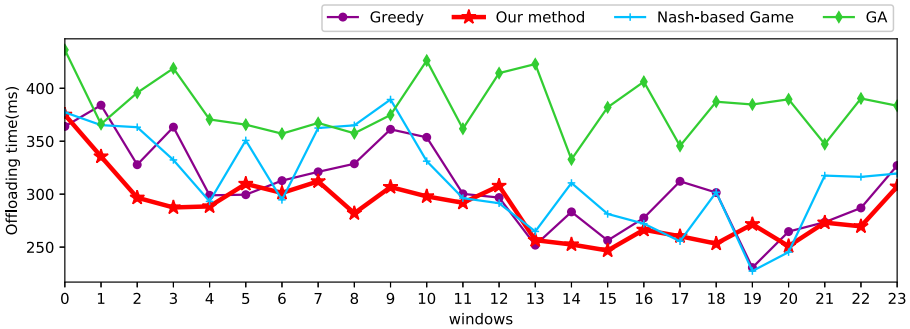

Fig. 7. Comparison of offloading time at different windows

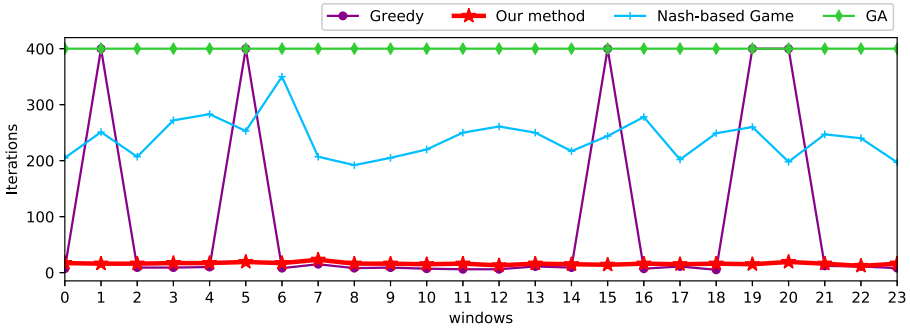

Fig. 8. Comparison of iterations at different windows

As shown in Fig. 10, players need to go through 10 iterations to reach new EE after mutation by using traditional game theory algorithm, while evolutionary game algorithm (our method) only needs one iteration to reach new EE in Fig. 9. Experimental results clearly suggest that our proposed approach outperforms traditional ones in terms of Anti-interference ability.

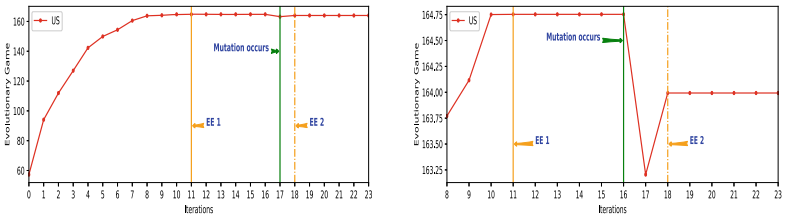

Fig. 9. Convergence of evolutionary game.

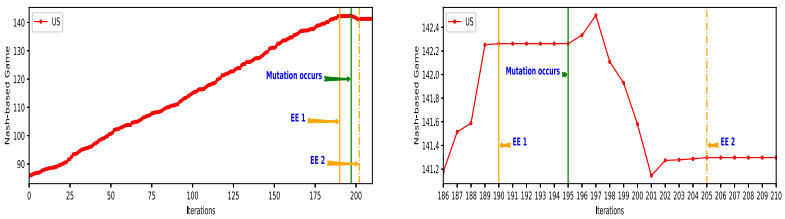

Fig. 10. Convergence of Nash-based game.

6 Conclusion

In this paper, we study the offloading problem for the cloud-edge-hybrid architecture and assume that geographically distributed edge servers are with fluctuating real-time performance. We develop a dynamic offloading strategy based on a probabilistic evolutionary game-theoretic model. For the model validation purpose, we test our method against other existing ones through simulations based on a well-known dataset of cloud/edge resource locations. It's clear to see that our proposed approach outperforms its peers in terms of multiple metrics.

In the future, we plan to carry out the following work: 1) We plan to consider the mobility of users and edge nodes, and study the strategy of multi-user task online offloading under irregular trajectory; 2) we intend to introduce a performance prediction model based on time series and neural network, and use trajectory prediction information and performance prediction information to drive the multi-user task offloading decision-making model; 3) We plan to consider the impact of task failure and transmission failure on task offloading under untrusted communication conditions, and design fault tolerant and fault-tolerant multi-user task offloading strategies and algorithms; 4) etc.

Acknowledgement. This work is supported in part by the Graduate Scientific Research and Innovation Foundation of Chongqing, China (Grant No. CYB20062 and CYS20066), and the Fundamental Research Funds for the Central Universities (China) under Project 2019CDXYJSJ0022.

References

1. Google maps platform. https://developers.google.cn/maps/documentation/javascript/tutorial. Accessed 6 May 2020
2. Tencent cloud platform. https://intl.cloud.tencent.com/zh/product/cvm. Accessed 16 Apr 2020
3. Al-Shuwaili, A.N., Simeone, O., Bagheri, A., Scutari, G.: Joint uplink/downlink optimization for backhaul-limited mobile cloud computing with user scheduling. CoRR abs/1607.06521 (2016). http://arxiv.org/abs/1607.06521
4. Alameddine, H.A., Sharafeddine, S., Sebbah, S., Ayoubi, S., Assi, C.: Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing. IEEE J. Sel. Areas Commun. **37**(3), 668–682 (2019). https://doi.org/10.1109/JSAC.2019.2894306
5. Alfakih, T., Hassan, M.M., Gumaei, A., Savaglio, C., Fortino, G.: Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA. IEEE Access **8**, 54074–54084 (2020). https://doi.org/10.1109/ACCESS.2020.2981434
6. Aslanpour, M.S., Gill, S.S., Toosi, A.N.: Performance evaluation metrics for cloud, fog and edge computing: a review, taxonomy, benchmarks and standards for future research. Internet Things **12**, 100273 (2020). https://doi.org/10.1016/j.iot.2020.100273. http://www.sciencedirect.com/science/article/pii/S2542660520301062
7. Chen, L., Zhou, S., Xu, J.: Computation peer offloading for energy-constrained mobile edge computing in small-cell networks. IEEE/ACM Trans. Networking **26**(4), 1619–1632 (2018)
8. Chen, M., Hao, Y.: Task offloading for mobile edge computing in software defined ultra-dense network. IEEE J. Sel. Areas Commun. **36**(3), 587–597 (2018)
9. Chen, X., Jiao, L., Li, W., Fu, X.: Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans. Networking **24**(5), 2795–2808 (2016). https://doi.org/10.1109/TNET.2015.2487344
10. Chen, Z., Cheng, S.: Computation offloading algorithms in mobile edge computing system: a survey. In: Data Science - 5th International Conference of Pioneering Computer Scientists, Engineers and Educators, ICPCSEE 2019, Proceedings, Part I, Guilin, China, 20–23 September 2019, pp. 217–225 (2019). https://doi.org/10.1007/978-981-15-0118-0_17
11. Cong, P., Zhou, J., Li, L., Cao, K., Wei, T., Li, K.: A survey of hierarchical energy optimization for mobile edge computing: a perspective from end devices to the cloud. ACM Comput. Surv. **53**(2), 38:1–38:44 (2020). https://doi.org/10.1145/3378935
12. Cuong, D., Tran, N., Tran, D., Pham, C., Alam, M.G.R., Hong, C.S.: Toward service selection game in a heterogeneous market cloud computing. In: Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management, IM 2015, pp. 44–52, June 2015. https://doi.org/10.1109/INM.2015.7140275
13. Dong, C., Wen, W.: Joint optimization for task offloading in edge computing: an evolutionary game approach. Sensors (Switzerland) **19**(3) (2019). https://doi.org/10.3390/s19030740
14. Du, W., Lei, T., He, Q., Liu, W., Lei, Q., Zhao, H., Wang, W.: Service capacity enhanced task offloading and resource allocation in multi-server edge computing environment. In: 2019 IEEE International Conference on Web Services, ICWS 2019, Milan, Italy,8–13 July 2019, pp. 83–90 (2019). https://doi.org/10.1109/ICWS.2019.00025

15. Fantacci, R., Picano, B.: A matching game with discard policy for virtual machines placement in hybrid cloud-edge architecture for industrial IoT systems. IEEE Trans. Ind. Informatics **16**(11), 7046–7055 (2020). https://doi.org/10.1109/TII.2020.2999880
16. He, Q., Cui, G., Zhang, X., Chen, F., Deng, S., Jin, H., Li, Y., Yang, Y.: A game-theoretical approach for user allocation in edge computing environment. IEEE Trans. Parallel Distrib. Syst. **31**(3), 515–529(2019). https://doi.org/10.1109/tpds.2019.2938944
17. Hofbauer, J., Sigmund, K.: Evolutionary game dynamics. Bull. Am. Math. Soc. **40**(4), 479–519 (2003)
18. Hosseinzadeh, M., Tho, Q.T., Ali, S., Rahmani, A.M., Souri, A., Norouzi, M., Huynh, B.: A hybrid service selection and composition model for cloud-edge computing in the internet of things. IEEE Access **8**, 85939–85949 (2020). https://doi.org/10.1109/ACCESS.2020.2992262
19. Hwang, S., Hsu, C., Lee, C.: Service selection for web services with probabilistic qos. IEEE Trans. Serv. Comput. **8**(3), 467–480 (2015). https://doi.org/10.1109/TSC.2014.2338851
20. Lan, Z., Xia, W., Cui, W., Yan, F., Shen, F., Zuo, X., Shen, L.: A hierarchical game for joint wireless and cloud resource allocation in mobile edge computing system. In: 10th International Conference on Wireless Communications and Signal Processing, WCSP 2018, Hangzhou, China, 18–20 October 2018. pp. 1–7 (2018). https://doi.org/10.1109/WCSP.2018.8555606
21. Li, W., Xia, Y., Zhou, M., Sun, X., Zhu, Q.: Fluctuation-aware and predictive workflow scheduling in cost-effective infrastructure-as-a-service clouds. IEEE Access **6**, 61488–61502 (2018). https://doi.org/10.1109/ACCESS.2018.2869827
22. Liu, C.F., Bennis, M., Debbah, M., Poor, H.V.: Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing. IEEE Trans. Commun. **67**(6), 4132–4150 (2019)
23. Niyato, D., Hossain, E.: Dynamics of network selection in heterogeneous wireless networks: an evolutionary game approach. IEEE Trans. Veh. Technol. **58**(4), 2008–2017 (2009)
24. Peng, Q., et al.: Mobility-aware and migration-enabled online edge user allocation in mobile edge computing. In: 2019 IEEE International Conference on Web Services (ICWS), pp. 91–98. IEEE (2019)
25. Taylor, P., Jonker, L.: Evolutionary stable strategies and game dynamics. Math. Biosci. **40**, 145–156 (1978). https://doi.org/10.1016/0025-5564(78)90077-9
26. Xia, X., Chen, F., He, Q., Grundy, J.C., Abdelrazek, M., Jin, H.: Cost-effective app data distribution in edge computing. IEEE Trans. Parallel Distrib. Syst. **32**(1), 31–44 (2021). https://doi.org/10.1109/TPDS.2020.3010521
27. Xia, X., Chen, F., He, Q., Grundy, J.C., Abdelrazek, M., Jin, H.: Online collaborative data caching in edge computing. IEEE Trans. Parallel Distrib. Syst. **32**(2), 281–294 (2021). https://doi.org/10.1109/TPDS.2020.3016344
28. Xu, J., Chen, L., Zhou, P.: Joint service caching and task offloading for mobile edge computing in dense networks. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, pp. 207–215. IEEE (2018)
29. Zaw, C.W., Ei, N.N., Im, H.Y.R., Tun, Y.K., Hong, C.S.: Cost and latency tradeoff in mobile edge computing: A distributed game approach. In: IEEE International Conference on Big Data and Smart Computing, BigComp 2019, Kyoto, Japan, 27 February–2 March 2019, pp. 1–7 (2019). https://doi.org/10.1109/BIGCOMP.2019.8679304

30. Zhao, H., Deng, S., Zhang, C., Du, W., He, Q., Yin, J.: A mobility-aware cross-edge computation offloading framework for partitionable applications. In: 2019 IEEE International Conference on Web Services, ICWS 2019, Milan, Italy, 8–13 July 2019, pp. 193–200 (2019). https://doi.org/10.1109/ICWS.2019.00041
31. Zhao, P., Tian, H., Fan, B.: Partial critical path based greedy offloading in small cell cloud. In: IEEE 84th Vehicular Technology Conference, VTC Fall 2016, Montreal, QC, Canada, 18–21 September 2016, pp. 1–5 (2016). https://doi.org/10.1109/VTCFall.2016.7881145
32. Zheng, Z., Zhang, Y., Lyu, M.R.: Investigating QoS of real-world web services. IEEE Trans. Serv. Comput. **7**(1), 32–39 (2014). https://doi.org/10.1109/TSC.2012.34