Hindawi
Discrete Dynamics in Nature and Society
Volume 2021, Article ID 3143434, 8 pages
https://doi.org/10.1155/2021/3143434

Research Article

Intelligent Scheduling Method Supporting Stadium Sharing

Lei Fang

Jilin Agricultural Science and Technology University, Jilin 132101, China

Correspondence should be addressed to Lei Fang; fanglei@jlnku.edu.cn

Received 6 October 2021; Revised 26 October 2021; Accepted 1 December 2021; Published 14 December 2021

Academic Editor: Lianbo Ma

At present, the fast-paced work and life make people under great pressure, and people's enthusiasm for fitness is getting higher and higher, which is in contradiction with the shortage of existing stadiums. So it is considerably significant to open shared stadiums near where citizens live for booking. Therefore, how to allow citizens to book a suitable stadium according to their own needs through mobile phones or computers is an urgent problem to be solved. The booking of the shared stadium can be regarded as a mobile edge computing (MEC) scenario, and the problem can be transformed into task scheduling research under MEC through intelligent scheduling method. When using edge computing (EC) technology for service calculation, the mobile terminal needs to offload the service to the edge computing server. After the server completes the calculation, the calculation results will be sent back to the mobile terminal. Therefore, the calculation time and system energy consumption in the calculation process can be further reduced through task scheduling to improve user satisfaction. In this study, joint scheduling of service caching and task algorithm is proposed to reduce the latency of booking shared stadium request and improve user experience. The simulation results show that the proposed algorithm with edge cooperation idea can achieve lower average system latency at lower load level and can significantly reduce the cloud offloading ratio under low and middle pressure. In addition, the proposed algorithm uses the secondary transfer of more tasks to reduce the pressure of local task running. Finally, the quality of experience (QoE) satisfaction rate under low pressure is guaranteed.

1. Introduction

Fitness is gaining popularity in China over the decade [1, 2]. One of the main problems encountered in the implementation is the lack of stadiums [3]. As an important guarantee for the development of sports, the sharing of stadium resources from college or high/middle school and society has become the direction under the condition of insufficient stadium resources, which can effectively alleviate the contradiction between the growing demand for fitness of citizens and the limited social stadium resources. Therefore, it is necessary to construct the resource-sharing model of stadiums from college or school opening to the society and ensure the sustainable opening of college or school stadium resources to the society.

Taking Beijing as an example, there are nearly about 1,000 stadiums available with different sizes in the whole city, which are free and charged, and how to make the citizens book a suitable stadium by using phones or computers for their own needs is in most urgent need to address the problem. The booking of the shared stadium can be regarded as task scheduling in MEC. The phones or computers are often deployed at the edge of the network [4–6]. Task offloading refers to the transfer of computing tasks from resource-constrained phones or computers to external platforms.

Mobile edge computing (MEC) is actually the cloud for real-time and personalized services, which offers the IT service environment and cloud computing capabilities. The related environment is characterized by proximity, low latency, and high bandwidth, and this service environment also offers exposure to real-time context information, and it can be leveraged by applications to create value. [7, 8].

For the problems of limited edge network resources and insufficient flexibility, it is necessary to introduce resource management and task scheduling mechanism to manage the edge network as a whole. Reasonable resource allocation and scheduling scheme are designed to make full use of edge resources to reduce latency and improve the user experience of booking stadiums. The scheme designing needs to

consider many factors such as the number of users, the distribution of computing resources, the number of computing resources, and so on, which maximize the utilization of resources at the edge. The optimized network latency makes it easier for citizens to book the most suitable stadium for them, which is bringing a better experience to users.

Accordingly, the contributions of this study are summarized as follows. (i) The offloading latency is modeled after analyzing the task computing latency and cloud computing latency. (ii) A joint scheduling of service caching and task algorithm is proposed to divide the problem into two relatively easy subproblems, which are joint scheduling of service caching and task.

The rest of this study is organized as follows. The related work is in Section 2. The modeling of resource allocation and task scheduling is studied in Section 3. In Section 4, joint scheduling of service caching and task algorithm is proposed. The experimental results are shown in Sections 5 and 6 that conclude this study.

2. Related Work

For the problems such as limited resources and insufficient elasticity of edge network, resource allocation and task scheduling mechanism should be introduced to manage the overall edge network. Currently, there are some research studies on resource allocation and task scheduling.

In order to cope with the problems of insufficient processing capacity and limited resources of terminal devices, computing offloading is introduced in MEC. EC offloading refers to the user-end offloading computing tasks to the MEC network to solve the device's shortcomings in resource storage, computing performance, and energy efficiency. Many strategies of resource allocation for MEC have been proposed. In [9], two resource allocation mechanisms were proposed, one is the auction-based mechanism, which generated the envy-free allocation, and the other is the linear program-based mechanism, which was no guarantee for envy. In order to recover the impact of limited caching service for task offloading, in [10], efficient convex programming was designed to solve the problem. In [11], a game-based distribution scheme was presented to allocate resources that offloading jointly and dynamically required in the mobile system. In [12], the nodes in the industrial internet of things using power transmission-scheduling scheme was proposed to lower the charging cost. In [13], the jointly partial task offloading and resource allocation of MEC with energy collection was proposed to take into account the network architecture of MEC. The computing tasks could be offloaded to the edge server, but the movability of users could not be predicted, and in [7], an effective solution based on relaxation and rounding off strategy was proposed to ensure the seamless migration problem by promising the movability of users. In order to safely perform the offloaded tasks, in [14], the Markov model was used to reduce the risk. In addition, a secure mechanism was proposed to lower the economic returns. There were most important two aspects in MEC, which were the security of task offloading and the connectivity of users, and in [15], the constraint of offloading rate and latency was studied to express the optimal solution of MEC. The power used in the edge server is huge, and in [16], the tasks were offloaded through searching the highest reward in order to solve the limited power of the edge server.

Task scheduling has been studied for a long time in cloud computing. Scholars usually optimize resource allocation and user task scheduling to reduce latency. With the development of MEC and the combination of cellular network, how to schedule EC tasks submitted by users has become an urgent problem to be solved. There are also some other methods for task scheduling in MEC. Based on the combination and strong coupling of resource allocation, in [17], a joint resource allocation and task scheduling method was proposed to divide the problem into two subproblems, which could optimize the coupling of MEC. Considering the cooperation of convolutional neural networks (CNNs) and MEC, in [18], the task scheduling problem could be divided into two problems, which were resource allocation and CNN task scheduling. In order to highlight the latency of task scheduling, the problem could be transformed as a minimum latency problem. It was necessary to transmit data in some special environments, and in [19], in order to transmit encrypted data, the task scheduling strategy was proposed. In [20], a dynamic planning energy-saved task scheduling algorithm was proposed to the minimum the latency constraint. In [21], MEC server was deployed at each area on the internet of vehicles, and authors proposed that the computing tasks could be scheduled to MEC server through wireless networks. In [22], the energy-based task scheduling method was proposed to optimize power consumption. In [23], a novel secure framework was proposed to offload the computing tasks to the MEC server in low latency and low power consumption. In [24], a random integer nonlinear planning method was proposed to schedule the computation-intensive tasks.

3. Modeling of Resource Allocation and Task Scheduling

Users book stadium on phones or computers and offload computing tasks to edge servers. The latency of task offloading can be divided into two parts, which are communication latency and computing latency. In the scenario of mobile edge network, wireless transmission latency is not the focus of this study that can be regarded as a constant. When the task of booking a shared stadium arrives at the base station, if the directly connected edge server has the ability to process the task, there is no transmission latency. If the task needs to be sent to other edge servers in the edge domain for processing, the related transmission latency needs to be increased. So, the problem of booking shared stadium is transformed into service caching and booking task scheduling.

3.1. Edge Caching. In this study, the tasks can be performed only if the resources exceed a certain threshold Th_m. There is a decision to be made about which types of services to cache so that the average latency is lowest. The decision variable a_{mn} is introduced to indicate whether service m runs on edge server n. The symbols in this study are shown in Table 1.

Table 1: Description of symbols.

Symbol	Description
P	Types of services to cache
Q	The number of servers that process booking shared stadium in the edge domain
P	$\{1,\ldots,p\}$, the set of services
Q	$\{1,\ldots,q\}$, the set of edge servers
Cf	Total CPU frequency
δ_{mn}	$m \in P$, $n \in Q$, direct request arrival rate of service m on edge server n
μ_m	$m \in P$, service rate coefficient for service m
L_m	$m \in P$, cloud computing latency for service m
l_m	$m \in P$, inner-domain forwarding latency of service m
Th_m	$m \in P$, minimum resources for service m operation
ω_m	m $\in P$, the weight of service m
task$_m$	$m \in P$, total task arrival rate of service m in edge domain
a_{mn}	$m \in P$, $n \in Q$, cache decision variable
e_{mn}	$m \in P$, $n \in Q$, arrival rate adjustment decision variable
ESf_{mn}	$m \in P$, $n \in Q$, resource allocation decision variable

3.2. EC Latency. Computing latency refers to the time required for the task of booking a shared stadium to be calculated in the edge server. The computing latency of a task is affected by multiple dimensions of computing resources at the same time. Only the CPU frequency is considered in this study. Let $1/\lambda$ and λ denote the parameters of negative exponential distribution and the service rate. It is assumed that when the CPU frequency allocated to the service exceeds the resource threshold Th_m, the service rate linearly increases with the increase in CPU frequency. For example, the coefficient of service m is μ_m, and the processor frequency allocated to edge server n is ESf_{mn}; then, the service rate is $\lambda_{mn} = \mu_m \times ESf_{mn}$. A simple M/M/1 queuing model can be used to model the computing latency of tasks [25].

3.3. Cloud Computing Latency. Due to the limitation of edge resources and the existence of sudden traffic, the edge cannot process the task of user offloading timely booking shared stadium. So the booking task may be sent to a remote data center for running, and the latency is called cloud computing latency in this study. The communication latency experienced by cloud computing is much larger than the communication latency L_{ic} in edge network. It is worth noting that cloud latency changes over time but over a shorter period of time, which can be assumed that the edge to cloud data center communication latency is a constant. Cloud data center has nearly infinite computing resources, which requires less computing latency than the edge and is relatively fixed. Therefore, for each service m, a constant L_m can be used to represent cloud computing latency within a period.

3.4. Optimization Model. The offloading latency is modeled after analyzing three kinds of latency. The purpose of this study is to optimize the latency performance of the edge network by adjusting the allocation of stadium resources and the scheduling of booking shared stadium tasks. The latency of each user of each service is a random variable, and a weighted average ω can be used to characterize the performance of the entire edge.

When a directly connected user of an edge server frequently uses a service, the resource quantity of the related service can be adjusted to increase ESf_{mn}. When there are many user requests for multiple services on the edge server, it needs to decide which services and what percentage of requests will be offloaded and where the offloading tasks will be performed. e_{mn} represents load adjustment for service m on edge server n. If e_{mn} is positive, it indicates that the load of the same service of other base stations in a certain domain needs to be offloaded to the local server. Otherwise, it means that a certain amount of user requests need to be transferred to other nodes for computing, which may be other edge servers or cloud data centers. When e_{mn} is determined, each node will forward the user task request to the controller according to the ratio $\eta_{mn} = e_{mn}/\delta_{mn}$, and then, the controller will decide the destination node to forward.

On the basis of the M/M/1 model in queuing theory, the computing latency of service m on edge server n is defined as follows.

$$l\_c_{mn} = \frac{1}{a_{mn} \times \mu_{mn} \times Esf_{mn} - (\delta_{mn} + \ln e_{mn})}. \quad (1)$$

Even for the same service, each edge server has its own queuing queue, and the l_c_{mn} is quite different. Weight balancing is required according to the number of users that need to be processed. The load proportion of each edge server is defined as follows.

$$\eta\_c_{mn} = \frac{\delta_{mn} + \ln e_{mn}}{\text{task}_m}. \quad (2)$$

So the average computing latency of service m at the edge is defined as follows:

$$\sum_{n \in q} l\_c_{mn} \times \eta\_c_{mn}. \quad (3)$$

In addition to the task of booking shared stadium performed on the edge, there are some booking tasks to offloading to the cloud for running, and the proportion of booking tasks performed on the cloud is determined by the adjustment of the arrival rate of each edge server δ_{mn}. The calculation of latency introduced during transmission is similar. Therefore, the transmission ratio can be defined as follows:

$$\eta\_\text{Cloud}_m = -\frac{\sum_{n\in q}\delta_{mn}}{\text{task}_m},$$
$$\eta\_l_m = -\frac{\sum_{n\in q}|\delta_{mn}|}{\text{task}_m}. \tag{4}$$

The average latency can be defined as follows.

$$l\_\text{Cloud}_m = \eta\_\text{Cloud}_m \times L_m,$$
$$\eta\_l_m = \eta\_l_m \times l_m. \tag{5}$$

To sum up, for a certain service m, the average latency can be defined as follows.

$$l\_a_m = \sum_{n\in q} l\_c_{mn} \times \eta\_c_{mn} + \eta\_\text{Cloud}_m \times L_m + \eta\_l_m \times l_m. \tag{6}$$

By using the weight of services to perform weighted average, the complete expression of average latency in the edge domain can be defined as follows.

$$l\_a_m L = \sum_{m\in p}\sum_{n\in q}\frac{\omega_m}{task_m}\left(\frac{\delta_{mn} + \ln e_{mn}}{a_{mn}\times\mu_{mn}\times Esf_{mn} - (\delta_{mn} + \ln e_{mn})} + l_m \times|\delta_{mn}| - \frac{2}{3}L_m \times e_{mn}\right). \tag{7}$$

There are certain constraints on decision-making, which are summarized as follows.

(i) In [26], the convergent divides the population into multiple clusters by using a clustering strategy. To ensure the convergence of the calculation queue, the service rate should be higher than the request arrival rate, which can ensure that the task of booking shared stadium can be performed faster and improve user experience; then,

$$a_{mn}\times\mu_{mn}\times Esf_{mn} > \delta_{mn} + \ln e_{mn}, \quad \forall m \in P, \forall n \in Q. \tag{8}$$

(ii) The edge server can forward the directly connected tasks at most, and the service arrival rate cannot be less than zero after adjustment; then,

$$\delta_{mn} + \ln e_{mn} \geq 0, \quad \forall m \in P, \forall n \in Q. \tag{9}$$

(iii) Only internal demands are processed in the edge domain, and the remaining demands are performed by the cloud. Therefore, the sum of service arrival rate adjustment should be less than or equal to zero; then,

$$\sum_{n\in q}\delta_{mn} \leq 0, \quad \forall m \in P. \tag{10}$$

(iv) The total resources allocated to different services on the same edge server cannot exceed the resource limitation; then,

$$\sum_{m\in p} a_{mn}\times Esf_{mn} \leq S, \quad \forall n \in Q. \tag{11}$$

(v) The CPU frequency allocated for a service should be greater than or equal to the minimum threshold; then,

$$Esf_{mn} \geq Th_m, \quad \forall m \in P, \forall n \in Q. \tag{12}$$

4. Joint Decision Algorithm

The intelligent scheduling method supporting the stadium sharing problem is a mixed-integer nonlinear programming, and the global optimal solution cannot be found in polynomial time. In this study, a joint scheduling of service caching and task algorithm is proposed to divide the problem into two relatively easy subproblems, which are joint scheduling of service caching and task. The subproblem of service caching is to make a decision on the decision variable a_{mn}. In service caching, a greedy strategy is proposed to determine the service that should be cached for each edge server. The joint task scheduling is to further optimize resource allocation and task scheduling on the basis of the determined service caching.

4.1. Service Caching Subproblem. The service caching subproblem is a zero-one programming problem, which is tightly coupled with subsequent subproblems. When the number of edge servers controlled by the controller is large, the time of the conventional solution will rapidly increase and it is difficult to solve. Therefore, this study presents a greedy strategy to determine the service caching on the edge server, which can greatly reduce the solution time and avoid excessive performance loss.

Caching of selected service at the edge is essential to allow more tasks of booking shared stadium that users offload to be performed at the edge without offloading to the cloud. Therefore, the task arrival rate of service is an important reference index. In addition, different tasks have different tolerance to latency, and tasks with lower tolerance and higher weight should be satisfied first. To balance the above two points, the "cache priority index" is introduced as shown in equation (13). Each edge server is ranked according to priority criteria and allocated computing resources according to minimum operating requirements until the server resources are exhausted. For the service to which computing resources are allocated, the service caching decision variable a_{mn} is set to 1.

Input: the number of iterations iN
Output: a_{mn}, ESf_{mn}, e_{mn}
(01) Initialization: logarithmic barrier $lb = 0.01$
(02) Initialization: $a_{mn} = 0$, $\forall m \in P, \forall n \in Q$
(03) **for** edge server $n \in Q$, **do**
(04) **for** service $m \in P$, **do**
(05) compute service cache priority index $cpi_{mn} = \omega_m \times \delta_{mn}$;
(06) **end for**
(07) Rank cpi_n in reverse order
(08) Resource surplus $rs = C$
(09) **for** service $m \in$ service cache order co_m, **do**
(10) **if** $rs \geq Th_m$, **then**
(11) amn = 1
(12) $rs = rs - Th_m$
(13) **end if**
(14) **end for**
(15) **end for**
(16) $ESf_{mn}^{(0)} = Th_m, e_{mn}^{(0)} = \delta_{mn}, \forall m \in P, \forall n \in Q$
(17) **for** $iN = 1 \longrightarrow iN$, **do**
(18) $ESf_{mn}^{iN} = ESf_{mn}^{iN-1} + (\partial L / \partial E\, Sf_{mn})$
(19) $e_{mn}^{iN} = e_{mn}^{iN-1} + (\partial L / \partial e_{mn})$
(20) **end for**

Algorithm 1: Joint scheduling of service caching and task.

$$cpi_{mn} = \omega_m \times \delta_{mn}, \quad \forall m \in P, \forall n \in Q. \tag{13}$$

4.2. Joint Scheduling of Service Caching and Task. There may be a resource surplus that needs to be allocated among the services that are cached by the edge server once the caching types are determined. The scheme of allocating computing resources will affect the waiting time of service queues. To optimize the latency, it is necessary to determine the proportion of computing tasks in the edge and cloud. Therefore, there are two decisions to be made. (i) The computing resources to be allocated for each service are determined by ESf_{mn}. (ii) The proportion of tasks that needs to be adjusted is determined by e_{mn}.

The two decision problems cannot be separately made, and the results need to be obtained in an optimization process. After the service caching decision variable a_{mn} is determined, the original objective function is transformed from the original mixed-integer nonlinear programming problem to the general nonlinear programming with constraint problem. The decision Algorithm 1 can be summarized as follows.

5. Experiment and Results Analysis

5.1. Performance Metrics. The algorithm in this study is proposed to reduce the latency of booking shared stadium request and improve user experience. The following performance metrics are set in this study. (i) The average latency of users in the system, which is also the target of direct optimization of the algorithm. (ii) The satisfaction rate of quality of experience (QoE) can reflect whether it is reasonable to take the average latency as the optimization goal. The settings for the simulation parameters are shown in Table 2.

Table 2: Simulation settings.

Parameter	Setting
Cf	4.0 GHz
δ_{mn}	0.75
L_m	0.06 s
l_m	0.02 s
Th_m	5
ω_m	0.24
e_{mn}	0.83
ESf_{mn}	2.9 GHz

A QoE latency limit is set in each edge decision sample. When the user's latency is smaller than the QoE latency limit, QoE requirements can be considered to be met. The average latency of each queue can be obtained through the queuing model, so this study directly takes whether the average latency is less than the QoE latency limit as the standard of whether users meet QoE. When machine learning meets congestion control, in [27], the performance of reinforcement learning-based congestion control algorithms was explored.

Edge load refers to the computing offloading requests generated by users in the edge domain within a unit time, that is, the arrival rate of requests. In this study, a benchmarking value is designed for the load size, which reflects the edge load relative to the edge affordability, so as to get rid of the influence of service type p and the number of edge servers q. It is assumed that each edge server has a computing resource of Cf (such as the basic frequency of CPU).

5.2. Result Analysis. Figure 1 shows the effect comparison of whether or not edge cooperation is performed in the edge domain. Three classical resource allocation and task

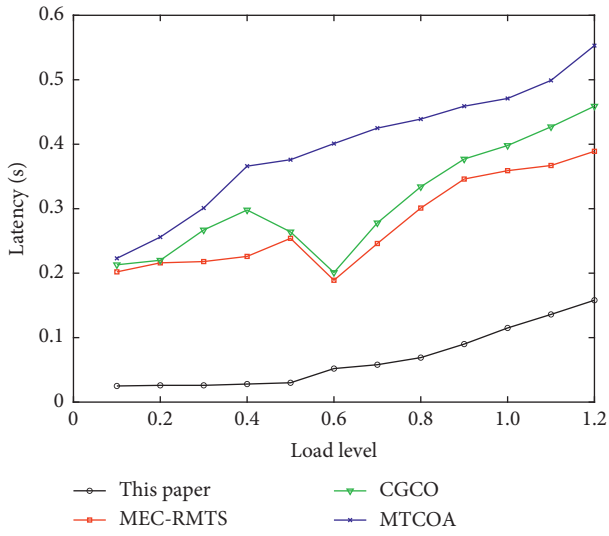

Figure 1: The comparison of latency.

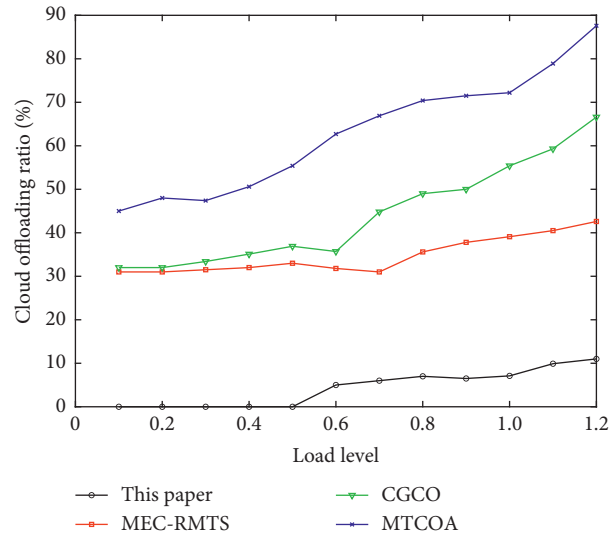

Figure 2: The comparison of cloud offloading.

scheduling algorithms in recent years are selected for comparison, which are MEC-based resource management and task scheduling (MEC-RMTS) [28], coalitional game-based cooperative offloading (CGCO) [29], and multiservice task computing offload algorithm (MTCOA) [30]. The MEC-RMTS framework is used for efficient task offloading in the internet of things, CGCO is a cooperative offloading algorithm based on the coalitional game, and MTCOA solves the multiservice task offloading problem. The data in the figure are measured when $p = 10$ and $q = 20$. Each edge server separately makes service caching and scheduling decisions.

It can be seen from Figure 1 that the proposed algorithm in this study with edge cooperation idea can achieve lower average system latency at lower load level. This is because the use of idle edge servers to provide certain computing services for neighbor nodes can improve the overall performance of the system. With the increase in edge load, the average latency of the three baselines gradually increases, which indicates that edge resources are unable to process edge load, and more tasks need to be transferred to the cloud for running. Edge cooperation can improve the stress resistance of the edge system. Through the analysis of the results, it is found that the proposed algorithm in this study using edge cooperation reduces the time latency by 70% on average compared with the algorithm without edge cooperation.

Figure 2 shows the proportion of the calculated tasks transferred to the cloud for running, which is called the proportion of cloud offloading. It can be seen from Figure 2 that the proposed algorithm in this study can significantly reduce the cloud offloading ratio under low and middle pressure, so as to achieve lower average latency. Under heavy load, edge cooperation cannot continue to reduce latency and degenerates into uncooperative mode.

Figure 3 shows the calculated proportion of tasks that are transmitted but not run in the above process. More network transmission may introduce more transmission

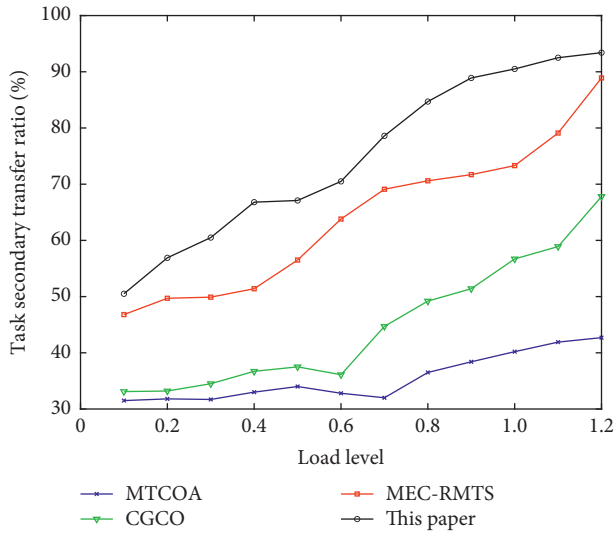

Figure 3: The comparison of task secondary transfer.

effort, but it is also more conducive to resource concentration. It can be seen from Figure 3 that the proposed algorithm in this study uses the method of secondary transfer of more tasks to reduce the pressure of local task running.

Figure 4 shows the statistics of the QoE satisfaction rate of users with different scheduling algorithms (latency is limited to 0.5 s). As can be seen from Figure 4, the proposed algorithm in this study can guarantee a better QoE satisfaction rate under low pressure. No matter which scheduling algorithm, the QoE satisfaction rate of users decreases with the increase in edge load pressure. However, no matter in which load range, the proposed algorithm in this study can achieve better performance, which also verifies that the proposed algorithm in this study meets the design goal of improving edge peak stress resistance.

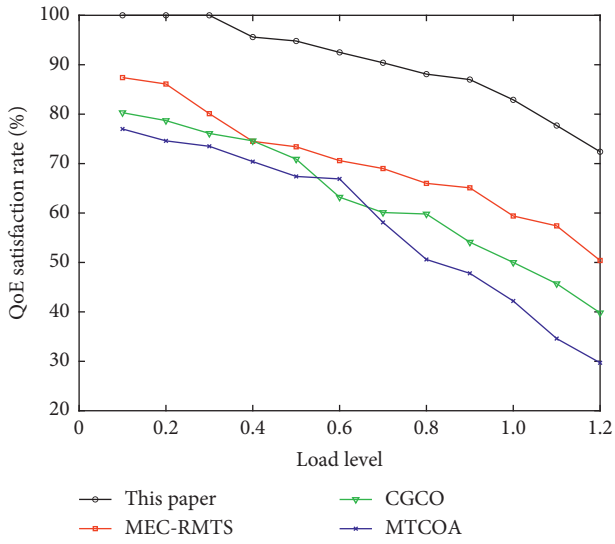

Figure 4: The comparison of QoE satisfaction rate.

6. Conclusions

National fitness is imperative, but a major problem encountered in the implementation is the lack of stadiums. The sharing of stadium resources from college or high school and society has become the direction of change under the current situation of insufficient stadium resources, which can effectively alleviate the contradiction between the growing fitness needs and the limited social stadium resources. In this study, the characteristics of edge resource allocation and task scheduling are analyzed, and joint scheduling of service caching and task algorithm is proposed. According to the minimum resource allocation requirements of service caching, the decision-making problem of the edge network is modeled as a joint resource allocation and task scheduling model, and the resource allocation and task scheduling scheme are obtained in an optimization process. Experimental results demonstrate that the proposed algorithm can effectively integrate and utilize edge resources, which can also improve the latency and QoE. Currently, there are few studies on task collaborative scheduling between mobile devices and MEC servers.

The algorithm proposed in this study is for the tasks that cannot be decomposable, that is, all the computing tasks can only be locally performed or on the MEC server. However, there are many mobile applications that can be fine-grained in practice. For such computing tasks, some subtasks can be offloaded to the MEC server for execution, while others are locally offloaded for execution. More detailed offloading decisions need to be reformulated, which can further reduce task latency.

Data Availability

The data used to support the findings of the study are included within the article.

Conflicts of Interest

The author declares no conflicts of interest.

Acknowledgments

This paper was supported by the Social Science Research Planning Project of Jilin Province Education Department (grant no. JJKH20210426SK).

References

[1] J. Wang, J. Li, and Y. Pan, "The spatial effect of Nanjing's fitness strategy: considering citizens' fitness status," *Arabian Journal of Geosciences*, vol. 14, no. 16, 2021.

[2] Q. Guo and B. Li, "Role of AI physical education based on application of functional sports training," *Journal of Intelligent & Fuzzy Systems*, vol. 40, no. 2, pp. 3337–3345, 2021.

[3] J. Kim and Y. Choi, "The effect of short-term off-season cross-country ski training on body composition, physical fitness, and isokinetic muscle functions of cross-country skiers," *Journal of Mens Health*, vol. 16, no. 1, pp. E63–E74, 2020.

[4] S. V. Simpson and G. Nagarajan, "A fuzzy based co-operative blackmailing attack detection scheme for edge computing nodes in MANET-IOT environment," *Future Generation Computer Systems*, vol. 125, pp. 544–563, 2021.

[5] C. Qu, P. Calyam, J. Yu et al., "DroneCOCoNet: learning-based edge computation offloading and control networking for drone video analytics," *Future Generation Computer Systems*, vol. 125, pp. 247–262, 2021.

[6] L. Zhao, K. Yang, Z. Tan et al., "Vehicular computation offloading for industrial mobile edge computing," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 11, pp. 7871–7881, 2021.

[7] Z. Liang, Y. Liu, T.-M. Lok, and K. Huang, "Multi-cell mobile edge computing: joint service migration and resource allocation," *IEEE Transactions on Wireless Communications*, vol. 20, no. 9, pp. 5898–5912, 2021.

[8] L. Qian, W. Wu, W. Lu, Y. Wu, B. Lin, and T. Q. S. Quek, "Secrecy-based energy-efficient mobile edge computing via cooperative non-orthogonal multiple access transmission," *IEEE Transactions on Communications*, vol. 69, no. 7, pp. 4659–4677, 2021.

[9] T. Bahreini, H. Badri, and D. Grosu, "Mechanisms for resource allocation and pricing in mobile edge computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 3, pp. 667–682, 2022.

[10] G. Zhao, H. Xu, Y. Zhao, C. Qiao, and L. Huang, "Offloading tasks with dependency and service caching in mobile edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 11, pp. 2777–2792, 2021.

[11] R. Roostaei, Z. Dabiri, and Z. Movahedi, "A game-theoretic joint optimal pricing and resource allocation for mobile edge computing in NOMA-based 5G networks and beyond," *Computer Networks*, vol. 198, Article ID 108352, 2021.

[12] W. Ejaz, M. Naeem, and S. Zeadally, "On-demand sensing and wireless power transfer for self-sustainable industrial internet of things networks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 10, pp. 7075–7084, 2021.

[13] J. Chen and Z. Wu, "Dynamic computation offloading with energy harvesting devices: a graph-based deep reinforcement learning approach," *IEEE Communications Letters*, vol. 25, no. 9, pp. 2968–2972, 2021.

[14] G. H. S. Carvalho, I. Woungang, A. Anpalagan, and I. Traore, "Optimal security-aware virtual machine management for mobile edge computing over 5G networks," *IEEE Systems Journal*, vol. 15, no. 3, pp. 3403–3414, 2021.

[15] W. Wu, X. Wang, F. Zhou, K.-K. Wong, C. Li, and B. Wang, "Resource allocation for enhancing offloading security in NOMA-enabled MEC networks," *IEEE Systems Journal*, vol. 15, no. 3, pp. 3789–3792, 2021.

[16] M. Song, Y. Lee, and K. Kim, "Reward-oriented task offloading under limited edge server power for multiaccess edge computing," *IEEE Internet of Things Journal*, vol. 8, no. 17, pp. 13425–13438, 2021.

[17] C. Xu, G. Zheng, and X. Zhao, "Energy-minimization task offloading and resource allocation for mobile edge computing in NOMA heterogeneous networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 16001–16016, 2020.

[18] R. Chai, X. Song, and Q. Chen, "Joint task offloading, CNN layer scheduling, and resource allocation in cooperative computing system," *IEEE Systems Journal*, vol. 14, no. 4, pp. 5350–5361, 2020.

[19] Y. Zhang, Y. Liu, J. Zhou, J. Sun, and K. Li, "Slow-movement particle swarm optimization algorithms for scheduling security-critical tasks in resource-limited mobile edge computing," *Future Generation Computer Systems*, vol. 112, pp. 148–161, 2020.

[20] Y. Zhang and J. Fu, "Energy-efficient computation offloading strategy with tasks scheduling in edge computing," *Wireless Networks*, vol. 27, no. 1, pp. 609–620, 2021.

[21] M. Pang, L. Wang, and N. Fang, "A collaborative scheduling strategy for IoV computing resources considering location privacy protection in mobile edge computing environment," *Journal of Cloud Computing*, vol. 9, no. 1, 2020.

[22] Y. Chen, Y. Zhang, Y. Wu, L. Qi, X. Chen, and X. Shen, "Joint task scheduling and energy management for heterogeneous mobile edge computing with hybrid energy supply," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8419–8429, 2020.

[23] S. Anoop and J. A. P. Singh, "Multi-user energy efficient secured framework with dynamic resource allocation policy for mobile edge network computing," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 7, pp. 7317–7332, 2020.

[24] Q. Zhang, L. Gui, F. Hou, J. Chen, S. Zhu, and F. Tian, "Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud RAN," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3282–3299, 2020.

[25] Z. Ma, C. Zhang, L. Zhang, and S. Wang, "Energy saving strategy and nash equilibrium of hybrid P2P networks," *Journal of Parallel and Distributed Computing*, vol. 157, pp. 145–156, 2021.

[26] L. Ma, S. Cheng, and Y. Shi, "Enhancing learning efficiency of brain storm optimization via orthogonal learning design," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 11, pp. 6723–6742, 2021.

[27] H. Jiang, Q. Li, Y. Jiang et al., "When machine learning meets congestion control: a survey and comparison," *Computer Networks*, vol. 192, Article ID 108033, 2021.

[28] M. T. Quasim, "Resource management and task scheduling for IoT using mobile edge computing," *Wireless Personal Communications*, 2021.

[29] X. Yang, H. Luo, Y. Sun, J. Zou, and M. Guizani, "Coalitional game-based cooperative computation offloading in MEC for reusable tasks," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12968–12982, 2021.

[30] S. Song, S. Ma, J. Zhao, F. Yang, and L. Zhai, "Cost-efficient multi-service task offloading scheduling for mobile edge computing," *Applied Intelligence*, 2021.