Unloading strategy with caching mechanism based on genetic-particle swarm optimization algorithm

Biying Peng[a], Taoshen Li[a,b], Yan Chen*[a]
[a]School of Computer, Electronics and Information, Guangxi University, Nanning 530004, China;
[b]China-ASEAN International Join Laboratory of Integrate Transport, Nanning University, Nanning 530200, China

ABSTRACT

For time-delay sensitive applications in moving edge computing scenarios, this paper proposes an offloading strategy with caching mechanism based on Genetic-Particle Swarm Optimization algorithm. The strategy combines the above two algorithms to obtain the optimal offloading ratio and caching decision in edge computing offloading. Caching completed and multiple requested tasks and their associated data to the edge cloud minimizes task unload delays. The simulation experiment results demonstrate that this strategy can significantly reduce the delay of mobile edge computing.

Keywords: Genetic-particle swarm optimization algorithm, delay, caching mechanism, calculation offloading strategy

1. INTRODUCTION

With the rapid development of wireless communication technology Powered by advanced wireless communication technology, more and more mobile devices have wireless network access requirements. Although unloading computation-intensive applications to the cloud can overcome the limitation of mobile devices' resources, for delay-sensitive applications, long latency is obviously not enough for users[1]. Mobile Edge Computing (MEC) deploys computing nodes to the edge network to meet users' requirements of low latency. This kind of research mainly focuses on uninstallation decision in multi-user environment and multi-server environment[2,3].

At present, some researchers use Particle Swarm Optimization (PSO) to solve the computational unloading problem. Wei et al.[4] adopted greedy strategy PSO algorithm and dynamic PSO algorithm to propose a task allocation strategy for single-user multilateral cloud devices based on dynamic PSO to realize task allocation of single-user multilateral cloud equipments through dynamic PSO, considering the interdependence of tasks. Bi et al.[5] used heuristic algorithms based on genetic simulated annealing and PSO to solve nonlinear constrained optimization problems, and jointly optimized the task unload ratio, CPU speed and transmission power, and bandwidth of available channels of each mobile device to achieve lower energy consumption. Luo et al.[6] coded the MEC server of the edge cloud, and finally determined the server number corresponding to each task through the PSO algorithm. Miao et al.[7] introduced compression factor and improved PSO by using simulated annealing algorithm to realize task unloading, in which particle coding corresponds to vehicle task allocation scheme.

Some researchers also use caching technology to improve the optimization effect when calculating unloading. Nath et al.[8] developed a dynamic scheduling strategy based on deep learning, which can cache popular tasks in MEC server to avoid repeated unloading. Lan et al.[9] established a task cache optimization problem using random theory, and solved it using task cache algorithm based on genetic algorithm. Guo[1] divided task cache and unload decision optimization into two sub-optimization problems, wherein task cache can be transformed into 0-1 integer programming problem.

Through the analysis of the above research work, it can be found that most of the existing work is focused on particle coding or fusion of other algorithms for the unloading problem based on PSO in MEC scenarios. In addition, the optimization results of PSO combined with other algorithms are better than those of traditional algorithms, and the unloading delay optimization with task cache strategy is better than that without cache. Based on this, this paper proposes an unloading strategy based on Genetic Particle Swarm Optimization Algorithm (GA-PSO) and caching mechanism to cache on edge cloud, so as to effectively reduce the time delay of moving edge computing. Finally, the

*cy@gxu.edu.cn

International Conference on Computer Application and Information Security (ICCAIS 2021),
edited by Yingfa Lu, Changbo Cheng, Proc. of SPIE Vol. 12260, 122600J · © The Authors.
Published under a Creative Commons Attribution CC-BY 3.0 License · doi: 10.1117/12.2637404

convergence and effectiveness of the proposed algorithm the convergence and validity of GA-PSO are proved by simulation.

2. SYSTEM MODEL

The goal of the system is to achieve the optimal task caching and unloading and minimize the system delay when the edge cloud caching capacity is limited and the mobile device energy consumption constraints exist. The system goal can be achieved by caching and unloading tasks based on GA-PSO, and the system model is shown in Figure 1. This paper mainly solves two problems, whose tasks can be cached to the edge cloud, and what percentage of tasks can be offloaded to the edge and performed there.

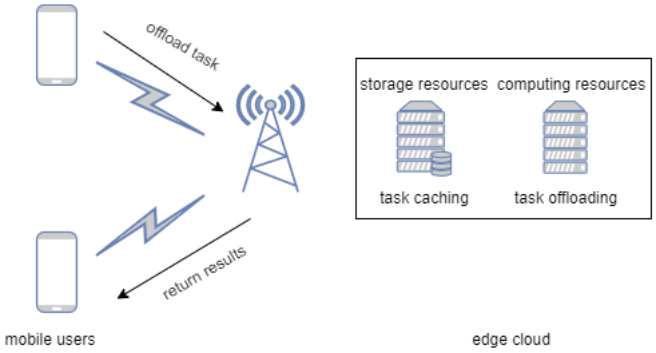

Figure 1. Multi-equipment system model.

2.1 Scene description

The mobile edge computing scenario in this paper consists of multiple mobile devices and a MEC edge cloud. Assume that the number of mobile devices is N, the number of computing tasks is I, user $N = \{1, 2, ..., n\}$, and task $I = \{1, 2, ..., i\}$. After the user sends the task request, the mobile device can connect with the edge cloud through the wireless channel. For task $U_{n,i}$ of mobile devices, it can be defined by binary groups (i.e., $U_{n,I} = \{C_i, d_i\}$), where C_i is the number of computing resources required to complete task I, that is, the total number of CPU cycles, and d_i is the data volume of task I, in bit.

2.2 Time delay model

Different devices will cause different delays when running tasks. This section considers two situations, namely, the current task is carried out locally and the task is offloaded. The delays involved in the above situations include: local computing delay, data uploading delay of mobile devices, computing delay of MEC server, and feedback delay, which can be ignored because the feedback delay is far less than the uploading delay.

(1) Local computing delay

where $f_{\mathrm{n}}^{\mathrm{local}}$ is the computing capability of the local CPU of mobile device n, and f_i is the clock cycle processing required for each bit of data.

$$T_i^{local} = \frac{C_i}{f_n^{local}} = \frac{di * f_i}{f_n^{local}} \tag{1}$$

(2) Transmission delay

Before calculating the time delay of uploading data by mobile devices, it is necessary to calculate the uploading rate first. According to Shannon formula, we can get the rate:

$$R_n = B * \log_2(1 + \frac{P_n * H_n}{\sigma^2}) \tag{2}$$

where, B is the wireless channel bandwidth between the mobile device and the edge cloud, P_n is the transmit power of the mobile device n, H_n is the wireless channel gain, and σ^2 is the noise power consumption.

The transmission delay thus obtained is:

$$T_i^{trans} = \frac{d_i}{R_n} \tag{3}$$

(3) MEC calculation delay

$$T_i^s = \frac{C_i}{f^s} = \frac{d_i * f_i}{f^s} \tag{4}$$

where f^s is MEC server CPU computing power. Then, the total delay of unloading the task to the MEC server for execution is:

$$T_i^e = T_i^{trans} + T_i^s \tag{5}$$

2.3 Energy consumption model

(1) Local calculated energy consumption

The computing capacity of CPU is closely related to its inherent chip architecture[10]. The energy consumption of a single CPU computing cycle can be expressed as $\varepsilon = \kappa * f^2$, and κ refers to the energy factor, which depends on the CPU chip technology[11]. Therefore, the energy consumption of local task execution is:

$$E_i^{local} = \kappa_u * (f_n^{local})^2 * C_i \tag{6}$$

(2) Transmission energy consumption

The energy consumed by mobile devices to upload data to the edge cloud is as follows:

$$E_{n,i}^{trans} = P_n * T_i^{trans} \tag{7}$$

2.4 Problem formalization

For the task caching problem, a decision variable $X1_i \in \{0,1\}$ can be defined. When $X1_i = 1$, it means that the task I is cached to the edge cloud for execution; if the value is 0, it means that the task is not cached. In this case, only the delay of task execution in the edge cloud needs to be considered. There is an edge cloud cache resource constraint, that is, the total amount of cached task data must be smaller than the edge cloud cache size. For the task offload problem, the unloading ratio can be defined as $X2_i \in [0,1]$. When $X2_i = 0$, tasks are executed locally; when $X2_i = 1$, all tasks are unloaded to the edge cloud; when $X2_i \in (0,1)$, part of tasks are unloaded to the edge cloud and the rest are executed locally.

To sum up, the total delay can be obtained as follows:

$$\begin{aligned} T_i &= X1_i * T_i^s + (1 - X1_i) * [X2_i * T_i^e + (1 - X2_i) * T_i^{local}] \\ &= X1_i * \frac{C_i}{f^s} + (1 - X1_i) * [X2_i * (\frac{d_i}{R_n} + \frac{C_i}{f^s}) + (1 - X2_i) * \frac{C_i}{f_n^{local}}] \end{aligned} \tag{8}$$

Total energy consumption of mobile devices:

$$\begin{aligned} E_i &= (1 - X1_i) * [X2_i * E_i^{local} + (1 - X2_i) * E_{n,i}^{trans}] \\ &= (1 - X1_i) * [X2_i * \kappa_u * (f_n^{local})^2 * C_i + (1 - X2_i) * \frac{P_n * d_i}{R_n}] \end{aligned} \tag{9}$$

The objective of the algorithm is to find the unload ratio and cache decision with the minimum delay of the system under the constraint of the maximum energy consumption of edge cloud resources and mobile devices. Therefore, the problem can be formalized as:

$$\min_{X1,X2}\sum_{n=1}^{N}\sum_{i=1}^{I}\{X1_i*\frac{C_i}{f^s}+(1-X1_i)*[X2_i*(\frac{d_i}{R_n}+\frac{C_i}{f^s})+(1-X2_i)*\frac{C_i}{f_n^{local}}]\}$$
$$subject to C1:\sum_{i=1}^{I}X1_i d_i \le E_c$$
$$C2: X1_i \in \{0,1\}, \forall i \in I$$
$$C3: X2_i \in [0,1], \forall i \in I \quad (10)$$
$$C4: E_i \le E_{\max}, \forall i \in I$$

The objective function means that the delay cost of the system is minimized through the task cache and the decision on the unloading ratio. Among them, constraint C1 indicates that the total amount of data for caching tasks cannot be greater than the caching capacity of the edge cloud, constraint C2 indicates that the task caching decision variables are binary variables, and 1 and 0 represent caching or not respectively, constraint C3 indicates that some divisible tasks are executed locally while the rest are executed on the edge cloud, and constraint C4 indicates that the power consumption of mobile devices should be controlled within the maximum power consumption constraint.

3. UNLOADING SCHEME

3.1 Particle coding

Individual particles are encoded in floating point numbers. Each particle element can take any floating-point number between 0 and 1.0001. The dimension of particle coding is the same as the total number of tasks. If the particle element value $pop = 1.0001$, it indicates that the task is cached to the edge cloud, and $pop \in [0,1]$ indicates that the value is the unloading ratio. The particle velocity represents how fast the task unloaded to the edge cloud, denoted as $V = \{v_1, v_2, \ldots, v_i\}$. The particle velocity is initialized as a random floating-point number from -0.1 to 0.1, and the dimension encoded by the particle velocity should also be the same as the total number of tasks. The optimal position of each particle in the evolutionary iteration process is G_{best}, and the optimal position of all particles is Z_{best}, which is the allocation method that minimizes the cost of the system.

3.2 Fitness function

Particle fitness represents the total delay cost of the system when the unloading decision and cache decision are adopted, and the function is as follows:

$$Fitness(pop) = \sum_{n=1}^{N}\sum_{i=1}^{I}T_i$$
$$= X1_i*\frac{C_i}{f^s}+(1-X1_i)*[X2_i*(\frac{d_i}{R_n}+\frac{C_i}{f^s})+(1-X2_i)*\frac{C_i}{f_n^{local}}] \quad (11)$$

3.3 Algorithm flow

For equation (10), that is, the minimization of the objective function, the unloading strategy based on GA-PSO with caching mechanism can be used to solve it.

The main process of the algorithm:

```
Initialization: pop_size,iter,popmin,popmax,c1,c2,w,oc,pm,V,pop
for i =1 to iter do
  for j =1 to pop_size do
    V(j,:)=V(j,:)+c1*rand*(gbest(j,:)-pop(j,:))+c2*rand*(zbest-pop(j,:));
    pop(j,:)=pop(j,:)+w*V(j,:);
    repeat
      mpick, cpick ← rand
      if mpick > pm then
```

Randomly select where to mutate
Mutation operation
end if
if *cpick* > *pc* **then**
Randomly select where to cross
Cross operation
end if
until Both chromosomes are available after mutating and crossing
if The constraints of cache and power consumption are not met **then**
Fitness(*pop*) ← *Inf*
else
Update *Fitness*(*pop*) based on equation (11)
if $Fitness_{gbest}$, $Fitness_{zbest}$ > *Fitness*(*pop*) **then**
Update $Fitness_{gbest}$, $Fitness_{zbest}$, G_{best}, Z_{best}
end if
end if
end for
end for

Output: Optimal unload ratio and cache decision Z_{best}, minimum delay *Fitness(pop)*

4. SIMULATION AND RESULT ANALYSIS

4.1 Simulation environment

In this section, an edge computing system composed of edge cloud and mobile devices is constructed by Matlab, and the unloading strategy based on GA-PSO and cache mechanism is implemented, and the performance of the strategy is evaluated in detail. Experimental parameters are shown in Table 1.

Table 1. Simulation parameters.

Parameter	Value	Parameter	Value
The data volume of task i (d_i)	7-40 (Mbit)	Transmission power of mobile device n (P_n)	0.5 (W)
Clock cycles required for each bit of data (f_i)	800-1000 (cycles/bit)	Radio channel gain (H_n)	$2*10^{-10}$-$2*10^{-6}$
Computing capability of the local CPU of device n (f_n^{local})	2-4 (GHZ)	The noise power (σ^2)	$1*10^{-9}$ (W)
Computing capability of the local CPU of MEC server (f^s)	5-8 (GHZ)	Wireless channel bandwidth between mobile devices and edge cloud (B)	1 (GHZ)
Edge cloud cache size (E_c)	500 (Mbit)	Inherent CPU coefficient of mobile devices (κ_u)	$5*10^{-27}$

4.2 Analysis of results

Figure 2 shows the comparison of the system delay cost of local computing and offloading all tasks to the MEC server, based on the GA-PSO algorithm without a caching mechanism, and an offloading strategy with a caching mechanism under different numbers of devices. It can be observed that the total system delay of the three types of strategies when the number of devices is 25, 50, 75, 100, 125, and as the number of devices increases, the delays of these four types of strategies are all showing an increasing trend, but the strategy in this article The system delay is less than the delay of the other three schemes.

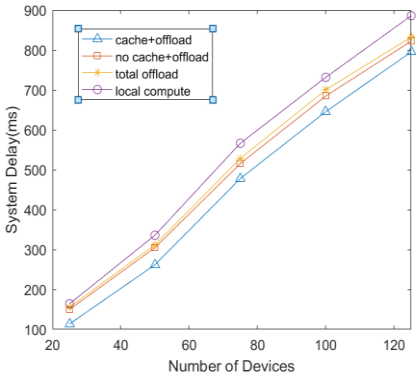

Figure 2. Impact of number of devices on delay.

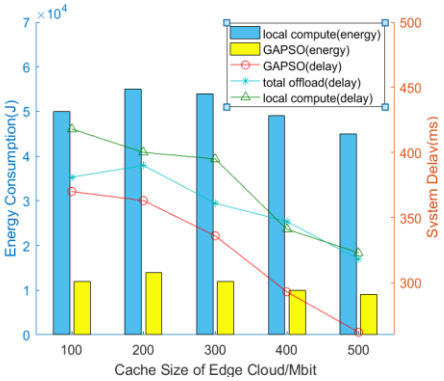

Figure 3. Impact of edge cloud caching capability on delay and power consumption.

Figure 3 shows the relationship between edge cloud cache capacity, energy consumption of mobile devices and total system delay when GA-PSO unloading strategy, local computing and all tasks are unloaded to MEC server in this paper. The left ordinate is equipment power consumption, and the right ordinate is system delay. As can be seen from Figure 3, with the increase of cache resources, energy consumption tends to decrease, that is, the more task data that edge cloud can cache, the lower the energy consumption of mobile devices will be. In addition, the energy consumption of local computing is significantly higher than that of this strategy, indicating that the GA-PSO unloading strategy in this paper reduces the energy consumption of mobile devices to a certain extent, because the relevant data of some tasks have been cached in the edge cloud, and the execution of such tasks has no energy consumption for mobile devices. It can also be seen from Figure 3 that the larger the edge cloud cache capacity is, the smaller the system delay is, indicating that cache has a great impact on the delay. This is because the larger the cache capacity is, the more cacheable tasks are, and the execution time of cached tasks is shorter than that of non-cached tasks that need to be unloaded.

5. CONCLUSION

In this paper, an unloading strategy based on GA-PSO and cache mechanism is proposed. This strategy combines the advantages of global search of genetic algorithm with the advantages of local search of PSO. The search speed is fast and the optimal unloading ratio and cache decision can be obtained. Simulation results indicate that compared with local computing, full offloading and no cache offloading strategies, the GA-PSO offloading strategy in this paper makes local devices and edge cloud cooperate in computing, which can reduce the delay cost of mobile edge computing. In the future, PSO can be combined with other optimization algorithms to find a more reliable unloading strategy.

ACKNOWLEDGMENT

These works are supported by the Guangxi science and technology plan project of China (No. AD20297125).

REFERENCES

[1] Guo, Y., “Task unloading strategy with caching mechanism in mobile edge computing,” Computer Applications and Software, 36(06), 114-119 (2019). (in Chinese)
[2] Chen, X., Jiao, L., Li, W. and Fu, X., “Efficient multi-user computation offloading for mobile-edge cloud computing,” Transactions on Networking, 24(5), 2795-2808 (2015).
[3] Chen, M., Hao, Y., Qiu, M., Song, J., Wu, D. and Humar, I., “Mobility-aware caching and computation offloading in 5G ultra-dense cellular net-works,” Sensors, 16(7), 974 (2016).
[4] Wei, Q., Wei, F., Ge, H., Feng, A., Wang, Y. and Li, W., “Computational offloading strategy based on dynamic particle swarm for multi-user mobile edge computing,” 2019 IEEE Symp. Series on Computational Intelligence (SSCI), 2890-2896 (2019).

[5] Bi, J., Yuan, H., Duanmu, S., Zhou, M. C. and Abusorrah, A., "Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," IEEE Internet of Things Journal, 8(5), 3774-3785 (2021).
[6] Luo, B. and Yu, B., "Computing unloading strategy based on particle swarm optimization in moving edge computing," Journal of Computer Applications, 40(08), 2293-2298 (2020). (in Chinese)
[7] Miao, Y., Xu, Y., Zhang, W., Liu, T. and Han, Z., "Task Unloading strategy of improved particle swarm optimization algorithm in vehicle networking," Application Research of Computers, 38(07), 2050-2055 (2021). (in Chinese)
[8] Nath, S. and Wu, J., "Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems," Intelligent and Converged Networks, 1(2), 181-198 (2020).
[9] Lan, Y., Wang, X., Wang, D., Liu, Z. and Zhang, Y., "Task caching offloading and resource allocation in D2D-aided fog computing networks," IEEE Access, 7, 104876-104891 (2019).
[10] Li, S., Ge, H. B., Chen, X. T., Liu, L., Gong, H. W. and Tang, R., "Computation offloading strategy for improved particle swarm optimization in mobile edge computing," 2021 IEEE 6th Inter. Conf. on Cloud Computing and Big Data Analytics (ICCCBDA), 375-381 (2021).
[11] Li, M., Zhou, X., Qiu, T., Zhao, Q. and Li, K., "Multi-relay assisted computation offloading for multi-access edge computing systems with energy harvesting," IEEE Transactions on Vehicular Technology, 70(10), 10941-10956 (2021).