A Hybrid Genetic Algorithm for Service Caching and Task Offloading in Edge-Cloud Computing

Li Li[1], Yusheng Sun[2], and Bo Wang[3]*
Guangdong Polytechnic of Science and Technology,
No.351, Kehua Street, Tianhe District, Guangzhou, China, 510640[1]
Software Engineering, Zhengzhou University of Light Industry
No. 136, Kexue Road, High-tech Industrial Development Zone, Zhengzhou, China 45000[2,3]

***Abstract*—Edge-cloud computing is increasingly prevalent for Internet-of-Thing (IoT) service provisioning by combining both benefits of edge and cloud computing. In this paper, we aim to improve the user satisfaction and the resource efficiency by service caching and task offloading for edge-cloud computing. We propose a hybrid heuristic method to combine the global search ability of the genetic algorithm (GA) and heuristic local search ability, to improve the number of satisfied requests and the resource utilization. The proposed method encodes the service caching strategies into chromosomes, and evolves the population by GA. Given a caching strategy from a chromosome, our method exploits a dual-stage heuristic method for the task offloading. In the first stage, the dual-stage heuristic method pre-offloads tasks to the cloud, and offloads tasks whose requirements cannot be satisfied by the cloud to edge servers, aiming at satisfying as many tasks' requirements as possible. The second stage re-offloads tasks from the cloud to edge servers, to get the utmost out of limited edge resources. Experimental results demonstrate the competitive edges of the proposed method over multiple classical and state-of-the-art techniques. Compared with five existing scheduling algorithms, our method achieves 11.3% to 23.7% more accepted tasks and 1.86% to 18.9% higher resource utilization.**

***Keywords*—Cloud computing; edge computing; genetic algorithm; service caching; task offloading**

I. Introduction

Internet-of-Thing (IoT) devices have become ubiquitous in our lives. IoT services are becoming increasingly needed in both the number and variety, as revealed by the Cisco Annual Internet Report [1]. Eventhough some resources are equipped on various IoT devices, they are not enough for satisfacting all requirements of users, due to the limited resource of a device.

To address the issue, mobile cloud computing (MCC) exploits the cloud computing with aboundent computing resources to extend the service ability of IoT devices, by offloading some tasks to the cloud [2]. But the cloud has a poor network performance as it deliveries services over wide area networks (WANs), and thus, it usully cannot process delay-sensitive request tasks. Therefore, edge computing is proposed to address the problem, by placing some resources close to user devices [3] to provide services with low latency.

By combing both benefits of edge and cloud computing, edge-cloud computing (ECC) has begun to be used more and more in both industrial and academic [4]. While the cooperation between the edge and cloud computing needs to be further enhanced for improving the resource efficiency and user satisfaction [5]. The purpose can be achieved by a joint task offloading and service caching strategy for edge-cloud computing.

unfortunately, the joint task offloading and service caching problem (TOSCP) is NP-hard [6], and there is no available method providing the optimal solution within a reasonable time for a large-scale ECC system. There mainly two kinds of ways to solve the TOSCP for providing an accepted solution in polynomial time, heuristics and meta-heuristics. Heuristics uses local search strategies for local optimal solutions with a very few time consumption. On the contrary, meta-heuristics exploit global search strategies inspired by natural and social phenomena. Meta-heuristics usually have better performances, but require much more time costs, compared with heuristics. In this paper, we design a hybrid heuristic method by combining the genetic algorithm (GA), a representative meta-heuristic method, and a dual-stage heuristic search strategy which is designed for improving the cooperation between edge and cloud computing. The introducing of the heuristic method can improve both the quality of chromosomes and the convergence velocity for GA. Different from existing related works [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], the main advantages of our work include the awareness of the heterogeneity between edge and cloud resources, providing the joint solution of the service caching and task offloading, and the integration of the dual-stage heuristic into GA.

The rest of this paper is organized as follows. Section II formulate the TOSCP concerned in this paper. Section III presents our proposed hybrid heuristic method. Section IV evaluates our proposed method by simulated experiments. And finally, Section V concludes the paper.

II. Problem Statement

As shown in Fig. 1, in edge-cloud computing, the service provider uses multiple edge servers (ESs) and a cloud to provide various services for its users, where the cloud provide its resources in the form of virtual machine (VM). Due to a very limited resource, each ES can be deployed only a few services at a time. On the contrary, the cloud has seemingly infinite resources, and can be provide all services all the time. In general, an ES has local area network connections to users close to it. While the cloud provides its services over wide area network, e.g., Internet. Thus, the cloud has a much poorer network performance than ESs. Briefly, the cloud has abundant computing and storage resources but a high data transmission delay, but ESs have limited resources with low network delays.

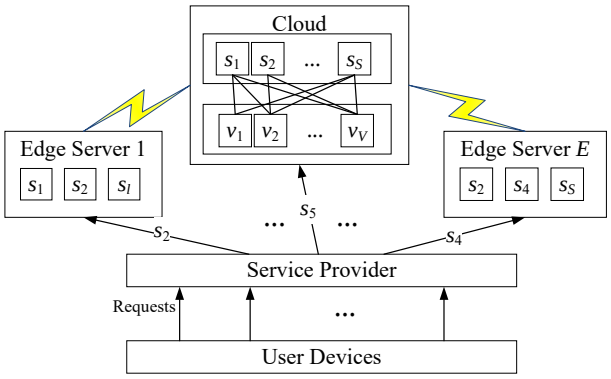

Fig. 1. The Architecture of the Edge-Cloud Computing System.

To deliver services efficiently, the service provider should carefully design the service caching and task offloading strategy, where the service caching decides which services each ES provides with its limited resources, and the task offloading strategy provide the computing node (ES or VM) where each user's request is processed. In this paper, we aim at designing a hybrid heuristic method for solving the TOSCP formulated as followings.

Given an edge-cloud computing system, there are E ESs, and the cloud provide V VMs. Without loss of generality, we use $E+V$ computing nodes, $n_1, ..., n_E$ and $n_{E+1}, ..., n_{E+V}$, to represent these ESs and VMs, respectively. For a computing node, say n_j, it has g_j computing capacity and b_j downlink bandwidth. For each ES, say n_j ($1 \leq j \leq E$), it has m_j storage capacity for service deployments.

The service provider provide S services, $s_1, ..., s_S$, by the edge-cloud computing system. The storage requirement of service s_k is r_k. There are T request tasks, $t_1, ..., t_T$, to be processed in the system. Each task require one of S services. We use the binary constants, $a_{i,k}$ ($1 \leq i \leq T$, $1 \leq k \leq S$), to represent the service that each task requires, where $a_{i,k}$ represents that task t_i require service s_k. For its processing,task t_i needs c_i computing resource, and has h_i amount of input data to be processed. The deadline of t_i is d_i, i.e., t_i must be completed before d_i.

For the formulation of TOSCP, we define binary variables by Eq. (1) to indicate the solution, where $x_{i,j} = 1$ means that t_i is scheduled to n_j for its processing. $x_{i,j} = 0$ represents t_i is not scheduled to n_j.

$$x_{i,j} = \begin{cases} 1, & \text{if } t_i \text{ is scheduled to } n_j \text{ for its processing} \\ 0, & \text{else} \end{cases}, \quad 1 \leq i \leq T, 1 \leq j \leq E+V. \quad (1)$$

As no more than one computing node that each task can be processed,

$$\sum_{j=1}^{E+V} x_{i,j} \leq 1, 1 \leq i \leq T. \quad (2)$$

In addition, we use binary variables $y_{j,k}$ to represent the service cache strategy for ESs, where $y_{j,k} = 1$ indicates service s_k is cached (deployed) on ES n_j ($1 \leq j \leq E$). Noticing that if a task is scheduled on an ES to be processed, its required service must be cached (deployed) on the ES. Then, following equations hold.

$$y_{j,k} = \min\{1, \sum_{i=1}^{T}(x_{i,j} \cdot a_{i,k})\}, 1 \leq j \leq E, 1 \leq k \leq S. \quad (3)$$

For each ES, the total storage space of all services cached on it cannot be larger than its storage capacity, i.e.,

$$\sum_{k=1}^{S}(y_{j,k} \cdot r_k) \leq m_j, 1 \leq j \leq E. \quad (4)$$

For tasks scheduled on a computing node, their finish time can be calculated using the pipeline execution model with the deadline decrease order [10], as shown by Eq. (5). where $ft_{i,j}$ is the finish time of t_i when it is scheduled on n_j, and $ft_{i,j} = 0$ if t_i is not scheduled on n_j. As the output data is usually much less than the input data for a task, this paper ignores the latency of the output data transfer.

$$ft_{i,j} = x_{i,j} \cdot \max\{\max_{d_{ii}<d_i} ft_{ii,j} + \frac{c_i}{g_j}, \max_{d_{ii}<d_i}(ft_{ii,j} - \frac{h_{ii}}{b_j}) + \frac{h_i}{b_j} + \frac{c_i}{g_j}\}, 1 \leq i \leq T, 1 \leq j \leq E+V. \quad (5)$$

Then, the deadline constraints can be formulated as Eq. (6).

$$ft_{i,j} \leq d_i, 1 \leq i \leq T, 1 \leq j \leq E+V. \quad (6)$$

For each computing node, its usage time is the latest finish time of tasks scheduled on it, which is

$$ut_j = \max_{1 \leq i \leq T}\{ft_{i,j}\}, 1 \leq j \leq E+V, \quad (7)$$

and thus, the occupied resource of each node is

$$or_j = ut_j \cdot g_j, 1 \leq j \leq E+V. \quad (8)$$

For a computing node, the amount of resources used for processing tasks is the accumulated resources required by these tasks, i.e.,

$$ur_j = \sum_{i=1}^{T}(x_{i,j} \cdot c_i), 1 \leq j \leq E+V. \quad (9)$$

The resource utilization of each computing node and the overall resource utilization of the edge-cloud computing can be calculated by Eq. (10) and (11), respectively.

$$U_j = \frac{ur_j}{or_j}, 1 \leq j \leq E+V. \quad (10)$$

$$U = \frac{\sum_{j=1}^{E+V} ur_j}{\sum_{j=1}^{E+V} or_j}. \quad (11)$$

Based on the above formulations, TOSCP can be modelled as following optimization objective with constraints (1)–(11).

$$\textbf{Maximizing} \quad N + U \quad (12)$$

Where $N = \sum_{i=0}^{T} \sum_{j=1}^{E+V} x_{i,j}$ is the number of tasks processed by the edge-cloud computing system. Noticing that U is no more than 1, and thus, the major objective is maximizing the number of processed tasks, and the utilization maximization is the minor one. Due to the discreteness of decision variables, this optimization problem is hard to be solved exactly for large-scale problems, as its complexity is exponentially increased with the number of decision variables. Thus, we present a hybrid heuristic method for solving this problem with a reasonable time.

III. Genetic Algorithm with Dual-Stage Heuristic

Our proposed method, DGA, exploits the global search ability of GA, where each chromosome represents a service caching solution for the edge-cloud computing, as shown in Algorithm 1. Meantime, DGA uses a dual-stage heuristic method, as shown in Algorithm 2, to solve the task offloading problem.

As shown in Algorithm 1, first, DGA initializes a population, a set of chromosomes, randomly (line 1). Then DGA repeats the evolution of the population using crossover, mutation, and selection operators (lines 5–7). After the maximum repeat time is reached, DGA provides a service caching and task offloading strategy by decoding the best chromosome with the best fitness. Where the fitness function used by DGA is the optimization objective (12), i.e., the finished task number plus the overall resource utilization.

Each chromosome corresponds to a service caching solution. The length of the chromosome, i.e., the number of genes, is the number of ESs. Genes have a one-to-one relationship with ESs, and the value of a gene indicates which services are cached on the corresponding ES. There are S services, then a gene is a binary string with length S, where binary bits have a one-to-one relationship with services to indicate whether services are cached on the ES (i.e., $y_{j,k}$ for ES n_k in Eq. 3). For example, in an edge-cloud computing, there are 5 services and 2 ESs. The chromosome (00111b, 11100b) indicates that the first three services are cached on the second ES, and the last three services are cached on the first ES.

To increase the rate of convergence, DGA see each gene value as an integer for the population evolution. To ensure the population diversity for large-scale systems, DGA uses uniform crossover, uniform mutation, and tournament selection operators to evolve the population.

Given a chromosome, DGA uses a heuristic method with a dual-stage to map the chromosome into a service caching and task offloading strategy, and calculates its fitness, as shown in Algorithm 2. There are two stages for a task. In the first stage, for each task, DGA finds whether a VM that can finish the task within its deadline. If there is such a VM, the task is pre-scheduled to the VM (lines 3–9). Otherwise, DGA searches whether there is an ES that caches the task's requested service and satisfies the task's deadline constraint. If there is such an ES, the task is scheduled to the ES (lines 10–18). Otherwise, the task cannot be processed by the edge-cloud computing for satisfying its requirements, and thus is rejected. After the first stage is completed, the next stage aims at improving the overall resource cost by re-scheduling some tasks from VMs to ESs (lines 20–26). For each task pre-scheduled to VMs, the second stage looks up an ES that can satisfy requirements of the task. If there is such an ES, the task is re-scheduled to the ES. Otherwise, the task will be processed by the VM where it is pre-scheduled.

Algorithm 1 DGA: The Improved GA with a Dual-Stage Heuristic Search

Input: The information of tasks, services, ESs, and cloud VMs;
Output: A service caching and task offloading strategy;
1: Randomly initializing chromosomes;
2: **while** the maximum iteration number is not reached **do**
3: For each chromosome, calculating its fitness value using Algorithm 2;
4: Updating the best fitness and the best chromosome;
5: Executing uniform crossover operator for chromosomes;
6: Conducting uniform mutation operator on each chromosome;
7: Selecting chromosomes by the tournament selection operator;
8: Increasing the iteration number by one;
9: **end while**
10: **return** the service caching and task offloading strategy decoded from the best chromosome by Algorithm 2;

Algorithm 2 The Decoding by a Dual-Stage Heuristic Search

Input: A chromosome;
Output: The service caching and task offloading strategy, the fitness;
1: Decoding the chromosome into the service caching strategy;
2: **for** Each task **do**
3: **for** Each VM **do**
4: **if** The VM satisfies requirements of the task **then**
5: Pre-scheduling the task to the VM;
6: Accumulating the processed task (N) and the used resource amount (ur_j) based on Eq. (9);
7: **break**;
8: **end if**
9: **end for**
10: **if** The task isn't Pre-scheduled to a VM **then**
11: **for** Each ES **do**
12: **if** The ES satisfies requirements of the task **then**
13: Scheduling the task to the ES;
14: Accumulating the processed task and the used resource amount;
15: **break**;
16: **end if**
17: **end for**
18: **end if**
19: **end for**
20: **for** Each task pre-scheduled to VMs **do**
21: **for** Each ES **do**
22: **if** The ES satisfies requirements of the task **then**
23: Scheduling the task to the ES;
24: **end if**
25: **end for**
26: **end for**
27: Calculating the resource utilization (U) by Eq. (11);
28: Calculating the fitness: $N + U$;
29: **return** the service caching and task offloading strategy, the fitness;

IV. Performance Evaluation

In this section, we evaluate the performance of DGA based on a simulated edge-cloud system, to verify the high efficiency

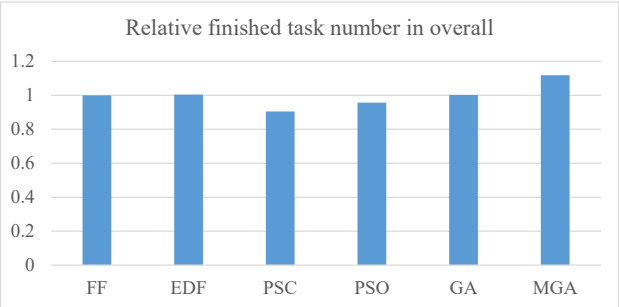

Fig. 2. The Normalized Numbers of Finished Tasks, Achieved by Various Methods.

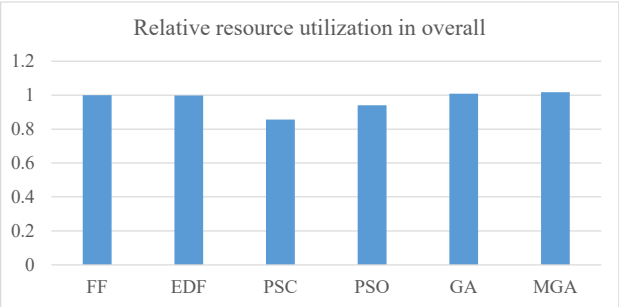

Fig. 3. The Normalized Resource Utilization Achieved by Various Methods.

of DGA in solving TOSCP.

A. Experiment Design

In the simulated edge-cloud system, we randomly generate 1000 tasks, 100 services, 10 ESs and one VM type. The computing resource required by a task is randomly set between 0.5GHz and 1.2GHz. The input data amount of each task is generated randomly in the range of [5MB, 6MB]. These two parameters are referring to [11]. The deadline is set to [1, 5] seconds for each task. Referring to [12], the storage space required by a service is in the range of [40, 80]GB, randomly. Each ES has 20GHz computing capacity. The network bandwidth between an ES and a user device is 60Mbps. The VM type is configured with 5GHz computing capacity and 15Mbps network bandwidth.

There are five benchmark methods in our experiments, First Fit (FF), Earliest Deadline First (EDF), Popularity-based Service Caching (PSC) [13], Particle Swarm Optimization (PSO) [5], and Genetic Algorithm (GA) [14]. The metrics used for the performance evaluation include the finished task number (the number of tasks whose requirements are satisfied) and the overall resource utilization, which used frequently to quantify the user satisfaction and the resource efficiency, respectively.

We repeat our experiment 100 times. In each repeat, we first generate an edge-cloud system with randomly set parameters. Then, we test the performance of our method and the five benchmark methods, respectively, in the generated system. To highlight the relative performance among these methods, we normalize each metric value of each method by dividing it by that of FF. In the following, we report the average normalized value for each metric and each method.

B. Experiment Results and Analysis

Fig. 2 shows the normalized numbers of tasks whose requirements are satisfied, when applying various methods. As shown in the figure, we can see that DGA can finish 11.3% to 23.7% more tasks than other methods. This verifies that our method has a better performance in optimizing the user satisfaction, compared with other method. This is benefited from the high population diversity of GA and the high efficiency of the dual-stage heuristic.

The normalized performances of various methods in resource utilization are shown in Fig. 3. From the figure, we can see that DGA achieves 1.86% to 18.9% high resource utilization than other methods. Thus, our method has a better resource efficiency.

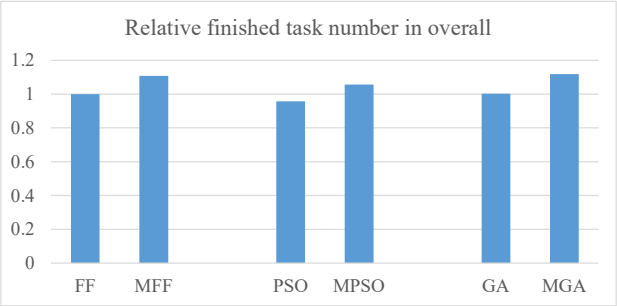

Fig. 4. The Improvement of the Dual-Stage Heuristic for Various Methods

One of the main advantages of our method is exploiting the dual-stage heuristic. The dual-stage heuristic can be applied for improving the performance of other methods, e.g., FF, PSO. Next, we conduct experiments for evaluating the improvement of the dual-stage heuristic for various methods, by comparing the performance differences between FF/PSO/GA and DFF/DPSO/DGA, where DFF/DPSO/DGA is the method of integrating FF/PSO/GA with the dual-stage heuristic. The results are shown in Fig. 4. DFF/DPSO/DGA can finish 10.8%/10.5%/11.6% more tasks than FF/PSO/GA. Thus the dual-stage heuristic can improve performance by about 10 percent for various methods.

V. Conclusions

We studied the service caching and task offloading problem for edge-cloud computing in this paper. We first formulated the problem as an binary non-linear programming with two optimization objectives, the finished task number and the resource utilization. Then, for solving the problem in polynomial time, we proposed a hybrid method which integrates GA and a dual-stage heuristic. Finally, we conducted extensive simulated experiments to evaluate the performance of our proposed method. Experiment results showed that our method can finish up to 23.7% more tasks and achieves up to 18.9% higher resource utilization, compared with five of classical and latest works.

In this paper, we focus on independent tasks without any data or logic dependency relationship. In the future, we will extent our work to support the workflow applications

consisting of interdependent tasks. In addition, we will exploit more global and local heuristic searching strategies to improve the performance of our method.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. The research was supported by the National Natural Science Foundation of China (Grant No. 61872043, 61975187, 62072414), the key scientific and technological projects of Henan Province (Grant No. 202102210149, 212102210096), the Key Scientific Research Projects of Henan Higher School (Grant No. 21A520050), Qin Xin Talents Cultivation Program, Beijing Information Science & Technology University (No. QXTCP B201904), and the fund of the Beijing Key Laboratory of Internet Culture and Digital Dissemination Research (Grant No.ICDDXN004).

REFERENCES

[1] *Cisco Systems, Inc.*, Cisco annual internet report (2018–2023) white paper, https://www.cisco.com/c/en/us/solutions/executive-perspectives/annual-internet-report/index.html, March 2020.

[2] *X. Jin, W. Hua, Z. Wang, and Y. Chen*, A survey of research on computation offloading in mobile cloud computing, Wireless Networks, vol. **28**, 1563–1585, May 2022.

[3] *P. Cruz, N. Achir, and A. C. Viana*, On the edge of the deployment: A survey on multi-access edge computing, ACM Computing Surveys, Mar 2022, just Accepted.(DOI 10.1145/3529758)

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

[5] *B. Wang, J. Cheng, J. Cao, C. Wang, and W. Huang*, Integer particle swarm optimization based task scheduling for device-edge-cloud cooperative computing to improve sla satisfaction, PeerJ Computer Science, vol. **8**, p. e893, 2022.

[6] *J. Xu, L. Chen, and P. Zhou*, Joint Service Caching and Task Offloading for Mobile Edge Computing in Dense Networks, IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, pp 207-215, **2018** (DOI 10.1109/INFOCOM.2018.8485977).

[7] *T. Gao, Q. Tang, J. Li, Y. Zhang, Y. Li and J. Zhang*, A Particle Swarm Optimization With Lévy Flight for Service Caching and Task Offloading in Edge-Cloud Computing, IEEE Access, vol. **10**, pp. 76636-76647, 2022.

[8] *K. Wang, W. Chen, J. Li, Y. Yang and L. Hanzo*, Joint Task Offloading and Caching for Massive MIMO-Aided Multi-Tier Computing Networks, IEEE Transactions on Communications, vol. **70**, no. 3, 1820-1833, 2022.

[9] *W. Fan, J. Han, Y. Su, X. Liu, F. Wu, B. Tang and Y. Liu*, Joint Task Offloading and Service Caching for Multi-Access Edge Computing in WiFi-Cellular Heterogeneous Networks, IEEE Transactions on Wireless Communications, vol. **21**, no. 11, pp. 9653-9667, 2022.

[10] *B. Wang, C. Wang, W. Huang, Y. Song, and X. Qin*, Security-aware task scheduling with deadline constraints on heterogeneous hybrid clouds, Journal of Parallel and Distributed Computing, vol. **153**, pp. 15-28, 2021.

[11] *Y. Dai, D. Xu, S. Maharjan, G. Qiao, and Y. Zhang*, Artificial intelligence empowered edge computing and caching for internet of vehicles, IEEE Wireless Communications, vol. **26**, no. 3: pp. 12–18, 2019.

[12] *G. Zhang, S. Zhang, W. Zhang, Z. Shen, and L. Wang*, Joint service caching, computation offloading and resource allocation in mobile edge computing systems, IEEE Transactions on Wireless Communications, vol. **20**, no. 8, pp. 5288–5300, 2021

[13] *X. Wei, J. Liu, Y. Wang, C. Tang, and Y. Hu*, Wireless edge caching based on content similarity in dynamic environments, Journal of Systems Architecture, vol. **115**, p. 102000, 2021

[14] *B. Wang, B. Lv, and Y. Song*, A hybrid genetic algorithm with integer coding for task offloading in edge-cloud cooperative computing, IAENG International Journal of Computer Science, vol. **49**, no. 2, pp. 503–510, 2022.