Peer-to-Peer Networking and Applications (2021) 14:1190–1206
https://doi.org/10.1007/s12083-021-01081-x

Computing tasks assignment optimization among edge computing servers via SDN

Chao Bu [1,2] · Jinsong Wang [2]

Received: 27 September 2020 / Accepted: 19 January 2021 / Published online: 1 February 2021

Abstract
As an extension of cloud computing, the edge computing has become an important pattern to deal with novel service scenarios of Internet of Everything (IoE) under 5G, especially for the delay sensitive computing tasks generated from edge equipment. The edge computing provides the key support to meet the characteristics of delay sensitivity by deploying servers near network edges. However, a great many uneven distributed computing tasks in different network edges usually lead to task processing delay bottleneck for single Edge Computing Server (ECS). Tasks assignment is mainly based on the local ECS status without the global network view considered, which also easily leads to unbalanced task loads among multiple ECSs. In this paper, the novel networking idea of Software Defined Network (SDN) is introduced into the edge computing pattern. The logically highly centralized control plane consists of multiple physically distributed ECSs, so as to collaboratively assign computing tasks in a global view. In order to optimize the task assignment and minimize the task processing delay, three schemes are proposed in this paper. The scheme of assessing the ECS's task computing features is firstly proposed, then the scheme of predicting the ECS's future unit task processing time is presented. Thus, different types of computing tasks can be assigned to appropriate ECSs that are better at dealing with them with processing delay minimized. Furthermore, the scheme of optimizing the delay of task processing time estimation is devised, so as to further improve task assignment efficiency. Experimental results show that the proposed mechanism is able to optimize the task assignment and minimize the task processing delay more efficiently than the state of the art. Specifically, our mechanism is capable of improving the average unit task processing delay and the ECS load balancing degree by about 14% and 23% respectively, compared with corresponding work.

Keywords Delay sensitivity · Edge computing · SDN · Task assignment efficiency

1 Introduction

With the rapid development of network communication technology, people's demands for higher performance services are becoming more and more urgent. The 5th Generation mobile system (5G) is gradually integrating into people's daily life. It is committed to satisfy the features such as ultra-high traffic density, ultra-high connection density, and ultra-high mobility [1]. Thus, the requirements of the novel service scenarios have changed significantly, which represent the characteristics of delay sensitively including reliable control, fast moving, real-time analysis and location awareness [2, 3]. Some research (i.e., Section 2) have been done on optimizing scheduling computing tasks among multiple servers in the cloud center, which significantly improve task processing time. However, the cloud computing servers usually locate far away from network edges. Considering the fact that the number of computing requests from mobile equipment is increasing rapidly with the developing of service scenarios under 5G, the conventional cloud computing may lead to much higher transmission delay and serious congestion. It is obvious that the classical cloud computing mode of centrally dealing with large-scale computing tasks no longer adapts to this situation. As an extension of cloud computing, the edge computing deploys the capabilities such as computing, networking and storing near to the data resource at the network edges [4]. Its distributed Edge Computing Servers (ECSs) are much closer to users so as to provide real-time computing services nearby. In this

✉ Chao Bu
bc_0722@163.com

[1] Key Laboratory of Computer Vision and System of Ministry of Education, School of Computer Science and Engineering, Tianjin University of Technology, Tianjin 300384, China

[2] Tianjin Key Laboratory of Intelligence Computing and Novel Software Technology, School of Computer Science and Engineering, Tianjin University of Technology, Tianjin 300384, China

paper, we leverage the key support of edge computing pattern to meet the special demands of the novel business scenarios under 5G.

The 5G can support high-speed data-transmission, which brings edge computing big advantages to significantly decrease the delivery delay of service data. However, facing the era of "Internet of Everything (IoE)" coming with 5G, there are some new challenges for the edge computing to deal with [5]. A number of research (i.e., Section 2) have proposed dealing with edge computing tasks by distributed ECSs, so as to improve task processing efficiency in the 5G environment. However, according to the edge computing pattern, distributed deployed ECSs locate in different Edge Network Domains (ENDs). Each ECS mainly carries out computing tasks based on its local network status. When a single ECS receives a large number of uneven distributed computing requests from its END, which frequently happens in the current network service scenarios, the delay bottleneck of processing computing tasks for the single ECS is easily caused [6]. While, the ECSs in other ENDs maybe idle at the moment. In this paper, we propose to collaboratively assign computing tasks that are non-uniformly distributed in different ENDs among multiple ECSs in a global view. In this approach, assigning tasks to appropriate ECSs with both the ECS working status and the network status considered, not only further optimizes the task processing efficiency but also balances ECSs' working loads.

As a novel networking idea, Software Defined Network (SDN) [7] has the features of conciseness, agility and openness that are naturally fit for the demand of 5G service scenarios [8, 9]. By introducing the decoupling control and forwarding of SDN into edge computing modelling, the logically highly centralized control plane can be constituted by multiple ECSs that are physically distributed [10]. Thus, the task processing pattern of edge computing, which manages services and resource based on each ECS's local information [11], can be well improved. In this paper, we introduce the idea of SDN into globally task assignment. By leveraging the advantages of SDN, some research (i.e., Section 2) have been done on optimizing edge computing tasks assignment among multiple ECSs in a global view. Thus, the ECSs' working loads are well balanced while improving the task processing efficiency. In fact, however, the working scenes and computing modes of ECSs in different ENDs usually have regional features. In other words, an ECS in a certain END may frequently deal with some types of computing tasks, which enables the ECS be better at processing these corresponding tasks with less time than other ECSs. The task processing efficiency can be further optimized by assigning tasks to the suitable ECSs. In this paper, we devise a scheme to assess each ECS's task processing features (i.e., the frequency of dealing with certain types of computing tasks) by mining and analyzing the historical records of its past processed tasks. In this approach, the candidate ECSs that are suitable to deal with one type of computing tasks can be preliminarily obtain, which enables tasks to be assigned to the more suitable ECSs with their processing delay further reduced.

Among multiple candidate ECSs, how to select the current most suitable one to deal with the task that is waiting to be assigned is still a challenge. Although an ECS is assessed to be suitable for one type of tasks, it may still bring serious delay problem to the new assigned task. For example, the ECS's current available computing capacity cannot afford a new task, thus this task will not be processed immediately after being assigned until sufficient computing capacity is released. In addition, a new assigned task may have a negative impact on an ECS's current computing efficiency. For example, the increased processing time brought to the ECS by the new assigned task may reduce the ECS's future unit task processing efficiency. In order to solve the challenges described above, we then propose a scheme to assess the ECS's unit task processing time before a task being assigned to it. In this scheme, the Future Unit task processing Time (FUT) of the ECS is defined. It is used as the evaluation criteria to select the ECS. By taking the past processing time of a type of computing tasks in an ECS into account, the method to predict the candidate ECS's FUT is presented. In this approach, the ECS with the minimum FUT is selected as the most suitable one for corresponding tasks in the current network status.

However, a task may last several estimation rounds without being assigned. Before each new assignment round, its future processing time needs to be estimated, which obviously brings much extra re-estimation time overhead. In this paper, we further devise an approach to solve the re-estimation time overhead problem for the task that participates multiple assignment rounds. Thus, re-estimation operations are greatly reduced before each assignment round, which further improves the whole task assignment efficiency among multiple ECSs.

In this paper, according to the devised schemes mentioned above, the mechanism of collaboratively Assigning Computing Tasks among multiple ECSs (ACTE) by the advantages of SDN is proposed. The major contributions are as follows:

- The system framework of the ACTE mechanism is presented, so as to provide the detail working principles and flows of the whole system.
- The scheme of assessing the ECS's computing task processing features is devised, so as to obtain the candidate ECSs for different types of tasks waiting to be assigned.
- The method of predicting the ECS's FUT is presented, so as to select the most suitable ECS with the minimum FUT for corresponding tasks.
- The approach of optimizing estimation time overhead is designed, so as to further improve the tasks assignment efficiency.

The remainder of this paper is structured as follows. Section 2 reviews the related work. Section 3 presents the system framework of the proposed ACTE. Section 4 devises the schemes of task assignment among multiple ECSs. Section 5 shows simulation results. Finally, section 6 concludes this paper.

2 Related work

Table 1 presents an overview of the related work. The detailed description on the above research and the difference between them and our work are shown in the following paragraphs.

The research of computing task scheduling problem have firstly been studied and done in the cloud computing center, so as to improve the task processing time. In [12], an optimization model for task scheduling is presented, so as to minimize task processing time and energy consumption for cloud computing. It formulates an integer programming optimization problem for task energy consumption, then devise a task scheduling algorithm to achieve small task processing time with energy consumption minimized. In [13], a new model of resource allocation to optimize task scheduling in cloud computing is proposed. By integrating the multi-objective optimization and the particle swarm optimization in the model, the tasks are scheduled to the virtual machines with waiting time minimized and system throughput maximized. In [14], an improved genetic algorithm for static task scheduling in the cloud environment is proposed. It uses the advantages of evolutionary genetic algorithm along with a heuristic-based Heterogeneous Earliest Finish Time (HEFT) search to assign subtasks to processors. In [15], based on the popular min-max normalization technique, a task scheduling algorithm is presented. It keeps a trade-off between the two parameters of makespan and resource utilization, so as to achieve multi-objective optimization by task scheduling. In [16], an alternative method for cloud task scheduling is proposed, so as to minimize the makspan that required to schedule a number of tasks on different virtual machines. It is based on the improvement of the moth search algorithm by using the differential evolution. In the above work, by scheduling tasks among multiple servers in the cloud computing center, the task makespan and resource utilization is greatly improved. However, the tasks are usually processed centrally, and the task processing center always locates far away from network edges where requests generate. This may lead to serious problems such as delay and congestion, especially when lots of tasks that are delay-sensitive arrive in a short time slot in the era of IoE.

Some research then have presented approaches of providing computing services by leveraging edge computing pattern, and devised schemes to schedule tasks to improve the task processing efficiency in the 5G environment. In [17], the task scheduling problem is formulated as mix-integer programming to minimize computation, communication and violation costs. Two efficient heuristic algorithms are proposed and evaluated for task scheduling, they significantly provide low makespan and decrease the total computation cost. In [18], by studying a novel D2D-enabled multi-helper mobile edge computing system, a time division multiple access transmission scheme is proposed. It minimizes the computation delay by optimizing the local user's task assignment jointly with the time and rate considered. In [19], the problems of long-term task assignment and resource coordination are studied. It designs an online algorithm to adaptively decide the task assignment, and an iterative wireless resource allocation algorithm to improve the joint wireless transmit power and sub-channel resource allocation. In [20], by studying the task assignment algorithm in data shared mobile edge computing systems, it proposes three algorithms to deal with holistic tasks and divisible tasks. It devises an approximation algorithm based on linear programming for holistic tasks, and two approximation algorithms to solve divisible tasks. In [21], a framework for task assignment is proposed, so as to offload the computationally intensive. It allows the offloaded tasks to be served at appropriate cloudlets, so that the latency in the network can be reduced and meanwhile the quality of service experienced can be improved. In [22], a distributed task unloading strategy to low load base station group under mobile edge computing environment is proposed. It models the communication resource, computing resource and task queue of low load base station group, and solves the problem of distributed task unloading by using the potential game model. These research supports task assignment among multiple edge servers by the advantages of edge computing pattern, so as to avoid computing latency and congestion problems. However, the conventional edge computing pattern assigns and deals with tasks mainly based on each distributed server's local resource and network status without a global view. Thus, some edge computing servers may take on heavy workload in some edge network domains due to lots of computing requests, while other servers are relatively idle with part of their computing resource wasted. Therefore, how to collaboratively assign tasks in a global view with further improving task processing efficiency and balancing server working load is still a challenge.

By introducing the advantages of SDN, some research have been done on computing tasks assignment among multiple edge computing servers in a global view with the dynamic network conditions considered. In [23], by leveraging the global view of the network at the SDN controller, the dynamic network load is taken into account when offloading tasks. It formulates the multi-hop task offloading problem as an integer linear program, then presents a greedy-heuristic-based approach to solve the problem, so as to efficiently reduce the average delay and energy consumption. In [24], by taking the load balancing of the computation resources at the edge servers into account, the task processing delay minimization problem is investigated. It proposes an SDN-based task

Table 1 Overview of the related works

Subject	Related work	Main method used	Key contributions
Cloud computing	Ref. [12]	Greedy Optimization	Proposing the most-efficient-server first greedy task scheduling algorithm
	Ref. [13]	Particle Swarm Optimization	Proposing a novel algorithm on multi-object PSO task-scheduling
	Ref. [14]	Genetic Algorithm	Proposing an improved genetic algorithm for static task scheduling
	Ref. [15]	Min-Max Normalization	Proposing the normalized multi-object min-min max-min scheduling for tasks
	Ref. [16]	Moth Search Algorithm Differential Evolution Algorithm	Proposing an alternative method for cloud task scheduling on different virtual machines
Edge computing	Ref. [17]	Mixed Integer Linear Programming	Proposing two efficient heuristic algorithms IoT task scheduling in volunteer computing systems
	Ref. [18]	Convex Relaxation Greedy Algorithm	Constructing a suboptimal task assignment solution approach, and developing a heuristic scheme based on the greedy task assignment
	Ref. [19]	Stochastic Optimization Lyapunov Optimization	Proposing a joint task assignment, wireless coordination and optimization, and computation resource allocation strategy
	Ref. [20]	Linear Programming	Proposing three algorithms on efficiently assigning holistic tasks and divisible tasks
	Ref. [21]	Logic-based Benders Decomposition	Defining and formulating the dynamic task offloading and scheduling problem, and proposing an efficient approach to optimally solve the problem
	Ref. [22]	Potential Game Model	Proposing a distributed task offload strategy to the low-load base station group, and Solving the problem off offloading distributed tasks by introducing the game theory
Edge computing and SDN	Ref. [23]	Greedy Algorithm	Proposing a dynamic task offloading scheme in software-defined fog for IoT applications
	Ref. [24]	Game Theory	Proposing an SDN-based task offloading architecture, and devising two task offloading schemes and a computing offloading scheme
	Ref. [25]	Reinforcement Learning	Introducing a novel SDN-based computation offloading framework, and Proposing two approaches for delay minimization problem
	Ref. [26]	Mixed Integer Linear Programming	Proposing a framework for a task deadline-awareness balanced distribution of tasks across the Cloudlets by leveraging SDN
	Ref. [27]	Poisson Process Admission Control Policy	Proposing an optimal task assignment scheme for a multi-cloudlet environment, and presenting an admission control approach for task assignment management

offloading architecture and three schemes to reduce the tasks' processing delay and balance the servers' load. In [25], a novel SDN based framework for computation offloading in mobile edge computing wireless networks is introduced. It proposes the Q-learning and cooperative Q-learning approaches to solve the task offloading and resource allocation problems. In [26], based on SDN technologies, a framework for task deadline-awareness balanced distribution of tasks is proposed. It optimally manages the resources and balances an equitable load across a network of Cloudlets. In [27], the task assignment problem in a multi-cloudlet network connected via a wireless SDN network is studied, and a novel task assignment scheme which makes task assignment decisions is proposed. It takes significant parameters that affect the latency in processing tasks into account, and aims at reducing the network latency in processing the offloaded tasks. By leveraging the advantages of controlling and programming the entire network as a unified network of SDN, the tasks from the network edges can be assigned to multiple edge servers in a global view with network load considered. Thus, the task processing efficiency is improved with edge servers' working load balanced. In this paper, we further take the different task processing features of servers into account when collaboratively assigning tasks among multiple servers. In addition, the historical records of each server are fully utilized to predict the FUT of each server and estimate the task processing delay, so as to optimize the task assignment and improve the task processing efficiency.

3 System framework

Based on the decoupling control logic and data forwarding of SDN, the ACTE mechanism is proposed. It is used to optimize large-scale computing tasks assignment with each ECS's computing features and real-time working load considered.

However, it is a complex problem to optimally assign multiple tasks to multiple servers, for example, there may be $n!/m!$ possibilities to assign n tasks to m servers. In this paper, we consider the problem as minimizing computing task processing delay under related constrains.

Involving the computing tasks collaboratively assignment among multiple ENDs, the goal is to optimize the unit task processing time under the limited processing capacity of each ECS. In this paper, the ACTE mechanism is modelled by the proposed empirical rules-based improved greedy strategy, which supports the establishment of the system framework and corresponding schemes. In detail, firstly, the historical computing task processing records are leveraged offline to assess each ECS's task processing features based on the regression analysis method. Then, according to the improved greedy strategy based on unit task delay assessment, the tasks waiting to be assigned are selected by the most suitable ECS at each assignment round. Finally, estimation time is further optimized by analyzing the time increment of re-estimating. The system framework of the proposed ACTE mechanism is shown in Fig. 1.

3.1 System detailed description

The logically highly centralized control plane consists of multiple ECSs that are physically distributed in different edge network domains. Each ECS serves as the controller of its located network domain. Computing requests from edge equipment are received by the data plane then sent to the control plane. The data plane contains a large number of interconnected switching devices located much closer to users. When receiving computing requests, the control plane collaboratively assigns the corresponding computing tasks to appropriate ECSs. The task assignment considers each ECS's current working load and computing features, so as to minimize the task processing time. For example, an ECS located in the edge network domain where a lot of financial institutions exist, is much better at dealing with financial computing requests. Because the ECS has plenty of such computing instances of processing the similar computing tasks, which enable it to complete these tasks with much short time than other ECSs.

According to the proposed system framework in Fig. 1, we devise three schemes to support computing task assignment optimization, so as to minimize task processing delay. By mining and analyzing each ECS's historical computing task processing records, the Computing Feature Assessment module (CFA) assesses the frequencies of an ECS dealing with different types of computing tasks. The candidate ECSs to be assigned some certain types of tasks can be preliminarily obtained. The Unit Task Delay Assessment module (UTDA) takes charge of predicting the processing time overhead of the tasks waiting to be assigned in each possible ECS. The most suitable ECS with the lowest FUT is selected. In order to further optimize the estimation time overhead in each task assignment round, the Estimation Time Optimization module (ETO) is in charge of improving the tasks assignment efficiency by reducing unnecessary estimation delay.

3.2 Intersystem interaction

By the cooperation of the three modules (i.e., CFA, UTDA, ETO) in the control plane, the computing tasks assignment among multiple ECSs is achieved. Before each new time period, the CFA have already offline assessed the ECS's computing features for different types of tasks. When the computing requests are received, according to the types of the corresponding tasks, the UTDA selected the most suitable ECS from the candidate ECSs that are obtained by the CFA. Then, the tasks are assigned to their appropriate ECSs to be dealt with. For the tasks that are not be assigned successfully in an assignment round, the ETO reduces the estimation time overhead of these re-assigned tasks, so as to further optimize the delay. The detailed working interactions between these modules of the system is shown in Fig. 2.

4 Tasks collaboratively assignment

In this section, the mathematical formulation is presented for the proposed ACTE mechanism. Table 2 shows the main notations used in this section. Specifically, three major schemes are devised as follows. They are the ECS's computing features assessment, the unit task processing time minimum, and the estimation time overhead optimization. Meanwhile, the method of task processing time estimation is also designed. By the ECS's computing features assessment, the candidate ECSs that are appropriate to deal with different types of computing tasks can be obtained. By the unit task processing time minimum, the most suitable ECS for a certain task in the current assignment round can be selected from multiple candidate ECSs. By the estimation time overhead optimization, the task assignment efficiency can be further improved with the total delay reduced.

4.1 ECS's computing features assessment

In practice, the computing task demands from a single user may be incidental to an ECS, however, the demands from a large number of user groups usually have certain regularity to an ECS. According to the task processing records over several time periods, we try to learn the regularity by combining the frequencies of tasks processed in an ECS based on multivariate linear regression, so as to predict the ECS's task processing features in the following time periods. In this section, we propose assessing the task processing features of an ECS by mining and analyzing the ECS's historical records of tasks

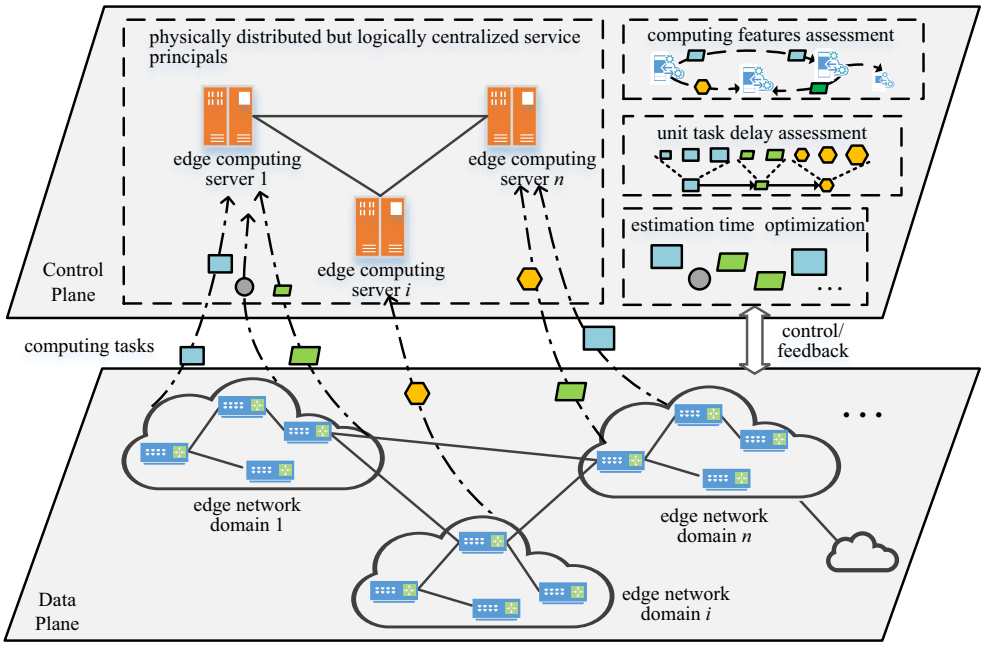


Fig. 1 CNF deployment situations

processed by it in the past t time periods. For different types of computing tasks, we firstly determine the preliminary appropriate ECSs that are better at dealing with which types of tasks. In this approach, each ECS's computing features (i.e., the frequency of dealing with certain types of tasks) in the $(t+1)th$ time period can be assessed.

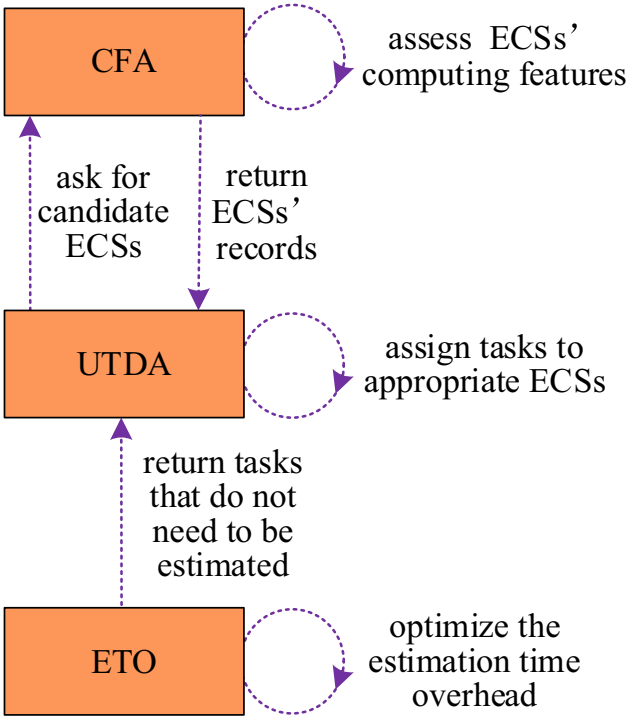


Fig. 2 The working interactions between modules

Assume that SCT_k denotes the k Service type of Computing Task, and $NTP_k^i(t)$ is defined as the Number of this type of Tasks Processed in the ECS i (i.e., ECS_i) in the $(t)th$ time period. The Actual Processed Frequency of SCT_k in ECS_i in the $(t)th$ time period is defined as follows:

$$APF_k^i(t) = \frac{NTP_k^i(t)}{\sum_{SCT_l \in SCT^i(t)} NTP_l^i(t)} \tag{1}$$

Here, $SCT^i(t)$ is the Set of Computing Tasks that have been processed in ECS_i in the $(t)th$ time period.

According to the actual processed frequencies of SCT_k in ECS_i in the last t time periods, the appropriate degree that SCT_k is assigned to ECS_i in the $(t+1)th$ time period (i.e., the current time period) can be assessed. $APF_k^i(t+1)$ denotes the appropriate degree that SCT_k can be assigned to SCT_k in the $(t+1)th$ time period, shown as follows:

$$APF_k^i(t+1) = \sum_{z=1}^{t} \gamma_z \cdot APF_k^i(z) + \beta \tag{2}$$

Here, γ_z is the regression coefficient of $APF_k^i(z)$, β is a constant. We use the typical least squares method to learn the values of $\gamma_1, \gamma_2, \ldots, \gamma_t$, and β. We define that $X = [\gamma_1, \gamma_2, \ldots, \gamma_t, \beta]$, and the approach is devised to obtain the values of its elements by mining and analyzing the historical processed frequencies of SCT_k in ECS_i shown as follows.

The values of $APF_k^i(u)$, $APF_k^i(u+1)$, $APF_k^i(u+2)$,…, $APF_k^i(u+2t)$ (i.e., the actual processed frequencies of SCT_k in ECS_i from the (u)th time period to the $(u+2t)$th time

Table 2 Summary of the main notations

Symbol	Description
SCT_k	the k Service type of Computing Task
ECS_i	the Edge Computing Server i
$NTP_k^i(t)$	the Number of Tasks (i.e., SCT_k) Processed in ECS_i in the $(t)th$ time period
$SCT^i(t)$	the Set of Computing Tasks processed in ECS_i in the $(t)th$ time period
$APF_k^i(t)$	the Actual Processed Frequency of SCT_k in ECS_i in the $(t)th$ time
ST_i^c	the Set of Tasks assigned to ECS_i but not yet completely processed
TCC_i	the Total Computing Capacity of ECS_i
SCT^c	the Set of Computing Tasks currently waiting to be assigned
TPD_k	the Task Processing Deadline of SCT_k
RCC_k	the Required Computing Capacity of SCT_k
WD_i^c	the Waiting Duration
$G^{ECS_i}(SCT_k)$	the function to estimate the processing time of SCT_k in ECS_i
$\lvert ST_i^c \cup \{SCT_k\}\rvert$	the number of the elements in $ST_i^c \cup \{SCT_k\}$
$\overline{PT_{i,t}(SCT_k)}$	the average Processing Time of SCT_k in ECS_i in the past t time periods
$APT_i^{m,p}(SCT_k)$	the $(p)th$ Abnormal Processing Time of SCT_k in ECS_i in the m time period
$\overline{PT_i^m(SCT_k)}$	the average normal Processing Time of SCT_k in ECS_i in the m time period
$\overline{\Delta PT_i^m(SCT_k)}$	the average difference between $APT_i^{m,p}(SCT_k)$ and $\overline{PT_i^m(SCT_k)}$

period) can be obtained from the historical records of ECS_i. We define a matrix G shown as follows:

$$G = \begin{bmatrix} APF_k^i(u) & APF_k^i(u+1) & \dots & APF_k^i(u+t) & 1 \\ APF_k^i(u+1) & APF_k^i(u+2) & \dots & APF_k^i(u+t+1) & 1 \\ \dots & \dots & \dots & \dots & \dots \\ APF_k^i(u+t) & APF_k^i(u+t+1) & \dots & APF_k^i(u+2t) & 1 \end{bmatrix} \tag{3}$$

Here, we use G_u, G_{u+1},…, G_{u+t} to replace the row elements of G:

$$\begin{cases} G_u = \left[APF_k^i(u) \quad APF_k^i(u+1) \quad \dots \quad APF_k^i(u+t) \quad 1\right] \\ G_{u+1} = \left[APF_k^i(u+1) \quad APF_k^i(u+2) \quad \dots \quad APF_k^i(u+t+1) \quad 1\right] \\ \dots \\ G_{u+t} = \left[APF_k^i(u+t) \quad APF_k^i(u+t+1) \quad \dots \quad APF_k^i(u+2t) \quad 1\right] \end{cases} \tag{4}$$

The Eq. (3) can be converted as follows:

$$G = [G_u, G_{u+1}, \dots, G_{u+t}]^T \tag{5}$$

Let $Y = \left[APF_k^i\ (u+t+1), APF_k^i(u+t+2), \dots, APF_k^i(u+2t+1)\right]$ be the actual processed frequencies of SCT_k in ECS_i in the $(u+t+1)$th, $(u+t+2)$th,…, and $(u+2t+1)$th time periods. According to the Eq. (2), let $[G_u \cdot X^{\mathrm{T}}, G_{u+1} \cdot X^{\mathrm{T}}, \dots, G_{u+t} \cdot X^{\mathrm{T}}]$ be the assessed processed frequencies of SCT_k in ECS_i. We define E_X as follows:

$$E_X = \left(Y - G \cdot X^{\mathrm{T}}\right)^{\mathrm{T}} \cdot \left(Y - G \cdot X^{\mathrm{T}}\right) \tag{6}$$

When E_X achieves the minimum, the X can be obtained shown as follows:

$$\frac{\partial E_X}{\partial X} = \frac{\partial \left(Y - G \cdot X^{\mathrm{T}}\right)^{\mathrm{T}} \cdot \left(Y - G \cdot X^{\mathrm{T}}\right)}{\partial X} = 0 \tag{7}$$

In this approach, the appropriate degree that SCT_k should be assigned to ECS_i in the $(t+1)th$ time period has been assessed. Here, let TS be the Threshold to judge that if an ECS is Suitable to dealing with a type of tasks. The set of preliminary appropriate ECSs (i.e., candidate ECSs) whose computing features are suitable for SCT_k in the $(t+1)th$ time period is defined as $ECS^k(t+1)$. Each ECS_i ($ECS_i \in ECS^k(t+1)$) must satisfy the condition $APF_i^k(t+1) \geq TS$. Because the above assessment happens at the beginning of a new time period, it does not bring extra delay in the real time.

4.2 Unit task processing time minimum

We assume that the task assignment happens in the $(t+1)th$ time period (i.e., the current time period). In a practical scenario, it is extremely rare to receive just a single computing task at a moment from multiple edge network domains. The usual scenario is that a large number of computing requests are received by the control plane in a short time slot. The computing tasks that are delay sensitive should be assigned appropriately as soon as possible. According to the above scheme, the set of ECSs that are good at dealing with different types of tasks have been obtained. How to assign the task to the most suitable ECS from multiple candidate ECSs, we devise a

scheme to solve the problem by considering minimizing the ECS's unit task processing time.

Let ST_i^c be the Set of Tasks assigned to ECS_i but not yet completely processed, and TCC_i be the Total Computing Capacity of ECS_i. Let SCT^c be the Set of Computing Tasks that are waiting to be assigned currently. Let TPD_k be the Task Processing Deadline of SCT_k (i.e., SCT_k need to be completed before TPD_k), and RCC_k be the Required Computing Capacity of SCT_k. The aim of selecting the most suitable ECS is to minimize the ECS's future unit task processing time if the task is assigned to it. Meanwhile, the assigned task should be completed as soon as possible before its deadline. Therefore, for ECS_i, the task SCT_k that tends to be assigned to ECS_i needs to satisfy the conditions shown as follows:

$$Minimize\left(\frac{G^{ECS_i}(SCT_k)+\sum\limits_{SCT_l\in ST_i^c} G^{ECS_i}(SCT_l)}{\left|ST_i^c\cup\{SCT_k\}\right|}\right) \tag{8}$$

s.t.

$$\forall ECS_i\in ECS^k(t+1):\ \ APF_i^k(t+1)\geq TS \tag{9}$$

$$TPD_k\geq WD_i^c+G^{ECS_i}(SCT_k) \tag{10}$$

$$RCC_k\leq TCC_i-\sum_{SCT_l\in SCT^c} RCC_l \tag{11}$$

Here, G^{ECS_i} is the estimation function of the task processing time in ECS_i, and can be assessed according to the task's historical processed time records in ECS_i, which is designed in the next part. $\left|ST_i^c\cup\{SCT_k\}\right|$ is the number of the elements in the set of $ST_i^c\cup\{SCT_k\}$, the set contains all already assigned tasks in ECS_i and SCT_k that is supposed assigned to ECS_i. WD_i^c is the Waiting Duration before a new assigned task start being processed in ECS_i currently. It can be calculated by the sum of two estimated time overhead, shown as $WD_i^c=(1-\delta)G^{ECS_i}(SCT_c)+\sum G^{ECS_i}(SCT_l)$, here, δ is the percentage of processing progress of the currently running task ECS_i, and SCT_l ($SCT_l\in\left\{SCT_c|ST_i^c\right\}$) is the task earlier assigned to ECS_i but not yet processed. Eqs. (9), (10), and (11) are the constraints on Eq. (8). If ECS_i is selected as the most suitable ECS to deal with SCT_k, ECS_i need to belong to $ECS^k(t+1)$ with $APF_i^k(t+1)\geq TS$ satisfied (i.e., Eq. (9)); the estimated time of completing SCT_k in ECS_i should less than or equal to the deadline of SCT_k (i.e., Eq. (10)); the current available capacity of SCT_k is greater than or equal to the required computing capacity of SCT_k (i.e., Eq. (11)).

According to Eq. (8), if a task (i.e., SCT_k) is assessed to minimize just one ECS's FUT, the task should be assigned to this ECS. If multiple candidate ECSs satisfy Eq. (8), the task should be assigned to the ECS that can complete the task earliest, shown as follows:

$$Minimize\left(TPD_k-\left(WD_i^c+G^{ECS_i}(SCT_k)\right)\right) \tag{12}$$

However, the situation may also exist that SCT_k cannot be assigned to any ECS in $ECS^k(t+1)$ in the current assignment round due to failing to satisfy the conditions such as Eqs. (9), (10), or (11). By considering load balancing among multiple ECSs in the control plane, SCT_k can be assigned to other ECS with the lowest working load currently, and the ECS need to satisfy both Eqs. (10) and (11). If none of the above is possible, SCT_k participates in the next task assignment round.

4.3 Task processing time estimation

The processing time of a type of tasks in an ECS usually remains stable due to the ECS's computing features. According to a task's past processing time in an ECS, the task's usual processing time in this ECS is not hard to estimate. However, it is impossible for an ECS to deal with only one task at a time, other already assigned tasks should also be considered at the same time. Because the actual processing time of the being processed tasks that have already been assigned in the ECS may lead to time fluctuation to the new assigned task.

In this paper, we consider two factors to define the estimation function of a task's processing time in a candidate ECS. One factor is the usual processing time of the task itself in the last t time periods, the other is the possibility of time fluctuation caused by the processing anomaly. In this approach, the estimation function of the processing time of SCT_k in ECS_i is defined as follows:

$$G^{ECS_i}(SCT_k)=\overline{PT_{i,t}(SCT_k)}+C\cdot\sum_{m=1}^{t}\overline{\Delta PT_i^m(SCT_k)} \tag{13}$$

Here, $\overline{PT_{i,t}(SCT_k)}$ is the average Processing Time of SCT_k in ECS_i in the past t time periods. C is the influence coefficient of the possible processing anomaly. $\overline{\Delta PT_i^m(SCT_k)}$ is the average difference between the average processing time and the abnormal processing time of SCT_k in ECS_i in the m time period. And $\overline{\Delta PT_i^m(SCT_k)}$ is defined as follows:

$$\overline{\Delta PT_i^m(SCT_k)}=\frac{q}{n}\cdot\sum_{p=1}^{q}\left(APT_i^{m,p}(SCT_k)-\overline{PT_i^m(SCT_k)}\right) \tag{14}$$

Here, n is the total processed number of SCT_k in ECS_i in the m time period, and q is its abnormal processed number of SCT_k in ECS_i in the m time period. $APT_i^{m,p}(SCT_k)$ is its $(p)th$ Abnormal Processing Time, and $\overline{PT_i^m(SCT_k)}$ is its average normal Processing Time.

4.4 Estimation time overhead optimization

In order to meet the demands of the delay sensitive computing tasks, the schemes are proposed above to assign different types of tasks to appropriate ECSs that are better at dealing with them. So the corresponding tasks' processing time will be well improved. On the other hand, the schemes not only optimize each ECS's following task processing efficiency, but also improve the load balancing among multiple ECSs. Therefore, the processing time estimation of a task in an ECS before the task having been assigned plays a critical role.

The situation mentioned above that some tasks may not have been successfully assigned to one ECS in the previous assignment round. However, their processing time in ECSs have already been estimated. It obviously leads to unnecessary time overhead of re-estimating these tasks' processing time in a new assignment round. We design an approach to reduce the estimation time overhead and further improve the assignment efficiency, shown as follows:

$$\exists (1 \le r < c) : Minimize^{r}_{SCT_k} \left(\frac{G^{ECS_i}(SCT_k) + \sum_{SCT_v \in SCT^r} G^{ECS_i}(SCT_v)}{|ST^r_i \cup \{SCT_k\}|} - \frac{\sum_{SCT_v \in SCT^r} G^{ECS_i}(SCT_v)}{|ST^r_i|} \right) > Minimize^{c}_{SCT_e \in (SCT^c / SCT_k)} \left(\frac{G^{ECS_i}(SCT_e) + \sum_{SCT_l \in SCT^c} G^{ECS_i}(SCT_l)}{|ST^c_i \cup \{SCT_e\}|} - \frac{\sum_{SCT_l \in SCT^c} G^{ECS_i}(SCT_l)}{|ST^c_i|} \right) \tag{15}$$

Here, the superscript r denotes the previous assignment round, and the superscript c denotes the current assignment round. If the future unit task processing time increment brought to ECS_i by SCT_k in a previous assignment round, is higher than the future unit task processing time increment brought to ECS_i by one of other waiting to be assigned tasks in the current assignment round, SCT_k will not be estimated for ECS_i in the current assignment round.

4.5 Computational complexity analysis

According to the working interactions between modules in the system shown in Fig. 2. The real-time computation of the proposed ACTE mechanism is making task assignment decision, which mainly happens in the UTDA module. For example, the ECS's task computing features assessment have already been calculated offline (i.e., the part of 4.1) before each new time period. Thus, the ECS's suitability degrees of dealing with different types of tasks have already been obtained before making task assignment decisions in the current time period. In addition, before each task assignment round, the estimated processing time of tasks in different ECSs have also already been calculated offline (i.e., the part of 4.3) according to ECSs' historical task processing records. Therefore, we mainly analyze the computational complexity of real-timely assigning tasks among multiple ECSs.

For the tasks waiting to be selected in each assignment round, the ECS's selection strategy for them is based on the greedy optimization. For each ECS, the tasks waiting to be assigned are sorted according to Eq. (8) and selected with the ECS's available computing capacity (i.e., Eq. (11)) and the task's processing deadline (i.e., Eq. (10)) satisfied. Assume that the number of the tasks (i.e., the elements in SCT^c) waiting to be assigned is n, it will take $O(n \lg n)$ at most. In addition, considering the problem of tasks being repeatedly selected by multiple ECSs, the task should be assigned to the one with the earliest completed time according to Eq. (12). In practice, the number of ECSs is much smaller than the tasks waiting to be assigned in each task assigned round. Assume that the number of ECSs is n at most and the number of tasks selected by each ECS is n at most, thus the time taken by Eq. (12) is defined as $O(f(n))$. It will take another $O(nf(n))$ at most. According to the analysis above, the overall computation complexity of real-timely task assignment is $O(n \lg n + nf(n))$ at most.

5 Performance evaluation

In this section, we evaluate the performance of the proposed ACTE mechanism and make an analysis to the computing tasks collaboratively assignment. The schemes are implemented in Python and all experiments are performed on a computer with one Intel(R) Core(TM) i7–6700 CPU @ 3.40 GHz and 16 GB of RAM. The network topologies used in the simulation are Geant and Interroute obtained from the Internet Topology Zoo [28], shown in Fig. 3. Specifically, Geant is a network topology with 41 nodes and 65 links, and Interroute is a network topology with 110 nodes and 148 links. We divide Geant and Interroute into 4 and 8 edge network domains respectively. Each domain contains about 10 switching devices and one ECS that controls these switching devices.

The simulation parameters refer to the existing works [29, 30]. A type of task is set to take up 2 to 5 units of computing capacity, and its normal computing time is randomly set to be 10 to 20 units of time while just is half of the above time (i.e., 5 to 10 units of time) in the ECS that is better at dealing this type of tasks. An ECS is set to have 4000 units of computing capacity. We also assume that computing tasks are divided into 10 types, each ECS is only better at dealing with 2 or 3 types of them. We compare the processed ACTE with two related recent schemes that are Distributed Task Offloading Strategy (DTOS) and Latency

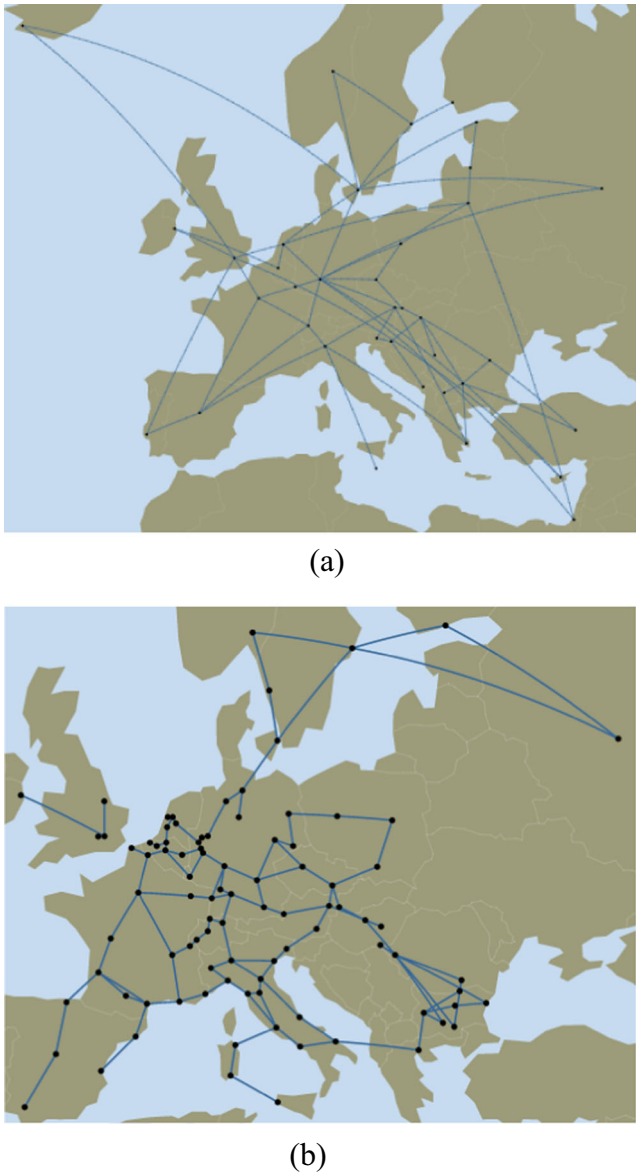


Fig. 3 The network topologies used in the simulation. **a** Geant Topology, **b** Interroute Topology

Aware Task Assignment (LATA). The DTOS is the approach on distributed task unloading to low load base station group under mobile edge computing environment, which is mainly simulated according to the related work [22]. The LATA is the approach on making task assignment decisions via SDN to reduce task processing latency, which is mainly simulated according to the related work [27]. We use the following performance metrics to compare the three approaches and evaluate their performance, the performance metrics used in the evaluation are the Unit Task Processing Delay (UTPD), the Task Processing Time Optimization Ratio (TPTOR), the ECS Load Balancing Degree (ELBD), the Task Migrating Efficiency (TME) and the Task Migrating Success Ratio (TMSR).

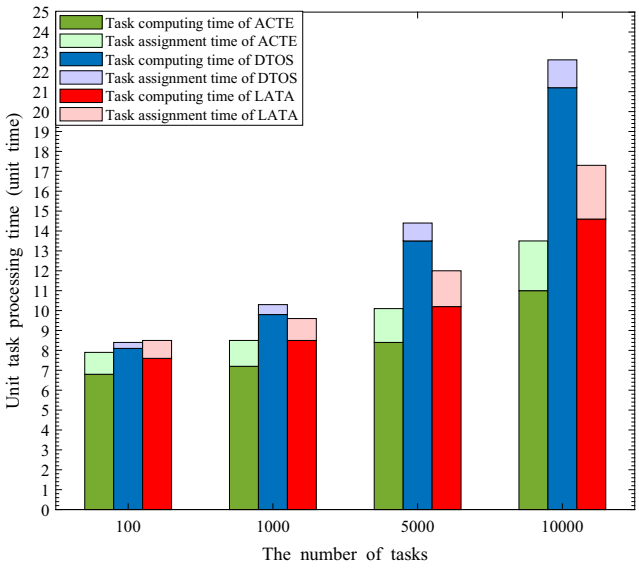
5.1 UTPD

The UTPD is defined as the time interval from the computing request being received to its corresponding computing task being successfully completed. It consists the task assignment time and the task computing time. We compare the UTPD of the three approaches, the results are shown in Fig. 4.

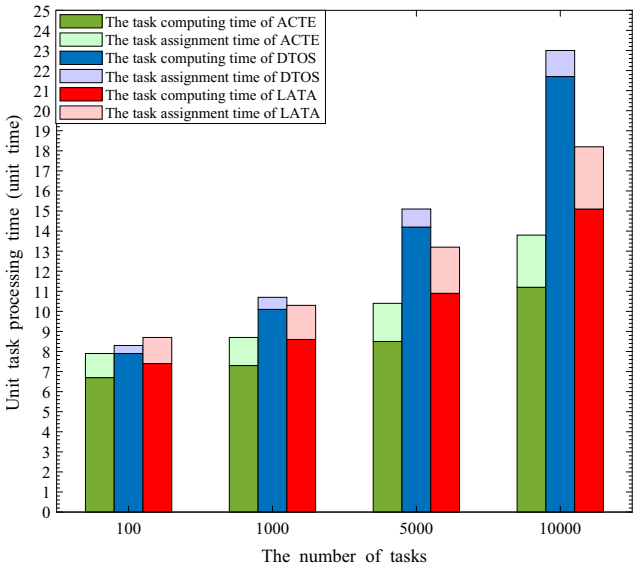
It can be seen that the average UTPD of ACTE is lower than that of DTOS and LATA, especially when the number of tasks increases rapidly. Although the task assignment time of ACTE is higher, the task computing time of it is always much lower than that of the two compared approaches. In more detail, when the number of tasks increases from 100 to 10,000, the average task

computing time of ACTE just increases by less than 5 unit time, while that of DTOS and LATA increase by more than 14 unit time and 9 unit time respectively. Moreover, under the peak load of tasks, the average time overhead to deal with a task by ACTE is about 60% of that by DTOS and 78% of that by LATA respectively. The reason is as follows. Before each time period, according to the historical records of each ECS, ACTE has already identified the ECS computing features for different types of computing tasks. Thus, the tasks can be processed by the ECSs that are good at dealing with them with less time. Although the task assignment time overhead of ACTE is relatively higher, once tasks have been assigned to suitable ECSs, the task computing time of ACTE is much lower.

Although LATA also assigns tasks based on the global view via the SDN controller, it mainly selects the ECSs of lower loads without the ECS task processing features considered. The unit task processing time cannot be further optimized. On the contrary, the task assignment time of DTOS is the lowest, because DTOS mainly assigns tasks to the local ECSs. When the number of tasks increases rapidly, the unit task processing



(a)



(b)

Fig. 4 The average unit task processing delay. **a** The average unit task processing delay over Geant, **b** The average unit task processing delay over Interroute

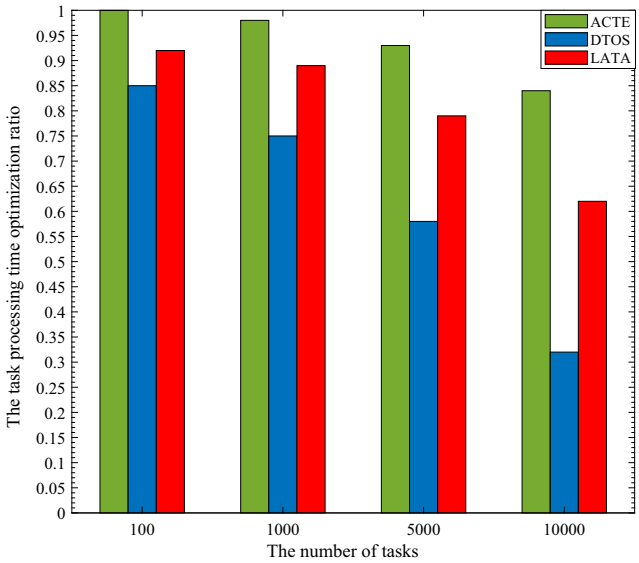


Fig. 5 The task processing time optimization ratio. **a** The task processing time optimization ratio over Geant, **b** The task processing time optimization ratio over Interroute

time of DTOS substantially increases due to the unbalanced ECS working loads.

5.2 TPTOR

The TPTOR is defined as the ratio of tasks with optimized processing time to the total tasks. Obviously, the more the number of tasks have been assigned to suitable ECSs to reduce their usual processed time, the higher the TPTOR is. We compare the TPTOR of the three mechanisms, the results are shown in Fig. 5.

The TPTOR of ACTE is always higher than that of DTOS and LATA. When the number of tasks increases, the TPTORs under the three approaches decreases. However, the decline rate of TPTOR of ACTE is obviously lower than that of DTOS and LATA, especially when the number of tasks reaches the maximum. In more detail, when the task load over the network is light, the TPTOR of ACTE can approach or reach 1, while the highest values of TPTORs of DTOS and LATA are just about 85% and 92% respectively. With the number of tasks increasing from 100 to 10,000, the TPTOR of ACTE still keeps beyond 84%, while the TPRORs of DTOS and LATA are lower than 35% and 60% respectively. The reason is as follows. ACTE not only assesses ECSs' usual computing features of dealing with different types of tasks, it also predicts the future unit task processing time if the task is assigned to an ECS before each assignment round. Thus, the

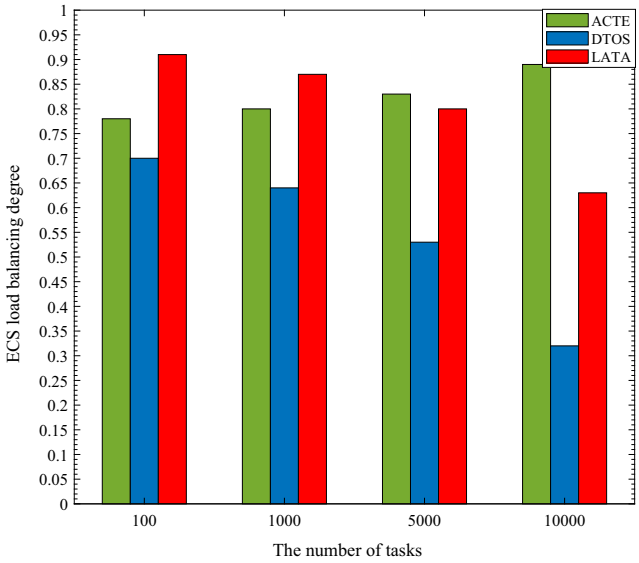


Fig. 6 The ECS load balancing degree. **a** The ECS load balancing degree over Geant, **b** The ECS load balancing degree over Interroute

ECS with the assessed minimum unit task processing time is selected among multiple candidate ECSs, which usually deals with the tasks assigned to it with less processing time. In addition, ACTE considers the load balancing among multiple ECSs in a global view, thus the ECS with more available computing capacity has higher possibility to deal with a new task, which also improves the TPTOR. On the contrary, DTOS and LATA assign tasks mainly according to the ECS's real-time working load without taking the ECS's computing features into account. Thus, the task processing time optimization mainly depends on the ECS's sufficient available computing capability. However, the number of tasks with optimized processing time greatly decreases with the overall ECSs' working loads becoming heavy. Especially, The TPTOR of DTOS is the lowest at the heaviest working load, because it cannot collaboratively assign excessive tasks to other ECSs with lighter working load in a global view.

5.3 ELBD

The ELBD is defined as the task load balancing degrees among multiple ECSs. It reflects the balance of each ECS's computing resource utilization. We compare the TPTOR of the three approaches, the results are shown in Fig. 6.

The ELBDs of ACTE and LATA are always higher than that of DTOS. The ELBD of ACTE becomes higher with the

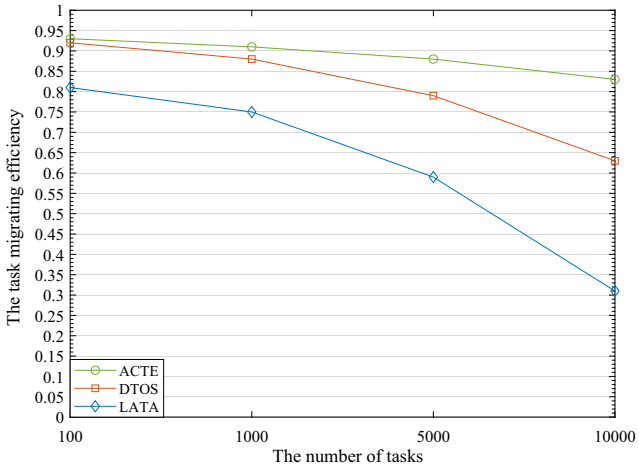


Fig. 7 The task migrating efficiency. **a** The task migrating efficiency over Geant, **b** The task migrating efficiency over Interroute

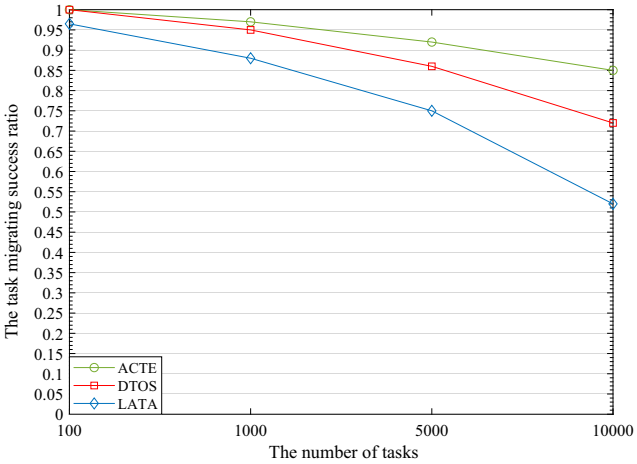
number of tasks increasing, while the ELBDs of DTOS and LATA decrease dramatically especially when the number of tasks increases rapidly. In more detail, when the number of tasks increases from 100 to 10,000, the ELBD of ACTE is always keeping beyond 75%, and its maximum value reaches 88%, while that of DTOS and LATA decline rapidly. The ELBDs of DTOS and LATA are only about 35% and 65% respectively at the highest task load. The reason is as follows. ACTE and LATA both take the overall ECSs' working loads into account to assign tasks with the global network status considered. While DTOS just considers the local network status, the large number of non-uniformly distributed tasks leads to lower ELBD of DTOS. Comparing with LATA, ACTE focuses the unit task processing minimum and considers ECSs' working load balancing as a further delay optimization method. Thus, when the number of tasks is low, the ELBDs of ACTE is lower than that of LATA. However, ACTE also takes the multiple possible scenarios for task assignment among multiple suitable ECSs into account, thus, the assignment is skewed toward the ECS with lower working load especially when the number of tasks is large. Therefore, the ELBD of ACTE is much higher than that of LATA when the number of tasks processed in the network reaches the maximum.

5.4 TME and TMSR

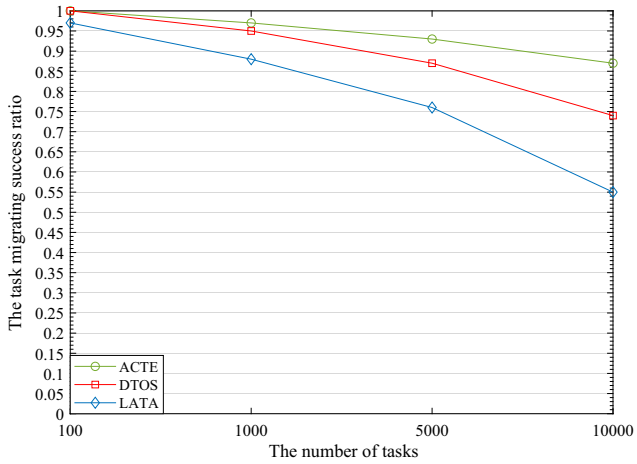
We also compare the TMEs and the TMSRs of ACTE, DTOS and LATA. The being processing tasks may need to be migrated to other ECSs to satisfy the changing demands of services in the real time. The TME is defined as 1 minus the ratio of the task migrating time to the task total processing time, and the TMSR is defined as the ratio of the successfully migrated tasks to the total tasks whose corresponding service demands are changed. In this simulation, we randomly select 20% of the total being

processed tasks to change their corresponding demands, and the results are shown in Figs. 7 and 8.

The TMEs and TMSRs of ACTE and LATA are always much higher than that of DTOS. When the number of tasks increases, the TMEs and TMSRs under the three approaches decreases. The values of TME and TMSR under LATA are very close to that under ACTE when the number of tasks is low, however, these values under LATA drop much faster than that under ACTE with the number of tasks increasing. Meanwhile, the values of TME and TMSR under DTOS drops dramatically by almost a half of their highest values. In more detail, when the number of tasks is low, the TMEs of ACTE and LATA are about 94% and 92% respectively, the TMSRs of ACTE and LATA are almost approaching 1, while the TME and TMSR of DTOS are about 81% and 96% respectively. When the number of tasks researches the maximum, the TME and TMSR of ACTE still keep beyond 80% and 85% respectively, while the TME and TMSR of LATA drop to about 65% and 75% respectively. Meanwhile, the TME and TMSR of DTOS are just about 33% and 53% respectively. The reason is as follows. ACTE and LATA support migrating tasks among multiple ECSs in a global view, and both tend to migrate tasks to the ECSs with lower working loads. Thus, the TME and TMSR of the two approaches can be well improved when the network task load is becoming heavy. However, LATA spends extra time re-analyzing reducing latency issues when re-assigning tasks, which affects its TME. In addition, the more the number of tasks there is, the less number of ECSs with low working loads there will be, which reduces the TMSR of LATA. ACTE supports quickly re-confirming suitable ECSs to re-assign tasks according to the changing demands, because the computing features of different ECSs have already been obtained, which enables tasks to be accurately migrated to appropriate ECSs. Thus, the TME of ACTE is improved. Furthermore, the re-estimation time can be saved if the task with the similar demands have ever been assessed due to the method devised in ACTE, which significantly optimize the TMSR of ACTE. On the contrary, DTOS has to



(a)



(b)

Fig. 8 The task migrating success ratio. **a** The task migrating success ratio over Geant, **b** The task migrating success ratio over Interroute

real-timely re-analysis potential game model, its TME reduces rapidly with the number of tasks increasing. Moreover, there are few choices to re-assign tasks to other ECSs in the same local network domain due to the local view of DTOS, which significantly affects the TMSR of DTOS.

6 Conclusions

In this paper, by leveraging the advantages of SDN, the mechanism called ACTE is proposed, it collaboratively assigns computing tasks among multiple ECSs in a global view. Three schemes are presented in the mechanism to optimize the task assignment and minimize the task processing delay. By mining the historical task processing data, the scheme to assess each ECS's task processing features is devised. Then, by estimating task processing time in different ECSs, the scheme to predict the ECS's future unit task processing delay is devised. Thus, each ECS's unit task processing time can be minimized by collaboratively assigning tasks among suitable ECSs at the current assignment round. In addition, in order to further improve the task assignment efficiency, an approach to optimize the task processing time estimation time overhead is designed. Simulation results have shown that the proposed mechanism is able to optimize the task processing delay more efficiently than the state of the art.

In this work, we mainly consider the static network topology, for example, the locations of switching devices and servers (i.e., the network nodes) are considered fixed. However, with the developing of IoE under 5G, the network nodes of dealing with tasks may be mobile. As a future research direction, we plan to extend the proposed mechanism in the scenario with a dynamic network topology, and consider the influence on the SDN-based task assigning brought by the mobile nodes.

Acknowledgments This work is supported by the National Natural Science Foundation of China under Grant no. 62002261 and 61802281, the Tianjin Municipal Education Commission Scientific Research Project under Grant No. 2018KJ145.

References

1. Hsieh HC, Chen JL, Benslimane A (2018) 5G virtualized multi-access edge computing platform for IoT applications. J Netw Comput Appl 115(1):94–102
2. Hassan N, Yau KLA, Wu C et al (2019) Edge computing in 5G: a review. IEEE Access 7:127276–127289
3. Taleb T, Samdanis K, Mada B, Flinck H (2017) On multi-access edge computing: a survey of the emerging 5G network edge cloud architecture and orchestration. IEEE Commun Surv Tut 19(3): 1657–1681
4. Pan J, McElhannon J (2018) Future edge cloud and edge computing for internet of things applications. IEEE Internet Things 5(1):439–449
5. Alvarez F, Breitgand D, Griffin D, Andriani P, Rizou S, Zioulis N, Moscatelli F, Serrano J, Keltsch M, Trakadas P, Phan TK, Weit A, Acar U, Prieto O, Iadanza F, Carrozzo G, Koumaras H, Zarpalas D, Jimenez D (2019) An edge-to-cloud virtualized multimedia service platform for 5G networks. IEEE T Broadcast 65(2):369–380
6. Zhao Y, Wang W, Li Y, Meixner CC et al (2019) Edge computing and networking: a survey on infrastructures and applications. IEEE Access 7:101213–101230
7. Salman O, Elhajj I, Chehab A, Kayssi A (2018) IoT survey: an SDN and fog computing perspective. Comput Netw 143(9):221–246
8. Schiller E, Nikaein N, Kalogeiton E, Gasparyan M, Braun T (2018) CDS-MEC NFV SDN-based application management for MEC in 5G. Comput Netw 135(22):96–107
9. Blanco B, Fajardo JO, Giannoulakis I, Kafetzakis E, Peng S, Pérez-Romero J, Trajkovska I, Khodashenas PS, Goratti L, Paolino M, Sfakianakis E, Liberal F, Xilouris G (2017) Technology pillars in the architecture of future 5G mobile networks NFV MEC and SDN. Comput Stand Inter 54(4):216–228
10. Barakabitze AA, Ahmad A, Mijumbi R, Hines A (2020) 5G network slicing using SDN and NFV: a survey of taxonomy, architectures and future challenges. Comput Netw 167(11):106984
11. Zaman FA, Jarray A, Karmouch A (2019) Software defined network-based edge cloud resource allocation framework. IEEE Access 7:10672–10690
12. Liu N, Dong Z, Roberto RC (2012) Task and server assignment for reduction of energy consumption in datacenters. In: IEEE international symposium on network computing and applications. USA, Cambridge, pp 171–174
13. Alkayal ES, Jennings NR, Abulkhair MF (2016) Efficient task scheduling multi-objective particle swarm optimization in cloud computing. In: IEEE conference on local computer networks workshops. Dubai, United Arab Emirates, pp 17–24
14. Keshanchi B, Souri A, Navimipour NJ (2017) An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing. J Syst Software 124:1–21
15. Gajera V, Shubham GR, Jana PK (2016) An effective multi-objective task scheduling algorithm using min-max normalization in cloud computing. In: IEEE international conference on applied and theoretical computing and communication technology. Bangalore, India, pp 812–816
16. Elaziz MA, Xiong S, Jayasena KPN, Li L (2019) Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution. Knowl-Based Syst 169:39–52
17. Farooq H, Sadoon A, Mohammad S, Tafazolli R (2020) Joint QoS-aware and cost-efficient task scheduling for fog-cloud resources in a volunteer computing system. ACM T Internet Techn, August: Article 05
18. Xing H, Liu L, Xu J, Nallanathan A (2019) Joint task assignment and resource allocation for D2D-enabled mobile-edge computing. IEEE T Commun 67(6):4193–4207
19. Sun Y, Wei T, Li H, Zhang Y, Wu W (2020) Energy-efficient multimedia task assignment and computing offloading for mobile edge computing networks. IEEE Access 8:36702–36713
20. Cheng S, Chen Z, Li J, Gao H (2019) Task assignment algorithms in data shared mobile edge computing systems. In: IEEE international conference on distributed computing systems. Dallas, USA, pp 997–1006
21. Alameddine HA, Sharafeddine S, Sebbah S, Ayoubi S, Assi C (2019) Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing. IEEE J Sel Area Comm 37(3):668–682

22. Li Y, Jiang C (2020) Distributed task offloading strategy to low load base stations in mobile edge computing environment. Comput Commun 164:240–248
23. Misra S, Saha N (2019) Detour: dynamic task offloading in software defined fog for IoT applications. IEEE J Sel Area Comm 37(5):1159–1166
24. Zhang J, Guo H, Liu j, Zhang Y (2020) Task offloading in vehicular edge computing networks: a load-balancing solution. IEEE T Veh Technol, 69(2): 2092–2104
25. Kiran N, Pan S, Yin C (2020) Joint resource allocation and computation offloading in mobile edge computing for SDN based wireless networks. J Commun Netw-S Kor 22(1):1–11
26. Shahryari S, Hosseini SA, Tashtarian F (2020) An SDN based framework for maximizing throughput and balanced load distribution in a cloudlet network. Future Gener Comp Sy 110:18–32
27. Chalapathi GSS, Chamola V, Tham CK, Gurunarayanan S, Ansari N (2020) An optimal delay aware task assignment scheme for wireless SDN networked edge cloudlets. Future Gener Comp Sy 102: 862–875
28. The Internet Topology Zoo. URL< http://www.topology-zoo.org/>
29. Yang S, Li F, Shen M, Chen X, Fu X, Wang Y (2019) Cloudlet placement and task allocation in mobile edge computing. IEEE Internet Things 6(3):5853–5863
30. Li D, Hong P, Xue K, Pei J (2018) Virtual network function placement considering resource optimization and SFC requests in cloud datacenter. IEEE T Parall Distr 29(7):1664–1677

Chao Bu received the B.S. degree in information security, and the M.S. and Ph.D degrees in software engineering from the Northeastern University, Shenyang, China. He is currently working at the School of Computer Science and Engineering, Tianjin University of Technology. His research interests include future Internet and service computing, etc. Email: bc_0722@163.com

Jinsong Wang received the B.Sc. degree from the Department of Computer Science, Tianjin University of Technology, Tianjin, China, and the M.Sc. and Ph.D. degrees from Nankai University, Tianjin. He is currently a Professor with the School of Computer Science and Engineering, Tianjin University of Technology. His research interests include computer networks, distributed computation, and evolutionary computation. Email: jswang@tjut.edu.cn