Efficiency-oriented Task Offloading with Quality Level Constraint in Green Edge Computing

Maosheng Zhu, Xi Li, Hong Ji, Heli Zhang
Key Laboratory of Universal Wireless Communications, Ministry of Education
Beijing University of Posts and Telecommunications. Beijing, P.R. China
Email: {zhumaosheng0712, lixi, jihong, zhangheli}@bupt.edu.cn

GLOBECOM 2022 - 2022 IEEE Global Communications Conference | 978-1-6654-3540-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/GLOBECOM48099.2022.10001279

***Abstract*—Edge computing ensures close processing in proximity to mobile devices (MD) via task offloading, resulting in a timely manner to support diversified services with low latency tolerance. Nevertheless, for the proliferation of services, the massive connection between MDs and edge servers (ES) results in huge energy consumption, leading to sustainability issues for greenhouse gas emissions. Besides, quality levels within the same service (e.g., accuracy of object detection, and clarity of video) become subdivided, requiring tighter service environment constraint of ESs for such subdivided quality levels. In this paper, we investigate efficiency-oriented task offloading for services with subdivided quality levels within quality level constraint of ESs. Specifically, we first devise a unified quality-aware service model to abstract out service structure, i.e., the quality-relation, number, and dependency of tasks within it. Then, the offloading algorithm is proposed based on an optimized version of non-dominated sorting genetic algorithm-II (NSGA-II), in which due to the heterogeneity of services, we also embed a scheduling algorithm for services in advance to improve tasks parallelism for concurrent execution and NSGA-II adaptation. Additionally, compared with other algorithms, simulation results demonstrate that our proposed algorithm not only significantly improves the convergence, but also optimizes its energy and latency efficiencies.**

***Index Terms*—Green edge computing, quality level constraint, NSGA-II, scheduling**

I. Introduction

With the persistent advances in Information and Communication Technologies, edge computing has proliferated over the years, which provides computing resources in proximity to mobile devices (MD) [1], [2]. Thus, diverse services with latency-sensitive and resource-hunger nature are flourishing on MDs [3], that MDs can offload all/partial of tasks in such services to edge servers (ES) for latency optimization. However, the massive connection between MDs and ESs is playing a major role in increasing energy consumption. According to Natural Resources Defense Council (NRDC) report [4], 100 million metric tons of carbon dioxide (CO_2) will be released from communication networks by 2022, and continue to increase due to the rolling out of 5G and beyond. Thus, renew challenges are brought by tackling sustainability issues for such greenhouse gas emissions, which emerge a requirement for energy efficient communications and computing.

In practical services, offloading tasks in them to ESs requires a service environment corresponding to this kind of service [5], indicating that tasks are differentiated by service type. Due to insufficient resources in ESs compared to public cloud, service environment deployed in ESs is limited. Thus, a ES can only provide computing resources to specific services, namely service environment constraint here. Moreover, tighter service environment constraint for the same service type is expected due to the subdivided quality levels within services, e.g., accuracy of object detection service, and clarity of video service. Indeed, quality levels are depend on subdivided service environment. For example, the offloading for object detection service [6], accuracy in Mask-RCNN [7] environment are better than in YOLOv3-Tiny [8] environment. It is uniformed namely quality level constraint here. Hence, emerging expectations are put forward for an efficient task offloading that considering the subdivided quality levels natural trend of services and within such quality level constraint.

So far, extensive researches on energy efficiency and latency optimization for task offloading are conducted in green edge computing. Ji et al. [2] devised offloading algorithm for energy efficiency in uncertain communication environment. Goudarzi et al. [3] proposed an application placement technique for concurrent IoT application to jointly minimize the execution time and energy consumption. Besides, service environment constraint is also investigated recently. Zhou et al. [5] proposed an edge intelligence empowered edge computing system by jointly optimizing computation offloading and service caching. Itsumi et al. [6] investigated the offloading of motion vectors for object detection services in edge cloud ensemble scenario. Koubaa et al. [9] study deployment strategies for face recognition in cloud versus edge scenario. Nevertheless, these works are limited to one specific service, lacking a unified abstraction of heterogeneous services with subdivided quality levels and quality level constraint for ESs, which limits the implement of such algorithms to other services in practical.

Different from existing studies, in this paper, we investigate efficiency-oriented task offloading for services with subdivided quality levels within quality level constraint of ESs, by jointly optimizing energy consumption of MDs and latency of services. We not only devise a unified representation of such services, namely quality-aware service model, to abstract out its general structure, but also embed a scheduling algorithm of heterogeneous quality-aware services to enhance execution parallelism. Specifically, a quality-aware service model is firstly established to abstract out the quality-relation, number,

978-1-6654-3540-6/22 © 2022 IEEE

Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on February 11,2023 at 07:22:55 UTC from IEEE Xplore. Restrictions apply.

and dependency of tasks in such services. Then, the offloading problem is formulated to jointly minimize energy consumption for MDs and latency for services with quality level constraint. Next, the offloading algorithm is proposed based on an optimized version of non-dominated sorting genetic algorithm-II (NSGA-II). Due to the heterogeneity of services, we also embed a scheduling algorithm for services in advance to improve tasks parallelism for concurrent execution and NSGA-II adaptation. Finally, compared with other offloading algorithms, simulation results demonstrate that our proposed algorithm not only significantly improves the convergence, but also optimizes its energy and latency efficiencies.

The rest of the paper is organized as follows: We describe the system model and formulate the offloading problem in Section II. Then, we propose the efficient offloading algorithm in Section III. In addition, the simulation results in Section IV reveal the outperformance of our algorithm as compared with other offloading algorithms. Finally, Section V concludes this paper.

II. System Model

We consider a green edge computing system consisting of multi-MDs, multi-ESs, and one public cloud, in which the service types and their quality levels deployed on ESs are limited, and the public cloud is an environment that has all the quality levels of all service types.

A. Quality-Aware Service Model

It is assumed that each service is generated on a MD and can be executed locally or partially offloaded to ESs and cloud for execution. We abstract the service types in this scenario into a set $\mathbb{S}_s = \{1, 2, ..., K\}$ and the service quality levels classification of type k-th service as a set $\mathbb{S}_k = \{1_k, 2_k, ..., L_k\}$. Without losing generality, the tasks generated in the k-th services with l_k-th quality level is denoted as a $DAG_{k,l} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V} = \mathbf{V_1^k} \cup \mathbf{V_2^{k,l}} = \{v_{1,f}, v_{2,f}, ..., v_{n,f}, ..., v_{N,f}\}$ is the set of tasks. It is divided into 2 subset $\mathbf{V_1^k}$ and $\mathbf{V_2^{k,l}}$, which represent the task set unrelated to quality and the task set related to quality, respectively. N tasks are generated in $DAG_{k,l}$, in which $v_{n,f}$ denotes n-th task, where $f \in \{1|2\}$ represents which subset it belongs to, i.e., $v_{n,1} \in \mathbf{V_1^k}$, $v_{n,2} \in \mathbf{V_2^{k,l}}$. $\forall v_{n,f} \in \mathbf{V_1^k}$, it can be executed on the devices deployed with k-th type service environment, while $\forall v_{n,f} \in \mathbf{V_2^{k,l}}$, it can only be executed on the devices deployed with k-th type service and l_k-th quality level environment. In addition, each task is modeled as a 2-tuple: $v_{n,f} = \{w_{n,f}, \varsigma_{n,f}\}$, where $w_{n,f}$ represents the number of instructions contained in $v_{n,f}$, $\varsigma_{n,f}$ is the offloading strategy of $v_{n,f}$, where $\varsigma_{n,f} \in \{s|0, 1, ..., h, ..., H-1, H\}$. If $\varsigma_{n,f} = 0$, $v_{n,f}$ is executed locally on MD, if $\varsigma_{n,f} \in \{s|0, 1, .., H-1\}$, $v_{n,f}$ is offloaded to corresponding ES for execution, and if $\varsigma_{n,f} = H$, $v_{n,f}$ is offloaded to public cloud. Meanwhile, $\forall v_{n,f} \in \mathbf{V}$, it must be executed after the execution of its predecessor tasks, which is denoted as $P(v_{n,f})$. $\mathbf{E} = \{e_{1,2}, e_{1,3}, .., e_{2,3}, ..., e_{i,j}, .., e_{N-1,N}\}$ is the set of directed edges between tasks in $DAG_{k,l}$, which represents the logical order and dependencies between tasks. $\forall e_{i,j} \in \mathbf{E}$, $e_{i,j}$ is modeled as a 3-tuple: $e_{i,j} = \{v_{i,f}, v_{j,f}, d_{i,j}\}$, where $v_{i,f}, v_{j,f}$ represent the start task and the end task of the directed edge connection respectively, and $d_{i,j}$ represents the amount of data transmitted between tasks.

B. Service Environment Model

ESs in this scenario form a set $\mathbb{S}_E = \{ES_1, ES_2, ..., ES_{H-1}\}$, where the service environment in the h-th ES is modeled as $\mathbf{\Theta}^h_{K \times L_k^{max}}$. It is a two-dimensional array with dimension $K \times L_k^{max}$, where K is the number of service types, and L_k^{max} indicates the maximum number of quality level L_k. $\forall \theta_{k,l} \in \mathbf{\Theta}^h_{K \times L_k^{max}}$, if $\theta_{k,l} = 1$, this ES dose have service environment with k-type and l_k-th quality level, otherwise $\theta_{k,l} = 0$, indicating this ES does not have the aforementioned service environment. It should be noted that $\forall v_{n,1} \in \boldsymbol{V}_i^k$, it requires only the line vector $\mathbf{\Theta}_k \neq \emptyset$ to execute on this ES. While $\forall v_{n,1} \in \boldsymbol{V}_i^{k,l}$, $\theta_{k,l} = 1$ must be achieved for its execution on this ES. Additionally, public cloud has much more richer service environment as compared to ES, we model the service environment as $\mathbf{\Theta}^H_{K \times L_k^{max}} = \mathbf{1}^H_{K \times L_k^{max}}$, indicating that public cloud has all the quality levels of all services.

C. Latency Model

Similar to previous works [10], the latency of a $DAG_{k,l}$ mainly includes three parts, i.e., processing delay, transmission delay, and waiting delay.

The processing delay is calculated as:

$$T_p^{k,l} = \sum_{n=1}^{N} \frac{w_{n,f}}{p} = \sum_{n=1}^{N} \begin{cases} \frac{w_{n,f}}{p_o}, & \varsigma_{n,f} = 0 \\ \frac{w_{n,f}}{p_h}, & \varsigma_{n,f} \in \{1, ..., H-1\} \\ \frac{w_{n,f}}{p_H}, & \varsigma_{n,f} = H \end{cases} \quad (1)$$

where p_o, p_h, and p_H are the processing capacity of MDs, ESs, and public cloud respectively.

The transmission delay is represented as:

$$\begin{aligned} T_t^{k,l} &= \sum_{i=0}^{j} \sum_{j=i+1}^{N} \frac{d_{i,j}}{R} \\ &= \sum_{i=0}^{j} \sum_{j=i+1}^{N} \begin{cases} \frac{d_{i,j}}{R_h}, & \varsigma_{i,f} = 0, \varsigma_{n,f} \in \{1, ..., H-1\} \\ \frac{d_{i,j}}{R_H}, & \varsigma_{i,f} = 0, \varsigma_{i,f} = H \\ 0, & otherwise \end{cases} \end{aligned} \quad (2)$$

where R_h and R_H are the transmission rate between cloud and MDs, and between ESs and MDs respectively. Similar to other works, it is assumed that the communication between public cloud and ESs, and among ESs are connected via core network. Thus, we set $R = \infty$ and $T_t^{k,l} = 0$.

Based on queuing theory [11], we adopt $M/M/1/\beta/\infty$ model to calculate the waiting delay on ESs, by which it is assumed that the average rate that a task arrives at an ES obeys the poisson distribution with the parameter λ, and the average service time of a task obeys the exponential

Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on February 11,2023 at 07:22:55 UTC from IEEE Xplore. Restrictions apply.

distribution with the parameter μ. Then, the average waiting time can be calculated as:

$$T_w^{k,l} = N\frac{\Psi}{\mu(1-Q^0)} \tag{3}$$

where

$$\Psi = \frac{\rho}{1-\rho} = \frac{(\beta+1)\rho^{\beta+1}}{1-\rho^{\beta+1}} \tag{4}$$

$$Q^0 = \frac{1-\rho}{1-\rho^{\beta+1}},\ \rho = \frac{\lambda}{\mu} \tag{5}$$

where β represents the maximum number of tasks that an ES can execute, and Q^0 is the probability while no task is executed.

Thus, the total latency of a $DAG_{k,l}$ is represented as:

$$T^{k,l} = T_p^{k,l} + T_t^{k,l} + T_w^{k,l} \tag{6}$$

D. Energy Model

The energy consumption on MD during the execution of a $DAG_{k,l}$ includes two parts, i.e., processing energy and transmission energy.

The processing energy can be calculated as:

$$E_p^{k,l} = \sum_{n=1}^{N}\frac{w_{n,f}}{p}\eta = \sum_{n=1}^{N}\begin{cases}\frac{w_{n,f}}{p_0}\eta_a, & \varsigma_{n,f}=0\\ \frac{w_{n,f}}{p_h}\eta_i, & \varsigma_{n,f}\in\{1,...,H-1\}\\ \frac{w_{n,f}}{p_H}\eta_i, & \varsigma_{n,f}=H\end{cases} \tag{7}$$

where η_a and η_i denote the CPU power of MD on which it is in active and idle mode, respectively.

The transmission energy can be calculated as:

$$E_t^{k,l} = \sum_{i=0}^{j}\sum_{j=i+1}^{N}\frac{d_{i,j}}{R}\eta_t \tag{8}$$

where η_t denote the CPU power of MD on which it is in transmission mode.

Thus, the total energy consumption of a $DAG_{k,l}$ on local MD is represented as:

$$E^{k,l} = E_p^{k,l} + E_t^{k,l} \tag{9}$$

E. Problem Formulation

In this paper, we formulate the efficiency-oriented offloading problem as an optimization problem within quality level constraint of ESs, aiming at jointly minimizing the total latency of quality-aware services with all the quality levels of all service types, and the energy consumption of MDs. It is depicted in the following:

$$\min_{l,\varsigma_{n,f}} \alpha\frac{T^{k,l}}{T_{local}} + (1-\alpha)\frac{E^{k,l}}{E_{local}}, \forall k\in\mathbb{S}_s, \forall l\in\mathbb{S}_k \tag{10}$$

s.t

$$C1: \alpha\in[0,1] \tag{11}$$

$$C2: Exe\left(P\left(v_{n,f}\right)\right) \le Exe\left(P\left(v_{n,f}\right)+v_{n,f}\right) \tag{12}$$

$$C3: \mathbf{\Theta}_k \neq \emptyset,\ \forall v_{n,1}\in\mathbf{V}_1^k, \varsigma_{n,1}\in\{1,2,...H-1 \tag{13}$$

$$C4: \theta_{k,l}=1,\ \forall v_{n,2}\in\mathbf{V}_2^{k,l}, \varsigma_{n,2}\in\{1,2,...H-1\} \tag{14}$$

where T_{local} and E_{local} represent local latency and energy consumption of k-th services with l-th quality level. Additionally, α is the wighted parameter for latency and energy consumption, which can be adjusted based on user requirements. $C1$ represents that to ensure the normalization of the result, the value of this parameter must be in the range of 0 to 1. $C2$ indicates that the execution of the $v_{n,f}$ must be established after all its precursors have been executed. Moreover, $C3$ and $C4$ indicate that the ES to which $v_{n,f}$ is offloaded must have the service environment for execution.

III. Proposed Algorithm

Problem (10) is NP-hard, as we design the offloading algorithm for dependent tasks in quality-aware services $DAG_{k,l}$. Moreover, it becomes more complex as the service types and their quality levels are considered in this paper. To make it trackable, we first design a lightweight scheduling algorithm for quality-aware services to support the parallel executing of tasks. Then, we apply an optimized version of NSGA-II [12] to devise the offloading algorithm for tasks.

A. Scheduling Algorithm for Quality-aware Services

Quality-aware services are heterogeneous with respect to the structure of $DAG_{k,l}$, i.e, the quality-relation, number, and dependency of tasks. The scheduling of quality-aware services can not only increase the degree of parallelism and determine the task execution sequence but also adapt to the CODING step of the proposed offloading algorithm.

Algorithm 1 demonstrates the details of how a quality-related service $DAG_{k,l}$ is scheduled. As multi-sources that a $DAG_{k,l}$ may has, it should be formulated to a signal-source DAG firstly to implement Breadth-First-Search (BFS) algorithm for the obtaining of scheduling index i.e., BFS level. Then, tasks with the same scheduling index can be executed in parallel. Specifically, we scan each task to identify whose $P(v_f^i)=\emptyset$, and add it to a source set S. If $|S|>1$, let add an auxiliary fixed-vertex $v_{o,f}$ as the source root, where we set $w_{o,f}=0$, $\varsigma_{o,f}=0$, and $\forall v_{i,f}\in S$, $d_{o,i}=0$. Thus, a $DAG_{k,l}$ is transformed into a standardized single-source DAG: $DAG_{k,l}^*$. Next, by leveraging BFS algorithm to DAG^*, starting at v_o^i, scheduling indexes of each task are obtained, which is stored in set $Schid_{k,l}$.

In this paper, multi-quality-aware services of multi-service types with multi-quality levels are offloaded at the same time. All $Schid_{k,l}$ form $K\times L_k^{max}$ services are required to be combined for overall scheduling. Thus, we iterate $DAG_{k.l}$ to establish a three-dimensional array named $Schid_{K\times L_k^{max}\times N^{max}}$, where the tasks in a $DAG_{k,l}$ are stored in row vector. Fig. 1a is an example of output from **Algorithm 1**, in which two heterogeneous $DAG_{k,l}$ are scheduled respectively. It can be seen that three source vertices, which are represented in gray, are included in the left DAG while another has one. Moreover, an auxiliary fixed-vertex $v_{o,f}$ is added as the source root on the left, such that both of them are formulated to $DAG_{k,l}^*$. Then, relying on the BFS method, scheduling indexes are obtained,

Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on February 11,2023 at 07:22:55 UTC from IEEE Xplore. Restrictions apply.

which are listed in each vertex. Fig. 1b depict the combination of two DAG in an array $Schid_{K\times L_k^{max}\times N^{max}}$.

B. Offloading Algorithm for quality-aware services

We propose the offloading algorithm for quality-aware services by applying an optimized NSGA-II to specify offloading strategies $\varsigma_{n,f}$ for all tasks in $Schid_{K\times L_k^{max}\times N^{max}}$. NSGA-II was first proposed in [12]. Compared to other genetic algorithms (GAs), it has a low-complexity sorting algorithm and incorporates elitism, which has a better performance in the preservation of off-spring and the acceleration toward Pareto Frontier convergence. Thus, we implement it in this paper. The proposed offloading algorithm is comprised of four phases, which are discussed in detail as follows.

Phase1: Integer Encoding

After the scheduling step for quality-aware services, $Schid_{K\times L_k^{max}\times N^{max}}$ is obtained. In this phase, we encode it to a chromosome. A chromosome is comprised of multi-tasks from $Schid_{K\times L_k^{max}\times N^{max}}$, named gene here. The value of gene denotes the integer offloading strategy $\varsigma_{n,f}$ in a task. In addition, multi-chromosomes make up a population.

Phase 2: Initialization

In this phase, basic parameters for this algorithm are initialized, i.e., population size N_{pop}, Number of iterations I_{num}, Number of DAGs I. Then, the population A_o is initialized with the random value of genes, i.e., $\varsigma_{n,f}$, in each chromosome. Additionally, genetic operator, i.e., mutation and crossover, is also initialized. Thus, a new population B_o is obtained with this operator.

Phase 3: Fast Non-Dominated Sort

We first combine population A_o and B_o as a new population C_o with size $2N_{pop}$. Then, we scan each chromosome p and initialize set $S_p = \emptyset$ and $n_p = 0$, where the chromosomes in C_o which are dominated by p are selected to form set S_p and counted in n_p. Next, chromosomes that are True for $n_p == 0$ are selected to form the first front F_1, and the $rank$ of these chromosomes are marked as 1. Repeat this process to get fronts $F_1, F_2, ..., F_i ....$ As for F_i, we scan chromosomes in S_p, let $n_p = n_p - 1$, if $n_p == 0$ is True, these chromosomes are selected to from front F_{i+1}. Based on this sort, the chromosomes with $rank = 1, 2, ...$ are selected to form a new population D. Additionally, this loop is stopped until the size of D is reached to N_{pop}.

Phase 4: Calculation of Crowding Distance and Screen for Off-springs

Crowding distance is the basis for the screen for off-spring, which are calculated as follows: firstly, the crowding distance of each chromosome d_p is initialized to 0. Then, the maximum and minimum of $Cost$, i.e., Eq.(10), are obtained. Let M represents the maximum number of $Cost$, i.e, $K \times L \times N_p$. Then, for all M chromosomes, we sort them via $Cost$ in ascending order, and set d_p of the first and last chromosomes to ∞. Then, d_p of other chromosomes are calculated as:

$$d_p = d_p + \frac{Cost(p+1) - Cost(p-1)}{Max[Cost()] - Min[Cost()]}, \forall p \in [1, N_p - 2] \tag{15}$$

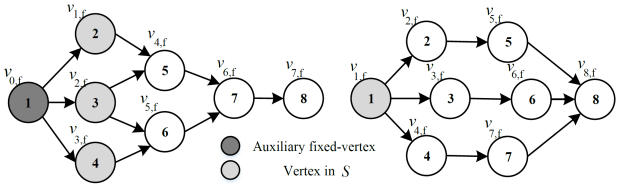
The screen for off-springs is based on the value of $rank$ for each chromosome. For the chromosomes in the last front that reaches the population size N_{pop}, the screen is based on crowding distance, i.e., the chromosomes with larger crowding distance in the last front have higher probability to be off-springs. After the screen, a new population with size N_{pop} is obtained, and genetic operator is implemented in this population. Thus, that next generation with off-springs is formed. Then, let go to **Phase 2** untill number of iterations I_{num} is achieved.

Based on the descriptions above, the details of the proposed offloading algorithm is demonstrated in **Algorithm 2**.

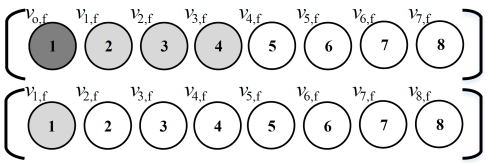
Algorithm 1: Scheduling Algorithm of $DAG_{k,l}$

Input : quality-aware service $DAG_{k,l}$;
Output: signal-source $DAG^*_{k,l}$, scheduling index $Schid_{k,l}$;

initialize $\mathbf{S} = \emptyset$ and $v_{o,f} = \{0, 0\}$;
for $n \in [1, N]$ **do**
  if $P(v_{n,f}) == \emptyset$ **then**
    $\mathbf{S} = \mathbf{S}.add(v_{n,f})$;
  end
  $n = n + 1$;
end
if $|\boldsymbol{S}| >= 1$ **then**
  for *each* $v_{n,f} \in \boldsymbol{S}$ **do**
    $P(v_{n,f}) = P(v_{n,f}).add(v_{o,f})$;
    $d_{o,n} = 0$;
  end
end
$DAG^*_{k,l} = (DAG_{k,l}, v_{o,f})$;
$Schid_{k,l}$=BFS($DAG^*_{k,l}$);
Return: $DAG^*_{k,l}$, $Schid_{k,l}$



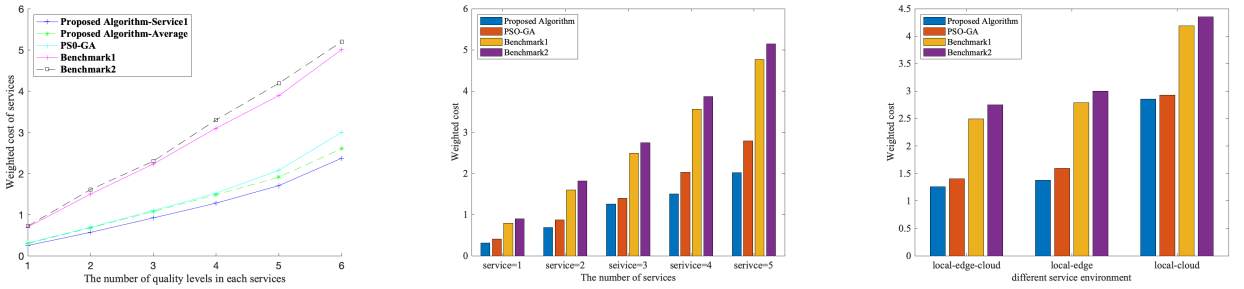
(a) Two scheduled DAGs via **Algorithm 1**.



(b) $Schid_{K\times L_k^{max}\times N^{max}}$ combines two DAGs.

Fig. 1: An example of scheduling for two DAGs.

Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on February 11,2023 at 07:22:55 UTC from IEEE Xplore. Restrictions apply.



(a) Weighted cost for quality levels (b) Weighted cost for services (c) Weighted cost for offloading scenario

Fig. 2: Performance evaluation for Proposed Algorithm, PSO-GA, Benchmark1, and Benchmark2.

Algorithm 2: The Proposed Offloading Algorithm

Input : quality-aware services, I_{num}, N_{pop};
Output: Offloading strategy $\varsigma_{n,f}$, Normalized cost $Cost$;

initialize $\varsigma_{n,f}$ randomly in $\{\varsigma|0,1,..,H\}$;
for $k \in [1, K]$ **do**
  for $l \in [1, L_k]$ **do**
    $Schid_{k,l}$ = **Algorithm 1** $(DAG_{k,l})$;
    $Schid_{K\times L_k^{max}\times N^{max}} += Schid_{k,l}$;
  end
end
while $i \leq I_{num}$ **do**
  initialize population A_i for $Schid_{K\times L_k^{max}\times N^{max}}$ with size N_{pop};
  B_i = genetic operation (A_i);
  $C_i = A_i + B_i$;
  F_i = **Fast Non-Dominated Sort** (C_i);
  initialize $A_{i+1} = \emptyset$, $j = 0$;
  while $len(A_{i+1}) + len(F_j) \leq N_{pop}$ **do**
    d_j = **Crowding Distance** (F_j);
    $A_{i+1} \longleftarrow A_{i+1} + F_j$;
    $j \longleftarrow j + 1$;
    $A_{i+1} \longleftarrow A_{i+1} + F_j[0 : N_{pop} - len(A_{i+1})]$;
    $i \longleftarrow i + 1$;
  end
  D_i = genetic operation (A_i);
  $A_i \longleftarrow D_i$;
end
Return: $\varsigma_{n,f}$, $Cost$

IV. Simulation Result

In this section, the system setup and parameters are first proposed. Additionally, as compared to other algorithms, we evaluate the performance of the proposed offloading algorithm in detail.

A. System Setup and Parameters

Unless otherwise stated, the evaluation parameters and their respective values in this paper are summarized in Table I. In order to reveal the superiority of our proposed algorithm, we implement some other algorithms below for comparison:

TABLE I: Parameter Setting

Evaluation parameters	Default value
Number of MDs, ESs, and public cloud	3, 5, 1
Number of quality levels of services	3
Power of MDs in active, idle, and transmission state	$\eta_a = 0.5W$, $\eta_i = 0.01W$, $\eta_t = 0.1W$
Processing rate of ESs	$p_h \in [2000, 4000]MHz$
Processing rate of public cloud	$p_H = 5000MHz$
Processing rate of MDs	$p_o = 250MHz$
Transmission rate of ESs	$R_h \in [3000, 4000]Kbps$
Transmission rate of public cloud	$R_H \in [500, 1000]Kbps$
Parameters of waiting time model	$\beta = 30$, $\rho = 0.75$
Wighted parameter	$\alpha = 0.5$
Default number of tasks in $DAG_{k,l}$	$N = 4$, if $l_k = 1$;$N = 8$, if $l_k = 2$; $N = 12$, if $l_k = 3$
Default weight of tasks	$w_{n,f} \in [20, 40]Kb$

PSO-GA: This algorithm introduces genetic operators, e.g., crossover, mutation, within GA, to balance both global and local search processes in Particle Swarm Optimization (PSO), which has been widely used in offloading technique in [13], [14] recently.

Benchmark1: We use a round-robin offloading algorithm for all tasks in all quality-aware services as a benchmark1, in which each task is executed in a round-robin manner across all devices in the edge computing networks.

Benchmark2: We use a random offloading algorithm for all tasks in all quality-aware services as a benchmark2, in which each task can be randomly selected to be executed on any device in the edge computing networks.

We evaluate the performance of these offloading algorithms for 100 runs within coverage. In addition, we implement the simulation via using Eclipse on JAVA language in a MacBook with a 4-core 2GHz Intel Core i5 processor and 16GB 3733MHz RAM.

B. Performance Analysis

First, we analyze the convergence of the proposed algorithm, where we set $K = 3$, $L = 3$, and assume that the first and last quartile of tasks is not quality-related, i.e, $v_{n,f} \in \mathbf{V_1^k}$.

Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on February 11,2023 at 07:22:55 UTC from IEEE Xplore. Restrictions apply.

The numerical results are shown in Table II, where we take the average of 100 results for each iteration. The results demonstrate that convergence is obtained after the number of iterations reaches 450. Compared to other researches [10] based on NSGA-II, the convergence is significantly optimized in this paper even though the service model is more complex. It is mainly due to the fact that **Algorithm 1** not only increases the parallelism among tasks, but also better adapts to the integer coding phase. Consequently, it results in a significant reduction of the initial solution space that accelerates the convergence of the proposed offloading algorithm.

Then, we analyze the weighted cost of services for the number of quality levels, where other default parameters remain unchanged. Fig. 2a reveals that the performance of PSO-GA and the proposed algorithm is significantly better than that of the benchmarks. Additionally, we individually evaluate the performance of the proposed algorithm for service 1, the result demonstrates our algorithm has good individual adaptability. Besides, we evaluate the weighted cost for the number of services. Fig. 2b depicts that our proposed algorithm outperforms other algorithms. Indeed, the performance of the proposed algorithm in this paper is better when the number of quality levels and services increases. This is mainly because more tasks need to be offloaded as the number increases, e.g, overall 72 tasks are contained when $l_k = 6$, and 60 tasks are contained when $k = 5$. Thus, the scheduling algorithm we devised in this paper improves the performance more obviously, via parallelism increasing and encoding adapting.

Moreover, we evaluate the performance in the different offloading scenarios. It can be seen from Fig. 2c that the weighted cost in the local-cloud scenario is relatively higher, mainly because the public cloud is far away from MDs, resulting in a lower transmission rate, thus it brings inevitable high transmission cost. The weighted cost in the local-edge scenario is smaller, due to richer computing resources compared to MDs and higher transmission rates for closer physical proximity compared to the cloud. However, the resources will become insufficient if large-scale tasks are generated. Considering the characteristic of public cloud and ESs, we adapt a local-edge-cloud collaborative green edge computing network in this paper to effectively optimize the energy and latency efficiencies.

V. Conclusion

In this paper, we study the efficiency-oriented task offloading problem for services with subdivided quality levels in green edge computing, and propose an efficient offloading algorithm with quality level constraint by jointly minimizing energy consumption of MDs and latency of services. To achieve this goal, we first establish a unified abstraction of services with subdivided quality levels, and formulate this offloading problem as an optimization problem. Then, the offloading algorithm is devised based on an optimized version of NSGA-II, in which due to the heterogeneity of services, a scheduling algorithm for services is embedded in advance to maximize the number of parallel tasks for concurrent execution and NSGA-II adaptation. Finally, compared with other offloading algorithms, simulation results reveals that our proposed algorithm not only accelerates the convergence of results, but also achieves energy and latency efficiencies.

TABLE II: Convergence Analysis

Number of iterations	50	100	150	200	250
Weighted cost	0.400	0.351	0.323	0.325	0.320
300	350	400	450	500	550
0.317	0.316	0.314	0.315	0.315	0.315

VI. Acknowledgement

This work is supported by National Natural Science Foundation of China under Grant 62171061.

References

[1] L. Yang, M. Li, H. Zhang, H. Ji, M. Xiao and X. Li, "Distributed Resource Management for Blockchain in Fog-Enabled IoT Networks," in IEEE Internet of Things Journal, vol. 8, no. 4, pp. 2330-2341, 15 Feb.15, 2021, doi: 10.1109/JIOT.2020.3028071.

[2] T. Ji, C. Luo and L. Yu et al., "Energy-Efficient Computation Offloading in Mobile Edge Computing Systems with Uncertainties," in IEEE Transactions on Wireless Communications, doi: 10.1109/TWC.2022.3142685.

[3] M. Goudarzi, H. Wu and M. Palaniswami et al., "An Application Placement Technique for Concurrent IoT Applications in Edge and Fog Computing Environments," in IEEE Transactions on Mobile Computing, vol. 20, no. 4, pp. 1298-1311, 1 April 2021, doi: 10.1109/TMC.2020.2967041.

[4] M. S. Mekala, Gaurav Dhiman and Gautam Srivastava et al., "A DRL-Based Service Offloading Approach Using DAG for Edge Computational Orchestration," in IEEE Transactions on Computational Social Systems, doi: 10.1109/TCSS.2022.3161627.

[5] Y. Zhou, X. Li and H. Ji et al., "Blockchain-based Trustworthy Service Caching and Task Offloading for Intelligent Edge Computing," 2021 IEEE Global Communications Conference (GLOBECOM), 2021, pp. 1-6, doi: 10.1109 GLOBECOM46510.2021.9685168.

[6] H. Itsumi, F. Beye and Y. Shinohara et al., "Edge Cloud Ensemble with Motion Vectors for Object Detection in Wireless Environments," ICC 2021 - IEEE International Conference on Communications, 2021, pp. 1-6, doi: 10.1109/ICC42927.2021.9500973.

[7] K. He, G. Gkioxari and P. Dollr et al., "Mask r-cnn," in Proceedings of 2017 IEEE International Conference on Computer Vision (ICCV), 2017.

[8] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," arXiv, 2018.

[9] A. Koubaa, A. Ammar and A. Kanhouch et al., "Cloud Versus Edge Deployment Strategies of Real-Time Face Recognition Inference," in IEEE Transactions on Network Science and Engineering, vol. 9, no. 1, pp. 143-160, 1 Jan.-Feb. 2022, doi: 10.1109/TNSE.2021.3055835.

[10] K. Peng , M. Zhu and Y. Zhang et al. "An energy- and cost-aware computation offloading method for workflow applications in mobile edge computing",in EURASIP Journal on Wireless Communications and Networking, (2019) 2019:207, doi:10.1186/s13638-019-1526-x.

[11] Vilaplana. J, Solsona. F et al. (2014). A queuing theory model for cloud computing. The Journal of Supercomputing, 69(1), 492-507.

[12] K. Deb, A. Pratap and S. Agarwal et al., "A fast and elitist multiobjective genetic algorithm: NSGA-II," in IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182-197, April 2002, doi: 10.1109/4235.996017.

[13] B. Lin, Y. Huang and J. Zhang et al., "Cost-Driven Off-Loading for DNN-Based Applications Over Cloud, Edge, and End Devices," in IEEE Transactions on Industrial Informatics, vol. 16, no. 8, pp. 5456-5466, Aug. 2020, doi: 10.1109/TII.2019.2961237.

[14] M. Xue, H. Wu and R. Li et al., "EosDNN: An Efficient Offloading Scheme for DNN Inference Acceleration in Local-Edge-Cloud Collaborative Environments," in IEEE Transactions on Green Communications and Networking, vol. 6, no. 1, pp. 248-264, March 2022, doi: 10.1109/TGCN.2021.3111731.

Authorized licensed use limited to: Beijing Information Science & Tech Univ. Downloaded on February 11,2023 at 07:22:55 UTC from IEEE Xplore. Restrictions apply.