Joint Optimization Scheme of Multi-Service Replication and Request Offloading in Mobile Edge Computing

Chenxi Li, Guanghui Li, Shihong Hu, Chenglong Dai and Dong Li

December 17, 2021

Joint Optimization Scheme of Multi-service Replication and Request Offloading in Mobile Edge Computing

Chenxi Li, Guanghui Li*, Shihong Hu, Chenglong Dai, and Dong Li

School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi, Jiangsu 214122, China
`corresponging author: ghli@jiangnan.edu.cn`

Abstract. To meet the ever-increasing service quality requirements of end-users and enable delay-sensitive applications to be completed within a tolerable time, Mobile Edge Computing (MEC) offloads the request of users to the edge servers that are closer to the end equipment. However, deploying a single service replication in an appropriate edge node is difficult to deal with all requests of users for multiple services. In addition, after the service replication is deployed at the edge node, a corresponding user request offloading scheme is also required. Considering the heterogeneity of edge servers, this paper studies the joint optimization problem of multi-service replication and request offloading. Firstly, we present an edge computing architecture with multi-service replication, and define the multi-service replication and request offloading as a joint optimization problem. Secondly, a multi-service replication algorithm called Multireplicas Greedy Best (MGB) is proposed to solve the joint optimization problem. Finally, the simulation experiments are carried out. The experimental results show that the proposed algorithm can effectively reduce the overall delay compared with the random strategy, the nearest node offloading strategy, the particle swarm algorithm, and the greedy algorithm.

Keywords: Mobile Edge Computing · Service Replication· Request Offloading· Multi-objective Optimization.

1 Introduction

In recent years, Internet of Things (IoT) technology has developed rapidly[1,2]. Statistics show that there are about 30 billion IoT devices in the world by 2021. Furthermore, it is estimated that by 2025, mobile IoT devices will grow to 75 billion[3]. Many time-sensitive applications on IoT devices require fast and safe services, such as video surveillance, traffic flow mapping, personalized multimedia, and data sharing. These requirements usually need interactive behavior with high Quality of Service(QoS)[4], but most of the embedded processors and storage capacity of IoT devices have limited resources and cannot be used for large-scale computing, storage, and communication[5]. The traditional method is to upload all the requirements to the cloud server data center for processing. As a typical example, Mobile Cloud Computing (MCC) was introduced as a solution. MCC is a resource scheduling framework using a resource-sharing model, which can solve the problem of insufficient resources used by IoT devices, especially those applications that require a lot of computing resources.

Supporting mobile applications through public Wi-Fi networks has received significant research attention. The 5th Generation (5G) Mobile Communication Technology and mobile IoT devices (i.e., mobile phones, tablets) have promoted delay-sensitive services such as AR/VR, healthcare, intelligent transportation, and location positioning[4]. However, MCC is not enough to support the communication and computing of IoT devices in 5G. Therefore, Mobile Edge Computing (MEC) has been attached to great importance in recent years. The core idea of MEC is to offload some requests to edge servers instead of offloading them to the cloud server for processing. In the MEC architecture, computing and storage resources are usually deployed at the edge of the network, and edge servers are deployed in the wireless access network. The edge nodes can be any computing resource of the network to offload the user requests. In this case, the offloading request from the user equipment to each service replication can be served by the nearest edge server first, thereby avoiding long-distance network transmission.

A challenging problem in the MEC system is the scarcity of resources. Compared with cloud servers, the resources of edge nodes are limited. Therefore, it is an unworkable scheme to run large-scale applications on a single edge node in MEC. Moubayed et al.[6] proposed an effective way to solve this problem, allowing users to run their application requests on multiple edge nodes. The essential issues that need to be resolved are to find the optimal replication location of the service on multiple nodes and find the location where users request to offload.

In the field of cloud computing, Thai et al.[7] proposed some service replication strategies, which tended to place service replications on the cloud. Compared with cloud servers, edge nodes are more widely distributed in the network topology, and the storage capacity of edge nodes is more limited. Therefore, the strategies proposed by Thai cannot be applied directly. Many studies have discussed service replication placement strategies. Naas et al.[8] proposed some service placement strategies in the edge computing environment. However, these strategies only focused on placing a single service replication to the appropriate edge node. When there are multiple data consumers from different locations requesting the same service, a single service replication cannot meet the latency requirements of all consumers. Multi-service replication was mentioned by the work in[9], the authors investigated multi-service replication as a stochastic game, but they ignored the user-requested offloading problem.

To reduce user request latency under the framework of MEC, researchers have done a lot of work. It can be roughly divided into two categories: One category is to optimize the scheme of service replications. For example, Naas constructed the replication problem as a Generalized Assignment Problem (GAP)[8], which used edge node location information to reduce the total delay. The second category is to find the optimal edge nodes to offload user requests.

There is a big difference between a single-user scenario and a multi-user scenario in reality. Making an optimal offloading decision under the condition of considering the resource allocation of edge nodes is a complex problem. However, they ignored the impact of edge node resource allocation on computation offloading.

The multi-service placement problem has been proven to be NP-hard, and use CPLEX MILP solver to find out the best replication placement will take a lot of time [8]. Therefore, to effectively solve the problem, we propose a joint optimization scheme of multi-service replication and request offloading mechanism based on an enhanced greedy algorithm. This mechanism realizes the optimal allocation of edge node resources, and makes an offloading calculation decision.

Our main contributions of this study are highlighted as follows:

(1) Considering the diversity of user requests and the heterogeneity of edge nodes in MEC, we model the multi-replication service placement problem as a joint optimization problem of edge node service replication and user request offloading.

(2) With the primary goal of minimizing total delay and algorithm execution time, we propose a heuristic algorithm Multireplicas Greedy Best (MGB) based on an enhanced greedy strategy.

(3) We evaluate the performance of the proposed algorithm by simulation experiments. The results show that MGB outperforms the baselines in terms of the reduction of total delay and execution time.

2 Multi-Edge Computing System Model

The edge computing architecture proposed in this paper is shown in Fig.1. The system consists of a remote cloud server, multiple edge nodes, and IoT devices.

Multi-replica edge computing architecture is a hierarchical structure. From the bottom to the top, IoT devices encompass wireless end-users objects such as sensors, robots, smartphones, and cameras, which are in the bottom layer. These devices generate continuous and periodic service requests, which need to be processed and stored in edge nodes or cloud servers. The middle layer above the IoT device layer is the edge network layer. This layer is divided into two parts: signal receiving and request processing. Edge nodes maintain computation services for users via various virtualization techniques (e.g., containers and virtual machines). Due to the heterogeneity of edge nodes, the network conditions and resources of edge nodes may be different. The request processing part represents the physical infrastructure for processing the request, which contains the physical machines called MEC nodes. Finally, there is a cloud layer composed of cloud servers on top of the edge network layer. The cloud servers store heterogeneous services and cover the geographic area where the edge nodes are located. The edge nodes and IoT devices are connected via wireless networks. Generally, services can be replicated at any edge node, and when an IoT device generates a request, the request can be processed by the edge node.

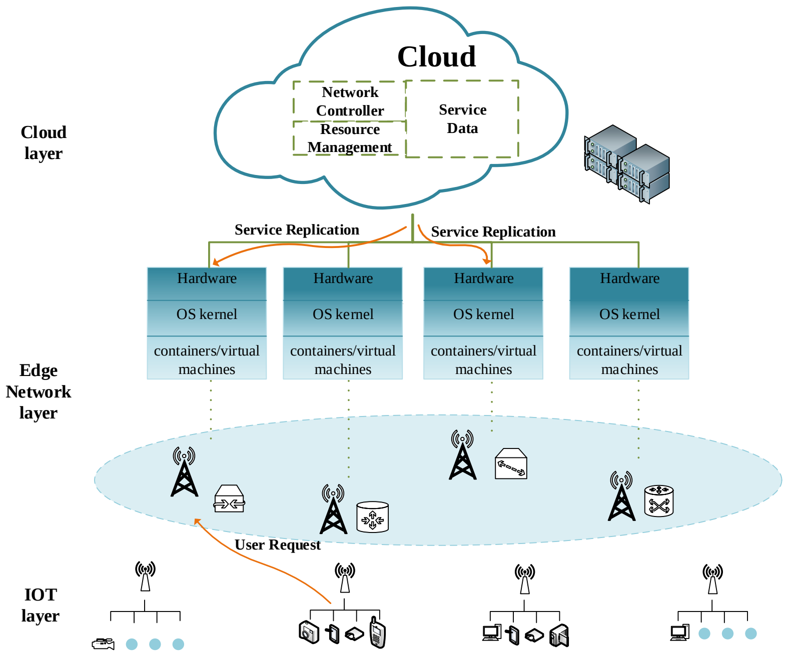


Fig. 1. Multi-replica Edge Computing Architecture.

Heterogeneous edge nodes may include gateways(GW), Region Point of Presences (RPOPs), and Local Point of Presences (LPOPs), etc., denoted by $C = \{C_1, ..., C_k, ..., C_m\}$.In the rest of this paper, when referring to edge nodes, k is used interchangeably with C_k to simplify the representation. The services in the cloud server can be replicated on the edge node, which is marked as $S = \{S_1, ..., S_i, ..., S_n\}$, and i will be used interchangeably with S_i later when referring to the service replication. IoT devices generate service requests, and these services will be processed on edge nodes. The generated user requests are denoted as $R = \{R_1, ..., R_j, ...R_q\}$, and we use j interchangeably when referring to request R_j. The service data block for service replication and user request is denoted by $data_i$, which is the unit block for data transmission.

The overall delay consists of the following three parts:

(1) V_{ik} : The delay required for the service replication i to be replicated on the edge node k. The edge nodes are usually heterogeneous, making the speed at which they download the service replication from the cloud server different. Therefore, the heterogeneity of edge nodes must be considered to calculate the service replication time. It can be formulated as follows:

$$V_{ik} = \frac{data_i}{\delta} \cdot td_{ik} \tag{1}$$

Here, td_{ik} is the delay required for the edge node k to download the unit data block of type i request from the cloud server, $data_i$ is the size of the replication i transferred between cloud and edge, δ is the unit data block to be transmitted.

(2) X_{ik} : The delay required to run the service i on the edge node k. Because the edge node is heterogeneous, the delay of processing unit data on the edge node is different, represented by tu_k. λ is the impact ratio of data block size to the service operation, which can be a given positive constant between [0,1] according to the actual situation. The relationship between them is given as follows:

$$X_{ik} = \frac{\lambda data_i}{\delta} \cdot tu_k \tag{2}$$

(3) T_{ijk} : The delay required of the service i to offload the user request j at the edge node k through wireless communication. The delay is proportional to the total amount of data requested $data_i$, and the delay is inversely proportional to the wireless communication transmission rate w_{jk} from the user request j to the edge node k.

$$T_{ijk} = \frac{\lambda data_i}{w_{jk}} \tag{3}$$

The wireless communication transmission rate w_{jk} can be represented by a wireless transmission model based on Shannon–Hartley theorem[11], which is shown in formula (4), where d represents the distance from the terminal device to the edge, v indicates the path loss exponent, and P represents the current device transmit power, h denotes the channel fading coefficient and ω_0 represents the Gaussian white noise power. Based on Shannon's theorem, when a request is offloaded to an edge node with bandwidth B, the transmission rate can be expressed as:

$$w_{jk} = B \log_2 \left(1 + \frac{P|h|^2}{\omega_0 d_{jk}^{\vartheta}} \right) \tag{4}$$

Therefore, the overall delay cost consists of the time required for the service replication i to be replicated on the edge node k, the time required to run the request i on the edge node k, and the user j to request service i at the edge node k through the wireless communication link. It is worth noting that the more service copies that are replicated on edge nodes, the more time it takes to store services, and the time for users to request services may decrease; on the contrary, if fewer service copies are replicated on edge nodes, the storage service time will be reduced, but at the same time the user request service time may increase.

There are multiple replication edge nodes in the edge computing architecture, and there may be multiple options when the service is replicated. We set a decision variable p_{ik}. Assuming that service i is stored in the cloud server, and the data block is transferred to the edge node k during service replication. In order to indicate the replicate state of service i on edge node k, the decision variable p_{ik} is assigned binary value: when the service i is replicated on the edge node k, $p_{ik} = 1$; otherwise, it is 0.

Furthermore, to indicate whether the user request j is offload to service i, the decision variable q_{ij} is defined as follows: when $q_{ij} = 1$, it means that the user j is to request the service i, and 0 otherwise.

Similarly, the decision variable r_{jk} represents the situation that the request j is processed on the edge node k. $r_{jk} = 1$ means that the request j is processed on the edge node k; otherwise, it is 0. Only when there is a service replication on the edge node k, the request j can be processed on the edge node k. In addition, for each user, the request only needs to flow to one edge node, so the following condition should be satisfied:

$$\sum_{k=1}^{K} r_{jk} = 1 \tag{5}$$

Based on the above discussion, we modle the joint optimization problem of service replication and user request offloading as follows:

$$\text{Minimize } \sum_{k=1}^{K} p_{ik} \cdot V_{ik} + \sum_{j=1}^{J}\sum_{i=1}^{I}\sum_{k=1}^{K} q_{ij} \cdot r_{jk} \cdot (X_{ik} + T_{ijk}) \tag{6}$$

s.t.

$$\sum_{i=1}^{I} data_i p_{ik} \le \mathrm{stor}_k, \forall k = 1, 2, 3... \tag{7}$$

$$\sum_{k=1}^{K} r_{jk} = 1 \tag{8}$$

$$p_{ik} = 0, \forall i, \forall k : data_i > stor_k, i = 1, 2, 3..., k = 1, 2, 3... \tag{9}$$

The objective of this function is to find the optimal service replication and user request offloading plan to minimize overall delay. The constraint (7) ensures that the service replication stored on the edge node must be smaller than the storage capacity $stor_k$ of the edge node. Constraint (8) can guarantee that each request is completely offloaded to an edge node. Constraint (9) indicates that when the storage capacity of the service replication is greater than the storage capacity of the edge node, no replication will perform at the current node.

3 Joint Optimization Algorithm of Service Replication and User Request Offloading

Service replication and user request offloading is a complex multi-objective optimization problem, which involves a trade-off between two conflicting objectives. This paper attempts to find an effective solution for this problem based on an enhanced greedy strategy. This strategy is more efficient and can reduce the total delay cost of service replication significantly.

The data transmission process of service replication and user request is shown in Fig.1. Before services are replicated, the cloud server stores services which can handle various user requests. In order to ensure that each user request can be offloaded to edge nodes, the number of service replications should exceed the number of user requests. At the same time, due to the limited storage capacity of the edge node, it is necessary to determine whether the remaining storage capacity of the edge node can accommodate a new one before replicating. We traverse all edge nodes and create the array of ready-to-use service replication solutions as a_i.

After determining the location of the service replication at the edge node, the user request needs to be offloaded to the edge node. The request generated by IoT devices can be offloaded to the edge node through wireless communication. In order to find the optimal edge node to offload the user request, an original idea is offloaded the request to the closest edge node of this user. However, there are two problems. One is that the nearest edge node may not replicate the service; the other is that the computing capability of the edge node closest to the user is uncertain. So it is necessary for each edge node to determine whether the service can handle the offloading request and consider each edge node computing capability.

We design a joint optimization algorithm for service replication and user request offloading based on an enhanced greedy strategy in order to improve efficiency while maintaining the quality of the solution. Algorithm 1 is the proposed service replication algorithm. The first line of the algorithm traverses all replication schemes and records all replication schemes as $AL = \{a_1, a_2, ..., a_n\}$. Lines 2-12 iteratively calculate the delay time of each service replication scheme. Before all the replicate schemes are traversed, the service delay time calculation will be repeated. In each iteration, the user request offloading time is updated to minimize total user request time according to Algorithm 2. Lines 6-8 determine whether the current replicate plan exceeds the upper limit of the edge node storage capacity. If it exceeds, no service replication will be performed. Finally, we find the optimal replicate edge nodes.

After the algorithm determines the replicate location of all services, it also needs to determine user request offloads to which edge node. This paper sets a heuristic rule: all IoT users need to consider the current requesting users and the remaining unallocated edge node users. We design an algorithm based on an enhanced greedy strategy called Multireplicas Greedy Best (MGB). The algorithm runs in two stages: In the first stage, it determines the best replication location among the edge nodes and finds the minimum replicate cost of the service on the edge node. In the second stage, the algorithm considers which edge node to offload the users request.

Algorithm 1 Optimal Multi-service Replication

```
Require: Set of edge nodes $C$, Service data size $DATA$, Edge node storage capacity
    $stor$
Ensure: Optimal strategy of replication and overall delay
 1: Enumerate all replication strategies $AL = \{a_1, a_2, ..., a_n\}$
 2: for all $a_i \in AL$ do
 3:     for all $k \in C$ do
 4:         Calculate the replicate delay $V_{ik}$ through Formula (1)
 5:         Call the Optimal Request Offloading algorithm
 6:         if $\sum_i^I data_i p_{ik} >$ stor$_k$ then
 7:             The service can not replicate on edge nodes, so offload the user request to
                the cloud server
 8:         end if
 9:     end for
10:     Calculate the replication delay for all edge nodes $V = \sum_k^K V_{ik}$
11:     According to Formula (6) calculate the minimum overall delay
12: end for
13: Find the optimal multi-service replication strategy with the minimum overall delay
    for all replications.
```

Algorithm 2 is the user request offloading algorithm. The input is the service replication strategy of Algorithm 1, and the overall delay of all user requests is output. Line 1 of the algorithm initializes the minimum offloading delay. Lines 2-7 describe the request offloading that takes into account the cost of communication between users and edges. According to formulas (2) and (3), the edge nodes are arranged according to the offloading delay and the service running delay. Line 8 indicates the offloading delay and the service running delay in an arrangement. To choose among candidate replications the best node to offload the request, lines 9-21 describe a feasible offloading solution for all user requests. To find these solution, we first use greedy algorithm. However, the smallest offloading delay of the user request for offloading may not be globally optimal in overall delay. So we proposed MGB, which takes into account the smallest offloading delay node and the second smallest offloading delay node. In MGB, we calculate the overall delay of the smallest offloading delay node and the second smallest offloading delay node. If the overall delay offloading at the second smallest offloading delay node which is less than overall delay of the smallest offloading delay node, the edge node with the second smallest delay will be offloaded.

Algorithm 2 **Optimal Request Offloading**

```
Require: Replication strategies $a_i$, Replication delay $V$, IoT user request $R$
Ensure: minimum offloading delay of all user request
 1: Initialize $minDelay = MaxValue$
 2: Traverse all user requests $R = \{R_1, ..., R_j, ..., R_q\}$
 3: for all  $R_j \in R$ do
 4:     Determine whether there is a replication store in node, and offloading the service
        when the edge node has a replication
 5:     According to Formulas (2), (3) calculate the offloading delay and the service
        running delay $X_{ik}$ and $T_{ijk}$
 6:     Calculate $retOffloadingDelay[j] = \sum_{k=1}^{K} \sum_{i=1}^{I} X_{ik} + T_{ijk}$
 7: end for
 8: Arrange the $retOffloadingDelay$ in order
 9: for all  $R_j \in R$ do
10:     Offloading the request $R_j$ on edge node with the smallest delay in arrangement
        $retOffloadingDelay$
11:     Calculate $OverallDelay = X + V + K$
12:     if $OverallDealy < minDelay$ then
13:         Take the current delay $minDelay = OverallDealy$
14:         Remove the edge node from the replicated edge node list
15:     else
16:         Offloading the request on edge node with the second smallest delay in ar-
            rangement $retOffloadingDelay$
17:         Calculate $OverallDelay = X + V + K$
18:         Take the delay $minDelay = OverallDealy$
19:         Remove the edge node from the replicated edge node list
20:     end if
21: end for
22: Find the optimal request offloading strategy with the minimum overall delay for
    all requests
```

4 Simulation and Performance Evaluation

To verify the effectiveness of the MGB strategy, we conducted simulation experiments. We compare the performance of the proposed algorithm with the random algorithm, greedy algorithm, nearest edge node offloading algorithm, and particle swarm algorithm.

4.1 Experimental Settings

The simulation experiments are conducted on an Intel 3.7 GHz Core i5 system with 8 GB RAM. The simulation scenario includes a set of IoT devices, edge nodes, and a cloud server infrastructure (as shown in Fig.1). The edge node is composed of heterogeneous devices such as GW, RPOP, and LPOP. Like literature [12]. This article sets the storage capacity of GW to 16GB, LPOP storage capacity is set to 32GB, the storage capacity of RPOP is set to 128GB, and the size of the cloud service center (Cloud Service, CS) is set to 128TB. As shown in Table 1:

Table 1. The storage capacity of Nodes.

DataHost	GW	LPOP	RPOP	CS
Storage capacity	16GB	32GB	128GB	128TB

To verify the effectiveness of the multi-service replication strategy, a wireless transmission model based on Shannon–Hartley theorem is used. To meet the needs of real users, this article uses 5G wireless transmission parameters as the wireless communication link. The channel bandwidth is denoted by B, assuming that there is no significant change in an edge-computing architecture, which is 3GHZ. The transmit power is denoted by P, which is uniformly distributed in the range of 100mW-320mW. Two different fading coefficients h are considered, which are 1.0 and 1.1, respectively. In the same channel, the Gaussian white noise power w conforms to the normal distribution. The consumption coefficient v is fixed to 1 in the experiment. As shown in table 2:

Table 2. Parameters of Wireless Communication Link.

Notation	Description	Value
B	wireless bandwidth	2MHz
P	transmission power	1mW-3.2mW
h	channel fading coefficient	1.0 and 1.1
w	white Gaussian noise power	Normal distribution $X - N(0, 50)$
v	path loss exponent	1

At the same time, we set the communication delay among different edge nodes as shown in Table 3.

Table 3. Delay of Different Wireless Communication Links.

Channel Type	IOT-GW	IOT-LPOP	IOT-RPOP	IOT-CS	GW-CS	LPOP-CS	RPOP-CS
Latency(ms)	5	10	20	100	5	10	20

4.2 Performance Index

The purpose of service replication and user request offloading in the edge computing architecture is to provide users a better experience when they request. Therefore, this experiment mainly evaluates two important indices for time-sensitive applications, namely, the overall delay of the edge replication and the execution time of MGB.

(1) Overall delay: This article defines the delay cost as the sum of the time required for the service copy to be replicated on the edge node, the time required to run the request on the edge node, and the time required for the user to request service at the edge node through the wireless communication link.

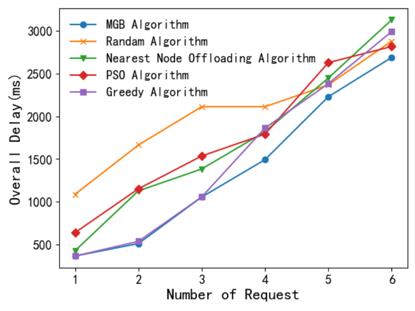
(2) Execution time: The execution time refers to the time from the simulation data generation to the completion of the service replication and user request offloading.

We compare the MGB algorithm with the following four baselines:

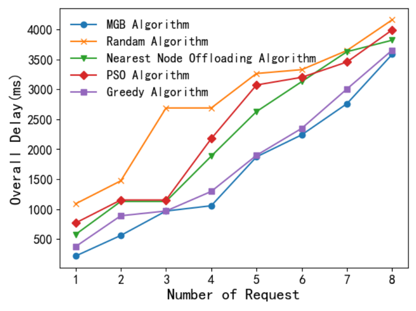
- Random algorithm: The replications are placed randomly in all edge nodes, and the user requests are offloaded randomly to all edge nods [13].
- The nearest edge node offload: This strategy offloads user requests to the edge node which is closest to the user, regardless of the service replication delay and edge node computing power [11].
- Greedy algorithm: Greedy algorithm is a classic heuristic algorithm, which can effectively solve the NP-hard problem [14]. This strategy finds the best offloading edge node for each user request as far as possible unless there is no available edge node.
- Particle swarm algorithm: Particle swarm algorithm (PSO) uses particle swarm optimization to perform user request offloading [15]. The algorithm finds the optimal solution by simulating the migration activities of natural bird groups. Each bird group represents a candidate solution to the optimization problem. The parameter settings in PSO have a great influence on the performance of the algorithm. Considering the execution time of the algorithm and the quality of the solution, we set 100 as the number of iterations and 0.5 as the maximum particle speed later in the experiment.

4.3 Experimental Results and Baselines

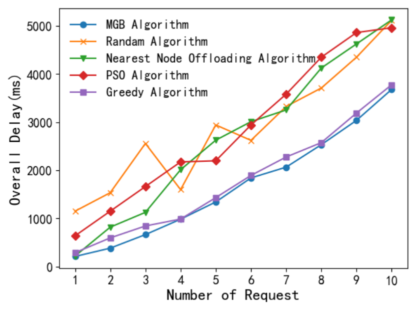
The total delay of various algorithms are shown in Fig. 2 when the number of service replication is 6, 8, 10, and 20, respectively. The x-axis is the number of offloading requests on the replicate service node, and the y-axis is the total delay between the service replication and the request offloading.



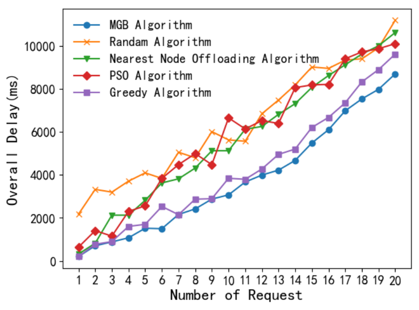
(a) Repliaction=6



(b) Repliaction=8



(c) Repliaction=10



(d) Repliaction=20

Fig. 2. Overall Delay under Different Number of Replications.

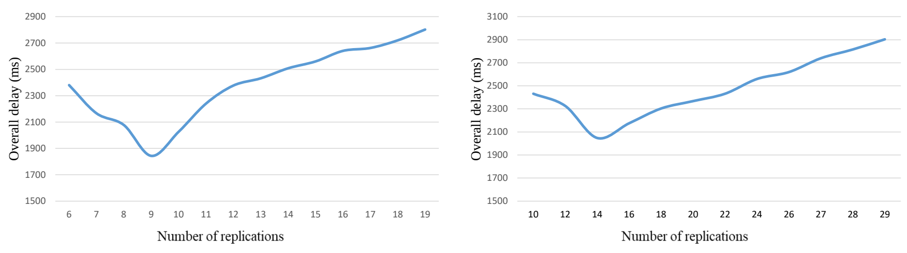
For each algorithm, we run ten times to get the average value of the overall delay time. It was found that when replication is 20, the MGB algorithm reduced the delay by 41.6% compared with random replication for service offloading. In addition, the MGB algorithm is reduced by 11.45%, which is 6.5% less than the particle swarm algorithm, and 1.9% less than the naive greedy algorithm. This is because the random algorithm does not consider the geographical distribution of user requests, the delay of offloading, and the processing capabilities of different edge nodes for services replication, so the delay is relatively high, and the nearest edge node offloading algorithm only considers the distance between the user request from the edge node. The PSO algorithm is a better solution, but the PSO algorithm has too many parameters, which makes it difficult to adjust those parameters quickly and effectively. The greedy algorithm can reach the optimal solution in some cases, but in more cases, it may easy to fall into a local optimal solution. In contrast, the MGB algorithm attempts to replicate fewer services to edge nodes and uses an enhanced greedy algorithm to offload user requests so as to avoid falling into a local optimum and greatly reduce the delay time. Table 4 shows the overall delay reduction ratio of MGB vs. baselines.

Where the second line represents the number of replications, and columns 2-5 give the delay reduction percentages compared with the baselines.

Table 4. Overall Delay Reduction Ratio MGB vs Baselines.

Baselines	Number of Replications			
	6	8	10	20
Random Algorithm	14.06%	23.84%	28.04%	41.60%
Nearest Node Offloading	10.13%	10.22%	13.64%	11.45%
PSO	6.68%	6.17%	7.43%	6.50%
Greedy Algorithm	4.54%	2.95%	2.53%	1.90%

Fig.3 shows the overall delay variation with the number of service replications. The x-axis is the number of user requests, and the y-axis is the overall delay between service replication and request offloading. We use random algorithm to replicate services and MGB algorithm to offload user requests. In Fig.3(a), we plot the overall delay for user request J=6. As expected, the overall delay is gradually decreasing until 9 service replications. However, after that point, the overall delay increase gradually. This is because more service replications may result in smaller user requests to offload, but too many service replications will cause the replicate time to be too long, thereby increasing the overall delay. And in Fig.3(b), the lowest latency is 14 service replications, when the user request is 10. Therefore, it is necessary to comprehensively consider service replication and user request offloading to reduce the overall delay time.



(a) User Request J=6 (a) User Request J=10

Fig. 3. The Overall Delay Variation with the Number of Replications.

In Fig.4, the random algorithm, greedy algorithm and particle swarm algorithm are selected as the baseline algorithm to compare the execution time with the MGB algorithm. The x-axis is the number of service replications, and the y-axis is the algorithm execution time. In the experimental setting, a total of 20 edge nodes have performed service replications. The results show that the execution time of the random algorithm is the shortest. And the execution time of the MGB algorithm is shorter, basically similar to the time of the random algorithm and greedy algorithm. However, the particle swarm algorithm has the longest execution time, especially when user requests is 20, the execution time is one order of magnitude higher than MGB.

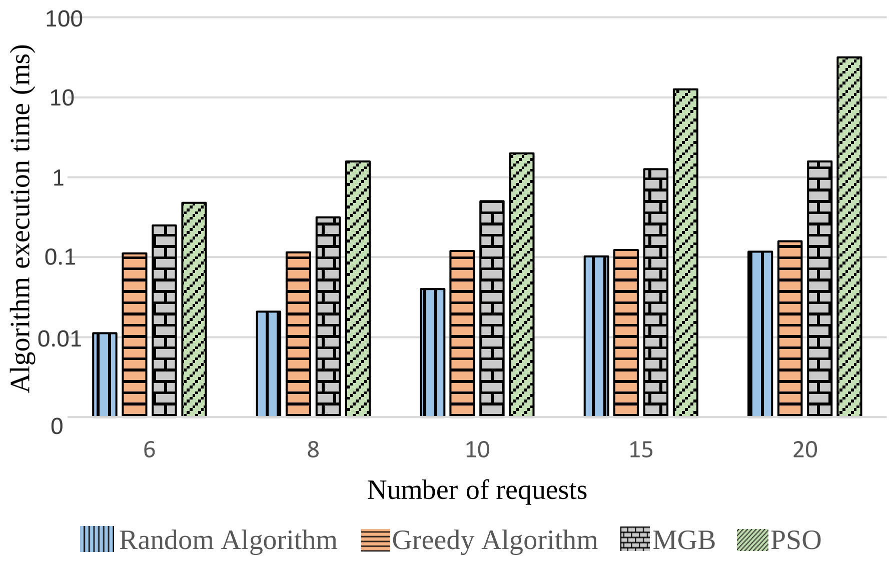


Fig. 4. Comparison of Algorithm Execution Time.

5 Conclusion

In this paper, the service replication and user request offloading in the MEC system is formalized as a joint optimization problem. Because the joint optimization problem is an NP-hard problem, as the number of service replications and user requests increase, the overhead of the optimization algorithm will increase dramatically. In order to solve the problem efficiently, we proposed the MGB algorithm and analyzed its performance. Our experiment compares the MGB algorithm with baselines algorithm. The result shows that the MGB algorithm has higher efficiency and can find a solution with lower overall delay for the above mentioned joint optimization problem. In the future, we intend to evaluate our algorithm in real scenarios. Also, we plan to develop multi-service replication based mechanisms for capacity provisioning and resource planning in edge computing systems.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (No. 62072216)

References

1. Hu, S., Li, G.: Fault-Tolerant Clustering Topology Evolution Mechanism of Wireless Sensor Networks. IEEE Access, **6**, 28085-28096 (2018)
2. Chen, M., Wang, T., Ota, K., Dong, M., Liu, A.: Intelligent resource allocation management for vehicles network: An A3C learning approach. Computer Communications, **151**: 485-494 (2020)
3. Cisco Systems, http://https://www.cisco.com/c/zh_cn.html. Last accessed 8 Aug. 2021
4. Wang, T., Cao, Z., Wang, S., et al.: Privacy-enhanced data collection based on deep learning for Internet of vehicles. IEEE Transactions on Industrial Informatics, **16**(10), 6663-6672 (2019)
5. Wang, T., Jia, W., Xing, G. et al.:Exploiting statistical mobility models for efficient Wi-Fi deployment. IEEE Transactions on Vehicular Technology, **62**(1), 360-373 (2012)
6. Moubayed, A., Shami, A., Heidari, P., Larabi, A.,Brunner, R.: Edge-Enabled V2X Service Placement for Intelligent Transportation Systems. IEEE Transactions on Mobile Computing. **20**(4), 1380-1392 (2021)
7. Thai, M., Lin, Y., Lai, Y., Chien, H.: Workload and Capacity Optimization for Cloud-Edge Computing Systems with Vertical and Horizontal Offloading. IEEE Transactions on Network and Service Management. **17**(1), 227-238 (2020)
8. Naas, M. I., Parvedy, P. R., Boukhobza, J., Lemarchand, L.: iFogStor: an IoT data placement strategy for fog infrastructure. In: Fog and Edge Computing (ICFEC), pp. 97-104 (2017)
9. Liu, X., Yu, J., Feng, Z., Gao, Y.: Multi-agent reinforcement learning for resource allocation in IoT networks with edge computing. China Communications. **17**(9), 220-236 (2020)
10. Yu, X., Tang, L.: Competition and Cooperation between Edge and Remote Clouds: A Stackelberg Game Approach. In: IEEE 4th International Conference on Computer and Communications (ICCC), pp. 1919-1923 (2018)
11. Wang, Y., Sheng, M., Wang, X., Wang, L., Li, J.: Mobile-edge computing: Partial computation offloading using dynamic voltage scaling. IEEE Transactions on Communications. **64**(10), 4268–4282 (2016)
12. Meye, P., Raipin, P., Tronel, F., Anceaume, E.: Toward a distributed storage system leveraging the DSL infrastructure of an ISP. In: 11th Consumer Communications and Networking Conference (CCNC). pp. 533-534 (2014)
13. Chang, W., Wang, P.: An adaptable replication scheme in mobile online system for mobile-edge cloud computing. In: IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). pp. 109-114 (2017)
14. Kiani, A., Ansari, N., Khreishah, A.: Hierarchical capacity provi- sioning for fog computing. IEEE Transactions on Networking. **27**(3), 962–971 (2019)
15. Lin, B., Zhu, F., Zhang, J., Chen, J., Chen, X., Xiong, N., Mauri, J. L.: A time-driven data placement strategy for a scientific workflow combining edge computing and cloud computing. IEEE Transactions on Industrial Informatics. **15**(7), 4254–4265 (2019)