future internet

MDPI

Article

A Novel Task Caching and Migration Strategy in Multi-Access Edge Computing Based on the Genetic Algorithm

Lujie Tang, Bing Tang *, Linyao Kang and Li Zhang

School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, Hunan, China

* Correspondence: btang@hnust.edu.cn; Tel.: +86-0731-58290474

Received: 17 July 2019; Accepted: 19 August 2019; Published: 20 August 2019

Abstract: Multi-access edge computing (MEC) brings high-bandwidth and low-latency access to applications distributed at the edge of the network. Data transmission and exchange become faster, and the overhead of the task migration between mobile devices and edge cloud becomes smaller. In this paper, we adopt the fine-grained task migration model. At the same time, in order to further reduce the delay and energy consumption of task execution, the concept of the task cache is proposed, which involves caching the completed tasks and related data on the edge cloud. Then, we consider the limitations of the edge cloud cache capacity to study the task caching strategy and fine-grained task migration strategy on the edge cloud using the genetic algorithm (GA). Thus, we obtained the optimal mobile device task migration strategy, satisfying minimum energy consumption and the optimal cache on the edge cloud. The simulation results showed that the task caching strategy based on fine-grained migration can greatly reduce the energy consumption of mobile devices in the MEC environment.

Keywords: edge computing; task migration; task caching; genetic algorithm

1. Introduction

With the rapid development of the mobile Internet and the Internet of Things (IoT), as well as the emergence of various new types of services, users are increasingly demanding better quality network services. Therefore, in order to effectively solve the challenges of high load and low latency caused by the development of the Internet, the concept of multi-access edge computing (MEC) has been proposed and it has attracted the attention of academics and industry. Originally defined as mobile edge computing, MEC offers application developers and content providers cloud-computing capabilities and an IT service environment at the edge of the network. This environment is characterized by ultra-low latency and high bandwidth as well as real-time access to radio network information that can be leveraged by applications. MEC provides a new ecosystem and value chain. Operators can open their Radio Access Network (RAN) edge to authorized third-parties, allowing them to flexibly and rapidly deploy innovative applications and services to mobile subscribers, enterprises and vertical segments. It is now considered to be one of the key technologies for the next generation network [1].

By migrating related computing tasks and data to the edge cloud, the energy consumption and latency of the mobile device can be reduced, which can greatly improve user experiences. In general, the key technology for computing migration is to take advantage of the edge cloud to provide the appropriate decision-making process for task offloading. Although the edge cloud has strong computing and storage capabilities, the resources in the edge cloud are still very limited in the context of the rapidly increasing number of mobile devices and services.

To the best of our knowledge, the existing work on MEC has mainly focused on the following two aspects: edge caching and task offloading. Unfortunately, because the main concern of edge caching is the storage capacity of the edge cloud, researchers have rarely considered the computing and storage capabilities of the edge clouds simultaneously. Furthermore, with regard to task offloading, most of the existing work has focused on task migration and has only considered coarse-grained migration or simple dependencies rather than complex dependencies. Therefore, in this paper, due to the low latency and relatively limited capability of the edge cloud, we study the problem of task caching for MEC, and propose a joint optimization of task caching and task offloading with complex dependencies using the fine-grained partitioning model to solve the problem of energy consumption optimization using the genetic algorithm, while satisfying user requirements.

The rest of this paper is organized as follows: Section 2 examines the relevant studies on MEC and task migration and offloading. Section 3 presents the problem definition and solution, in which a task execution time model, energy model, and genetic algorithm (GA)-based optimization algorithm are given. Section 4 presents the simulation results and analysis, and the final section concludes the paper and discusses future work.

2. Related Work

Edge computing is a general term for a cloud-based IT service environment located at the edge of a network. The purpose of edge computing and MEC is to bring real-time, high-bandwidth, low-latency access to latency-dependent applications distributed at the edge of the network. The primary goal of edge computing is to reduce network congestion and improve application performance by executing related task processing closer to the end user, thereby improving the delivery of content and applications to those users. Various examples that have already been realized include augmented reality (AR), virtual reality (VR), connected cars, IoT applications and so on [1]. Large public venues and enterprise organizations also benefit from edge computing. Enterprises are increasingly motivated to use small cell networks to transmit data at sizable locations such as offices, campuses, or stadiums.

Edge computing lets operators host content and applications close to the edge of the network. It also brings new levels of performance and access to mobile, wireless, and wired networks. The technology is routinely mentioned in conversations about the infrastructure of 5G networks and Network Function Virtualization (NFV) technology, particularly for handling the huge number of IoT devices (commercial and industrial) that are constantly connected to the network [2].

Currently, task migration and task offloading algorithms are key research topics in MEC or mobile cloud computing. Some studies have presented task migration and task offloading strategies in terms of minimizing delay, minimizing energy consumption or minimizing both of these at the same time [3].

An approach for minimizing delay was proposed in [4]; this adopted a Markov decision process approach to handle transmission decisions, which include where the computation tasks are scheduled based on the queueing state of the task buffer, the execution state of the local processing unit, and the state of the transmission unit. In [5], Liu et al. studied the task offloading problem from a matching perspective and aimed to optimize the total network delay. They proposed a pricing-based one-to-one matching algorithm and pricing-based one-to-many matching algorithms for the task offloading. Chen et al. investigated the task offloading problem in ultra-dense network aiming to minimize the delay while saving the battery life of user's equipment, and formulated the task offloading problem as a mixed integer nonlinear program [6]. A control-theoretic approach and a two-level resource allocation and admission control mechanism for a cluster of edge servers in the MEC environment is presented in [7].

In terms of minimizing energy consumption, Kwak et al. considered network traffic, cloud workloads and also the ability of CPU frequency scaling and network interface selection between WiFi and cellular to minimize energy when offloading tasks [8]. Cao et al. proposed an energy-optimal offloading algorithm of mobile computing to achieve the maximum saving energy based on combinatorial optimization method [9]. Jiang et al. discussed the tradeoff between energy optimization

and delay optimization, and presented a Lyapunov optimization-based scheme for offloading scheduling, as well as download scheduling for cloud execution output for multiple applications running in mobile devices [10]. In [11], Wang et al. proposed an energy-efficient and deadline-aware task offloading strategy based on the channel constraint, with the goal of minimizing the energy consumption of mobile devices while satisfying the deadline constraints of mobile cloud workflows They employed an adaptive inertia weight-based particle swarm optimization to solve the optimization problem.

In addition, several studies have considered the caching problem in MEC, for example, caching content in small cell base stations [12,13] or mmWave-cellular networks [14] have been proposed to enhance the quality of the experience and minimize delay. However, we rarely see research on both task caching and migration.

The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. It has been widely used in scheduling optimization problems. For task migration problems, several strategies have been proposed based on GA, such as in [15,16]. Shi et al. proposed the mobility-aware computing offloading strategy of distributed mobile cloud computing called MAGA, and an integer encoding based adaptive genetic algorithm is implemented for offloading decisions [15]. Deng et al. considered the complex dependencies between service components and introduced a mobility model and a trade-off fault-tolerance mechanism for the offloading system, which provides robust offloading decisions for mobile services and effectively optimizes the time and energy consumption of these services [16]. However, neither of these two papers addressed the issue of edge caching in their offloading strategies, which is considered in our paper.

3. Architecture

In order to minimize the total energy consumed by mobile devices while satisfying delay requirements, we first propose a fine-grained task partitioning model, then we propose a caching algorithm based on the task model.

3.1. Fine-Grained Task Migration Model

There are two main task migration models that are often adopted in MEC. One is coarse-grained task migration and the other is fine-grained task migration. Coarse-grained task migration considers the entire application as a migration object, thus we no longer divide it into multiple subtasks, and the entire task migrates to the edge cloud. On the contrary, fine-grained task migration means that the application can be divided into multiple subtasks, and these subtasks have dependencies on each other, not simple dependencies, but complex dependencies which can be expressed as a directed acyclic graph (DAG). However, both of them have obvious shortcomings. The former does not make full use of the low latency advantage of the edge cloud, while the latter simplifies the complex dependencies between subtasks. Therefore, in order to resolve the shortcomings of the above two task migration models, we propose a new fine-grained task partitioning model. The topology model is shown in Figure 1.

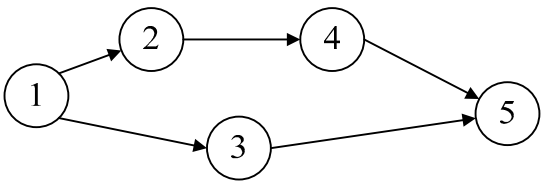

Figure 1. Topology model of fine-grained task partitioning.

As shown in Figure 1, the input data source of subtask 5 comes from subtask 3 and subtask 4. When subtask 3 and subtask 4 are both completed, then subtask 5 starts execution. Here, we introduce a real example, a facial expression recognition application, which can benefit from MEC [16,17].

The facial expression recognition application task consists of eight subtasks, including (1) input video, (2) face detection, (3) facial feature detection, (4) face registration, (5) facial feature extraction, (6) action unit recognition, (7) expression recognition, and (8) output class. When we make an offloading decision for this application task, the following concern should be addressed: the above subtasks can be organized as a workflow with data dependencies. Each subtask in this facial expression recognition application task can be either executed locally on mobile devices or remotely (offloaded) to the edge cloud. Because of their dependencies, if a subtask is offloaded, its following subtask cannot start until the result is returned, even if local resources are available. This is different from traditional independent offloading services where multiple services can be simultaneously offloaded or executed remotely or locally at any time.

3.2. Task Caching Model

Task caching refers to the caching of completed task applications and their related data in the edge cloud. The service provider caches the appropriate data according to the capacity of the edge cloud and the popularity of the task, so as to reduce the delay and energy consumption of user requested data. The main problem faced by task caching is that the storage capacity of the edge cloud is limited, which directly affects the task completion time and energy consumption.

3.3. Problems and Scenarios

Considering that an MEC system consists of N mobile devices and the edge cloud, we assume that there are K computation tasks, and the number of users is greater than the number of tasks ($N \geq K$). This is because some computing tasks are highly popular and they may be repeatedly requested and processed multiple times. Thus, we assume that one mobile device only requests tasks once, while different users can request the same task based on their preferences.

For heterogeneous computing tasks, we define three parameters to model the computing task. We define $U_{n,k} = \{w_k, s_k, t_n\}$ to denote that the user n requests task k, where w_k represents the amount of computing resources required for $U_{n,k}$, and s_k means the data size of the computing task, and t_n is the completion time requirement that user n requests task k. Furthermore, since the storage capacity of the edge cloud is limited, we assume that the cache size is C_e. We define $x_k \in \{0, 1\}$ to indicate whether task k is cached on the edge cloud ($x_k = 1$) or not ($x_k = 0$).

We divide task k into multiple subtasks with complex dependencies based on the fine-grained task partitioning model, which can be represented by $k = \{k_1, k_2, \ldots, k_v\}$. This generates a fine-grained topology model of task k, which is shown in Figure 2. For a graph G_k, we define it as $G_k = (V_k, E_k)$, where V_k is the set of subtasks to be processed in task k, and E_k is the set of directed edges that represents dependencies between subtasks. We also define $e_{ku,m} \in E_k$ as the amount of data transferred from the subtask u to the subtask m. Subtask m starts only after subtask u has been completed and the data transmitted has been received.

As can be seen from Figure 2, the subtasks can be divided into two groups. In the first group, all subtasks must be executed locally on the devices, such as subtask 0, subtask u, and subtask v, which belong to the non-migratory task set V_{local}. In the second group, subtasks can be migrated to the edge cloud, such as subtask 1, subtask 2, and subtask m, which belong to the migratable task set $V_{offload}$. In the second group, not all of the m must be migrated to the edge cloud for execution, and the decision on migration depends on several factors such as the network condition and data size the task requested.

Regarding the dependencies between subtasks, we use a binary variable $R_{ku,m} \in \{0, 1\}$ to represent the dependencies, where $R_{ku,m} = 1$ indicates that the subtask u is a predecessor of the subtask m, while $R_{ku,m} = 0$ indicates that the subtask u has no predecessor relationship with the subtask m.

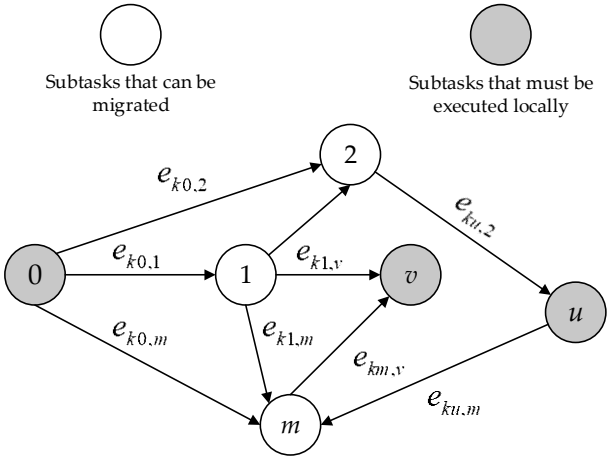

Figure 2. The fine-grained topology model for task k.

3.4. Time and Energy Model

The objective of the energy-saving algorithm in task migration based on task caching is to reduce the energy consumption of computation migration as much as possible, while satisfying the task completion time requirement through caching some tasks on the edge cloud. Before migration, the algorithm compares the time and energy consumption on mobile devices and the edge cloud, so as to make a task migration decision. Therefore, it is necessary to establish a corresponding task energy model and time model.

The migration decision variable $D_{km} \in \{0, 1\}$ is used to represent the execution location of each subtask of the task k, and the location could be on the mobile device or on the edge cloud. When $D_{km} = 1$, the subtask m is migrated to the edge cloud and then executed, while when $D_{km} = 0$, the subtask m is executed on the device locally.

3.5. Task Execution Time and Energy Model

3.5.1. Execution Time on Mobile Devices

If the subtask m of the task k is executed on the mobile device, that is $D_{km} = 0$, and the execution time T^l_{km} of the subtask m has a strong relationship with the task size w_{km} and the CPU capacity f_l of the mobile device. The execution time T^l_{km} of the subtask m can be expressed as follows:

$$T^l_{km} = \frac{w_{km}}{f_l} \tag{1}$$

3.5.2. Execution Time on the Edge Cloud

If subtask m of the task k is executed on the edge cloud, that means $D_{km} = 1$, and the execution time T^e_{km} of the subtask m is determined by the task size w_{km} and the CPU capacity f_e of the edge cloud. The execution time T^e_{km} of the subtask m can be expressed as follows:

$$T^e_{km} = \frac{w_{km}}{f_e} \tag{2}$$

3.5.3. Energy Consumption When Executed on Mobile Devices

If subtask m of the task k is executed on the mobile device, that means $D_{km} = 0$. In this situation, the mobile device is in a working state. The local execution energy consumption E^l_{km} of the task

is determined by its execution time T^l_{km} and the power p_l when the mobile device performs the computation. The execution energy consumption E^l_{km} of the subtask m can be expressed as follows:

$$E^l_{km} = T^l_{km} p_l \tag{3}$$

3.5.4. Sleep Energy Consumption on Mobile Devices

If subtask m of the task k is executed on the edge cloud, that means $D_{km} = 1$. In this situation, the mobile device is in a sleep state, and the power in the sleep state of the mobile device is p_h. The sleep energy consumption E^e_{km} of the subtask m can be expressed as follows:

$$E^e_{km} = T^e_{km} p_h \tag{4}$$

3.6. Data Transfer Time and Energy Model

Suppose that subtask u is the predecessor of subtask m, that is to say $R_{ku,m} = 1$, we know that:

(1) If subtask u and subtask m are executed in the same location (both are on the edge cloud or on the device), so we have $D_{ku} = D_{km}$. We assume there is no data transmission overhead between the two subtasks, so we have $T_{ku,m} = 0$ and $E_{ku,m} = 0$.

(2) If the locations where subtask u and subtask m are executed are different (one is on the edge cloud and the other is on the device), we have $R_{ku,m} = 1$, $D_{ku} = 0$, and $D_{km} = 1$. When the subtask u is executed on the mobile device and the subtask m is executed on the edge cloud, computation result transmission delay $T_{ku,m}$ is generated, and it can be calculated by Equation (5), which depends on the result data amount $e_{ku,m}$ and the transmission rate B_s.

$$T_{ku,m} = \frac{e_{ku,m}}{B_s} \tag{5}$$

Furthermore, the energy consumption $E_{ku,m}$ for transmitting the result data is shown as follows:

$$E_{ku,m} = T_{ku,m} p_s \tag{6}$$

(3) If subtask u and subtask m are executed in different locations, we have $R_{ku,m} = 1$, $D_{ku} = 1$ and $D_{km} = 0$. When subtask u is executed on the edge cloud and subtask m is executed on the mobile device, this results in a data receiving delay, which depends on the amount of data transmission $e_{ku,m}$ and the data transmission rate of the result B_r. Thus, the data receiving delay $T_{ku,m}$ between the two tasks can be expressed as follows:

$$T_{ku,m} = \frac{e_{ku,m}}{B_r} \tag{7}$$

The energy consumption $E_{ku,m}$ for receiving the computation result data is shown in the following:

$$E_{ku,m} = T_{ku,m} p_r \tag{8}$$

3.7. Minimizing Energy Consumption

The goal of using a combination of caching and an energy-saving strategy based on fine-grained task migration is to consider whether to cache the task on the edge cloud and refine the task into smaller-grained subtasks when the migration destination is uncertain. The migration optimization objective is to reduce the energy consumption of devices and save energy for these devices. At the same time, the user requested task delay time must be satisfied. Therefore, when we model minimizing energy consumption, the task completion time constraints must be ensured and the quality of user experience should be guaranteed.

The transmission time T_k^t of the task k depends on the size of the data s_{km} transferred from the device to the edge cloud and the execution location D_{km} of the subtasks, which is calculated as the following:

$$T_k^t = \sum_{m=1}^{v} D_{km} \frac{s_{km}}{B_s} \tag{9}$$

The energy consumed by task k to transfer data to the edge cloud is denoted as E_k^t, which is shown as the following:

$$E_k^t = T_k^t p_s \tag{10}$$

The execution time T_{km}^x of the subtask m of task k is related to the execution location and it can be calculated according to the following equation:

$$T_{km}^x = (1 - D_{km}) T_{km}^l + D_{km} T_{km}^e \tag{11}$$

The start time of subtask m of the task k is denoted as T_{km}^b, which depends on the sum of its predecessors' completion time and the result data transmission delay between them. If the subtask m has multiple predecessors, the execution starts until the last predecessor subtask and all data transmission have been completed. The start time T_{km}^b of subtask m of task k is shown as follows:

$$T_{km}^b = \max_{u \in V_k} R_{ku,m} (T_{ku}^f + |D_{ku} - D_{km}| T_{ku,m}) \tag{12}$$

Then, the time when the subtask m is completed can be obtained according to the following equation using T_{km}^b and T_{km}^x:

$$T_{km}^f = T_{km}^b + T_{km}^x \tag{13}$$

We define a variable $x_k \in \{0, 1\}$ to indicate whether task k is cached on the edge cloud ($x_k = 1$) or not ($x_k = 0$). The completion time of task k is the time when the last subtask is completed plus the data transmission time of task k. The execution time of task k is presented in the following equation:

$$T_{n,k} = x_k T_{kv}^f + (1 - x_k)(T_{kv}^f + T_k^t) \tag{14}$$

The energy consumption of the mobile device is mainly composed of three parts: the task execution energy consumption, the result data transmission energy consumption and the task data transmission energy consumption. If the task data is cached on the edge cloud, we assume the data transmission energy consumption as $E_k^t = 0$.

The execution energy E_{km}^x of the subtask m is related to the execution location D_{km}, and it is calculated as follows:

$$E_{km}^x = D_{km} E_{km}^e + (1 - D_{km}) E_{km}^l \tag{15}$$

Therefore, the energy consumption of the mobile device when the entire task k is executed is shown as follows:

$$E_{n,k} = (1 - x_k) E_k^t + \sum_{m=1}^{v} E_{km}^x + \sum_{(u,m) \in V_k} |D_{ku} - D_{km}| E_{ku,m} \tag{16}$$

Based on the above analysis, a model for minimizing the energy consumption of mobile devices under the task completion time constraints is proposed, which can be expressed as follows:

$$\begin{aligned} &\text{minimize} \sum_{n=1}^{N} E_{n,k} \\ &subject\ to: \sum_{k=1}^{K} x_k s_k \le C_e \\ &T_{n,k} \le t_n \\ &w_k = \sum_{m=1}^{v} w_{km} \\ &s_k = \sum_{m=1}^{v} s_{km} \\ &D_{km} \in \{0,1\}, R_{ku,m} \in \{0,1\}, x_k \in \{0,1\} \end{aligned} \tag{17}$$

The decision array $D[N][v+1]$ is the solution of the optimization problem we introduced above, where N means a total of N users, and the first v variables are the subtask decision variables, and the last variable indicates whether the user task is cached on the edge cloud or not. Since there are two choices for the execution location of each migratable subtask, and multiple types for the cached task, if the enumeration method is used to find the optimal solution, the computational complexity is too high and increases exponentially. Through theoretical analysis, we found that the array $D[N][v+1]$ of decision variables is a set of binary variables and we can use the genetic algorithm for an approximate solution. The benefit of the genetic algorithm is that it avoids the high complexity of the enumeration solution and also ensures the accuracy of the solution.

3.8. GA-Based Task Caching and Migration Strategy

In order to get the minimum energy consumption, it is necessary to find the best decision variable array composed of $N \times (v+1)$ binary variables. If subtasks are restricted to being executed locally on the devices, no decision is required and the decision variable is always 0. In this paper, we choose the genetic algorithm of bit coding to solve the problem of minimizing energy consumption. The basic components common to almost all genetic algorithms are: a fitness function for optimization, a population of chromosomes, selection of which chromosomes will reproduce, crossover to produce the next generation of chromosomes, and random mutation of chromosomes in the new generation.

3.8.1. Fitness Function

The fitness value refers to the degree of adaptability of the natural population to the environment. The fitness function is the criterion for evaluating the performance of the individual. The function value is used to distinguish the merits of each individual. The larger the function value, the more favorable the individual chromosomes are in the evolutionary iterative operation of the genetic algorithm, the better it is to be retained, and the corresponding migration decision is better. In this paper, we define the fitness function using the total energy consumption of mobile devices, which is shown as follows:

$$fitness = 1/\sum_{n=1}^{N} E_{n,k} \tag{18}$$

3.8.2. Initialization

In the initialization phase, we choose binary encoding. The initial chromosome population size is set to 50, the crossover probability is 0.6, the mutation probability is 0.15, the maximum population iteration number is 200, and the minimum evolution rate of the continuous five generations of population is 0.2%, that is, the algorithm terminates when the population is iteratively evolved up to 200 times, or the evolution rate of the fitness function value of the most adaptive chromosome individual in the continuous five generations of population is less than 0.2%.

3.8.3. Selection

In the selection phase, the probability of occurrence of each individual in the offspring is calculated according to the fitness value of the individual. Then, we use the roulette selection method to select a part of the individuals from the parent population to form a descendant population. The probability of selecting a chromosome for recombination is related to its fitness value.

3.8.4. Crossover

The crossover operation imitates the process of reproduction and mating in nature. By optimizing the combination of parental chromosomes, some genes are exchanged to produce new excellent individuals, and their excellent quality is retained in the new population. In order to increase the diversity of the population and the excellent characteristics of the genetic individual, the crossover method we adopt is to use a random algorithm with equal probability to select single-point crossing, double-point crossing, and uniform crossing to cross each of them, which avoids unity of the population due to only using single-point crossing.

3.8.5. Mutation

The mutation operation replaces certain gene values in the chromosome coding string with other gene values according to the mutation probability, thereby forming a new individual, which is a supplementary method for generating new individuals, and enhances the local search ability of the genetic algorithm. In order to maintain population diversity, several commonly used mutations include: basic bit mutation, uniform mutation, boundary mutation and Gaussian approximation mutation. We use a mutation method suitable for binary coding, that is, basic bit mutation strategy.

4. Simulations and Results

4.1. Experiment Configurations

The simulation scenarios are described as follows. There are N users, and K types of task, one base station, and the edge cloud is deployed in the base station to provide services. Each user performs one type of task. The simulation parameters are given empirical values, which are listed in Table 1 [18].

Table 1. The value of parameters.

Parameter	Value	Meaning of the Parameter
f_l	300 MIPS	Computation rate on mobile device
f_e	5000 MIPS	Computation rate on edge cloud
B_s	2 Mbps	Transmission speed on mobile device
B_r	2 Mbps	Receiving speed on mobile device
p_l	0.80 W	Computation power on mobile device
p_h	0.04 W	Sleep power on mobile device
p_s	0.03 W	Transmission power on mobile device
p_r	0.01 W	Receiving power on mobile device

In this section, we study the effectiveness of the algorithm. We first explore the impact of the caching strategy on task execution through measuring user task completion time and energy consumption. Then, we adjust different task data transmission volume, and explore the change of task execution energy consumption with the increase of data transmission amount when using the caching strategy. We also set different cache capacities of the edge cloud, and study the impact of cache capacity on the energy consumption. Finally, by comparing three strategies: the strategy where cache-based subtasks are migrated; the strategy where non-cached tasks are all executed locally; and the strategy where non-cached tasks are all executed on the edge cloud, we evaluate the energy-saving result of the algorithm proposed in this paper.

4.2. Results and Analysis

The algorithm was implemented using Matlab 2012a. In the first evaluation, the maximum population iteration number for GA was set to 200. The values of user number N and task type K were set to 20 and 5, respectively. Each user is supposed to submit one task. The cache capacity of the edge cloud is 5000 MB, the amount of data transferred by all subtasks follows the normal distribution. In order to intuitively evaluate the complexity of the algorithm, we recorded the running time of the simulation program. The running time was around 11 seconds in a Windows 10 Operating System with Intel Core i7 CPU and 8 GB memory. In order to analyze the impact of the task cache on task execution performance, we compared the task completion time and energy consumption in two scenarios, with a cache policy and without a cache policy. The results are shown in Figures 3 and 4.

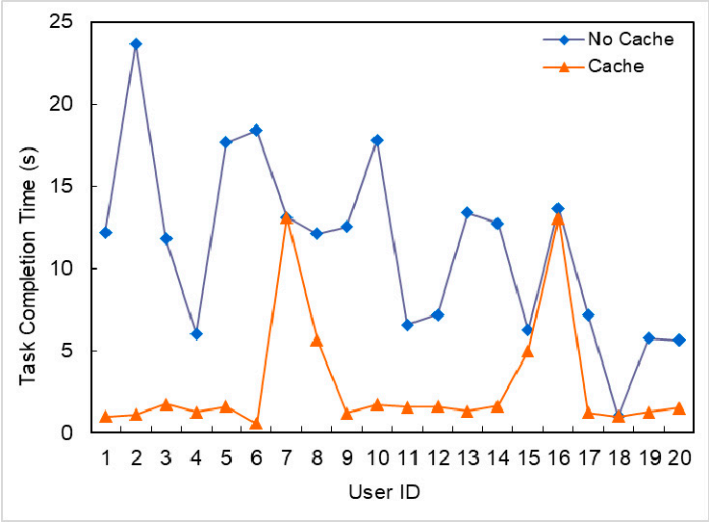

Figure 3. Task completion time for different users.

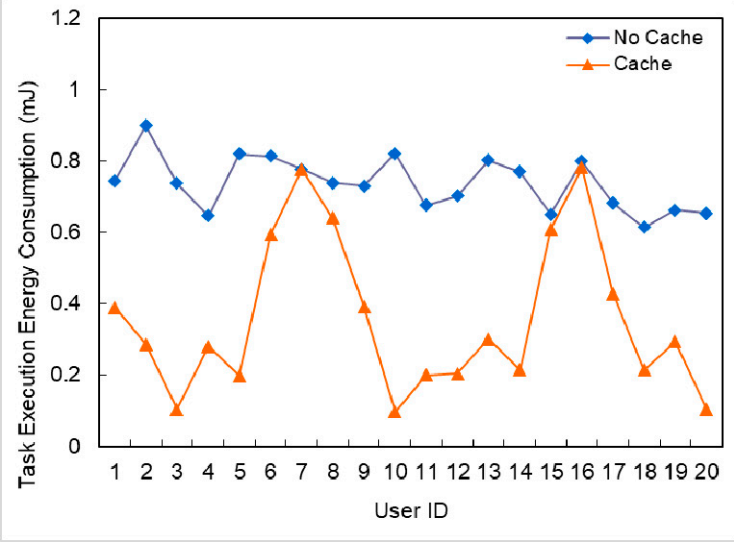

Figure 4. Energy consumption for different users.

Because the cache capacity on the edge cloud is limited, only some types of tasks can be cached. As can be seen from Figures 3 and 4, the tasks performed by User 7 and User 16 were not cached on the edge cloud. Therefore, the optimal subtask migration strategies solved by the genetic algorithm indicates that for the tasks performed by User 7 and User 16, the execution time and energy consumption are the same as the non-cache strategy. It can be seen from these two figures that the task completion time for

User 2 decreased by 95.23%, and the task execution energy consumption for User 3 decreased by 85.84%. The cache strategy has a great influence on task execution time and energy consumption. Caching tasks on the edge cloud can reduce the time and energy consumption required for task data transmission.

In the second evaluation, we also set 20 users and 5 different type of tasks. The average data size means the average size of data transmissions for all tasks performed by 20 users. The amount of data transferred by all subtasks also follows the normal distribution.

As shown in Figure 5, in terms of energy consumption, using the cache policy outperforms using the non-cache policy, the energy consumed by the task is much less when using the cache policy. As the task data volume increases, the rate of increase in energy consumed by task execution using the non-cache policy is much faster than using the cache policy. When the average data size was 9600 MB, the energy consumption decreased by 55.38% when using the cache policy. This indicates that the energy saving effect brought by the caching strategy becomes more and more obvious for tasks with a large amount of task data transmission.

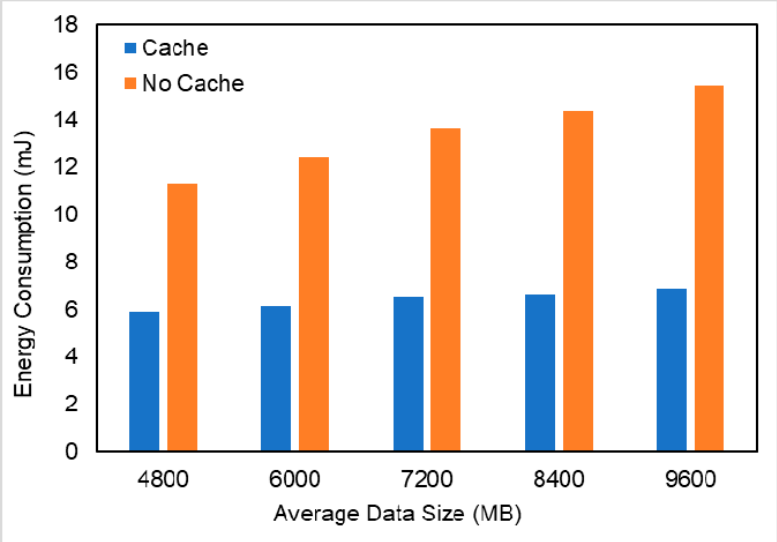

Figure 5. Energy consumption as the average size of data transmission increases when two strategies are adopted.

We updated the cache capacity of the edge cloud, and the number of users was 60 and the types of task was set to 8. As is shown in Figure 6, as the cache capacity of the edge cloud increased, the energy consumption of the 60 users was reduced. The average data sizes were set to 36000 MB and 54000 MB, respectively, and when the edge cloud size was 4500 MB, the energy consumption decreased by 33.29%. The reduction in task execution energy consumption is due to the support of the cache capacity on the edge cloud. At the same time, it can be seen that the greater the amount of task data transmission, the more energy is consumed.

We compared the energy consumption in terms of three strategies, (a) cache-based subtask migration algorithm (GA); (b) all tasks executed in the edge cloud (Edge); and (c) all tasks only executed on mobile devices (Local). As can be seen from Figure 7, the cache-based subtask migration algorithm outperformed the others. Using cache and task offloading can effectively reduce the energy overhead of mobile devices. At the same time, it can be seen that when the amount of data for a task is small, the difference in energy consumption between strategy GA and Edge are also small. In addition, with an increase in the amount of data transmission, the energy consumed is gets closer and closer for the Edge and Local strategies. Compared with the Edge and Local strategies, the maximum improvement in energy consumption for the GA strategy is21.16% (when the average data size is 144000 MB) and 31.6% (when the average data size is 5400 MB).

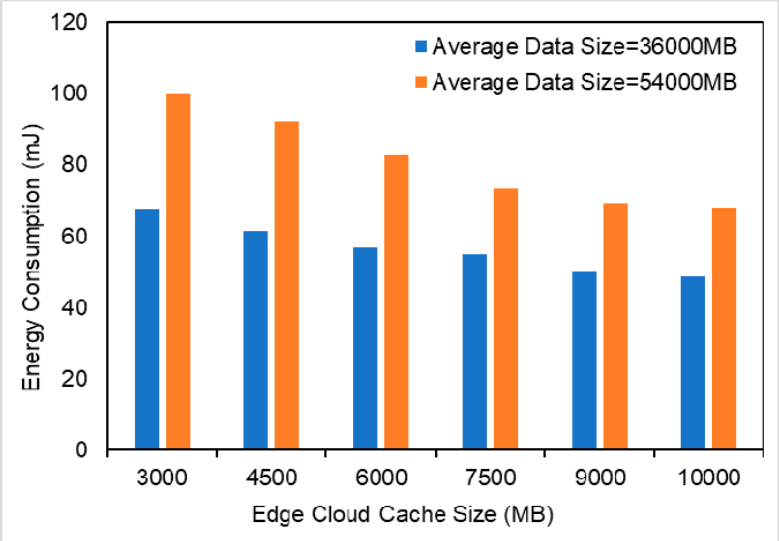

Figure 6. The impact of the edge cloud cache size on energy consumption.

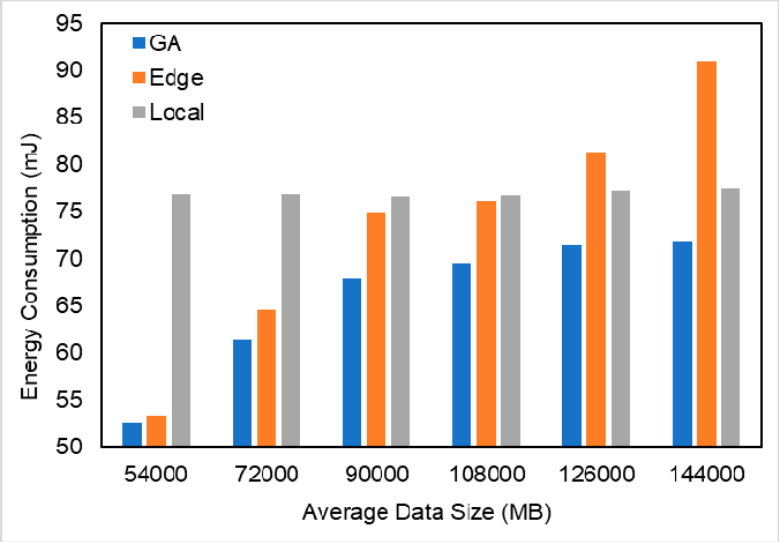

Figure 7. The impact of average data size on energy consumption.

5. Conclusions

In this paper, we proposed a multi-user and fine-grained task migration caching strategy in MEC that satisfies the completion time constraints and the goal of minimizing energy consumption. First, we established a fine-grained task partitioning model to transform user tasks into directed graphs with multiple subtasks. Then, based on the fine-grained task model, we studied the impact of the caching strategy on task execution energy consumption. The genetic algorithm was used to solve the above optimization problem and to obtain the optimal task migration and task caching solution. The simulation results showed that the caching strategy can greatly reduce the time required for task execution and the energy consumption, and the fine-grained task migration caching algorithm proposed in this paper is more efficient in energy-saving than other strategies, such as all tasks being executed in the edge cloud or mobile devices.

In future work, we plan to improve task migration in MEC in the following three ways: (1) The task migration strategy proposed in this paper only considers the environmental state whereby the mobile device is always within signal coverage, and the network status is stable. The mobility of the mobile device is not considered, which could be considered in future work with migration decisions made by predicting the movement and trajectory of devices. (2) This paper only considers the task characteristics

when investigating the fine-grained task migration energy-saving strategy. However, channel changes and battery remaining were not considered. In future work, more factors should be considered, and the migration strategy should be dynamically adapted to more complicated situations. (3) This paper adopts GA to solve optimization problems. However, in practical applications, GA has a long running time. In the future, new, fast algorithms should be studied to improve task migration performance.

Author Contributions: Conceptualization, L.T. and B.T.; Data curation, L.Z.; Formal analysis, L.T. and L.K.; Funding acquisition, B.T.; Investigation, L.Z.; Methodology, L.T.; Project administration, B.T.; Software, L.T.; Supervision, B.T.; Validation, L.K.; Visualization, L.K.; Writing—original draft, L.T.; Writing—review & editing, B.T.

Funding: This research is supported by the National Natural Science Foundation of China under grant no. 61602169, 61702181, and 61876062, and the Natural Science Foundation of Hunan Province under grant no. 2018JJ2135 and 2018JJ3190, as well as the Scientific Research Fund of Hunan Provincial Education Department under grant no. 18A186.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge computing: Vision and challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [CrossRef]
2. Sarrigiannis, I.; Kartsakli, E.; Ramantas, K. Application and Network VNF migration in a MEC-enabled 5G Architecture. In Proceedings of the 2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Barcelona, Spain, 17–19 September 2018; pp. 1–6.
3. Mach, P.; Becvar, Z. Mobile Edge Computing: A Survey on Architecture and Computation Offloading. *IEEE Commun. Surv. Tut.* **2017**, *19*, 1628–1656. [CrossRef]
4. Liu, J.; Mao, Y.; Zhang, J.; Letaief, K.B. Delay-optimal computation task scheduling for mobile-edge computing systems. In Proceedings of the IEEE International Symposium on information Theory, Barcelona, Spain, 10–15 July 2016; pp. 1451–1455.
5. Liu, P.; Li, J.; Sun, Z. Matching-Based Task Offloading for Vehicular Edge Computing. *IEEE Access* **2019**, *7*, 27628–27640. [CrossRef]
6. Chen, M.; Hao, Y. Task Offloading for Mobile Edge Computing in Software Defined Ultra-Dense Network. *IEEE J. Sel. Area. Comm.* **2018**, *36*, 587–597. [CrossRef]
7. Avgeris, M.; Dechouniotis, D.; Athanasopoulos, N.; Papavassiliou, S. Adaptive Resource Allocation for Computation Offloading: A Control-Theoretic Approach. *ACM T. Internet Techn.* **2019**, *19*, 23. [CrossRef]
8. Kwak, J.; Kim, Y.; Lee, J.; Chong, S. Dream: Dynamic Resource and Task Allocation for Energy Minimization in Mobile Cloud Systems. *IEEE J. Sel. Area. Comm.* **2015**, *33*, 2510–2523. [CrossRef]
9. Cao, S.; Tao, X.; Hou, Y.; Cui, Q. An energy-optimal offloading algorithm of mobile computing based on HetNets. In Proceedings of the IEEE International Conference on Connected Vehicles and Expo, Seattle, WA, USA, 12–16 September 2016; pp. 254–258.
10. Jiang, Z.; Mao, S. Energy Delay Tradeoff in Cloud Offloading for Multi-Core Mobile Devices. *IEEE Access* **2015**, *3*, 2306–2316. [CrossRef]
11. Wang, Y.; Wu, L.; Yuan, X.; Liu, X.; Li, X. An Energy-Efficient and Deadline-Aware Task Offloading Strategy Based on Channel Constraint for Mobile Cloud Workflows. *IEEE Access* **2019**, *7*, 69858–69872. [CrossRef]
12. Bastug, E.; Bennis, M.; Kountouris, M. Cache-Enabled Small Cell Networks: Modeling and Tradeoffs. In Proceedings of the 2014 11th International Symposium on Wireless Communications Systems (ISWCS), Barcelona, Spain, 26–29 August 2014.
13. Blasco, P.; Gunduz, D. Learning-Based Optimization of Cache Content in A Small Cell Base Station. In Proceedings of the 2014 IEEE International Conference on Communications (ICC), Sydney, Australia, 10–14 July 2014; pp. 1897–1903.
14. Giatsoglou, N.; Ntontin, K.; Kartsakli, E.; Antonopoulos, A.; Verikoukis, C.V. D2D-Aware Device Caching in MmWave-Cellular Networks. *IEEE J. Sel. Area. Comm.* **2017**, *35*, 2025–2037. [CrossRef]
15. Shi, Y.; Chen, S.; Xu, X. MAGA: A Mobility-Aware Computation Offloading Decision for Distributed Mobile Cloud Computing. *IEEE Internet Things J.* **2018**, *5*, 164–174. [CrossRef]

16. Deng, S.; Huang, L.; Taheri, J.; Zomaya, A.Y. Computation offloading for Service Workflow in Mobile Cloud Computing. *IEEE Trans. Parallel Distrib. Syst.* **2015**, *26*, 3317–3329. [CrossRef]
17. Whitehill, J.; Serpell, Z.; Lin, Y.-C.; Foster, A.; Movellan, J.R. The Faces of Engagement: Automatic Recognition of Student Engagement from Facial Expressions. *IEEE Trans. Affective Comput.* **2014**, *5*, 86–98. [CrossRef]
18. Deng, M.; Tian, H.; Fan, B. Fine-granularity based application offloading policy in cloud-enhanced small cell networks. In Proceedings of the 2016 IEEE International Conference on Communications Workshops (ICC), Kuala Lumpur, Malaysia, 23–27 May 2016; pp. 638–643.