A Hybrid Genetic Algorithm with Integer Coding for Task Offloading in Edge-Cloud Cooperative Computing

Bo Wang, *Member, IAENG,* Bin Lv, and Ying Song*

***Abstract*—Edge-Cloud Cooperative Computing (E3C) is an efficient way to address the resource scarcity issue of mobile devices by edge servers with low network delays and the cloud with abundant computing resources. To improve the user satisfaction which directly affects the profit and the reputation of service providers, we focus on the task offloading problem for E3C in this paper. We design an integer genetic algorithm incorporating earliest deadline first (EDF) to map each chromosome as a task offloading solution. Our method uses each gene of a chromosome to represent the computing core that the corresponding task assigned to, and decides the task execution order by EDF in each computing core. Experiment results show that our method achieves better accepted ratio and resource efficiency than several offloading methods based on heuristics, the particle swarm optimization, and the binary genetic algorithm.**

***Index Terms*—edge cloud, genetic algorithm, task offloading, task scheduling**

I. Introduction

Nowadays, mobile devices and applications are all around us. But most of the time, a mobile device cannot satisfy the requirements of its user. This is mainly because the device has limited capabilities of resources and energy. By exploiting the cloud resources for extending the capability of devices, Mobile Cloud Computing (MCC) can address this issue [1], [2]. While, the cloud computing has a poor network connection with devices. This can result in that the demands of network delay-sensitive applications cannot be met. To address this problem, edge computing is exploited to reduce the network distance between devices and clouds, by placing some edge servers close to user devices [3]. By combining both advantages of MCC and edge computing, the Edge-Cloud Cooperative Computing (E3C) is a promising way to satisfy various requirements of Quality of Service (QoS) in both the computation delay and the network latency for mobile users.

For a service provider, how to improve the user satisfaction with high resource efficiency is a problem that must be solved. This is because the user satisfaction and the resource efficiency affect the provider's service cost and reputation [4], [5]. Task offloading is one of the most efficient way to address the problem. Task offloading decides which location, which computing node, and what order for task processing. These three sub-decision problems are respectively named as offloading decision, task assignment, and task ordering [3]. Most of existing works only concerned one or two of these decision problems. Some works didn't concern the offloading decision problem. They ignore the capacity of device resources even though user devices have comparable resources to personal computer nowadays [6]. Some other works left the offloading decision making to users. These works suppose users have expertise in the offloading decision. Thus, we study on the task offloading for jointly solving the three sub-decision problems to improve the user satisfaction and the resource efficiency for providing services by E3C.

Unfortunately, the task offloading problem is strongly NP-hard [7]. This means there is no applicable method to solve the problem precisely. Thus, to solve this problem in polynomial time complexity, two kinds of methods, heuristics [8] and meta-heuristics [9], are exploited by existing works. A heuristic method is exploiting a local search strategy for a specific problem to provide a local optimal solution. Heuristics usually take little time. Meta-heuristics use some random methods and generalized local search or global search strategies. In general, meta-heuristic methods cost much more computation time but can provide a better performance, compared with heuristic methods. Thus, in this paper, we combine the benefits of both heuristics and meta-heuristics to design a hybrid method for addressing the task offloading problem in E3C. To be specific, we exploit the genetic algorithm (GA) [10] to address the offloading decision and the task assignment sub-problems, and apply the earliest deadline first (EDF) to solve the task ordering sub-problem. This is because GA is one of the most representative meta-heuristics, and usually has a good performance. EDF is a heuristic method specially designed for scheduling deadline-constrained tasks. In addition, GA can be implemented easily, and has been used for solving optimization problems in various fields [11]–[16].

In this paper, we concern the task offloading for E3C environments with deadline constraints, with the exploitation of computing resources in local devices. We aim at

Manuscript received December 13, 2021; revised March 28, 2022. This work is supported by the key scientific and technological projects of Henan province grant number 212102210096, 212102210410, the key scientific research projects of Henan higher school grant number 21A520050, National Natural Science Foundation of China grant number 61872043, 61975187, 62072414, Qin Xin Talents Cultivation Program, Beijing Information Science & Technology University grant number QXTCP B201904, the fund of the Beijing Key Laboratory of Internet Culture and Digital Dissemination Research grant number ICDDXN004.

B. Wang is a lecturer of Software Engineering College, Zhengzhou University of Light Industry, China (e-mail: wangb@zzuli.edu.cn).

B. Lv is a network engineer of Linyi Branch of China Mobile Communications Group Shandong Co., Ltd., China (e-mail: lvbin@sd.chinamobile.com).

Y. Song is an associate professor of Beijing Advanced Innovation Center for Materials Genome Engineering, and Beijing Key Laboratory of Internet Culture and Digital Dissemination, Beijing Information Science and Technology University, China (corresponding author to provide e-mail: songying@bistu.edu.cn).

optimizing the user satisfaction and the resource efficiency. In this paper, we use the number of completed tasks and the resource utilization for quantifying the user satisfaction and the resource efficiency, respectively. Our method is also applicable for other metrics. To address the problem in reasonable time, we propose a hybrid heuristic method by integrating EDF into GA. The proposed method exploits the global search ability of GA with integer coding to decide the assignment of tasks to computing cores. In the implementation of GA, a chromosome represents a solution of joint offloading decision and task assignment sub-problems. There is a one-to-one correspondence between tasks and genes of each chromosome. The value of a gene represents the core where the corresponding task is assigned. For the sub-problem of task ordering, the proposed method applies EDF for each computing core. To evaluate the proposed method, we conduct extensive simulated experiments. Experiment results show that our proposed method achieves 20.7%–21.2x better user satisfaction and 3.1%–120% better resource efficiency than several existing heuristics and meta-heuristics.

In the following, we present the formulation of the task offloading problem and our hybrid heuristic method respectively in section II and section III. We illustrate the simulated experiment environment and analysis experiment results in section IV. We illustrate related works in section V. In section VI, we conclude our work.

II. Problem Statement

We consider a E3C system, as shown in Fig. 1. The system consists of multiple user devices, several edge computing centres (edge for short), and one cloud. Each edge is equipped with one or more servers (ES). The cloud provides several types of cloud servers (CS). For each user, its request tasks can be processed locally if its device has enough resources. Otherwise, some tasks can be offloaded to ESs or CSs for their executions. In general, each device has a wireless network connection with an edge. Each ES processes tasks offloaded by user devices that have network connections with the ES. When there is no available resources in the local device or ESs for some tasks, these tasks will be offloaded to CSs that is rented in the cloud.

In the considered E3C system, there are M user devices, E ESs, and C CSs. Without loss of generality, we use s_i, $1 \leq i \leq M+E+C$ to respectively represent these devices/servers. When $1 \leq i \leq M$, s_i represents ith device. If $M+1 \leq i \leq M+E$, s_i is $(i-M)$th ES. When $M+E+1 \leq i \leq M+E+C$, s_i represents $(i-M-E)$th CS. For s_i, there are n_i computing core, and each core has g_i computing capacity. We use binary constants z_{il}, $1 \leq i,l \leq M+E+C$ to represent the network topology of the system. The value of z_{il} is the bandwidth between s_i and s_l. if there is no network connection between s_i and s_l, $z_{il}=0$. In addition, $z_{ii}=\infty$, which means there is no data transmission delay within a device/server.

There are T tasks (denoted by t_j, $1 \leq j \leq T$) needed to be processed by the system. We use binary constants o_{ij}, $1 \leq i \leq M, 1 \leq j \leq T$ to identify the ownership relationship between tasks and devices. If t_j is launched by s_i, $o_{ij}=1$, and otherwise $o_{ij}=0$. We use d_j to represent the deadline of t_j, which means t_j must be finished before d_j for the user satisfaction. Without loss of generality, we assume $d_1 \leq d_2 \leq \ldots \leq d_T$. We concern hard deadline tasks in this paper, i.e., a task is accepted only when its deadline constraint is met. We denote the amounts of the input data and the output data as in_j and out_j for t_j, respectively. We represent the amount of computing resource required by t_j as com_j. Then the processing time (pt_{ij}) of t_j when it is assigned to one core of s_i can be calculated by Eq. (1)-(4). Where ct_{ij} is the computing time. $\sum_{l=1}^{M} o_{lj}z_{li}$ is the network bandwidth between s_i and the device that launches t_j. it_{ij} and ot_{ij} are the data transmission delays of the input data and the output data.

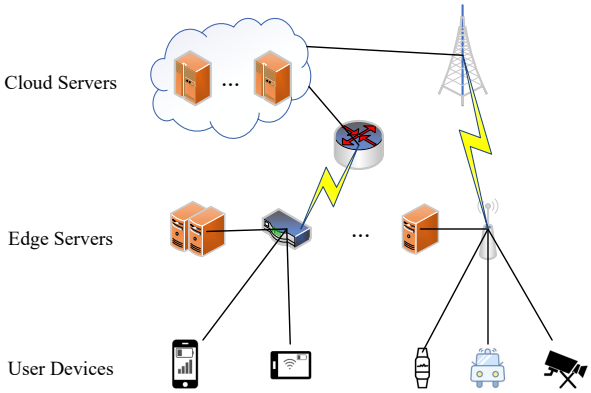

Fig. 1. The edge-cloud cooperative computing (E3C) environment.

$$pt_{ij} = it_{ij} + ct_{ij} + ot_{ij} \tag{1}$$

$$it_{ij} = \frac{in_i}{\sum_{l=1}^{M} o_{lj}z_{li}} \tag{2}$$

$$ct_{ij} = \frac{com_j}{g_i} \tag{3}$$

$$ot_{ij} = \frac{out_i}{\sum_{l=1}^{M} o_{lj}z_{li}} \tag{4}$$

To formulate the task offloading problem, we express the solution by the binary variables defined as Eq. (5), where x_{ijk} represents whether t_j is assigned to kth computing core of s_i for the task processing.

$$x_{i,j,k} = \begin{cases} 1, & \text{if } t_j \text{ is assigned to } k\text{th core of } s_i \\ 0, & \text{else} \end{cases}, \quad 1 \leq i \leq M+E+C, 1 \leq j \leq T, 1 \leq k \leq n_j \tag{5}$$

Then the number of tasks that are assigned to a core for processing in the system, i.e., the number of accepted tasks, is $N = \sum_{j=1}^{T}\sum_{i=1}^{M+E+C}\sum_{k=1}^{n_j} x_{i,j,k}$.

For tasks assigned to a computing core, with deadline constraints, they can be finished only when each of them can be finished in the increasing order of deadline [17]. Then for all tasks assigned to a core, we can derive their finish time, on the basis of the earliest-deadline-first execution sequence. In this paper, we assume the bottleneck resource is either the computing or the input network bandwidth for tasks' processing. This is because the amount of the output data is usually much less than that of the input data in the real world.

In a computing core, the computing of a task can be started only when the transfer of its input data is completed and the core is available to the task. When t_j assigned to kth core of s_i, The earliest finish time of transferring its

input data is $\sum_{m=1}^{j} x_{i,m,k} \cdot it_{i,m}$. Where $\sum_{m=1}^{j-1} x_{i,m,k} \cdot it_{i,m}$ is transfer time exhausted by tasks with earlier deadline. The earliest available time of a computing core for a task is the completion time of the computing of tasks with earlier deadline. Hence, for t_j assigned to kth core of s_i, the earliest available time of computing resources is $\max_{1 \le m < j}\{x_{i,m,k} \cdot (ft_m - ot_{i,m})\}$. Where ft_m is t_m's finish time, and therefore, $ft_m - ot_{i,m}$ is the computing completion time of t_m. Then the earliest start time of t_j's computing when it is assigned to kth core of s_i is $\max\{\sum_{m=1}^{j} x_{i,m,k} \cdot it_{i,m}, \max_{1 \le m < j}\{x_{i,m,k} \cdot (ft_m - ot_{i,m})\}\}$. Thus its finish time ft_j can be calculated by Eq. (6).

$$ft_j = \sum_{i=1}^{M+E+C} \sum_{k=1}^{n_j} x_{i,j,k} \cdot (\max\{\sum_{m=1}^{j} x_{i,m,k} \cdot it_{i,m}, \max_{1 \le m < j}\{x_{i,m,k} \cdot (ft_m - ot_{i,m})\}\} + ct_{ij} + ot_{ij}), \quad 1 \le j \le T \tag{6}$$

For each computing node (device or server), the occupied time is the latest time of finishing tasks processed in the node, i.e., $\max_{1 \le k \le n_i} \max_{1 \le j \le T}\{x_{i,j,k} \cdot ft_j\}$. So the accumulated amount of occupied computing resources is $\sum_{i=1}^{M+E+C}(n_i \cdot \max_{1 \le k \le n_i} \max_{1 \le j \le T}\{x_{i,j,k} \cdot ft_j\})$. While the effective computing resources for task processing is the computing resources consumed by accepted tasks, i.e., $\sum_{j=1}^{T} \sum_{i=1}^{M+E+C} \sum_{k=1}^{n_j}(x_{i,j,k} \cdot com_j)$. Therefore, the overall CPU utilization of the E3C system, U, can be calculated by Eq. (7).

$$U = \frac{\sum_{j=1}^{T} \sum_{i=1}^{M+E+C} \sum_{k=1}^{n_j}(x_{i,j,k} \cdot com_j)}{\sum_{i=1}^{M+E+C}(n_i \cdot \max_{1 \le k \le n_i} \max_{1 \le j \le T}\{x_{i,j,k} \cdot ft_j\})} \tag{7}$$

Based on the above formulating, we can model the task offloading problem as the following optimization problem.

$$\text{Maximizing} \quad N + U \tag{8}$$

Subject to:

$$(2) - (7) \tag{9}$$

$$ft_j \le d_j \tag{10}$$

$$x_{i,j,k} \in \{0, 1\}, \quad 1 \le i \le M + E + C, 1 \le j \le T, 1 \le k \le n_j \tag{11}$$

In this optimization problem, the decision variables include $x_{i,j,k} \in \{0,1\}, 1 \le i \le M+E+C, 1 \le j \le T, 1 \le k \le n_j$. And there are two objectives, maximizing the number of accepted tasks and the overall computing utilization, as shown in Eq. (8). The utilization is no more than 1, thus, the number of accepted tasks is the major optimization objective. Constraints (9) is calculating the objective values and the finish time of each task. Constraints (10) restricts deadlines of accepted tasks. Constraints (11) represent the binary requirements for the decision variables. There are some solvers, e.g., lp_solve [18], can be applied for solving the problem. But these solvers are exponential computational complexity at worst, and thus are not suitable to large scale systems. Thus, in the next section, we design a hybrid method by combining GA and EDF to provide the solution for the task offloading problem.

III. Hybrid Heuristic Task Offloading Method

In this section, we exploit the global search ability of GA and the heuristic search of EDF to provide a task offloading solution for E3C systems. Our hybrid GA method is outlined in Algorithm 1.

As shown in Algorithm 1, our method first initializes chromosomes with randomly generated genes. A chromosome is representing an assignment solution of tasks to computing cores in the E3C. We exploit an integer encoding method to encode an assignment solution into a chromosome. In each chromosome, genes have a one-to-one corresponding relationship with tasks, and denote the computing cores where tasks are assigned.

Algorithm 1 The hybrid heuristic task offloading method

Input: The information of tasks and resources of the E3C system;
Output: The task offloading solution with optimized accepted ratio and resource efficiency;
1: initializing chromosomes randomly;
2: **while** the terminal condition is not satisfied **do**
3: calculating the fitness value for each chromosome using Algorithm 2;
4: updating the best fitness $fitness_{best}$ and the corresponding chromosome ch_{best};
5: crossing chromosomes using single point crossover with a certain probability;
6: mutating chromosomes using uniform mutation with a certain probability;
7: selecting chromosomes by tournament selection;
8: **end while**
9: **return** the solution decoded from ch_{best} based on Algorithm 2;

As exampled in Fig. 2, there are 2 user devices, one ES, and one CS with one core in an E3C environment. Each device or the ES has 2 cores. dc_{ij} is jth core of ith device. ec_i is ith core of ES. vc_i represents the CS core. Three tasks are launched by a device. Tasks launched by the first device are t_1, t_2, t_3, respectively. Tasks launched by the second device are represented by t_4, t_5, t_6, respectively. In such situation, five candidate cores can be used for processing each task, i.e., two device cores, two ES cores and the CS core. These candidate cores are respectively numbered as 1–5 for each task. Then, the chromosome (2, 3, 2, 1, 4, 5) is representing that t_1 and t_3 will be processed in the second local core, t_4 is assigned to the first local core, t_2 and t_5 are respectively offloaded to the first and the second ES cores, and t_6 is offloaded to the ES.

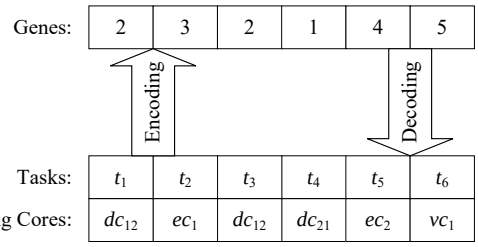

Fig. 2. An example of the integer encoding method.

Then, our method calculates the fitness value for each chromosome to evaluate the performance of the corresponding task offloading solution. As we focus on optimizing the user satisfaction and the resource efficiency in this paper, we

define the fitness ($fitness$) of a chromosome as the number of accepted tasks (N) plus the overall resource utilization (U) adopting the corresponding offloading solution, i.e.,

$$fitness = N + U \tag{12}$$

Where N can be achieved in the procedure of decoding the chromosome into the offloading solution, as shown in line 4 in Algorithm 2. The overall resource utilization (U) can be calculated by dividing the exhausted resource amount for task processing by the total resource amount.

As a chromosome is representing an assignment of tasks to cores, a task ordering method for each core is needed for decoding the chromosome into a task offloading solution. We concern tasks with hard deadline requirements, thus adopt the earliest deadline first (EDF) ordering method because of its optimal accept ratio in such scenario [17], see line 3 in Algorithm 2.

Algorithm 2 Calculating fitness value for a chromosome

Input: The chromosome;
Output: The fitness value;
1: decoding the chromosome into the mapping between tasks and computing cores;
2: **for** each computing core **do**
3: ordering the execution of tasks assigned to the core in the increasing order of deadline;
4: accumulating the number of accepted tasks;
5: **end for**
6: calculating the fitness value using Eq. (12);
7: **return** the fitness value;

After calculating fitness values for all chromosomes, our method selects and records the chromosome with the best fitness (see in line 4 of Algorithm 1), and evolves the population adopting operators of crossover, mutation and selection, respectively shown in lines 5 and 7 of Algorithm 1. We adopt single point crossover to cross two random chromosomes, which generating two new chromosomes. The uniform mutation is applied to guarantee the chromosome diversity. After generating offspring by crossover and mutation, we use tournament selection to eliminate bad chromosomes for population evolution. To improve the chromosome diversity, we view integers as binary when conducting the crossover and mutation operators. Such as, for two genes of 3 (011 in binary) and 4 (100 in binary), there is only two possible offspring when doing crossover operator in integer, while six in binary (011, 100, 000, 111, 010, 101 in binary). Our method repeats these above procedures until the terminal condition is satisfied. There are two terminal conditions can be set: (i) the best fitness is not improved several times continuously, (ii) the number of evolution repeats reaches the maximum. In this paper, we adopt the second terminal condition.

The benefits of our method mainly include the following three aspects. (1) We encode an assignment of tasks to computing cores into a chromosome. This can make better use of the global search ability of GA than existing meta-heuristic methods which encoded the assignment of tasks to computing servers. (2) We exploit an integer encoding method to represent the solution space, which results in a much less solution space than binary encoding-based methods for a task offloading problem. This leads to a much more possibility for achieving a global best solution. (3) We integrate the EDF method into GA procedures to combine the benefits of both local and global search, which is helpful for improving the performance of GA in both the convergence speed and the solution quality.

IV. Performance Evaluation

In this section, for evaluating our method, we first illustrate established simulated experiment environment. Simulation parameters are set referring to existing works. Then, we deeply analyse the experiment results to verify the performance superiority of our method.

A. Experiment Design

In our simulated E3C environment, there are 20 devices, 2 edge computing centres, and 1 CS type. The connectivity between every device and each edge is set randomly. An edge is equipped with 1–4 servers. A core of a device has 1–2GHz computing capacity. For each ES or CS, the computing capacity is set as 2–3GHz for each core. Every device launches 1–100 tasks. Thus, in the simulated environment, there are about 1000 tasks on average. For each task, it requires 100–200GHz computing resources, has 20–500MB input data, and must be finished within 100–1000 seconds. The bandwidth between a device and an ES is set as 10–100Mbps. The bandwidth between a device and a CS is 1–100Mbps. For our method, we set both the population size and the maximum repeat number for the terminal condition as 1000. The probabilities of crossover and mutation are set to 0.1 and 0.5, respectively.

We compare our method (represented as GA_EDF) with the following classical and up-to-date methods, First Fit (FF), First Fit Decreasing (FFD), Earliest Deadline First (EDF), Earliest Finish Time Frist (EFTF) [19], Least Average Completion Time (LACT) [20], Least Slack Time First (LSTF) Mahmud et al. [21], Particle Swarm Optimization (PSO) [22], Hybrid Genetic and EDF with binary encoding (GA_EDF_BIN) [23].

We focus on the following performance aspects of these methods.

- **User satisfaction** strongly affects the income and the reputation of service providers [24]. We use three metrics to quantify the user satisfaction, the accept ratio, the accumulated computing size of accepted tasks, and the total amount of data processed by accepted tasks. For each metric, a greater value is better.
- **Resource efficiency** largely determines the cost of service provision. Two metrics are used for quantifying the resource efficiency in this paper. One is the overall utilization of computing resources, which is one of the most commonly used metric for performance evaluation. Another is the cost efficiency which is the ratio between the accumulated computing size of finished tasks in the cloud and the cost of rented cloud resources.
- **Processing efficiency** is the amount of workload processed per unit time. We use two metrics for its quantifications, which are the computing rate and the data processing rate. The computing rate is the processed computing size per unit time, which is calculated by dividing the completed computing size by the the latest

finish time. The data processing rate is the ratio between the total amount of input data processed by accepted tasks and the latest finish time.

The experimental results are shown in fig. 3–9. We scale each value of each method by one of FF for each metric. Each experiment is repeated 11 times, and the median is reported.

Fig. 3. The relative accepted ratios achieved by various task offloading methods.

B. Experiment Results

1) User Satisfaction: Fig. 3–5 show the relative performance of various methods in user satisfaction. As shown in these figures, we can see that our method achieves 20.7%–9.18x greater accepted ratio, 42.8%–21.2x larger finished computing size, and 32.8%–9.32x more processed data amount, compared with other methods. These results prove that our method has better performance in user satisfaction.

GA_EDF has 20.7%–60.2% better user satisfaction than heuristic methods, as shown in Fig. 3–5. This proves that meta-heuristics can achieve better performance than heuristics in task offloading. The main advantage of meta-heuristics is their global searching abilities which benefit from exploited randomness methods. But if meta-heuristic based methods don't designed carefully, they may achieve a very poor performance. For example, GA_EDF_BIN achieves above 86% less accepted ratio than heuristic methods.

The main reason that GA_EDF has much better performance than GA_EDF_BIN is that the solution space of GA_EDF is much smaller than that of GA_EDF_BIN, which makes great improvement in search efficiency. For example, considering an edge-cloud system with eight tasks and each task has eight candidate core for its processing. When using integer encoding of GA_EDF, the size of the solution space is 8^8. While the solution space size is 2^{64} when using binary encoding of GA_EDF_BIN, as there are $8 * 8 = 64$ genes for each chromosome and each gene has two possible values (0 or 1). In this case, GA_EDF_BIN has a 2^{30} time larger solution space than GA_EDF. In addition, the size of solution space is increased with the task number and the candidate core number of each task, exponentially.

Compared with PSO, our method can achieve much better performance in user satisfaction. The main reason is that PSO employs the meta-heuristic searching strategy for the solution of task assignments to servers, and applies heuristic searching strategy for the task assignment and the task ordering in each server. This can dramatically shrink the search space for meta-heuristics, but don't make full use of their global searching ability [24]. As proved by previous works [25], fine-grained resource allocation helps to improve the resource efficiency. Thus, our method uses the computing core as the resource granularity when making the offloading decision by GA meta-heuristic searching strategy, which helps to achieve a better user satisfaction, as proved by the experiment results.

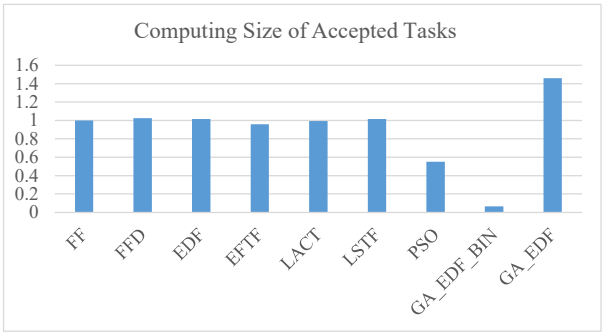

Fig. 4. The relative accepted task sizes achieved by various task offloading methods.

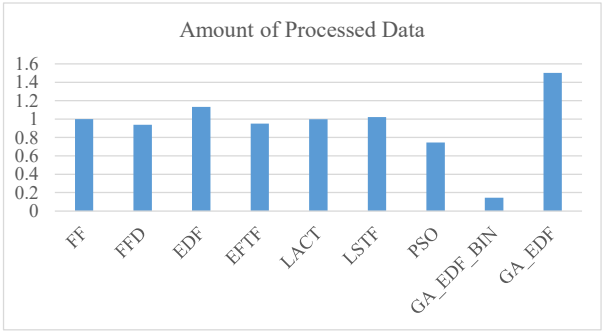

Fig. 5. The relative processed data amounts achieved by various task offloading methods.

2) Resource Efficiency: Fig. 6 and 7 give the resource efficiency achieved by various offloading methods. Compared with other methods, our method has 3.11%–120% higher resource utilization and 4.84%–131% better cost efficiency[1], as shown in Fig. 6 and Fig. 7, respectively. Benefiting from the integration of EDF heuristic into GA meta-heuristic, our method achieves the best resource efficiency.

In general, the resource utilization is related to the ratio between the waiting time of input data and the computing time exhausted for task executions. When there are a longer time for waiting input data, more computing resources are idle, and thus the resource utilization is lower. Thus, GA_EDF_BIN has the worst resource utilization, because it has much grater rDC than other methods. rDC represents the ratio between the input data amount and the computing size of accepted tasks. rDC achieved by GA_EDF_BIN is more

[1]GA_EDF_BIN doesn't have a valid value of the cost efficiency because no task is offloaded to the cloud when employing it for task offloading.

than twice than that by FF, while others achieve comparable rDC to FF. In addition, in our experiment, the correlation coefficient of rDC and the resource utilization is -0.81. Thus, it is a promising research issue to study on the measurement of the resource efficiency by task characteristics in distributed computing systems. This is very helpful for e.g. resource allocation and performance evaluation. Meanwhile, a greater degree of parallelism can lead to a shorter accumulated waiting time for input data. This is because it is more likely to appear the situation that the input data transmission of a task is completed before its previous task is finished. Thus because our method can finish the most tasks, it achieves the best resource efficiency.

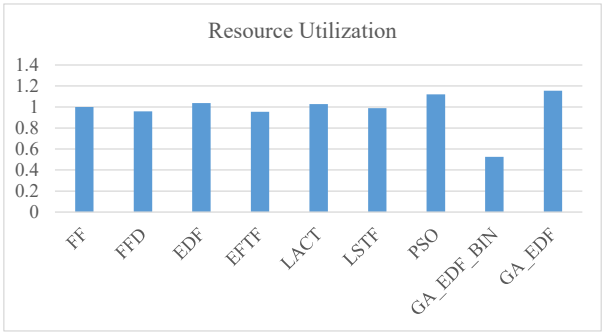

Fig. 6. The relative overall resource utilizations achieved by various task offloading methods.

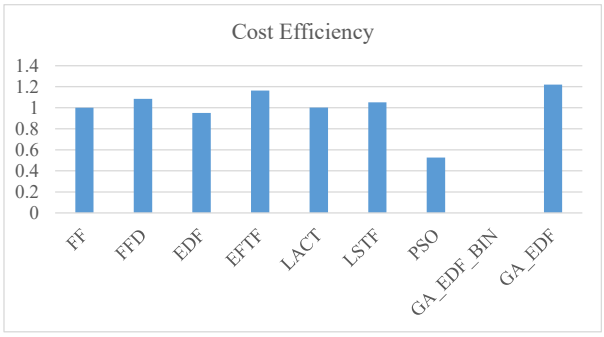

Fig. 7. The relative overall resource utilizations achieved by various task offloading methods.

3) Processing Efficiency: Fig. 8 and 9 are respectively showing the relative computing rates and data processing rates achieved by various offloading methods. As shown in these figures, our method has 40.3%–20.5x greater computing rate and 32.3%–9.17x faster data processing rate, compared with other methods. The processing efficiency is decided by the paralleling speed ratio. Our method provides the maximum accepted ratio, as shown in Fig. 3, i.e., finishes the most tasks. This implies that our method achieves a higher degree of parallelism than other methods in overall. And thus, our method has the best processing efficiency.

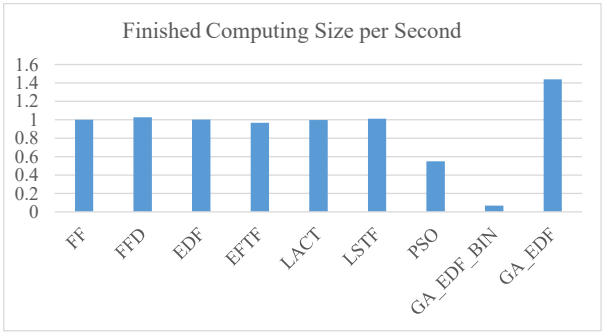

Fig. 8. The relative computing rates achieved by various task offloading methods.

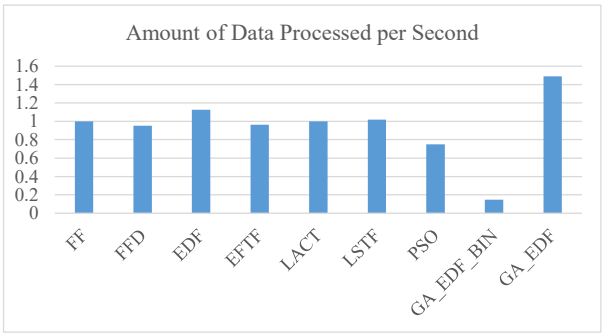

Fig. 9. The relative data processing rates achieved by various task offloading methods.

C. Discussion

According to the experimental results, the hybrid of GA and EDF can achieve better performance than each of them, by combining the global search ability of GA and the heuristic local search ability of EDF. Meantime, GA_EDF use the integer encoding method which improve the search efficiency by reducing the solution space size. These are two main benefits of GA_EDF, the integration of meta-heuristic algorithm and heuristic method and the reduction of the solution space. Thus, there are two research directions for improving the performance of task scheduling strategies. The first one is studying on the hybrid approach of various meta-heuristic algorithms and heuristic methods, such as integrating the crossover and mutation operators into PSO for its particle updating. Another is to design a new encoding method to reduce the size of solution space as much as possible.

V. Related Work

As edge computing and cloud computing are two promising ways to address the resource scarcity problem of mobile devices, there have been several works focusing on the task offloading problem to improve the performance of task execution for these computing environments. Such as, Zakaryia et al. [26] proposed a generic algorithm based offloading method for optimizing the overall execution time of offloaded tasks in an edge computing environment with deadline constraints. Their proposed GA-based method encoded a mapping between task and edge servers into a chromosome. In addition, this work considered that each edge node provided its resources in form of homogeneous virtual machines, which limited the scope of application. The method proposed

by Liu et al. [19] greedily offloaded a task to the edge server providing the minimum finish time, and reduced the amount of the data transferred between servers by task redundant executions, for optimizing the finish time of a workflow. The redundant execution can greatly increase the resource consumption, which leads to a low resource efficiency. These works focused on the execution time optimization, which may cost many resources.

By combining the benefits of both edge computing and cloud computing, Thanh et al. [27] proposed a heuristic method to place the Virtual Network Functions (VNFs) of linear Service Function Chains (SFCs) in E3C environments with Fat-Tree network topology for optimizing the acceptance ratio of offloaded requests. The proposed method first made the offloading decision by gradually moving VNFs from edges to the cloud with the satisfaction of resource capacity constraints. Then, the method placed each VNF with the idea of First Fit (FF) in each edge or the cloud, and heuristically conducted VNF migrations to improve the network distance of VNFs within each SFC. At last, the heuristic method applied the breadth first search to find the shortest network path for a pair of VNFs that belonged to a SFC and were allocated to different computing tiers, and constructed a set of physical network links according to VNF locations for each edge. Dedas [20] was designed for optimizing the number of tasks that meet the deadlines and minimizing the average completion time (ACT) of the tasks. In an edge server or the cloud, Dedas inserted the new task in a server or replaced a scheduled task by the new task to generate a feasible schedule with less ATC. By the scheduling idea, Dedas iteratively assigned the new task to the edge server such that the number of completed tasks is maximized and ACT is minimized. A task is dispatched to the cloud only if edge resources cannot satisfy its requirements. These proposed methods are designed for only one kind of applications, which may lead to a low resource efficiency without exploiting the complementarity of resource requirements for various applications [28]. Xie et al. [22] designed an improved PSO scheduling method for executing a workflow on E3C concerning the tradeoff between the finish time and the resource cost. While for simplification, they assumed all network bandwidths between any two servers are identical. Aburukba et al. [23] studied on the offloading problem optimizing the weighted sum of all request delays with deadline constraints. They formulated the problem as a Mixed Integer Linear Programming (MILP) problem which is NP-Hard, and proposed a binary GA algorithm to solve the MILP problem, where each gene represented a map between a task and a computing core of an E3C server. Chen et al. [29] designed a task scheduling method based on PSO which using the the crossover and mutation operators for the particle updating. Ma et al. [30] adopted a load balance strategy for improving the revenue, which ignoring the resource heterogeneity in E3C systems.

All of above works did not exploit the resources of user devices for processing tasks. Thus, Fizza et al. [31] as well as Elashri and Azim [32] focused on the offloading problem with applying the computing resources of local devices for processing some tasks. The method proposed by Fizza et al. [31] first scheduled tasks to local devices in the priority order of hard, firm, and soft deadline tasks, and then offloaded as many as tasks to the edge processors with optimized communication delay when all task cannot be finished by local resources. For tasks whose requirements cannot be satisfied by local or edge processors, their method offloaded them to the cloud. Elashri and Azim [32] tried to improve the computing energy consumption for real-time embedded systems with the periodic resource model. Their method adjusted the processor speed of each device into the minimal value such that hard real-time tasks can be finished within their respective deadline locally. They also processed weak real-time tasks locally when these tasks' requirements cannot be met to edge servers. For soft real-time tasks with high power consumptions and processor usages, they considered to offload these tasks to the cloud. This work did not concern the economic cost of cloud resources, when making offloading decisions. This may lead to many tasks offloaded to the cloud, and thus result in a high cost for cloud resources.

Different from these existing works, our work focuses on optimizing both the user satisfaction and the overall resource efficiency for E3C environments, considering the exploitation of the local device resources for processing tasks. Our work focuses on jointly solving the problems of offloading decision, task assignment, and task ordering.

VI. Conclusions

In this paper, we focus on the task offloading problem in E3C environments. We aim at optimizing the user satisfaction and the resource efficiency. We first formulate the problem as a binary non-linear programming, and then propose a hybrid method by combining the global search ability of GA and the heuristic search strategy of EDF to solve the problem. Simulated experimental results prove that our method has better performance in both user satisfaction and resource efficiency, compared with several existing heuristic and meta-heuristic methods.

In this paper, we focus on independent tasks, as done in many published works. Independent tasks are a kind of very common application in real world. While the inter-dependent relationship among tasks of workflow jobs remarkably increases the complexity of task scheduling with the complex network topology in a E3C. In the next, we will study on the scheduling of dependent tasks in E3C, to extend the application scope of our method.

References

[1] A. Aliyu, A. H. Abdullah, O. Kaiwartya, S. H. H. Madni, U. M. Joda, A. Ado, and M. Tayyab, Mobile Cloud Computing: Taxonomy and Challenges, *Journal of Computer Networks and Communications*, *2547921*, 23 pages, 2020.

[2] Hua-Ping Wu, Hui Li, and Xiao-Lan Sun, "Evolutionary Game for Enterprise Cloud Accounting Resource Sharing Behavior Based on the Cloud Sharing Platform," IAENG International Journal of Applied Mathematics, vol. 51, no.1, pp125-132, 2021

[3] B. Wang, C. Wang, W. Huang, Y. Song, and X. Qin, A survey and taxonomy on task offloading for edge-cloud computing, *IEEE Access*, *8*, 186 080–186 101, 2020.

[4] A. Chakraborty, S. Misra, and A. Mondal, QoS-aware dynamic cost management scheme for sensors-as-a-service, *IEEE Transactions on Services Computing*, *in Press*, 12 pages, 2020.

[5] Y. Wang, J. T. Zhou, and X. Song, A utility game driven qos optimization for cloud services, *IEEE Transactions on Services Computing*, *in Press*, 14 pages, 2021.

[6] C.-J. Wu, D. Brooks, K. Chen, D. Chen, S. Choudhury, M. Dukhan, K. Hazelwood, E. Isaac, Y. Jia, B. Jia, T. Leyvand, H. Lu, Y. Lu, L. Qiao, B. Reagen, J. Spisak, F. Sun, A. Tulloch, P. Vajda, X. Wang, Y. Wang, B. Wasti, Y. Wu, R. Xian, S. Yoo, and P. Zhang, Machine learning at facebook: Understanding inference at the edge, *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp331–344, 2019.

[7] J. Du and J. Y.-T. Leung, Complexity of scheduling parallel task systems, *SIAM Journal on Discrete Mathematics*, *2(4)*: 473–487, 1989.

[8] R. NoorianTalouki, M. H. Shirvani, and H. Motameni, A heuristic-based task scheduling algorithm for scientific workflows in heterogeneous cloud computing platforms, *Journal of King Saud University - Computer and Information Sciences*, *in press*, 12 pages, 2021.

[9] E. H. Houssein, A. G. Gad, Y. M. Wazery, and P. N. Suganthanc, Task Scheduling in Cloud Computing based on Meta-heuristics: Review, Taxonomy, Open Challenges, and Future Trends, *Swarm & Evolutionary Computation*, *in press*, 41 pages, 2021.

[10] S. Katoch, S.S. Chauhan, and V. Kumar, A review on genetic algorithm: past, present, and future, *Multimedia Tools and Applications*, *80(5)*: 8091–8126, 2021.

[11] Nan Wang, Jie-Sheng Wang, Yong-Xin Zhang, and Tian-Zhu Li, "Two-dimensional Bin-packing Problem with Rectangular and Circular Regions Solved by Genetic Algorithm," IAENG International Journal of Applied Mathematics, vol. 51, no.2, pp268-278, 2021.

[12] G.T. Reddy, M.P.K. Reddy, K. Lakshmanna, D.S. Rajput, R. Kaluri, and G. Srivastava, Hybrid genetic algorithm and a fuzzy logic classifier for heart disease diagnosis, *Evolutionary Intelligence*, *13(2)*: 185–196, 2020.

[13] C. Kim, R. Batra, L. Chen, H. Tran, and R. Ramprasad, Polymer design using genetic algorithm and machine learning, *Computational Materials Science*, *110067*, 6 pages, 2021.

[14] F. Rosso, V. Ciancio, J. Dell'Olmo, and F. Salata, Multi-objective optimization of building retrofit in the Mediterranean climate by means of genetic algorithm application, *Energy and Buildings*, vol. 216, 109945: 18 pages, 2020.

[15] Yechun Yu, Xue Deng, Chuangjie Chen, Kai Cheng. Research on Fuzzy Multi-objective Multi-period Portfolio by Hybrid Genetic Algorithm with Wavelet Neural Network. Engineering Letters, vol. 28, no.2, pp594–600, 2020.

[16] Glen Cancian, and Wayne Pullan, "Evaluation of Tailored Mutation Operator in a Parallel Genetic Algorithm for Pavement Maintenance Treatment Scheduling," Lecture Notes in Engineering and Computer Science: Proceedings of The International MultiConference of Engineers and Computer Scientists 2021, 20-22 October, 2021, Hong Kong, pp7-12.

[17] M. L. Pinedo, Deterministic Models: Preliminaries, *Scheduling: theory, algorithms, and systems*, 5th ed.; Publisher: Springer, pp. 13–32, 2016.

[18] lpsolve: A mixed integer linear programming (MILP) solver. Available online: http://sourceforge.net/projects/lpsolve, (accessed on 24/02/2022).

[19] L. Liu, H. Tan, S. H.-C. Jiang, Z. Han, X.-Y. Li, and H. Huang, Dependent task placement and scheduling with function configuration in edge computing, *Proceedings of the International Symposium on Quality of Service (IWQoS'19)*, pp1–10, 2019.

[20] J. Meng, H. Tan, X. Li, Z. Han, and B. Li, Online deadline-aware task dispatching and scheduling in edge computing, *IEEE Transactions on Parallel and Distributed Systems*, *31(6)*: 1270–1286, 2020.

[21] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, Profit-aware application placement for integrated fog–cloud computing environments, *Journal of Parallel and Distributed Computing*, *135*, pp177–190, 2020.

[22] Y. Xie, Y. Zhu, Y. Wang, Y. Cheng, R. Xu, A. S. Sani, D. Yuan, and Y. Yang, A novel directional and non-local-convergent particle swarm optimization based workflow scheduling in cloud-edge environment, *Future Generation Computer Systems*, *97*: 351–378, 2019.

[23] R. O. Aburukba, M. AliKarrar, T. Landolsi, and K. El-Fakih, Scheduling internet of things requests to minimize latency in hybrid fog-cloud computing, *Future Generation Computer Systems*, *111*: 539 – 551, 2020.

[24] B. Wang, J. Cheng, J. Cao, C. Wang, W. Huang. Integer particle swarm optimization based task scheduling for device-edge-cloud cooperative computing to improve SLA satisfaction. *PeerJ Computer Science*, *8:e893*, 22 pages, 2022.

[25] M. Tirmazi, A. Barker, N. Deng, M. E. Haque, Z. G. Qin, S. Hand, M. Harchol-Balter, and J. Wilkes. Borg: the next generation. *Proceedings of the Fifteenth European Conference on Computer Systems (EuroSys '20)*. Association for Computing Machinery, New York, NY, USA, pp30:1–14, 2020.

[26] S. A. Zakaryia, S. A. Ahmed, and M. K. Hussein, Evolutionary offloading in an edge environment, *Egyptian Informatics Journal*, *22(3)*: 257-267, 2021.

[27] N. H. Thanh, N. T. Kien, N. Van Hoa, T. T. Huong, F. Wamser, and T. Hossfeld, Energy-aware service function chain embedding in edge-cloud environments for IoT applications, *IEEE Internet of Things Journal*, *8(17)*: 13465 - 13486, 2021.

[28] B. Wang, C. Wang, Y. Song, J. Cao, X. Cui and L. Zhang, A survey and taxonomy on workload scheduling and resource provisioning in hybrid clouds, *Cluster Computing*, *23(3)*: 2809–2834, 2020.

[29] X. Chen, J. Zhang, B. Lin, Z. Chen, K. Wolter, and G. Min, Energy-efficient offloading for DNN-based smart IoT systems in cloud-edge environments, *IEEE Transactions on Parallel and Distributed Systems*, *33(3)*:, 683-697, 2022.

[30] Z. Ma, S. Zhang, Z. Chen, T. Han, Z. Qian, M. Xiao, N. Chen, J. Wu, and S. Lu, Towards Revenue-Driven Multi-User Online Task Offloading in Edge Computing, *IEEE Transactions on Parallel and Distributed Systems*, *33(5)*: 1185-1198, 2022.

[31] K. Fizza, N. Auluck, and A. Azim, Improving the schedulability of real-time tasks using fog computing, *IEEE Transactions on Services Computing*, *in press*, 14 pages, 2019.

[32] S. Elashri and A. Azim, Energy-efficient offloading of real-time tasks using cloud computing, *Cluster Computing*, *23(4)*: 3273–3288, 2020.