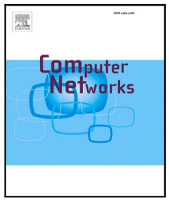
Computer Networks 214 (2022) 109164

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Utility Maximization for Splittable Task Offloading in IoT Edge Network

Jiacheng Wang [a,1], Jianhui Zhang [a,1], Liming Liu [a], Xuzhao Zheng [b], Hanxiang Wang [a], Zhigang Gao [a,*]

[a] School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, 310018, China
[b] Hangzhou Hikvision Digital Technology Co., Ltd., Hangzhou, 310052, China

ARTICLE INFO

Keywords:
Internet of Things
Edge Networks
Time-Expanded Graph
Utility Maximization
Task Offloading

ABSTRACT

This paper comprehensively investigates spatio-temporal dynamics for task offloading in the Internet of Things (IoT) Edge Network (*i*TEN) in order to maximize utility. Different from the previous works in the literature that only consider partially dynamic factors, this paper takes into account the time-varying wireless link quality, communication power, wireless interference on task offloading, and the spatiotemporal dynamics of energy harvested by terminals and their charging efficiency. Our goal is to maximize utility during the task offloading by considering the above-mentioned factors, which are relatively complex but closer to reality. This paper designs the Time-Expanded Graph (TEG) to transfer network dynamics and wireless interference into some static weight in the graph so as to devise the algorithm easily. This paper firstly devises the Single Terminal (ST) utility maximization algorithm on the basis of TEG when there is only one terminal. In the case of multiple terminals, it is very complicated to directly solve the utility maximization of the task offloading. This paper adopts the framework of Garg and Könemann and devises a multi-terminal algorithm (MT) to maximize the total utility of all terminals. MT is a fast approximate algorithm and its approximate ratio is 1-3ς, where $0 < \varsigma < 1/3$ is a positive small constant. The comprehensive experiments are conducted to illustrate that our algorithm significantly improves the overall utility compared to the three basic algorithms.

1. Introduction

With the rapid proliferation of the IoT, the IoT terminals and their produced data grows dramatically in recent years [1]. However, due to the limitation of the computation and storage capabilities of the IoT terminals, they are inadequate for computation-intensive tasks [2]. Furthermore, IoT terminals are limited by their own power supply [3,4]. Fortunately, the fast development of wireless networks has enabled more and more IoT terminals to connect to the Internet and share their information. IoT terminals move some computation tasks to the specified devices, which have relatively rich computation and storage capability and sufficient energy supply [5,6]. This way of migrating tasks is often described as computation offloading [7].

In recent years, a growing number of works have been contributed to computation offloading [1,8,9]. Cloud computing emerges in late 2007, which provides flexible computation services by remote cloud [10,11]. The Cloud integrates abundant computation and storage resources in a specific place and offers diversified services to the IoT terminals [12]. Nevertheless, the number of IoT terminals and their produced tasks are growing rapidly, which brings tremendous pressure to the computation load of the Cloud [13,14]. On the other hand, the location of the cloud is generally far away from the IoT terminals, so network transmission would cause high network delay and energy consumption [15]. It is not feasible for time-sensitive tasks especially, such as real-time control in the industrial IoT [16–18]. To cope with the constrained high network latency and energy consumption of remote cloud, Mobile Edge Computing (MEC) has been proposed as a feasible paradigm [19,20]. In MEC, edge servers at the edge of the wireless networks receive and complete tasks from the IoT terminals. They have the advantage of being geographically close to IoT terminals and thus highlight its advantages in latency and bandwidth to effectively improve the reliability and quality of services [21–23]. This paper adopts a three-tier collaboration architecture called *i*TEN, which is composed of IoT terminals, edge servers, and the cloud. The previous works only consider part of the network dynamic properties such as power or link quality during task offloading. However, we need to take extra comprehensive factors into account to make it closer to reality [24,25]. This paper considers not only the network dynamic properties but also the time variability of the IoT terminal energy

* Corresponding author.
E-mail addresses: jcwang@hdu.edu.cn (J. Wang), jh_zhang@hdu.edu.cn (J. Zhang), limingliu@hdu.edu.cn (L. Liu), zhengxuzhao66@163.com (X. Zheng), hx_wang@hdu.edu.cn (H. Wang), gaozhigang@hdu.edu.cn (Z. Gao).
[1] Equal contribution and shared co-first authorship.

https://doi.org/10.1016/j.comnet.2022.109164
Received 22 December 2021; Received in revised form 27 June 2022; Accepted 4 July 2022
Available online 11 July 2022

collection and the imperfect charging efficiency. Integrating these factors, this problem becomes more complicated but more realistic. This paper further considers the problem of wireless interference during multi-terminal offloading and how to effectively characterize the above factors. These factors bring great challenges to the task offloading of IoT terminals.

This paper first comprehensively considers the dynamic characteristics of the network, the dynamic energy collection of the IoT terminals and their imperfect charging efficiency, as well as the interference factor of wireless transmission, and then try to characterize the above dynamic factors in a suitable way. To maximize the utility of task offloading under constraints of wireless interference is an NP-hard problem, and it is quite challenging to solve directly. This paper devises Time-Expanded Graph (TEG), which is a novel and fascinating way to handle the above-mentioned problem. Finally, the paper studies how to maximize the total utility of task offloading in two cases. Specifically, this paper first studies the case of a single terminal and designs a Single Terminal (ST) algorithm based on TEG. Secondly, each terminal has a certain amount of tasks to offload in multi-terminal cases. This paper adopts Garg and Könemann's method [26] to handle its dual problem rather than the primal one, which allows us to design an approximation algorithm: Multiple Terminal (MT) algorithm.

Contributions. This paper considers the task offloading from the IoT terminals to the edge servers or cloud by introducing TEG to devise an interesting method for the utility maximization problem and analyze it theoretically. The main contributions are listed below:

1. This paper considers network dynamics and wireless interference factors comprehensively, which are much more complicated than the previous one which only considers partial network dynamics. This paper introduces TEG to dynamically represent network dynamics and wireless interference, which simplifies the problem and enables us to have a good perspective to handle the issue.
2. On the basis of TEG, this paper proposes ST algorithms to solve the path selection of task offloading of a single IoT terminal, and expounds on the feasibility of the algorithm in the *i*TEN utility maximization. For scenarios where multiple IoT terminals offload tasks simultaneously, this paper simplifies the problem by introducing Garg and Könemann's framework and proposing MT algorithms with an approximate ratio of 1-3ς.
3. The proposed ST and MT algorithms are evaluated by extensive simulation experiments and compared with three baseline algorithms. The results show that our algorithm is significantly superior to these baseline algorithms in terms of utility, throughput, and energy consumption.

This paper is organized as follows. Section 2 summarizes the related work of task offloading. Section 3 proposes the system model to describe the dynamics of the network and formulates the problem of utility maximization. The TEG is constructed in Section 4, and the utility maximization problem is transformed into the corresponding TEG problem. Sections 5 and 6 design the corresponding ST and MT algorithms for a single terminal and multiple terminals respectively and carry out the theoretical analysis. This paper evaluates the performance of our proposed algorithms through extensive experiments in Section 7. Section 8 summarizes the work of this paper.

Table 1 lists the main symbols used in the paper.

Table 1
The main symbols used in the paper.

Sym.	Explanation	Sym.	Explanation
G	Graph	G^{Γ}	TEG
v	Node	e	Edge
τ	Time slot	T	Period
V	Node set	V^{Γ}	TEG node set
E	Edge set	E^{Γ}	TEG edge set
M	# of nodes	N	# of edges
h	Node process ability	θ	Remaining energy
ϕ	Consumed energy	ρ_r	Receiving power
ρ_t	Transmission power	f	Throughput
u	Utility	c	Edge capacity
p	Single offloading path	l	Length function
I	Interference function	r	Link quality
P	Path set	Q	Objective value function
F	Set of task	K	$\|F\|$
$\epsilon, \varsigma, \xi, \delta$	Constants	z, λ	Variable for dual problem

2. Related works

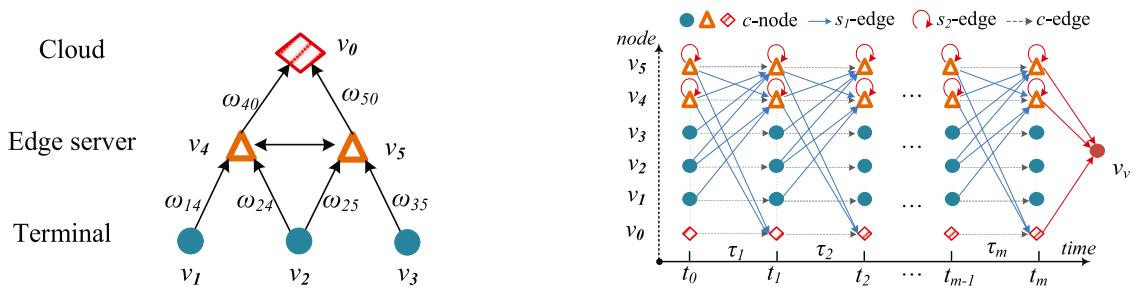
2.1. IoT Edge Computing

Extensive research was devoted to the field of computing offloading in IoT edge computing [27,28]. Guo *et al.* arranged multiple mobile terminals to offload their computation tasks to the remote cloud. The clock frequency and transmission power of mobile terminals were optimized to minimize the energy consumption and the computation delay while considering the task priority requirements [29]. Wang *et al.* investigated energy and task causality constraints due to task dynamic arrival and channel fluctuations and studied the terminal collecting energy from the energy beamforming, which could be used to perform computing tasks locally or offload tasks to the edge servers [30]. In [31], Zhang *et al.* proposed a stochastic mixed integer nonlinear programming problem based on joint optimization of the task allocation decision, flexible computational resource scheduling and wireless resource allocation. Guo *et al.* considered representing the energy-efficient computation offloading problem as a mixed integer non-linear programming problem [32].

Existing works were devoted to the problem of computation offloading in multi-hop scenarios [33,34], where edge servers have a connection with some other edge servers or clouds and share resources with each other [33]. Funai *et al.* proposed a heuristic algorithm for iterative task assignment, which could optimize the collaborative network of computing task allocation in the multi-hop collaborative network [34]. However, these efforts focused on independent tasks and usually did not jointly take network flow scheduling into account. In [35], the problem of fine-grained task offloading in edge computing of low-power IoT systems was studied. The goal was to minimize the average task duration of all IoT applications. The unique task topologies and schedules of the IoT network had a great impact on the performance and resource utilization of the whole task offload [36,37]. Wang *et al.* proposed a multi-user task offloading scheme based on non-orthogonal multiple access technologies, which could make multi-users offload tasks simultaneously to increase efficiency [36]. Apostolopoulos *et al.* considered the impact of users' risk-seeking or loss-aversion behaviors on mobile edge computing policies under the influence of current uncertainties such as unstable network connectivity [37].

2.2. iTEN Dynamics

The network dynamics were studied including some time-varying factors separately such as energy harvested, link quality, and power consumption. Zhan *et al.* considered the variation of transmission rate because the dynamic environment disturbs the channel [38]. Some works considered online offloading in iTEN, where task arrival time and channel state were time-varying [31,39]. He *et al.* considered that each user could only have one task per time slot in the multi-access edge computing system [40]. Liu *et al.* considered the network topology changes at different time slots due to IoT terminal or edge server movement [41]. In paper [42], Liu *et al.* considered that the geographical location of the users may change at different time slots. As users move, their tasks were migrated to nearby edge servers. In these works, dynamic factors such as transmission rate and dynamic



(a) Original Network (b) TEG Network

Fig. 1. (a) represents the original network, (b) constructs the corresponding TEG, the horizontal axis represents the time slot, and the time span is represented by the slot, such as $[t_{i-1}, t_i]$ is denoted by slot τ_i.

task arrival were considered, but in the *i*TEN, wireless interference and network dynamics were not fully considered.

From the existing works, we can find that the dynamics are the inherent features of *i*TEN and usually coexist. The existing related works are considered a part of such features separately so as not able to meet the fact that the dynamics and wireless interference coexist.

3. System model and problem formulation

3.1. System model

This paper constructs an actual architecture for collaborative task offloading in *i*TEN, as shown in Fig. 1(a), which includes three layers, namely terminal layer, edge layer and cloud layer. The terminal layer composes of diversified IoT terminals and the edge layer consists of a certain number of edge servers. This section introduces a graph $G(V, E)$ to represent the *i*TEN, where $V = \{v_j, j = 1, \ldots, M\}$, $E = \{e_{jk}, \forall v_j, v_k \in V\}$ and $|E| = N$. V is a node-set including the cloud, edge servers and terminals. Without loss of generality, v^0 represents the cloud node, and v^s and v^m represent the edge server node and the terminal node respectively. The node-set V is divided into two sets V_s and V_m. The former contains the cloud and edge server while the latter contains the terminals, *i.e.*, $\forall v^0$, $v^s \in V_s$ and $\forall v^m \in V_m$. V_m can be divided into two subsets: $V_{m,inf}$ and $V_{m,col}$, *i.e.* $V_m = \{V_{m,inf}, V_{m,col}\}$. $v^m \in V_{m,inf}$ has a fixed power supply while the other in $V_{m,col}$ may harvest energy from the environment and store it in its own battery of capacity B_i. In the energy model, the energy harvested by each terminal node is different from others due to the spatial inhomogeneity and time-variation of the environmental energy distribution [43]. Denote the energy collected by terminal node $v_i \in V_{m,col}$ in the time slot τ as $\varphi_i(\tau)$. Assume that the remaining battery energy of terminal node v_i at the start time t_k of time slot τ_{k+1} is $\theta_i(t_k)$. Therefore, the energy that terminal node v_i can use to transmit tasks in time slot τ cannot exceed the sum of its remaining and harvested energy. This paper further considers the imperfect charging efficiency of the battery, denoted by σ, and gets the following formula:

$$\theta_i(t_k) = \begin{cases} \min\{B_i, \theta_i(t_{k-1}) + \\ \qquad \sigma(\varphi_i(\tau_k) - \phi_i(\tau_k))\}, & \varphi_i(\tau_k) \geq \phi_i(\tau_k); \\ \max\{0, \theta_i(t_{k-1}) + \\ \qquad \varphi_i(\tau_k) - \phi_i(\tau_k)\}, & \varphi_i(\tau_k) < \phi_i(\tau_k). \end{cases} \tag{1}$$

E contains all directional edges between nodes and each edge denotes a communication link. For example, e_{jk} denotes a link from v_j to v_k and is associated with a capacity $c(e)$. The communication link can be either wireless or wired. Notice that $c(e)$ represents the communication bandwidth limitation of edge e as the following constraint.

$$f \leq c(e), \qquad \forall e \in E; \tag{2}$$

where f is the throughput going through the edge e and denotes the task load in this paper.

This paper studies the scenario where the *i*TEN is dynamic in several properties, including the link quality, the transmission power, and the task processing ability, which change over time because of variable physical reasons. At time t, this paper denotes the edge e's link quality by $r_e(t)$, $\forall e \in E$, as well as node v_j has a transmission power and node v_k has a receiving power, represented by $\rho_t^j(t)$ and $\rho_r^k(t)$ respectively, where $\rho_t^j(t) > 0$ and $\rho_r^k(t) > 0$, $\forall v_j, v_k \in V$.

For any edge e whose task load is f, it requires the transmitter and receiver to spend energy on communication, which is time-dependent and the equation is given as follows:

$$\begin{aligned} \phi_{\text{tx}}^j(f, t_{tx}) &= \int_{t_{tx}} f\rho_t^j(t)/r_e(t); \\ \phi_{\text{rx}}^k(f, t_{rx}) &= \int_{t_{rx}} f\rho_r^k(t)/r_e(t). \end{aligned} \tag{3}$$

where $\phi_{\text{tx}}^j(f,t)$ and $\phi_{\text{rx}}^k(f,t)$ represent the energy consumptions of the v_j and v_k and $0 \leq \phi_{\text{tx}}^j(f, t_{tx}) \leq \theta_j(t_{tx})$, $0 \leq \phi_{\text{rx}}^k(f, t_{rx}) \leq \theta_j(t_{rx})$. t_{tx} and t_{rx} are the time moments to begin the transmission and reception.

Some nodes also consume energy except that on communication when it processes tasks and suppose that node v_i is one of them. This paper also assumes that energy consumption on task processing is dynamic and lets $\psi_p^i(t)$ denote the energy consumption of node v_i to process unit task load at time t. Given the task with the load f_p, the time slot is t_p to process f_p and the consumed energy is given as the following equation.

$$\phi_{\text{c}}^i(f, t_p) = \int_{t_p} f_p \psi_p^i(t), \forall v_i \in V_s; \tag{4}$$

Each node may spend its energy on both communication and task processing. Let $\phi_i(\tau)$ denote the total energy consumptions of node v_i in the time slot τ, and get the components of the consumed energy as the following equation.

$$\phi_i(\tau) = \phi_{\text{tx}}^i(f_{tx}, t_{tx}) + \phi_{\text{rx}}^i(f_{rx}, t_{rx}) + \phi_{\text{c}}^i(f_p, t_p), \forall v_i \in V; \tag{5}$$

where f_{tx}, f_{rx}, and f_p are transmitted, received, and processed task load respectively. For each node in $V_{m,inf}$, the energy consumed on the communication cannot exceed the battery capacity in the whole period T. With the definitions of energy consumption in Eq. (5), The overall energy consumed by each terminal node cannot exceed the remaining energy in its battery as the following energy constraint:

$$\int_T \phi_i(\tau) \leq \theta_i(\tau), \forall v_i \in V_{m,inf}; \tag{6}$$

Consider that terminal node i can interfere with other terminal nodes in the process of offloading tasks through wireless transmission. The transmission rate $\mu(t)$ of terminal node i at time t can be obtained by the following Shannon formula [44,45]:

$$\mu_k^i(t) = B_k^i \log_2 (1 + \frac{\rho_t^i(t) g_k^i}{\omega + \sum_{v_j \in V_m, j \neq i} \rho_t^j(t) g_k^j}); \tag{7}$$

where B_k^i and g_k^i denote the bandwidth and channel gain between terminal node i and node k respectively. ω denotes the white Gaussian noise. Formula (7) represents the way to calculate the terminal transmission rate when the other terminals may transmit simultaneously.

Each node in V_s has its own computing capacity denoted by $h_i(t)$, which refers to the maximum available ability of node v_i to process tasks at time t. When each node v_i in the set V_s processes a task with the load of f, the time consumption Δt can be calculated by the following formula.

$$\Delta t_p^i(f, t_p) = \int_{t_p} \delta_1 \frac{f}{h_i(t)}, \qquad \forall v_i \in V_s; \tag{8}$$

where δ_1 is a constant, and t_p is the time moment to begin the task processing. When there is no task to process, $\Delta t_p^i(f, t_p) = 0$.

Each terminal may have a series of tasks to the process by offloading. This paper assumes that the task can be splittable into some segments with different lengths, and these segments forms a task f, *i.e.*, each splittable task contains a certain amount of load and different from others. When the task f is sent to node v_i, such as a edge server or cloud to process, which has time variable computing power h_i to process, it gets some utility as the following utility function if f is processed successfully.

$$u_i(f, t) = \delta_2 f h_i(t), \qquad \forall v_i \in V_s; \tag{9}$$

$$u_i(f, t) = \begin{cases} \dfrac{\delta_2}{\Delta t_p^i(f, t_p)}, & \Delta t_p^i(f, t_p) > 0; \quad (10) \\ 0, & \Delta t_p^i(f, t_p) = 0. \quad (10') \end{cases}$$

where δ_2 is a given positive constant. The utility function is the reciprocal of the time to process the task f, which means that the higher computing power leads to the better utility when given the task.

For each node in V_s, the constrained resource is the limited time. Given a time duration δt and the processed ability h, the task load f which can be processed should follow the below constraint:

$$f \le \int_{\delta t} h(t); \tag{11}$$

Interference function. In iTENs, some nodes may communicate by wireless so the wireless interference cannot be evitable. This paper assumes that the node cannot transmit and receive simultaneously when it communicates by wireless and adopts a general interference model I. Let I_e represents the interference set of edge e, including edge e and edges within the interference range of e, and I'_e represents the set of edges that would interfere with edge e. When the node transmits tasks with wired, the corresponding edge would not interfere with other edges, and the interference set of edge e is empty, *i.e.*, $I_e = \emptyset$.

3.2. Problem formulation

This paper studies the problem that the splittable tasks are offloaded from the terminals to the edge servers or the cloud so as to maximize the utility in the dynamic iTEN. Given the set V_m of terminals, each of which has a series F of tasks to process by offloading to those nodes in V_s, it costs some energy on the offloading and receives some utility. Assume that the energy consumption on communication and task processing is dynamic, and the related parameters, *i.e.*, $h_i(t)$ and $\rho(t)$, are previously known or can be estimated. The problem studied in this paper is to offload the task in F_i, $\forall v_i \in V_m$ to some nodes in V_s so as to maximize the overall utility under the constraints given in the previous subsections. The problem is formulated as below.

$$\mathcal{P}_1: \quad \max \quad \sum_{v_i \in V_s} u_i$$

$$s.t. \quad \text{Constraints (6) and } I; \tag{12}$$

$$\forall f \in F_j, v_j \in V_m. \tag{13}$$

When constraints in (12) are simplified to the wireless interference I, the problem $\mathcal{P}_1$ convert to the set coverage of the optimization problem and so that $\mathcal{P}_1$ is evidently NP-hard.

4. TEG and problem transformation

To describe the network dynamics, this section constructs the TEG, with which the problem $\mathcal{P}_1$ can be transformed as the static one so as to easy us to design the solution.

4.1. TEG

This section defines the graph $G^\Gamma(V^\Gamma, E^\Gamma)$ to represent the TEG of $G(V, E)$, where V^Γ and E^Γ denote the sets of the nodes and edges in TEG. This paper discretizes the system of the iTEN, where the period T is consists of m time slots of equal duration and assumes that the network properties keep constant including the processing ability, communication power, and link quality in each duration τ. The horizontal time coordinate is labeled with the time moments t_k, $k = 0, \dots, m$. The time slot τ_k represents the time duration between t_{k-1} and t_k. The TEG of G is constructed through the following steps:

1. Every node v in V is copied $m+1$ times denoted by $v(t_k), k = 0, \dots, m$, and named as c-node. All the c-nodes form V^Γ. Each c-node is assigned the parameter vector $[c(v(t_k)), \rho_t(t_k), \rho_r(t_k)]$, which are computing power and transmission power and receiving power respectively.
2. Create a directional link between c-node $v(t_k)$ and its next c-node $v(t_{k+1})$, and call it s_1-edge denoted by $e^{s_1}(\tau_{k+1})$. Create another s_2-edge from each edge server node $v(t_k) \in V_s$ to itself, and denote it by $e^{s_2}(\tau_{k+1})$. Set their weights as $[0, c(e^{s_1})]$ and $[1, c(e^{s_2})]$ respectively. The set E^s is consists of these s-edges.
3. For each node $v \in V$ in G, create c-edge from $v(t_k)$ to $v(t_{k+1})$, $k = 0, \dots, m$, denoted by $e^c(\tau_{k+1})$. Set parameter vector $[r_{e^c}(\tau), c(e^c)]$ for each of them and represent link quality and edge capacity respectively. The set E^c is consists of these c-edges.
4. Create a node v as the virtual node (v_v for short). Let all of "final" c-nodes from V_s, *i.e.*, $v_i(t_m)$, $\forall v_i \in V_s$, connect to v_v. Set the weights of the newly created edges to be zero and include the edges into E^c.

The above process leads to $V^\Gamma = \{v(t_j)$, j = 0, 1, 2, ⋯, m, $v \in V\} \cup \{v_v\}$ and $E^\Gamma = E^s \cup E^c$ in G^Γ. The reason to create s_2-edge for each node in V_s means that the edge server has the ability to relay tasks to other nodes and no energy consumption on task processing. By the above steps, this section obtains a new graph $G^\Gamma(V^\Gamma, E^\Gamma)$. In G^Γ, each terminal has more than one copies and the set V_m is extended as a new set. The c-node in TEG can be c-terminal, c-edge server or c-cloud. Let the first c-terminal $v(t_0)$ of each terminal in V_m be the source to offload the task, which is called the source c-terminal. All of the source c-terminals are included in a new set, which is also denoted by V^Γ when no confusion. The capacity of s-edge and c-edge in TEG is denoted by the $|\tau|$ and $\epsilon|\tau|$ respectively, where $\epsilon > 1$ is a small positive constant.

Fig. 1 gives an example to transfer a sample original network to its corresponding TEG. The original network $G(V, E)$ in Fig. 1(a) consists of $V = \{v_k, k = 0, \dots, 5\}$ and $E = \{e_k, k = 1, \dots, 8\}$. In Fig. 1(b), each node in $G(V, E)$ is copied of $m+1$ times, *i.e.*, the triangular node v_5 in the top row, v_5 has $m+1$ c-nodes, $v_5(t_k)$, $k = 0, \dots, m$. Each c-node has a gray dash-line arrow pointing itself from the current slot to the next slot, such as $v_5(t_0)$ to $v_5(t_1)$. v_5 has a series of s_1-edges, such as $v_4(t_k)$ to $v_0(t_{k+1})$, $k = 0, 1, \dots, m-1$. The second step creates the s_2-edge for the edge server. For example, the red circular arrow on $v_5(t_k)$, $k = 0, \dots, m$.

Path in TEG. Recall that the purpose of the task offloading is to maximize the utility. Given a task, the purpose can be transferred to find the shortest path in TEG. The task offloading consists of two parts, to find the path between the source and target and to implement the task. To obtain the utility in the dynamic iTEN, it requires to finding the

path with a certain cost and the target feeding back with some utility while the dynamic factors should be concerned in this paper. TEG can support the requirement and includes the network dynamic and wireless interference by finding the shortest path with the maximum flow. In TEG, each path contains the c-edge, s_1-edge, s_2-edge and c-node. The s_1-edge weight represents the cost to deliver the task while the s_2-edge weight refers to whether to offload tasks from the current edge node to other edge nodes or cloud nodes. Each c-node has the capacity to guarantee the consumed energy below the battery capacity. Notice that the edges linked to the virtual node have zero weight and the capacity of the virtual node is very large. This section gives the definition of the single offloading path in TEG.

Definition 1 (*Single Offloading Path*). An offloading path is an edge set from the source c-node $v_k(t_0)$ to the v_v in the TEG, including the s_1-edges, c-edges or the s_2-edges, denoted by p_k.

Through every single offloading path, the source c-node can send tasks to v_v. Let P_i denote the set of the single offloading path from $v_i(t_0)$ to v_v, $v_i \in V_m$. The task loads of them are thus $f(v_i(t_0), v_v) = \sum_{p\in P_i} f_p$, where $v_i \in V_m$. Let P be the set of all paths from all first c-nodes in V_m, *i.e.*, $P = \cup_{v_i\in V_m} P_i$.

Discrete constraints. Recall that each edge has the bandwidth and each s_1-edge has limited capacity accordingly.

$$\sum_{p\in P}\sum_{e\in p} f_p \le |\tau| c(e), \quad \forall \tau \in T, e \in E^\Gamma; \tag{14}$$

Some edges represent wireless communication and their corresponding s_1-edges have capacities affected by the wireless interference. So each s_1-edge has the constraint more general than that in (14) as the following one.

$$\sum_{p\in P}\sum_{e\in p} f_p + \sum_{e'(\tau)\in I'_{e(\tau)}} f_{e'(\tau)} \le c(e), \quad \forall \tau \in T, e', e \in E^\Gamma; \tag{15}$$

The above constraints stipulate that within the interference range of one s_1-edge in each time slot, the total throughput of the wireless s_1-edges should not be greater than the capacity of the s_1-edges. Note that any c-edge, s_2-edge, or wired s_1-edge is not affected by the interference model, *i.e.*, $I'_{e(\tau)} = \emptyset$, so it naturally satisfies the constraints. The constraint (15) is suitable for any edge $e \in E^\Gamma$, *i.e.*, the constraint is the generalization of the constraint (14).

The constraint (11) shows the limitation on the time available for the task processing. Since each node v_i has at most time $|\tau|$ to process task during each time slot τ, each c-node $v_i(\tau)$ can process the load $h_i(\tau)|\tau|$ at most, where $h_i(\tau)$ is the discretized processing ability of v_i in τ. The capacity of each c-node $v_i(\tau)$ is $h_i(\tau)|\tau|$, and the throughput is constrained as the following equation.

$$f \le h_i(\tau)|\tau|; \tag{16}$$

The parameters $\rho^i_t(t)$, $\rho^i_r(t)$ and $h_i(t)$ keep constant in each time slot in TEG. In order to simplify the representation, the time moments t_{tx}, t_{rx} and t_p in Eq. (5) are omitted here. Let $\phi^i(\tau)$ denote the energy consumed by node v_i in the time slot τ. The discrete version of Eq. (5) is given as the following one.

$$\phi^i(\tau) = \phi^i_{\text{tx}}(f_{tx}) + \phi^i_{\text{rx}}(f_{rx}) + \phi^i_{\text{c}}(f_p), \forall v_i \in V^\Gamma; \tag{17}$$

Accordingly Eq. (6) is discretized as the following energy constraint:

$$\sum_{\tau\in T} \phi_i(\tau) \le \theta_i(\tau), \forall v_i \in V_{m,inf}; \tag{18}$$

4.2. Discretized Utility Maximization problem

Some nodes have limited energy and their corresponding c-nodes update their energy by Eq. (17). Discrete the problem $\mathcal{P}_1$ into the corresponding $\mathcal{P}_2$ in TEG under the condition of discrete constraint, as shown below.

$$\begin{aligned} \mathcal{P}_2: \quad & \max \sum_{v_i(t_0)\in V^\Gamma_m} u(v_i(t_0), v_v) \\ & s.t. \quad \text{Constraints (15), (16) and (18);} \\ & f_i \geqslant 0, \quad \forall p \in P_i, v_i(t_0) \in V^\Gamma_m. \end{aligned}$$

5. Single terminal scheme

The preliminary purpose of task offloading is to maximize the utility and minimize the cost. This section studies the basic case where there is only one terminal to offload task and designs the ST algorithm.

5.1. Single offloading path construction

This subsection firstly investigates the construction of a single offloading path through the two steps of avoiding circulation and distributing energy in order to maximize utility and minimize cost.

Circle avoidance. The $G^\Gamma(V^\Gamma, E^\Gamma)$ may comprise cycles but any path that contains cycles would decrease the utility and increase the cost instead. Avoid any cycles when looking for a single offloading path in the task offloading process. Fortunately, it is not very difficult to determine whether a path contains cycles or not. Any path containing a cycle must pass through a node in $G(V, E)$ twice. Therefore, there are two c-nodes contained in the single offloading path and there is at least one s_1-edge among them. For example, as shown in Fig. 1(b), there is a path $v_1 \to v_4 \to v_5 \to v_4$ in G. Its corresponding path is $v_1(t_0) \to v_4(t_1) \to v_5(t_2) \to v_4(t_3)$. There are two c-nodes $v_4(t_1)$ and $v_4(t_3)$ of the same node v_4 and there is at least one s_1-edge, such as $(v_1(t_0), v_4(t_1))$. Therefore, the cycle in the path can be identified by using the following claim.

Claim 1 (*Cycle Identification*). *If any single offloading path in the original network G contains cycles, it must contain at least two same c-nodes and at least one s_1-edge between them in corresponding TEG.*

Energy allocation. Every single offloading path starts from one source c-terminal to the v_v and contains c-edge server or c-cloud. In order to maximize the utility, it has to deliver and process the task with the task load as much as possible according to the definition in Eq. (9). Accordingly, the task processing and delivering require the energy on transmission, receiving or task processing on the single offloading path with appropriate proportion so that there is an equal load to be delivered and processed. Concretely, the source c-terminal only spends energy on the transmission so it can devotes all its remaining energy on transmission to maximize the delivered task load. When one node acts as a relay node, it must use its energy for transmission and reception. When one node takes the task processing, it has to spend its energy on both receiving and processing. This section proposes a way to allocate energy for the latter two cases. Suppose that a c-node v_i is a relay node in the TEG. When in the time slot τ_k, it serves as a receiver to receive tasks through edge e, the remaining energy it can use is $\theta_i(t_{k-1})$, the receiving power is $\rho^i_{rx}(\tau_k)$, and the wireless link quality is $r_e(\tau_k)$. When in the time slot τ'_k, it serves as a transmitter to transmit tasks through edge e', the remaining energy it can use is $\theta_i(t_k)$, the transmission power $\rho^i_{tx}(\tau_{k'})$, and the wireless link quality is $r_{e'}(\tau_{k'})$, where $k < k'$. Let x, f, and f' be the amounts of energy spent on task receiving, data received by $v_i(t_k)$, and data transmitted by $v_i(t_{k'})$ respectively, and then get the following equation.

$$f = \frac{x r_e(\tau_k)}{\rho^i_{rx}(\tau_k)}; \tag{19}$$

$$f' = \frac{[\theta_i(t_{k-1}) - x] r_{e'}(\tau_{k'})}{\rho^i_{tx}(\tau_{k'})}. \tag{20}$$

To enable the two c-nodes to have the equal delivered task load, this paper assigns the energy for transmission and receiving satisfying the

Algorithm 1 Single c-node energy allocation

Input: Set transmission, receiving or task processing power, and the remaining energy $\theta_i(t_{k-1})$ for c-node $v_i(t_k)$.
Output: Allocate the energy to the c-node $v_i(t_k)$ for task transmission, receiving or processing.

1: **if** $v_i(t_k)$ is the source c-terminal **then**
2: It spends all remaining energy $\theta_i(t_{k-1})$ on task transmission;
3: **end if**
4: **if** $v_i(t_k)$ is the relay **then**
5: It allocates energy according to Equation (21);
6: **end if**
7: **if** $v_i(t_k)$ is the task processor **then**
8: It allocates energy according to Equation (24).
9: **end if**

following relationship while exploring all the remaining energy $\theta_i(t_{k-1})$ to maximize the delivered load. To set $f = f'$, and gets the amounts of energy on receiving and transmission are given as follows:

$$\begin{aligned}\theta^i_{rx}(\tau_k) &= \frac{r_{e'}(\tau_{k'})\rho^i_{rx}(\tau_k)\theta_i(t_{k-1})}{r_e(\tau_k)\rho^i_{tx}(\tau_{k'}) + r_{e'}(\tau_{k'})\rho^i_{rx}(\tau_k)};\\ \theta^i_{tx}(\tau_{k'}) &= \frac{r_e(\tau_k)\rho^i_{tx}(\tau_{k'})\theta_i(t_{k-1})}{r_e(\tau_k)\rho^i_{tx}(\tau_{k'}) + r_{e'}(\tau_{k'})\rho^i_{rx}(\tau_k)}.\end{aligned} \tag{21}$$

The method to allocate energy for the cloud and edge server to process tasks is quite similar. Suppose that a c-node is a processing node in the TEG. When in the time slot τ_k, it serves as a receiver to receive tasks through edge e, the remaining energy it can use is $\theta_i(t_{k-1})$, the receiving power is $\rho^i_{rx}(\tau_k)$, and the wireless link quality is $r_e(\tau_k)$. When it serves as a task processor to processing tasks, the remaining energy it can use is $\theta_i(t_k)$, the processing power $\rho^i_p(\tau_{k'})$. Let x, f and f' be the amount of energy spent on task receiving, the amount of data received by $v_i(t_k)$ and the amount of data processed by $v_i(t_{k'})$ respectively.

$$f = \frac{x r_e(\tau_k)}{\rho^i_{rx}(\tau_k)}; \tag{22}$$

$$f' = \frac{[\theta_i(t_{k-1}) - x]}{\rho^i_p(\tau_{k'})}. \tag{23}$$

Let $f = f'$ and lead to the result in the below equation.

$$\begin{aligned}\theta^i_{rx}(\tau_k) &= \frac{\rho^i_{rx}(\tau_k)\theta_i(t_{k-1})}{r_e(\tau_k)\rho^i_p(\tau_{k'}) + \rho^i_{rx}(\tau_k)};\\ \theta^i_p(\tau_{k'}) &= \frac{r_e(\tau_k)\rho^i_p(\tau_{k'})\theta_i(t_{k-1})}{r_e(\tau_k)\rho^i_p(\tau_{k'}) + \rho^i_{rx}(\tau_k)}.\end{aligned} \tag{24}$$

Determine the path utility and cost. When each c-node is assigned energy by the above equations, the c-edge server or c-cloud may be able to deliver or process tasks with a different load. However, every single offloading path delivers and processes one same task so the load keeps invariable. So we have the following claim for the task load on every single offloading path and summarize the above energy allocation as Algorithm 1.

Claim 2. *For a task delivered on a single offloading path p, its load, denoted by f_p is determined by the following way: $f_p = \min\{f_e, e \in p\}$, where f_e is calculated by the available energy for edge e, which is determined by Eqs. (21) and (24).*

5.2. Single terminal algorithm design

The section designs an algorithm, named Single Terminal algorithm (ST), to offload tasks when there is only one terminal. The basic mechanism of task offloading is to find an edge server or cloud server for the terminal and search for a path with the least consumption for task offloading. In TEG, the task communication power and the task computing power of the c-node have time-varying characteristics, so the cost and utility of task delivery are also time-varying. The core mechanism of ST is to look for the paths with the maximal utility and the minimum cost, and then renew the remaining energy of the c-node. The process is iterative till there is no such path existing. Given a source c-terminal $v_i(t_0)$ and the v_v in G^Γ, the algorithm ST first looks for all paths from source terminal $v_i(t_0)$ to the destination v_v and includes these paths in P_s. In order to save costs, ST searches for paths without cycles, so the Breadth First Search (BFS) can be used to find P_s. Next, according to the energy distribution method in Algorithm 1, and according to Definition 1, find the maximum utility path p_{max}. Note that if c-node $v_i(t)$ is the relay node on path p_i, it receives the task in the previous time slot, and transmits it in the next time slot. And the processing node c-node receives the task in the previous time slot and then processes it in the next time slot. Their energy distribution method is given in Algorithm 1. Finally, the remaining battery energy of the c-node on the path p_{max} and the remaining transmission capacity of the edges interfered by these edges on the path p_{max} are updated. Delete all paths that contain s_1-edges and whose updated capacity is negative or 0 from P_s, and repeat the above steps until there are no paths in P_s.

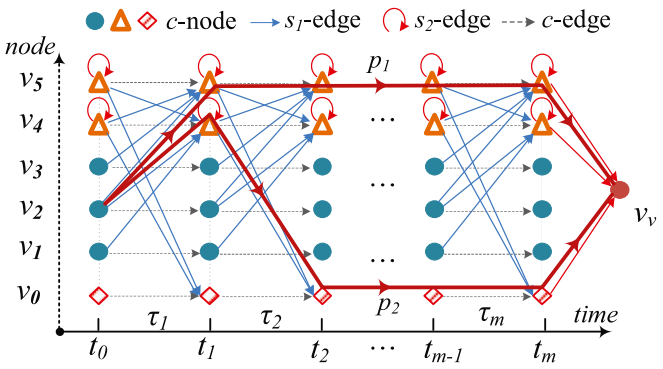


Fig. 2. There are two p_i, p_1 and p_2, in TEG.

This section illustrates the above process through the source terminal c-terminal $v_2(t_0)$ in Fig. 2. First, the algorithm ST searches all the paths from $v_2(t_0)$ to v_v through BFS and puts them into the set P_2. Second, calculate the utility of each path in the set P_2 and find the offloading path p_i with the largest utility. For example, path p_1 in Fig. 2 has the largest offloading utility, where p_1 contains the following c-nodes: $p_1 = v_2(t_0) \to v_5(t_1) \to v_5(t_2) \to \cdots \to v_5(t_{m-1}) \to v_5(t_m) \to v_v$. Finally, update the remaining energy of nodes $v_2(t_i)$, $i = 0, \ldots, m$, and $v_5(t_j)$, $j = 1, \ldots, m$. Furthermore, the edge interfered by path p_1 needs to update the capacity. Assuming that the maximum load of the edge e is $c(e)$, and it is interfered by the edge $e(v_2(t_0), v_5(t_1))$ in p_1, the offloading amount of this edge is f, and the remaining maximum load of edge e is $c(e)$ -= f. Perform the above steps iteratively until there is no path in the set P_2. Algorithm 2 describes the steps of the algorithm in detail.

5.3. Algorithm analysis.

Theorem 1. *The time complexity of Algorithm 2 is $O(m(N + 2M))$.*

Proof. There are $(m + 1)M$ c-nodes, mM s_1-edges, and mN c-edges in TEG. Step 3 in Algorithm 2 needs to spend $O(m(N + 2M))$ time to search all paths of the source terminal node $v_s(t_0)$ through BFS at time t_0. More than one bottleneck s_1-edge is found in each round, so the "while" loop has no more than mM rounds to complete. Therefore, the time complexity of Algorithm 2 is $O(m(N + 2M))$.

Algorithm 2 ST

Input: $G^{\Gamma}(V^{\Gamma}, E^{\Gamma})$, I, and F

Output: From $v_s(t_0)$ to v_v, the set of paths P_s that maximizes the overall task offloading utility.

1: The initial flow of all edges in G^{Γ} is set to be zero, *i.e.* $f = 0$;
2: Set $P_s = \emptyset$, and define $P' = \emptyset$ as a temporary path set;
3: Find all feasible paths of $v_s(t_0)$ to v_v through BFS, and add them to the P';
4: Use the method in Claim 1 to exclude paths containing loops from P';
5: According to Algorithm 1, calculate the maximum load transferred or processed by each s_1-edge in P_s with its maximum available energy, and then calculate the maximum load each path;
6: **while** $P' \neq \emptyset$ **do**
7: Search for the path p_{max} with the greatest task offloading utility, and obtain its corresponding load f_{max};
8: In Step 5, the energy allocation plan of the node has been calculated, and each c-node's remaining energy on p_{max} is updated according to this, and the remaining capacity of the edge on p_{max} is updated to $c(e) - f_{max}$;
9: The s_1-edge interfered by edge on p_{max} needs to update the capacity, *i.e.*, $c(e) - f_{max}$, $\forall e \in I_{p_{max}}$;
10: Move p_{max} from P' to P_s;
11: **end while**

6. Multiple terminals dual method

This section studies the case where there are multiple terminals, each of which has its own task set to offload. This paper introduces Garg and Könemann's framework [26] to transfer the problem $\mathcal{P}_2$ into a Maximum Concurrent Flow (MCF) problem, and then a simple and quick approximate solution is designed.

6.1. Garg and Könemann's framework

The MCF problem is a variation of the network flow problem and allows each commodity flow to deliver a flow demand for itself. Garg and Könemann's framework provides a fast and simple solution to it. Based on the TEG $G^{\Gamma}(V^{\Gamma}, E^{\Gamma})$ with the set V_m of all source c-terminals, each terminal source $v_i(t_0)$ has its own task q_i to deliver, which can be spitted into a set of tasks. In this paper, this algorithm needs to search task offloading paths P_i for each terminal under the energy and capacity constraints to maximize the overall utility while minimizing the costs. Notice that each source c-terminal may find more than one path to v_v. Let P_i represent the path set of the c-terminal $v_i(t_0)$ in $G^{\Gamma}(V^{\Gamma}, E^{\Gamma})$ and $P = \cup_{v_i(t_0) \in V_m} P_i$. Let f_p represent the task load on path p, $p \in P$, to indicate the load contained in the splittable task. The MCF problem can be shown by $\mathcal{P}_3$, as shown below.

$$
\begin{aligned}
\mathcal{P}_3: \quad & \text{Original problem} && \text{Dual problem} \\
& \max \sum \lambda && Q(l) \triangleq \min \sum_{e \in E^{\Gamma}} l(e)c(e) \\
& s.t. \sum_{e \in p_i} f_p \le c(e), \forall e \in E^{\Gamma}; && s.t. \sum_{e \in p} l(e) \ge z_i, \quad \forall p \in P_i; \\
& \sum_{p \in P_i} f_p \ge \lambda q_i, \forall v_i(t_0) \in V_m; && \sum_{i=1}^{K} q_i \cdot z_i \ge 1; \\
& f_p \ge 0, \forall p \in P. && l(e) \ge 0, \forall e \in E^{\Gamma}.
\end{aligned}
$$

The way to construct the dual problem of $\mathcal{P}_3$ is given as follows. Define $l(e)$ for each edge $e \in E^{\Gamma}$ which represents the length function of the edge, and define a positive throughput parameter z_i for c-terminal $v_i(t_0)$. Ensure that the length of each path in P_i is not less than z_i. The total amount of the product of throughput parameter z_i and demand q_i is not less than 1. Minimize the objective function $\sum_{e \in E^{\Gamma}} l(e)c(e)$ of the dual problem. An approximation algorithm for $\mathcal{P}_3$ [26] is proposed by Garg and Könemann.

The detailed procedure of Garg and Könemann algorithm is presented as below. Initially, $f = 0$, $\forall v_i(t_0) \in V_m$ and $l(e) = \frac{\xi}{c(e)}$, $\forall e \in E^{\Gamma}$, and $\xi = ((1-\varsigma)/|E^{\Gamma}|)^{1/\varsigma}$ is a small value with the previously given constant $\varsigma < 1$, which means that no task is delivered. The algorithm runs in multiple stages, and each stage contains multiple iterations. In each iteration, it needs to deliver q_i units of tasks for the source c-terminal $v_i(t_0)$. First, use the edge length function $l(e)$ to find the current shortest path p_i from $v_i(t_0)$ to v_v, where the parameter e is one of s_1-edge, s_2-edge and c-edge. Then, find the bottleneck load f_b through path p_i. Determine the size f_i of the task as the minimum between f_b and the remaining demand q_i' of the terminal $v_i(t_0)$, *i.e.*, $f_i = \min\{f_b, q_i'\}$. If $f_b > 0$, the length function $l(e)$ is multiplied by $1 + \varsigma \frac{f_i}{c(e)}$ and then obtain the shortest path in P_i, denote it by z_i, $z_i = \min_{p_j \in P_i} l(p_j)$. Use the length function $l(e)$ to represent the shortest path from $v_i(t_0)$ to v_v, which is represented by l^i_{min}. Let $\alpha(l) = \sum_{i=1}^{K} q_i z_i$ for all source c-terminal in V_m. So minimizing $Q(l)$ under the dual constraints is equivalent to minimizing $\beta \triangleq \min_l Q(l)/\alpha(l)$ by calculating the length $l(e)$ for each edge. When the target value is not less than 1, the algorithm stops under the specific conditions given in the following lemma, namely $Q(l) \ge 1$ and $\beta \ge 1$. When $\beta < 1$, Fleischer *et al.* raise a standard procedure to convert it to $\beta \ge 1$ [46]. It is worth noting that the final task load obtained through the above process may overflow on some edges in the path. Therefore, in order to obtain a feasible solution, it is necessary to cut down the final size f of the edge exceeding the capacity to the maximum f_m. When the path length is less than 1, the algorithm increases the throughput, and when the throughput overflows, the algorithm guarantees that the length of the edge grows exponentially. Therefore, it follows from the following theorem that the maximum throughput on the path is not large.

Lemma 2 (*see [26]*). *The size of the flow acquired through the above procedure is denoted by f, and it is reduced by $\log_{1+\varsigma} \frac{1}{\xi}$. The final size then is feasible.*

Lemma 3 (*see [26]*). *If $\beta \ge 1$, $\frac{|f|}{\log_{1+\varsigma} \frac{1}{\xi}} \ge (1-3\varsigma)OPT$, ς is a constant and OPT represents the size of the optimal flow.*

Lemma 4 (*see [26]*). *For any $\varsigma > 0$, there exists a way to calculate the approximate value of $(1-3\varsigma)$ for the MCF problem within the time complexity of $O(K(mN\xi)^2)$.*

$\mathcal{P}_2$ Transformation. Using the above-mentioned network flow, the throughput-related path is expressed as a network flow problem, and $\mathcal{P}_2$ can be converted into an MCF problem. For each source c-terminal $v_i(t_0)$ in the TEG, the throughput of the path from $v_i(t_0)$ to v_v is represented by the flow f_i. Each flow ensures that constraint constraints (6) and (15) are established. The solution of $\mathcal{P}_2$ is equal to finding a suitable flow path in TEG for all the tasks of all source c-terminals. In this section, $\mathcal{P}_2$ is transformed into the maximum multi-commodity flow problem of linear programming. Let P_i denote all possible task offloading paths related to the task in each source c-terminal $v_i(t_0)$, and P denote the union of all path sets, that is, $P = \cup_{v_i(t_0) \in V_m} P_i$. The problem $\mathcal{P}_2$ becomes the following formula.

$$
\begin{aligned}
\mathcal{P}_4: \max \quad & \lambda \\
s.t. \quad & \sum_{p \in P_i} f_p \ge \lambda q_i, \qquad \forall v_i(t_0) \in V_m; \\
& \text{Constraints (6) and (15);} \\
& f \ge 0, \qquad \forall f \in P_i, v_i(t_0) \in V_m.
\end{aligned}
$$

6.2. Networked energy allocation

Solving the maximum multi-commodity flow problem $\mathcal{P}_4$ directly is quite complicated. This paper adopts the framework of Garg and Könemann [26] and regards TEG as a directed graph with its own capacity for each edge. This section does not directly solve the solution of the original problem $\mathcal{P}_3$, but finds the solution of the dual problem of $\mathcal{P}_4$ and designs a fast approximation algorithm MT. For any path p_i, the throughput of multiple edges $e \in p_i$ on it is not only constrained by (15) but also affected by the energy consumption of its receiver in the time slot τ and that of its transmitter in the time slot $\tau+1$ when considering the available energy for transceiving. Recalling the above statement, we can know that c-edge does not need to consume energy. Suppose that the available throughput is f' under the energy constraints of Eq. (6). Replace f' in $\mathcal{P}_4$ with f, the constraint (6) can be omitted, and an equivalent problem is obtained, denoted as $\mathcal{P}_4'$. Therefore, the $\mathcal{P}_4'$ and the dual problem of $\mathcal{P}_3$ are the same types of problem.

We can easily know that the dual problem of $\mathcal{P}_3$ is NP-hard, so an approximate algorithm MT is designed for it to obtain an approximate solution in polynomial time. Approximation algorithms for optimization problems usually use the approximation ratio for theoretical evaluation. *Appro* and *Opt* are used to represent the approximation of the problem and the theoretical performance of the optimization algorithm. For the maximization problem, the approximation ratio ρ satisfies $\frac{Appro}{Opt} \geq \rho$, and for the minimization problem, the approximation ratio ρ satisfies $\frac{Appro}{Opt} \leq \rho$.

The idea of $\mathcal{P}_3$'s dual MT algorithm is to iteratively find the shortest path at the c-terminal of each source. Under the limitation of available energy and wireless interference, the available throughput is determined through the edge of the path. The MT algorithm includes three steps. In the first step, use the *Dijkstra* algorithm to find the shortest path of each source c-terminal from the set F. Secondly, it allocates energy for the shortest path to find the available throughput, and determines the final throughput in the dual problem of $\mathcal{P}_3$ under the given constraints. Finally, the remaining energy of the c-node in the shortest path is updated, as are the capacity and length of the edges. At the same time, it is also necessary to update the edges interfered by the edges on the shortest path. These three steps are repeated iteratively until all source c-terminals meet the requirements or $Q(l) \triangleq \min \sum_{e\in E^c} c'(e)l(e) \geq 1$. Algorithm 3 describes the details of MT in detail.

It can be seen from the 17th step of Algorithm 3 that the set $I'_{e:e\in p_i}$ contains the edges interfered by the wireless interference model I on the path p_i, and the available throughput of these edges is reduced. For example, assume that the s_1-edge $e_{25} : v_2(t_0) \rightarrow v_5(t_1)$ interferes with the edge $e_{35} : v_3(t_0) \rightarrow v_5(t_1)$ and the throughput on e_{25} is $f_{e_{25}}$ in Fig. 2. The available throughput of e_{35} takes the larger value between $c(e_{35}) - f_{e_{25}}$ and 0. Next, this section goes through a complete flow in Fig. 2 to illustrate the mechanism of Algorithm 3. After initializing the length of all s_1-edges, assume that the path p_1 is the shortest path from the source $v_2(t_0)$ to the target v_v. The algorithm MT calculates the available energy of all c-nodes on the path p_1 through the energy distribution method given in Algorithm 1 and then calculates the available throughput of each s_1-edge on the path p_1. It can be found that the bottleneck capacity of p_1 is $f'(p_1)$, and the increased throughput $\Delta f'(p_1)$ from $v_2(t_0)$ to v_v can be calculated. It is worth noting that the length of each s_1-edge on the path p_1 and the length of the edge interfered by it need to be updated at the same time, for example, $e_{35} : v_3(t_0) \rightarrow v_5(t_1)$. Under the constraints of the wireless interference model I, update the maximum available capacity of the edges interfered by the edges on p_1, all of which are obtained in the set $I'_{e:e\in p_i}$. Increase Q by $\varsigma \cdot \frac{\Delta f'_i}{c(e)}$ and decrease the demand q_i for the source c-terminal v_i through interval $\Delta f'_i$. MT repeats above the process until $Q \geq 1$.

According to Ref. [26], the time complexity of the algorithm proposed by the Garg and Könemann is $O(\varsigma^{-2}km \log L \cdot T_{sp})$, where T_{sp} represents the time needed to find the shortest path from the source terminal to the destination in the graph with non-negative edge weights, and L represents the maximum value of the sum of the number of edges of any path from the source terminal to the destination. In this paper, the theoretical performance of algorithm 3 is obtained by using the existing results. According to Claim 1 and Lemmas 2, 3, 4, the following theorem can be obtained. The number of c-nodes and the corresponding number of edges in TEG are $(m+1)M$ and $m(N+M)$ respectively.

Algorithm 3 MT

Input: $G^\Gamma(V^\Gamma, E^\Gamma)$; q for each terminal and $f = 0$.
Output: All source c-terminals tasks offload throughput load.

1: **for** each source c-terminal $v_i(t_0) \in V_m$ **do**
2: Set the throughput f_i of each source c-terminal to 0, and assign a demand value q_i for each source c-terminal, where $q_i > 0$;
3: **end for**
4: $Q = 0$; $\xi = (\frac{1-\varsigma}{|E^\Gamma|})^{1/\varsigma}$;
5: **for** each edge $e \in E^\Gamma$ **do**
6: $l(e) = \frac{\xi}{c(e)}$; $Q+ = l(e) \cdot c(e)$;
7: **end for**
8: Set a temporary variable q' to represent the requirements of each source c-terminal $v_i(t_0)$.
9: **while** $Q < 1$ **do**
10: **for** each $v_i(t_0) \in V_m$ **do**
11: **if** $q'_i \neq 0$ **then**
12: Set $q'_i = q_i$; /*q'_i is the remaining demand of $v_i(t_0)$*/
13: Use the length function $l(\cdot)$ to search for the shortest path p_i from each source c-terminal $v_i(t_0)$ to v_v in the G^Γ;
14: Configure energy for each c-node on path p_i according to algorithm 1 and get the real throughput f'_e of each e;
15: Search the practicable capacity threshold $f'_i(e')$ of e' on path p_i, and set $f'_{p_i} \leftarrow f'_i(e')$, where $e' \in p_i$;
16: Add practicable throughput to each source c-terminal $v_i(t_0)$: $\Delta f'_i \leftarrow \min\{f'_i(e), q'_i\}$;
17: **for** any edge $e \in p_i \cup I'_{e:e\in p_i}$ **do**
18: $l(e) *= (1 + \varsigma \cdot \frac{\Delta f'_i}{c(e)})$; $Q+ = \varsigma \cdot \frac{\Delta f'_i}{c(e)}$; $q'_i - = \Delta f'_i$;
19: **end for**
20: **end if**
21: **end for**
22: **end while**
23: **for** each source c-terminal $v_i(t_0) \in V_m$ **do**
24: $f'_i = f'_i|\tau|/\log_{1+\varsigma}\frac{1+\varsigma}{\xi}$; /*throughput scaling*/
25: **end for**

Theorem 5. *The time complexity of the algorithm MT is* $O(\varsigma^{-2}km^2(N+M)^2 \log m \log[(m+1)M])$ *in the TEG, and an approximate solution with an approximate ratio of* $1-3\varsigma$ *is designed,* $0 < \varsigma \leq 1/3$.

7. Evaluation

This section conducts two experimental cases to evaluate the performance of our proposed algorithms. In the first case, the number of edge servers is fixed to evaluate the impact on the number of terminals on the algorithm performance. The second case verifies the effect on algorithm performance when the number of edge servers increases in proportion to the number of terminals.

7.1. Simulation setting

Simulation scenarios. This paper conducts experiments in two different scenarios. In the first case, a relatively simple setting is adopted, in which the number of terminals increases from 10 to 120

Table 2
Experimental parameters in simulation.

Parameters	Values
Experiment range	1000*1000
The amount of cloud servers	1
The amount of edge servers in first case	3
The amount of IoT terminals in first case	[10,120]
The ratio of edge servers and IoT terminals in second case	1/10
The amount of IoT terminals in first case in second case	[10,50]
The communication range of edge servers	200
The communication range of IoT terminals	100
The wireless interference range of IoT terminals	150
The average of harvested energy of IoT terminals	[30, 150]
The variance of harvested energy of IoT terminals	[5, 30]
Transmission link quality	[0.5, 0.95]

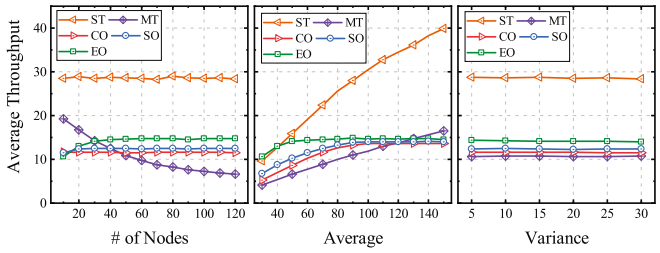


Fig. 3. Throughput under the different parameters.

and the interval increases by 10, while the number of edge servers and cloud servers are fixed at 3 and 1 respectively, and the average and variance are set as the values between 30 and 250 and those between 5 and 30. The energy collected in each time slot is obtained by the normal distribution. The second case considers a more complex situation, in which the number of edge servers varied as the IoT terminals dynamically, the terminals increase from 10 to 50, the interval 10, and the average and variance are 30–150 and 5–30, with interval 10 and 5, respectively. In ST and three basic algorithms, there is only one terminal. In MT, the numbers of terminals are randomly selected from the interval $[2, M/2]$ and the MT curve is the average value of the corresponding performance indicators of each terminal. The simulation programs of the two cases of experiments are running 100 times each to make the results more stable.

Parameter setting. This paper conducts extensive simulation with different numbers of nodes deployed in a 1000 × 1000 square area [47, 48]. It includes IoT terminals, edge servers and cloud servers, which are evenly distributed over the range. The number of cloud servers is set as 1, which can connect all edge servers. The transceiving range of the terminal node is set as 100 and the interference distance is set as 150. Since both the transceiving power and transmission link quality are time-varying, the simulation program randomly generates corresponding values within a certain range. So is the conversion efficiency of the harvested energy in each time slot. The main experimental parameters in the simulation are summarized in Table 2.

To comprehensively evaluate our proposed algorithm, this paper leverages three basic methods to make comparisons with our ST and MT methods.

1. Edge Offloading(EO): The user offloads all tasks to the edge server. Assume that the network channel condition is optimal and transceiving powers are static.
2. Cloud Offloading(CO): The user handles all tasks in the cloud, with other settings the same as EO.
3. Stochastic Offloading(SO): The user chooses to randomly offload tasks to edge servers or cloud servers. In this experimental setting, 50% of tasks are offloaded to edge servers and 50% to cloud servers respectively.

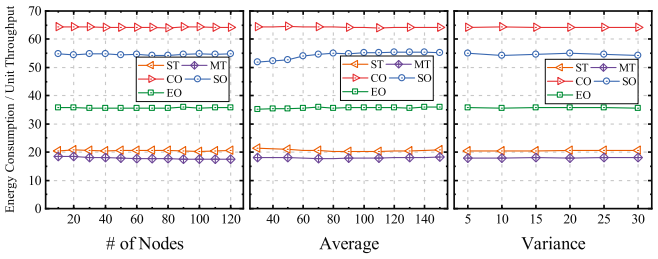


Fig. 4. Energy consumption per unit throughput under the different parameters.

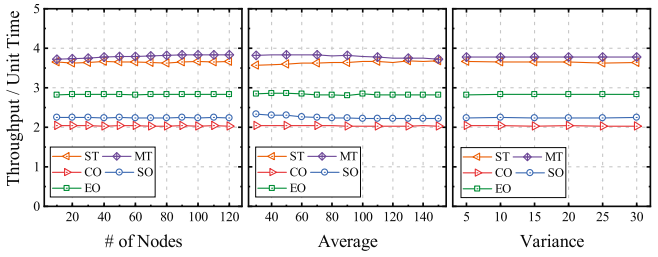


Fig. 5. Throughput per unit time under the different parameters.

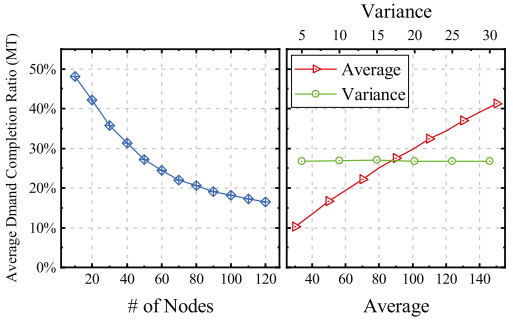


Fig. 6. Demand completion ratio under the different parameters.

7.2. Performance results

This subsection presents our proposed ST and MT and three basic algorithms under different parameter settings in terms of throughput, energy consumption per unit throughput, throughput per unit time, and utility.

Since this paper sets demand for each c-terminal in algorithm MT, this section can evaluate the average demand completion rate of terminals. In the TEG, the transmission time of the offloaded task can be calculated by Eq. (7), and the computing time of the task can be obtained by Eq. (8). The utility value adopts multiple criteria decision making and simple additive weighting method, where throughput is taken as the positive standard and execution time is the negative standard [49].

This section evaluates the performance of Algorithm ST and MT in comparison to three basic algorithms based on the simulated task of the first experimental case as shown in Fig. 3 to Fig. 8.

Throughput. Fig. 3 shows that the impact on throughput as the number of terminals, the average of the harvested energy, and the variance of the harvested energy. In the first subgraph, the throughput of both ST and three basic algorithms tends to be stable with the increase of terminals. However, with the increase of terminals, the average throughput of MT gradually decreases due to more interference. The last two subgraphs show that the throughput of ST and MT as well as the three basic algorithms tend to be stable as the average increases. In addition, variance has no effect on throughput.

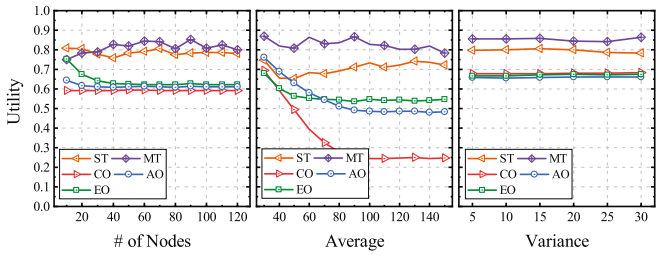


Fig. 7. Utility under the different parameters.

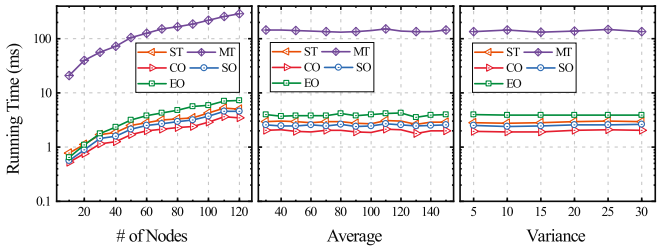


Fig. 8. Running time under the different parameters.

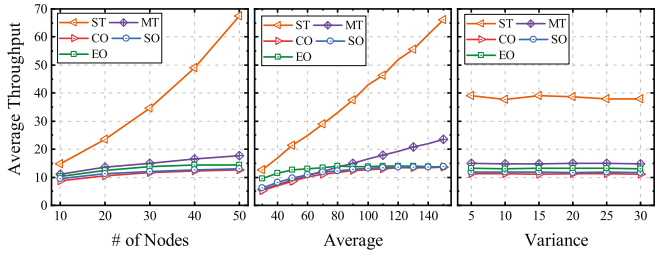


Fig. 9. Throughput under the different parameters.

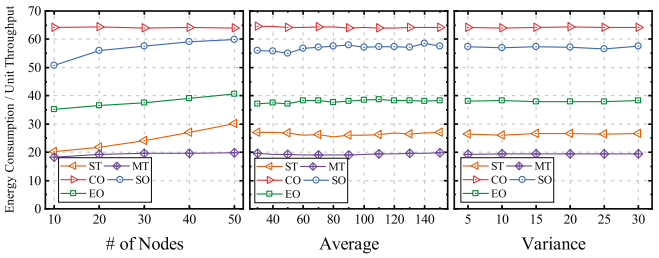


Fig. 10. Energy consumption per unit throughput under the different parameters.

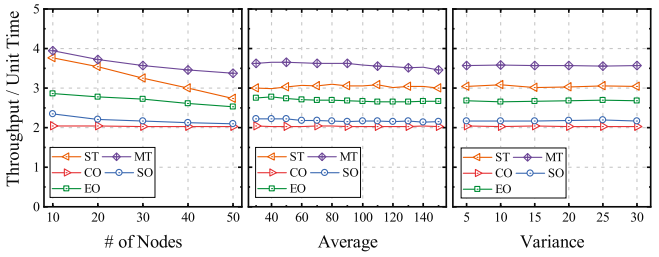


Fig. 11. Throughput per unit time under the different parameters.

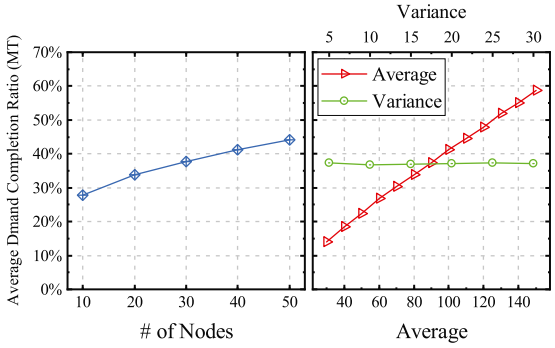


Fig. 12. Demand completion ratio under the different parameters.

Energy consumption per unit throughput. In Fig. 4, it is obvious that ST and MT perform better than the three basic algorithms in terms of energy consumption per unit throughput, while MT is slightly better than ST. The increase in the number of nodes, the average value and variance of harvested energy have little effect on energy consumption per unit of throughput.

Throughput per unit time. It can be seen from Fig. 5 that in terms of throughput per unit time, ST and MT are significantly higher than the three basic algorithms, while MT is slightly better than ST. At the same time, changes in the three independent variables, the number of terminals, the average and the variance of harvested energy, have not very obvious effects on the throughput per unit time.

Average demand completion ratio. The average demand completion ratio plotted in Fig. 6 is the percentage of tasks that are offloaded through MT in a given number of tasks. It decreases as the number of terminals increases and increases as the average value increases. In addition, variance has little effect on it.

Utility. According to the analysis in Fig. 7, the utility value of ST and MT under the influence of the three parameters is significantly better than that of the three basic algorithms. In the first subgraph, the utility of MT increases slightly with the increase of terminals. This is because the simultaneous offloading of multiple terminals reduces the average time cost. In the second subgraph, the utility of ST rises slightly after fluctuating, as energy increases, ST tries its best to offload tasks to maximize throughput. At the same time, MT is affected by the demand value of terminals, and its utility drops slightly. The variance has a small effect on their utility.

Running Time. Fig. 8 shows the running time of all algorithms. Note that this experiment runs under single thread on Apple M1 Pro chip in the MacBook Pro 2021 laptop. The running time of all algorithms is positively correlated to the number of nodes and independent of energy average and variance, which is consistent with our analysis on the time complexity of the algorithm ST, MT. It is obvious that ST has similar running time to other three algorithms while MT consumes more time, but also in millisecond level. This is because the time complexity of ST is proportional to the sum of the number of edges and nodes, while MT is proportional to the polynomial of the sum of them approximately, as analyzed above.

This section evaluates the performance of Algorithm ST and MT in comparison to the three basic algorithms based on the simulated data of the second experimental case as shown in Fig. 9 to Fig. 14.

Throughput. In Fig. 9, the results show that the ST is significantly better than the other four algorithms and the growth trend is obvious with the number of terminals and the average of the harvested energy on throughput. Meanwhile, the MT is better than the three basic algorithms but the growth trend is slow, mainly because the average throughput of multiple terminals is obtained and there is interference between multiple terminals.

Energy consumption per unit throughput. In Fig. 10, the energy consumption per unit throughput of ST and MT is significantly lower

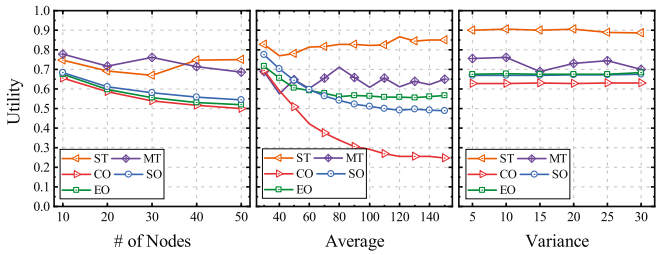


Fig. 13. Utility under the different parameters.

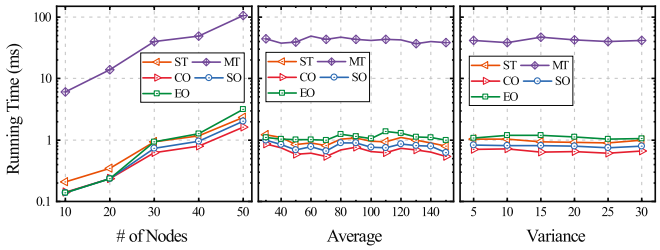


Fig. 14. Running Time under the different parameters.

than that of the three basic algorithms under the influence of three different factors, while MT is slightly better than ST. The main reason is that multiple terminals in MT can amortize the energy cost per unit throughput. In the first subgraph, although the number of terminals increases, the energy consumption per unit throughput of ST increases slightly because the current terminal node chooses some edge servers or cloud servers with large energy consumption to offload in the process of maximizing throughput.

Throughput per unit time. In Fig. 11, the overall trend shows that the throughput per unit time of MT is superior to that of ST, while MT is superior to the three basic algorithms. With the increase of terminals, the throughput per unit time of ST and MT gradually decreases, while the average and variance of the harvested energy have no effect on them.

Average demand completion ratio. Fig. 12 shows that as the number of terminals increases, the average demand completion ratio of multiple terminals increases. The difference from Fig. 6 is that the number of edge servers increases as the number of terminals increases in this case so that the terminals can offload tasks more effectively and increase the average demand completion rate. The average and variance of harvested energy are similar to Fig. 6.

Utility. Fig. 13 shows the utility of the three algorithms. Among them, with the increase of the number of nodes and the average value of collected energy, the utility of ST and MT fluctuate slightly, which is different from the obvious upward or downward trend in Fig. 13, indicating that the simultaneous growth of terminals and edge servers can enable each terminal node to get a better offloading of tasks. The situation of the three basic algorithms is the same as that in Fig. 7. The variance of harvested energy has little effect on them.

Running Time. Fig. 14 shows the running time of all algorithms on Apple M1 Pro chip. Although the number of edge servers is varied in this case, the trend of running time of all algorithms is similar to the fixed edge servers case. The running time at the millisecond level makes the algorithms practical in real-world applicability.

8. Conclusion

This paper investigates the splittable task offloading utility maximization in complicated time-varying network and wireless interference environments. We introduce a feasible scheme named TEG to character dynamic factors in the process of task offloading. For a single IoT terminal and multiple IoT terminal task offloading, this paper respectively devises practical algorithms based on TEG called ST and MT, which endeavor IoT terminal task offloading utility maximization under the constraint of the surplus energy. This paper provides a mathematics analysis to demonstrate the time complexity of ST and MT, guarantee algorithm gains the feasible solution and the approximate solution respectively. This paper considers task offloading in offline scenarios. In order to be closer to the online scenario, our next step is to study the online terminal task offloading on the basis of this paper.

CRediT authorship contribution statement

Jiacheng Wang: Writing – review & editing, Software, Validation, Visualization, Investigation. **Jianhui Zhang:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Funding acquisition. **Liming Liu:** Writing – original draft, Software, Validation, Visualization. **Xuzhao Zheng:** Validation, Investigation. **Hanxiang Wang:** Visualization, Investigation. **Zhigang Gao:** Supervision, Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work is supported by the National Key R&D Program of China under No. 2021YFC3320301, the National Natural Science Foundation of China under No. 61877015.

References

[1] W. Yu, F. Liang, X. He, W.G. Hatcher, C. Lu, J. Lin, X. Yang, A survey on the edge computing for the Internet of Things, IEEE Access 6 (2018) 6900–6919, http://dx.doi.org/10.1109/ACCESS.2017.2778504.

[2] J. Lin, L. Huang, H. Zhang, X. Yang, P. Zhao, A novel latency-guaranteed based resource double auction for market-oriented edge computing, Comput. Netw. 189 (2021) 107873, http://dx.doi.org/10.1016/j.comnet.2021.107873, URL https://www.sciencedirect.com/science/article/pii/S1389128621000426.

[3] F. Mashhadi, S.A.S. Monroy, A. Bozorgchenani, D. Tarchi, Optimal auction for delay and energy constrained task offloading in mobile edge computing, Comput. Netw. 183 (2020) 107527, http://dx.doi.org/10.1016/j.comnet.2020.107527, URL https://www.sciencedirect.com/science/article/pii/S1389128620311841.

[4] J. Zhang, W. Zhang, J. Wang, J. Feng, Z. Gao, S. Zheng, Rechargeable battery cabinet deployment for public bike system, IEEE Transactions on Intelligent Transportation Systems (2022) 1–14, http://dx.doi.org/10.1109/TITS.2022.3180079.

[5] S. Dong, H. Li, Y. Q.U., Z. Zhang, H. Lei, Survey of research on computation unloading strategy in mobile edge computing, Comput. Sci. 046 (11) (2019) 32–40.

[6] X. Shen, X. Shao, Q. Ge, L. Liu, RARS: recognition of audio recording source based on residual neural network, IEEE ACM Trans. Audio Speech Lang. Process. 29 (2021) 575–584, http://dx.doi.org/10.1109/TASLP.2020.3039597.

[7] K. Kumar, J. Liu, Y. Lu, B.K. Bhargava, A survey of computation offloading for mobile systems, Mob. Netw. Appl. 18 (1) (2013) 129–140, http://dx.doi.org/10.1007/s11036-012-0368-0.

[8] X. Hu, K. Wong, K. Yang, Wireless powered cooperation-assisted mobile edge computing, IEEE Trans. Wirel. Commun. 17 (4) (2018) 2375–2388, http://dx.doi.org/10.1109/TWC.2018.2794345.

[9] J. Wang, J. Hu, G. Min, A.Y. Zomaya, N. Georgalas, Fast adaptive task offloading in edge computing based on meta reinforcement learning, IEEE Trans. Parall. Distrib. Syst. 32 (1) (2021) 242–253, http://dx.doi.org/10.1109/TPDS.2020.3014896.

[10] L. Wang, G. von Laszewski, A.J. Younge, X. He, M. Kunze, J. Tao, C. Fu, Cloud computing: a perspective study, New Gener. Comput. 28 (2) (2010) 137–146, http://dx.doi.org/10.1007/s00354-008-0081-5.

[11] F. Saeik, M. Avgeris, D. Spatharakis, N. Santi, D. Dechouniotis, J. Violos, A. Leivadeas, N. Athanasopoulos, N. Mitton, S. Papavassiliou, Task offloading in edge and cloud computing: A survey on mathematical, artificial intelligence and control theory solutions, Comput. Netw. 195 (2021) 108177, http://dx.doi.org/10.1016/j.comnet.2021.108177, URL https://www.sciencedirect.com/science/article/pii/S1389128621002322.

[12] B. Hayes, Cloud computing, Commun. ACM 51 (7) (2008) 9–11, http://dx.doi.org/10.1145/1364782.1364786.

[13] L. Liu, Z. Chang, X. Guo, S. Mao, T. Ristaniemi, Multiobjective optimization for computation offloading in fog computing, IEEE Internet Things J. 5 (1) (2018) 283–294, http://dx.doi.org/10.1109/JIOT.2017.2780236.

[14] W. Fan, J. Han, L. Yao, F. Wu, Y. Liu, Latency-energy optimization for joint WiFi and cellular offloading in mobile edge computing networks, Comput. Netw. 181 (2020) 107570, http://dx.doi.org/10.1016/j.comnet.2020.107570, URL https://www.sciencedirect.com/science/article/pii/S1389128620312159.

[15] W. Hou, W. Li, L. Guo, Y. Sun, X. Cai, Recycling edge devices in sustainable Internet of Things networks, IEEE Internet Things J. 4 (5) (2017) 1696–1706, http://dx.doi.org/10.1109/JIOT.2017.2727098.

[16] Y. Chen, Z. Liu, Y. Zhang, Y. Wu, X. Chen, L. Zhao, Deep reinforcement learning-based dynamic resource management for mobile edge computing in industrial Internet of Things, IEEE Trans. Ind. Inf. 17 (7) (2021) 4925–4934, http://dx.doi.org/10.1109/TII.2020.3028963.

[17] K. Cao, L. Li, Y. Cui, T. Wei, S. Hu, Exploring placement of heterogeneous edge servers for response time minimization in mobile edge-cloud computing, IEEE Trans. Ind. Inf. 17 (1) (2021) 494–503, http://dx.doi.org/10.1109/TII.2020.2975897.

[18] Y. Guo, Z. Zhao, K. He, S. Lai, J. Xia, L. Fan, Efficient and flexible management for industrial Internet of Things: A federated learning approach, Comput. Netw. 192 (2021) 108122, http://dx.doi.org/10.1016/j.comnet.2021.108122, URL https://www.sciencedirect.com/science/article/pii/S1389128621001961.

[19] Y. Mao, C. You, J. Zhang, K. Huang, K.B. Letaief, A survey on mobile edge computing: The communication perspective, IEEE Commun. Surv. Tutor. 19 (4) (2017) 2322–2358, http://dx.doi.org/10.1109/COMST.2017.2745201.

[20] P. Mach, Z. Becvar, Mobile edge computing: A survey on architecture and computation offloading, IEEE Commun. Surv. Tutor. 19 (3) (2017) 1628–1656, http://dx.doi.org/10.1109/COMST.2017.2682318.

[21] X. Jiang, F.R. Yu, T. Song, V.C.M. Leung, A survey on multi-access edge computing applied to video streaming: Some research issues and challenges, IEEE Commun. Surv. Tutor. 23 (2) (2021) 871–903, http://dx.doi.org/10.1109/COMST.2021.3065237.

[22] B. Xiang, J. Elias, F. Martignon, E. Di Nitto, Resource calendaring for mobile edge computing: Centralized and decentralized optimization approaches, Comput. Netw. 199 (2021) 108426, http://dx.doi.org/10.1016/j.comnet.2021.108426, URL https://www.sciencedirect.com/science/article/pii/S138912862100390X.

[23] B. Ahat, A.C. Baktir, N. Aras, I. Altinel, A. Özgövde, C. Ersoy, Optimal server and service deployment for multi-tier edge cloud computing, Comput. Netw. 199 (2021) 108393, http://dx.doi.org/10.1016/j.comnet.2021.108393, URL https://www.sciencedirect.com/science/article/pii/S1389128621003716.

[24] A.S. Tan, E. Zeydan, Performance maximization of network assisted mobile data offloading with opportunistic device-to-device communications, Comput. Netw. 141 (2018) 31–43, http://dx.doi.org/10.1016/j.comnet.2018.05.011, URL https://www.sciencedirect.com/science/article/pii/S1389128618302214.

[25] Y. Su, W. Fan, Y. Liu, F. Wu, Game-based distributed pricing and task offloading in multi-cloud and multi-edge environments, Comput. Netw. 200 (2021) 108523, http://dx.doi.org/10.1016/j.comnet.2021.108523, URL https://www.sciencedirect.com/science/article/pii/S1389128621004539.

[26] N. Garg, J. Köenemann, Faster and simpler algorithms for multicommodity flow and other fractional packing problems, in: Proceedings of the 39th Annual Symposium on Foundations of Computer Science, FOCS, IEEE Computer Society, Washington, DC, USA, 1998, pp. 300–339.

[27] J. Xu, L. Chen, P. Zhou, Joint service caching and task offloading for mobile edge computing in dense networks, in: 2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, April 16-19, 2018, IEEE, 2018, pp. 207–215, http://dx.doi.org/10.1109/INFOCOM.2018.8485977.

[28] X. Xu, Y. Li, T. Huang, Y. Xue, K. Peng, L. Qi, W. Dou, An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks, J. Netw. Comput. Appl. 133 (2019) 75–85, http://dx.doi.org/10.1016/j.jnca.2019.02.008.

[29] S. Guo, B. Xiao, Y. Yang, Y. Yang, Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing, in: IEEE INFOCOM 2016 - the 35th Annual IEEE International Conference on Computer Communications, 2016, pp. 1–9, http://dx.doi.org/10.1109/INFOCOM.2016.7524497.

[30] F. Wang, J. Xu, S. Cui, Optimal energy allocation and task offloading policy for wireless powered mobile edge computing systems, IEEE Trans. Wirel. Commun. 19 (4) (2020) 2443–2459, http://dx.doi.org/10.1109/TWC.2020.2964765.

[31] Q. Zhang, L. Gui, F. Hou, J. Chen, S. Zhu, F. Tian, Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud RAN, IEEE Internet Things J. 7 (4) (2020) 3282–3299, http://dx.doi.org/10.1109/JIOT.2020.2967502.

[32] F. Guo, H. Zhang, H. Ji, X. Li, V.C.M. Leung, An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing, IEEE/ACM Trans. Netw. 26 (6) (2018) 2651–2664, http://dx.doi.org/10.1109/TNET.2018.2873002.

[33] Z. Hong, W. Chen, H. Huang, S. Guo, Z. Zheng, Multi-hop cooperative computation offloading for industrial IoT-edge-cloud computing environments, IEEE Trans. Parall. Distrib. Syst. 30 (12) (2019) 2759–2774, http://dx.doi.org/10.1109/TPDS.2019.2926979.

[34] C. Funai, C. Tapparello, W.B. Heinzelman, Computational offloading for energy constrained devices in multi-hop cooperative networks, IEEE Trans. Mob. Comput. 19 (1) (2020) 60–73, http://dx.doi.org/10.1109/TMC.2019.2892100.

[35] C. Shu, Z. Zhao, Y. Han, G. Min, H. Duan, Multi-user offloading for edge computing networks: A dependency-aware and latency-optimal approach, IEEE Internet Things J. 7 (3) (2020) 1678–1689, http://dx.doi.org/10.1109/JIOT.2019.2943373.

[36] F. Wang, J. Xu, Z. Ding, Multi-antenna NOMA for computation offloading in multiuser mobile edge computing systems, IEEE Trans. Commun. 67 (3) (2019) 2450–2463, http://dx.doi.org/10.1109/TCOMM.2018.2881725.

[37] P.A. Apostolopoulos, E. Tsiropoulou, S. Papavassiliou, Risk-aware data offloading in multi-server multi-access edge computing environment, IEEE/ACM Trans. Netw. 28 (3) (2020) 1405–1418, http://dx.doi.org/10.1109/TNET.2020.2983119.

[38] W. Zhan, C. Luo, J. Wang, C. Wang, G. Min, H. Duan, Q. Zhu, Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing, IEEE Internet Things J. 7 (6) (2020) 5449–5465, http://dx.doi.org/10.1109/JIOT.2020.2978830.

[39] Y. Chen, N. Zhang, Y. Zhang, X. Chen, Dynamic computation offloading in edge computing for Internet of Things, IEEE Internet Things J. 6 (3) (2019) 4242–4251, http://dx.doi.org/10.1109/JIOT.2018.2875715.

[40] X. He, R. Jin, H. Dai, Peace: Privacy-preserving and cost-efficient task offloading for mobile-edge computing, IEEE Trans. Wirel. Commun. 19 (3) (2020) 1814–1824, http://dx.doi.org/10.1109/TWC.2019.2958091.

[41] B. Liu, Y. Cao, Y. Zhang, T. Jiang, A distributed framework for task offloading in edge computing networks of arbitrary topology, IEEE Trans. Wirel. Commun. 19 (4) (2020) 2855–2867, http://dx.doi.org/10.1109/TWC.2020.2968527.

[42] C. Liu, M. Bennis, M. Debbah, H.V. Poor, Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing, IEEE Trans. Commun. 67 (6) (2019) 4132–4150, http://dx.doi.org/10.1109/TCOMM.2019.2898573.

[43] S. Guan, J. Zhang, Z. Song, B. Zhao, Y. Li, Energy-saving link scheduling in energy harvesting wireless multihop networks with the non-ideal battery, IEEE Access 8 (2020) 144027–144038.

[44] M. Keshavarznejad, M.H. Rezvani, S. Adabi, Delay-aware optimization of energy consumption for task offloading in fog environments using metaheuristic algorithms, Cluster Comput. 24 (3) (2021) 1825–1853.

[45] L. Liu, Z. Chang, X. Guo, S. Mao, T. Ristaniemi, Multiobjective optimization for computation offloading in fog computing, IEEE Internet Things J. 5 (1) (2017) 283–294.

[46] L. Fleischer, Approximating fractional multicommodity flow independent of the number of commodities, SIAM J. Discret. Math. 13 (4) (2000) 505–520, http://dx.doi.org/10.1137/S0895480199355754.

[47] G. Yang, L. Hou, X. He, D. He, S. Chan, M. Guizani, Offloading time optimization via Markov decision process in mobile-edge computing, IEEE Internet Things J. 8 (4) (2021) 2483–2493, http://dx.doi.org/10.1109/JIOT.2020.3033285.

[48] T. Liu, Y. Zhang, Y. Zhu, W. Tong, Y. Yang, Online computation offloading and resource scheduling in mobile-edge computing, IEEE Internet Things J. 8 (8) (2021) 6649–6664, http://dx.doi.org/10.1109/JIOT.2021.3051427.

[49] X. Xu, C. He, Z. Xu, L. Qi, S. Wan, M.Z.A. Bhuiyan, Joint optimization of offloading utility and privacy for edge computing enabled IoT, IEEE Internet Things J. 7 (4) (2020) 2622–2629, http://dx.doi.org/10.1109/JIOT.2019.2944007.

Jiacheng Wang received the B.S. degree in Electronic Information Science and Technology from the School of Electronics and Electrical Engineering, Wenzhou University, Wenzhou, China, in 2020. He is currently pursuing the M.S. degree in computer technology with Hangzhou Dianzi University, Hangzhou, China.

Jianhui Zhang received the B.S. degree in mechanotronics and the M.S. degree in fluid mechanics from Northwestern Polytechnical University, China, in 2000 and 2003, respectively, and the Ph.D. degree in control theory and engineering from Zhejiang University, Hangzhou, China, in 2008. He is currently working as a full professor with the School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou. He is the leader of the group of internet of things including 7 faculties and 50+ graduated students. His research interests spans internet of things, machine learning, wireless mobile sensing and city computing.

Liming Liu is currently pursuing the M.S. degree in computer technology with Hangzhou Dianzi University, Hangzhou, China.

Hanxiang Wang received the B.S. degree in Computer Science and Technology from the School of Information, North China University of Technology, Beijing, China, in 2020. He is currently pursuing the M.S. degree in computer technology with Hangzhou Dianzi University, Hangzhou, China.

Zhigang Gao received the B.S. degree in physics and the M.S. degree in computer application from Lanzhou University, China, in 1996 and 2000, respectively, and the Ph.D. degree in Computer Science and Technology from Zhejiang University, Hangzhou, China, in 2008. He is currently working as a full associate professor with the School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou. His research interests include internet of things, machine learning, and mobile computing.