International Journal of Latest Engineering Research and Applications (IJLERA) ISSN: 2455-7137

Volume – 07, Issue – 07, July 2022, PP – 39-44

Mobile Edge Computing Task Offloading Based on Particle Swarm Optimization

Xiulan Sun[1], Wenzao Li[1,2*], Yue Wang[3], Xue Tang[1]

[1](College of Communication Engineer, Chengdu University of Information Technology, China)
[2](Network and Data Security Key Lab. of Sichuan Pro. University of Electronic Science and Technology of China, China)
[3](Educational Informationization and Big Data center, Education Department of Sichuan Province, China)

Abstract: With the proliferation of IoT devices, the pressure of cloud computing is emerging. In order to save energy and reduce data transmission delays, mobile terminals usually require the multi-tasking load to be shifted to more powerful servers. Mobile edge computing, as an emerging solution, can meet the requirements of mobile edge devices to expand computing power by offloading tasks. In the scenario of a single terminal device and a single mobile edge server, this paper analyzes and studies the multi-task offloading problem in the scenario, and then proposes an optimization goal to reduce the system load. The system load is determined by the task completion delay and terminal device energy consumption weighted composition. Then a task offloading strategy based on particle swarm optimization is designed to solve the proposed problem. The simulation results of the algorithm are compared with the simulation results of other benchmark algorithms, which show that the algorithm proposed in this paper can effectively reduce the load.

Keywords: Mobile edge computing, system load, task completion delay, terminal device energy consumption

I. INTRODUCTION

Mobile edge computing is a network architecture that integrates cloud computing with strong data processing capabilities into the edge to provide an IT service environment [1], allowing mobile devices to offload computing to Base Stations (BSs). After the task is offloaded to the base station, it is calculated by the edge server. By offloading computing, mobile terminal devices can obtain larger processing and storage resources, and can achieve lower energy consumption or latency. In addition, offloading all tasks results in wasted communication resources, causing network congestion. As the number of tasks increases, the number of offloading strategies also grows exponentially, making it unrealistic to traverse all possible outcomes. Therefore, it is difficult to find a suitable offloading situation within the scope of conditions. The strategic joint processing of mobile devices and mobile edge servers is the key to solving this problem. Therefore, this requires the strategy to fully search and converge under numerous offloading situations.

In computing offloading, there are many factors that affect the total effectiveness and overhead of the system, among which important indicators include energy consumption, delay, quality of service (QoS), quality of experience (QoE), response time and cost [1]. Scholars utilize different research methods to achieve their optimization goals. Literature [2] attempts to minimize system latency (maximum server latency). The author decomposes the problem into a task offloading problem and a transmission power allocation problem, which are solved using matching theory and heuristics, respectively. Literature [3] proposes a power-constrained delay minimization problem by analyzing the average delay of each task and the average power consumption of mobile devices, and proposes an efficient one-dimensional search algorithm to find the optimal task scheduling strategy. Literature [4] formulated the problem of transmit power allocation for mobile users to minimize energy consumption. Using the quasi-convex technique, a sub-gradient-based non-cooperative game model is proposed. Literature [5] studies an efficient computational offloading mechanism for MEC in 5G heterogeneous networks. Considering the task computation and file transfer energy consumption, an optimization problem is proposed to minimize the energy consumption of the offloading system. Combined with the multi-access feature of 5G heterogeneous networks, an energy-efficient computation offloading (EECO) scheme is designed, which jointly optimizes offloading and radio resource allocation to obtain minimum energy consumption under delay constraints. Literature [6] proposes an optimization framework for offloading from a single mobile device (MD) to multiple edge devices. The optimization goal is to minimize the total task execution latency and MD's energy consumption. The literature [7] weights time and energy consumption in user experience as a hybrid overhead and performs joint optimization. The authors propose a Mixed Time and Effort (MOTE) minimization problem, using piecewise coordinate descent to process each variable level by level.

The contributions of this paper are summarized as follows.

1. This paper considers a mobile edge computing scenario of a terminal device and a MEC server, and models the offloading of multiple independent tasks generated by the terminal device.
2. In this paper, the task delay and terminal energy consumption are considered at the same time, and the optimization objective of the system load is formed by linear weighting.
3. Based on the proposed optimization problem, this paper designs an optimization algorithm based on particle swarm. The simulation results of the algorithm show that the offloading strategy of the algorithm can effectively optimize the system load, and this paper observes the performance of the offloading strategy in terms of delay and energy consumption.

The rest of this article is organized as follows. In Section 2, the system model and problem formulation are elaborated. In Section 3, detailed solutions are given. The performance of the proposed scheme is evaluated in Section 4. Finally, the conclusion of this paper is given in Section 5

II. SYSTEM MODEL

As shown in Fig.1, in a network consisting of a single terminal equipment and an edge server, the edge server is deployed beside the base station, and multiple independent tasks are generated on the terminal equipment. The terminal equipment and the base station can communicate wirelessly, offload the generated tasks to the base station, and perform calculations through the edge server. Each edge server has a CPU for task calculation. In this paper, terminal equipment generates m tasks, which are represented by task queue as$T = \{t_1, t_2, \dots, t_m\}$, where the input data size of task t_m is represented as I_m (bit). This paper defines the uplink data transmission rate of task t_m from the terminal device to the edge server as r_m. In this paper, α is used to indicate that the task tm is computed locally or offloaded to the edge server for computing. $\alpha = \{0,1\}$, when the value of α is 0, it means that the task is calculated locally, and when the value of α is 1, it means that the task is offloaded to the edge server for calculation. In the task queue, the tasks computed locally can be represented as set $T^l = \{t_1, t_2, \dots, t_a\}$; the tasks offloaded to the edge server can be represented as set $T^e = \{t_{a+1}, t_2, \dots, t_m\}$. Considering that the data amount of the calculation result of each task is relatively small, the download result is faster [8]. Therefore, in order to simplify the calculation, this paper ignores the delay and terminal energy consumption caused by the downlink transmission of each task, and focuses on the delay and terminal energy consumption caused by the uplink transmission.

Fig.1 system model

When the task t_m is calculated on the terminal, the delay D_m generated by the task is the calculation time delay. Therefore, the delay of the terminal device to complete the task t_m is expressed as D_m^l.

$$D_m^l = I_m / f^l \ \#(1)$$

And the terminal energy consumption is expressed as E_m^l.

$$E_m^l = P^l \times D_m^l \ \#(2)$$

Among them, f^l represents the computing resources of the terminal device, and P^l represents the power of the terminal processor.

When the edge server calculates the offloaded task t_m, the generated delay D_m^e consists of two parts, the transmission delay and the calculation delay. Therefore, the delay and energy consumption of task t_m computing on the edge server are expressed as:

$$D_m^e = D_m^{tran} + D_m^{com} \#(3)$$

$$E_m^e = E_m^{tran} + E_m^{com} \#(4)$$

The transmission delay D_m^{tran} is the delay of uploading the task to the edge server, expressed as:

$$D_m^{tran} = I_m / r_m \ \#(5)$$

The transmission energy consumption E_m^{tran} is the terminal device energy consumption of the task uploaded to the edge server, expressed as:

$$E_m^{tran} = D_m^{tran} \times P^{tran} \ \#(6)$$

In the formula, P^{tran} represents the transmission power of the terminal device. The calculation delay D_m^{com} is the time for the task to be calculated on the edge server after the task t_m is successfully uploaded, expressed as:

$$D_m^{com} = I_m / f^{com} \ \#(7)$$

where f^{com} is the computing resource of the edge server. Computing energy consumption E_m^{com} is the standby energy consumption generated by terminal device when the task is calculated on the edge server. Because the edge server has abundant computing resources, the computing task delay of the edge server is low, and the standby power of the terminal device is low, so the standby energy consumption of the terminal device is very low at this time and can be ignored. Therefore, equation (4) can be written as:

$$E_m^e = E_m^{tran} \#(8)$$

Therefore, the delay and energy consumption generated by the completion of the calculation of m independent tasks generated by the terminal device can be summarized as:

$$T = \sum_{i=1}^{a} D_i^l + \sum_{j=a+1}^{m} D_j^e \ \#(9)$$

$$E = \sum_{i=1}^{a} E_i^l + \sum_{j=a+1}^{m} E_j^e \ \#(10)$$

In the research, our goal is to reduce the system load composed of task completion delay and terminal energy consumption. Therefore, our objective function is formulated as follows.

$$Minimize\ B = \beta \times T + (1-\beta) * E \ \#(11)$$

In the formula, β and $(1-\beta)$ represent the weight of task completion delay and energy consumption, respectively. The larger the value of β, the more important the task completion delay is in the system load. The larger the value of $(1-\beta)$ is, the more the system load is concerned with the energy consumption of the terminal equipment.

III. PARTICLE SWARM-BASED OPTIMIZATION ALGORITHM DETAILS

The Because particle swarm optimization has the characteristics of few parameters, simple principle and easy implementation, it is widely used in the search for optimal strategy [9][10]. The particles in the particle swarm optimization algorithm have memory, and by changing the speed and position of the particles, the position and speed of the excellent particles are updated, and new individuals can be obtained, which can achieve rapid convergence [11]. In order to solve the problem raised in the previous section, we make some adjustments to the algorithm model, including particle position vector, velocity vector and fitness function, etc.

Particle position vector: Particle encoding adopts integer encoding, the dimension of particles is the same as the number of task sets, which is m, and the number of particle populations is M. The position vector of any particle $X_i = [x_1, x_2, \dots, x_j]$, where $i \in M, j \in m$.The particle's position vector represents the set of offloading decisions for all tasks.Since the offloading decision of the task is only 0 and 1, the values of the elements in the position vector are only 0 and 1.

Particle Velocity Vector: The velocity vector represents the change in the task offloading decision set, denoted by $V_i = [v_1, v_2, \dots, v_j]$.

Fitness function: For individual particles, the fitness value describes the quality of the task offloading decision expressed by the particle. Using equation (12) as the fitness function f, the smaller the fitness value is, the higher the system load will be caused by the particle offloading decision, and the worse the quality of the offloading decision set represented by the particle. Therefore, our goal is to find the particle with the largest fitness value during the algorithm iteration.

$$f = \frac{1}{B} \#(12)$$

The optimization algorithm flow based on particle swarm (PSO) is as follows:

(1). Initialize the particle swarm according to the parameter constraints, and obtain the initial velocity and initial position of each particle.

(2). The task completion delay, terminal energy consumption and system load are calculated through the offloading decision set represented by the particle position vector.

(3). Calculate the fitness value of each particle, save or update the historical best fitness value $Pbest$ of each particle.

(4). If the minimum fitness value in the particle swarm is smaller than the group's historical optimal fitness value, update the group's historical optimal fitness value $Gbest$.

(5). Because of the coding limitation of particle position vector and velocity vector, the common position and velocity update method of particle swarm algorithm is no longer applicable. This section updates the location information through the Sigmoid function.

$$S\left(v_j^{h+1}\right) = \frac{1}{1 + e^{\left(-v_j^{h+1}\right)}} \quad \#(13)$$

$$x_j^{(h+1)} = \begin{cases} 1, r_j < S\left(v_j^{h+1}\right) \\ 0, \quad else \end{cases} \quad \#(14)$$

In the formula, the superscript $h+1$ represents the number of iterations of the particle swarm. r_j is a value in a uniform distribution from 0 to 1. When $r_j < S\left(v_j^{h+1}\right)$, the value of x_j in the particle position is 1.

(6). When the number of iterations is less than or equal to the maximum number of iterations, repeat steps (2) (3) (4) (5); otherwise, the iteration terminates.

IV. FIGURES SIMULATION EXPERIMENT AND RESULT ANALYSIS

Several simulation experiments are carried out in this section, and the experimental results verify the effectiveness of the algorithm. The simulated scenario includes an edge server and a mobile terminal device with multiple tasks. The number of tasks m of the mobile terminal increases from 5 to 40, and increases by 5 in turn. For computing tasks, the input data size I is in [100KB, 400KB], [2100KB, 2400KB], [3100KB, 3400KB] respectively. The uplink data transmission rate r of the mobile terminal equipment is 1.5MB/s [12], the transmission power P^{tran} is 23dBm[13].And the computing resource of the mobile terminal equipment f^l is 1.5GHz [14]. The computing resource f^e of the mobile edge server is 10GHz [14]. The task completion delay weight β is 0.5.

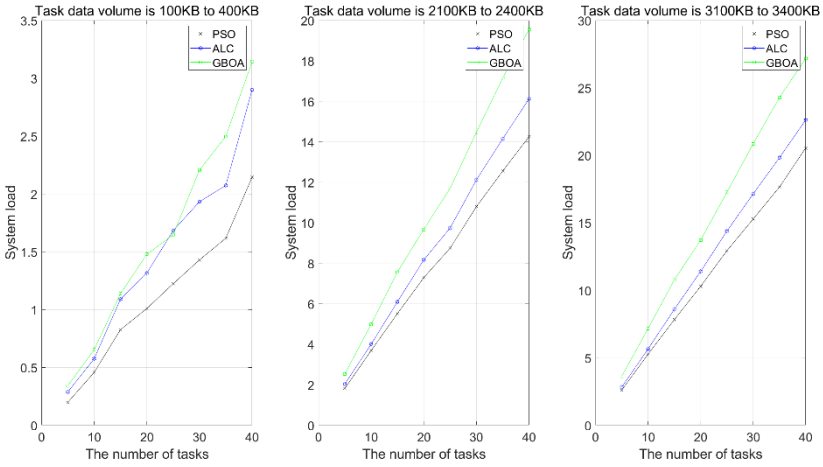

Fig.2 system load performance of several algorithms

In order to study the performance of the particle swarm optimization algorithm proposed in this paper in reducing the system load, the proposed algorithm is compared with the following offloading algorithms: (1) All local computing (ALC): All tasks generated by the terminal device are calculated on the terminal CPU without task offloading; (2) Greedy based offloading algorithm (GBOA): Mainly consider the situation of the task offloading module and the task processing module of the current terminal device. If the task offloading module is in an idle state, the terminal device offloads the task to the edge server for computing. Likewise, if the task processing module of the terminal device is in an idle state, the task is calculated and processed by the CPU of the terminal device.

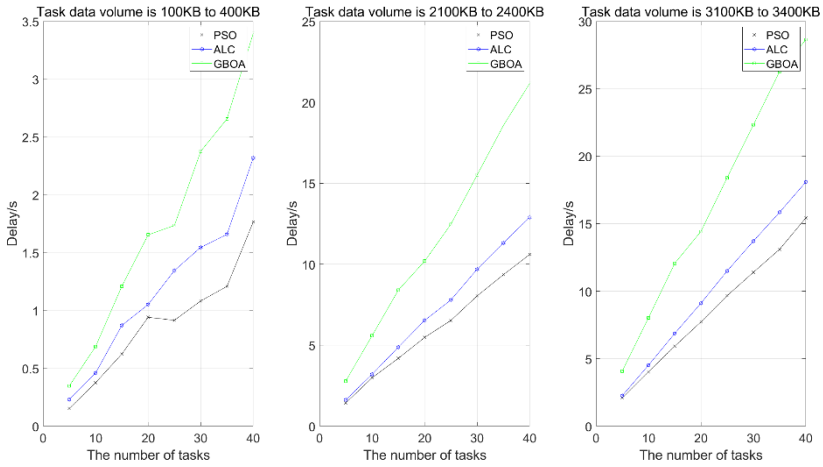

Fig.3 delay performance of several algorithms

Fig.2 depicts the performance of different algorithms in terms of system load for tasks with different amounts of data. It can be clearly seen that the average system load of the PSO algorithm is significantly lower than that of ALC and GBOA under different data volumes. It shows that the reasonable task offloading strategy proposed by the PSO algorithm can significantly reduce the system load. Among several algorithms, the offloading strategy of GBOA leads to the highest system load, which indicates that the unreasonable task offloading strategy results in a system load even higher than that obtained by all the tasks locally calculated.

Figures 3 and 4 show the performance of different algorithms in terms of task completion delay and terminal device energy consumption, respectively. The observation results show that the delay obtained by the PSO algorithm is lower than that of the other two methods in the case of different task data amounts; in terms of terminal energy consumption, the energy consumption obtained by the PSO algorithm is lower than that of the ALC algorithm. But when the amount of task data is large, the energy consumption of GBOA is relatively close to that of PSO, but the energy consumption of both algorithms is lower than that of ALC. In each subgraph of Fig.3, with the increase of the number of tasks, the delays of all offloading algorithms are gradually increasing. Among them, the delay of GBOA algorithm is the highest, and with the increase of the number of tasks and the amount of task data, the delay gap between GBOA, ALC and PSO also gradually increases. In each subgraph of Fig.4, with the increase in the number of tasks, the energy consumption of all offloading algorithms also increases gradually. Among them, the energy consumption of the ALC algorithm is the highest, and as the amount of task data increases, the energy consumption gap of the ALC algorithm and the other two gradually increase.

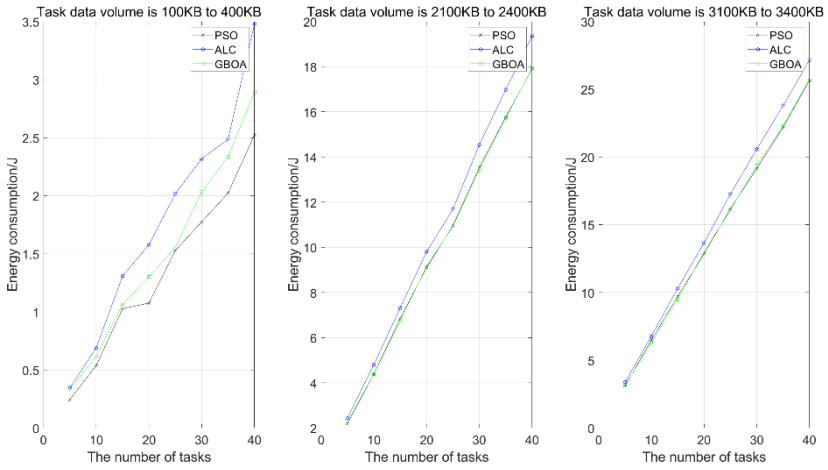

Fig.4 energy consumption performance of several algorithms

V. CONCLUSION

In this paper, we consider the scenario of multiple independent tasks generated by a single terminal device in a single edge server network, and optimize the system load composed of task completion delay and terminal device energy consumption.After constructing the optimization problem, an improved PSO optimization algorithm is proposed and simulation experiments are carried out. The simulation results show that the proposed algorithm can effectively optimize the system load, and can effectively reduce the system load, task completion delay and terminal equipment energy consumption.

REFERENCES

[1] Shakarami A, Ghobaei-Arani M, Masdari M, et al. A survey on the computation offloading approaches in mobile edge/cloud computing environment: a stochastic-based perspective[J]. Journal of Grid Computing, 2020, 18(4): 639-671.

[2] Xiao S, Liu C, Li K, et al. System delay optimization for mobile edge computing[J]. Future Generation Computer Systems, 2020, 109: 17-28.

[3] Liu J, Mao Y, Zhang J, et al. Delay-optimal computation task scheduling for mobile-edge computing systems[C]//2016 IEEE international symposium on information theory (ISIT). IEEE, 2016: 1451-1455.

[4] Hu S, Li G. Dynamic request scheduling optimization in mobile edge computing for IoT applications[J]. IEEE Internet of Things Journal, 2019, 7(2): 1426-1437.

[5] Zhang K, Mao Y, Leng S, et al. Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks [J]. IEEE access, 2016, 4: 5896-5907.

[6] Dinh T Q, Tang J, La Q D, et al. Offloading in mobile edge computing: Task allocation and computational frequency scaling[J]. IEEE Transactions on Communications, 2017, 65(8): 3571-3584.

[7] Tang Q, Lyu H, Han G, et al. Partial offloading strategy for mobile edge computing considering mixed overhead of time and energy[J]. Neural Computing and Applications, 2020, 32(19): 15383-15397.

[8] Liu J, Zhang Q. Offloading schemes in mobile edge computing for ultra-reliable low latency communications[J]. Ieee Access, 2018, 6: 12825-12837.

[9] Alfakih T, Hassan M M, Al-Razgan M. Multi-objective accelerated particle swarm optimization with dynamic programing technique for resource allocation in mobile edge computing[J]. IEEE Access, 2021, 9: 167503-167520.

[10] Zhou W, Chen L, Tang S, et al. Offloading strategy with PSO for mobile edge computing based on cache mechanism[J]. Cluster Computing, 2021: 1-13.

[11] Ebadifard F, Babamir S M. A PSO-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment [J]. Concurrency and Computation: Practice and Experience, 2018, 30(12): e4368.

[12] Fan W, Liu Y, Tang B, et al. Computation offloading based on cooperations of mobile edge computing-enabled base stations[J]. IEEE Access, 2017, 6: 22622-22633.

[13] Tang X, Wen Z, Chen J, et al. Joint Optimization Task Offloading Strategy for Mobile Edge Computing[C]//2021 IEEE 2nd International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA). IEEE, 2021, 2: 515-518.

[14] Lyu X, Tian H, Ni W, et al. Energy-efficient admission of delay-sensitive tasks for mobile edge computing[J]. IEEE Transactions on Communications, 2018, 66(6): 2603-2616.

