# Energy Consumption and Time Delay Optimization of Dependency-aware Tasks Offloading for Industry 5.0 Applications

Chen Xu, Mengzhuo Lv, Kun Zhang, Kui Cao, Gang Wang, Mingzhu We, and Bei Peng*

***Abstract*—With the development of network and communication technology, artificial intelligence, distributed computing and beyond fifth-generation communications, Industry 5.0 is booming and obtains rapid growth. To improve the processing efficiency of intensive tasks, Mobile Edge Computing (MEC) technology can facilitate task offloading from mobile devices to edge servers. Traditional methods do not fully consider that applications are usually composed of dependency-aware tasks, which neglect the impact of task dependencies on offloading strategies and lead to low efficiency in task scheduling. This paper proposes a joint optimization of energy consumption and time delay for dependency-aware task offloading with mobile edge computing. First, in order to minimize the energy consumption and task processing of mobile device, a dependency-aware task offloading model is established. Secondly, the dependencies between tasks are analyzed to construct a Directed Acyclic Graph (DAG), and an algorithm based on topological ordering is introduced to obtain possible solutions for task scheduling. Furthermore, to minimize the total cost, an improved Particle Swarm Optimization (PSO) algorithm is used to obtain the optimal task offloading decision and MEC server selection optimization. Experimental results demonstrate that the proposed strategy can reduce the time cost and energy consumption compared to existing typical methods for tasks with different dependencies effectively.**

***Index Terms*—Mobile edge computing, Industry 5.0, task scheduling, offloading strategy, particle swarm optimization.**

## I. INTRODUCTION

WITH the development of cutting-edge technologies such as complex networks and artificial intelligence, human society has gradually entered the industrial 5.0 era. By utilizing the technologies of digital twins, parallel systems, AR/VR/MR and brain computer interfaces, the enterprises reconstruct themselves into a virtual factory that synchronizes with real data from factories. The entire elements, business and process of the enterprise will be visualized and analyzed accurately, and quasi control can be achieved while greatly reduce the operation, maintenance and production costs of the entity. Faced with increasingly complex market competition and the growing demand for speed and customization from consumers, automated production alone cannot meet customer needs. Industry 5.0 aims to promote cooperation between humans and intelligent devices, combining human creativity with the accuracy and speed of automation. With the rapid development of Industrial Internet, the number of industrial devices on the cloud is increasing. The continuous expansion of network scale has led to increasingly high requirements for real-time analysis of massive data. Edge computing has gradually become the focus of attention in the field of Industrial Internet. Due to the low computing capacity of terminal devices and the constraints of their own battery energy, they cannot provide users with a relatively satisfactory computing experience [1].

Aiming to solve the above problem, Mobile Edge Computing (MEC) is introduced to avoid long delay by locating computing/processing near the terminal device [2]. MEC servers with strong computing power near mobile users can improve the computing power at the edge of the network. In this way, mobile devices can offload particular tasks to a nearby base station or Access Points (APs) equipped with MEC server. Relying on the powerful computing capacity of MEC servers, it can significantly reduce the energy consumption and processing delay of computationally intensive tasks, meet latency sensitive requirements, and improve the Quality of Service (QoS) of mobile applications [3]. The emergence of the fifth generation of mobile communication system (5G) has led to an explosive growth of applications for mobile terminals [4]. The scale of application scenario is also increasing, for example, natural language processing, virtual reality, and highly interactive online applications. Many computing intensive tasks are delivered to terminal devices for processing, such as artificial intelligence and massive information. Although handheld terminal devices have made significant progress after decades of development, they may not be able to keep up with the requirements of endlessly emerging application in aspects of communication and computing resources [5]. Moreover, the limitations of battery power, computing capacity and physical size of mobile devices prevent smooth execution of computing intensive applications on mobile devices. By transferring the computing intensive tasks from Internet of Things (IoTs)

This work was supported by Sichuan Science and Technology Program No. 23ZDYF0738, No. 2022YFG0343, and the National Natural Science Foundation of China with Grant 51975107 and Sichuan Science and Technology Major Project No. 2022ZDZX0039, No.2019ZDZX0020.

Chen Xu, Mengzhuo Lv, Kun Zhang, Kui Cao and Bei Peng are with the School of Mechanical and Electrical Engineering, University of Electronic Science and Technology of China, Chengdu, 611731, P R China (e-mail: piano_2009@uestc.edu.cn; bitlmz@163.com; khunzhang@foxmail.com; 1054970191@qq.com; beipeng@uestc.edu.cn).

Gang Wang is with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, 611731, P R China (e-mail: wanggang_hld@uestc.edu.cn).

Mingzhu Wei is with the School of Aeronautics and Astronautics, University of Electronic Science and Technology of China, Chengdu, 611731, P R China(e-mail: mingzhu.wei@uestc.edu.cn).

* Corresponding author: Bei Peng.

devices to cloud servers or MEC servers, mobile terminals can provide better user experience [6]. However, the task offloading process can be affected by various factors in different fields, such as communication channels, link quality, customer's preferences, the mobility of IoTs devices, and the availability of MEC servers [7]. Therefore, the task scheduling and the selection of MEC servers are the most critical issues for optimization of the cost and delay in total task processing. Offloading decisions require dynamic decisions about whether to offload tasks to the MEC server or the remote data processing center. When plenty of tasks are handled to offload towards the MEC server, excessive bandwidth resources will be consumed and also lead to unexpected transmission delay [8]. Thus, it is crucial to design an optimal offloading decision scheme for scheduling the tasks to the servers with adequate processing power.

To avoid excessive task offloading from massive mobile devices to edge nodes and balance the load on MEC servers being elected of task processing by multiple mobile devices simultaneously, this paper proposes a joint optimization of energy consumption and time delay for dependency-aware task offloading with mobile edge computing. The contributions of this work are as follows:

(1) To minimize the energy consumption and task processing of mobile device, we formulate a dependency-aware task offloading model.

(2) We construct a Directed Acyclic Graph (DAG) to analyze the dependencies between tasks, and designed an algorithm based on topological ordering to obtain possible solutions for task scheduling.

(3) To minimize the total cost, we propose an improved Particle Swarm Optimization (PSO) algorithm to achieve the optimal task offloading decision and MEC server selection optimization.

The rest of the paper is organized as follows: In Section 2, we list the relevant studies. Section 3 elaborates the system model. The formulated problems and solutions are presented in Section 4. The results and analysis are presented in Section 5. Section 6 summarizes the conclusions.

## II. Related works

In recent years, some scholars have studied task offloading in MEC scenarios from different perspectives. To solve the problems of task offloading and resource allocation in dynamic environments, Ranadheera et al. [9] proposed a reinforcement learning framework based on software defined edge clouds to optimize the utilization of distributed computing resources. Liu et al. [10] proposed a two-step joint clustering and scheduling scheme to optimize resource utilization between cells. By deploying a large number of micro base stations with MEC servers at the edge of the network, it can support efficient and flexible large-scale connections. Huang et al. [11] designed a heuristic task offloading algorithm to optimize the allocation of the channels and presented power distribution strategy to reduce the system computing overhead. To obtain the low-latency of 5G communications and optimize the MEC resource's allocation, Pang et al. [12] proposed a task offloading algorithm based on a convex programming model. Zhang et al. [13] implemented an energy-aware offloading scheme to jointly optimize the assignment of computing capacity and communication resources under the constraint of energy and delay sensitivity. By exploiting the prior knowledge of future motion trajectories and channel states of mobile devices, Wu et al. [14] proposed a simulated annealing algorithm to reduce system service delay.

Due to the exorbitant expenses in the deployment and maintenance of infrastructures, it is impractical to scatter abundant servers and access points in the edge network. Moreover, some edge computing server can only serve a limited number of applications. In addition, the diversity of different mobile devices has led to the instability of the edge computing environment [15]. Therefore, how to assign computing and bandwidth resources adaptively in the ever-changing edge computing environment is facing challenges. Wang et al. [16] proposed a resource assignment algorithm based on deep reinforcement learning to optimize the allocation of the resources under different MEC conditions. The proposed method can greatly reduce the average service time and balance the utilization of various resources. He et al. [17] designed a general framework and presented smart contract within a private blockchain network to improve the security and privacy of edge computing-enabled IoTs. To deal with different application requirements in multi-user wireless networks and multiple wireless access modes of communication between device, Lin et al. [18] employed reinforcement learning technology to address the problem of limited and inaccurate network information and presented an optimal resource allocation strategy to minimize the total cost. Lin et al. [19] introduced the AI service placement problem for multiple users and formulated a mixed integer nonlinear planning problem. By optimizing the resource allocation and service layout, the total computing time and energy consumption of all users can be minimized. Aiming at the delay problem in real-time image processing, Shen et al. [20] designed an intelligent recognition technology based on deep learning to process real-time image. To improve the efficiency of scheduling decision for the application with dependent tasks, Sowndarya et al. [21] proposed an Individual Time Allocation method with Greedy Scheduling (ITAGS) to minimize the application execution cost.

Making the task offloading decision is the most critical issue for improving efficiency and user experience. It is important to analyze the dynamic decisions about whether tasks should be offloaded to the MEC server or be processed locally. Moreover, appropriate bandwidth allocation and resource scheduling are crucial to minimize total computing time and energy consumption. Traditional methods are inefficient in solving dependent task offloading problems and ignore the impact of task dependencies on offloading strategies. This paper proposes a dependency-aware task offloading for joint optimization of delay and energy consumption (DTO-JODE) under the constraints of limited computing resources in MEC servers.

## III. System model

### A. Network model

The system of Mobile Edge Computing includes MEC servers, APs and mobile devices, as shown in Figure 1. Mobile

users can submit local tasks through WiFi, 4G or 5G communication methods to MEC servers through wireless channels for task offloading. $M=\{M_1,M_2,\cdots,M_L\}$ represents the set of MEC servers, where $L$ is the number of edge servers. Each server is equipped with limited computing resources, which can be quantified by CPU cycles. The major notations used in the rest of the paper in Table 1. The specific task offloading and resource allocation process is as follows:

(1) The mobile device sends an offload request to a nearby connectable base station. After receiving the offload request, the base station obtains the task scheduling based on the latest completion time of each task.

(2) The mobile device can formulate task offloading and resource allocation strategies according to the CPU frequency and transmit power of the MEC server, computing capacity of nearby edge servers, the resources required by tasks and the current channel environment.

(3) The MEC server receives offloading and resource allocation policies, sets CPU frequency and transmission power. The tasks should be scheduled to be executed locally or offloaded to a designated server.

(4) The MEC server executes the task and sends the execution results to the mobile device.

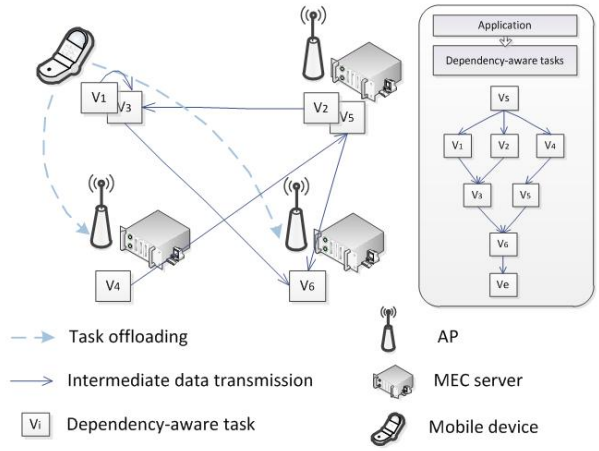



Fig. 1. The system of mobile edge computing.

Table 1. Notations,

| Notation | Description |
|---|---|
| $M=\{M_1,M_2,\cdots,M_L\}$ | MEC server |
| $V=\{V_s,V_1,\cdots,V_n,V_e\}$ | Task set |
| $V_s$ , $V_e$ | Virtual task to represent the beginning and the end of the entire task |
| $e_{V_i,V_j}$ | The dependency relationship between tasks |
| $S_i$ | The offloading strategy |
| $s_{V_i,M_k}$ | Flag used to indicate task $V_i$ being offloaded to MEC server or be executed locally |
| $d_{V_i}$ , $c_{V_i}$ , $l_{V_i}$ | Input data size of the task $V_i$ , CPU's calculation resources per bit for task processing and the CPU cycle required |
| $Rt_{V_i}^{local}$ , $Rt_{V_i,M_k}$ | Ready time for executing tasks locally and on MEC servers |
| $Ft_{V_i}^{local}$ , $Ft_{V_i,M_k}$ | Completion time of the task $V_i$ being executed locally and on edge servers |
| $R_{up}$ , $R_{down}$ | The transmission rate of data uploaded and downloading the results by mobile devices to the MEC server |
| $h_{up,M_k}$ , $h_{down,M_k}$ | Channel gain between the MEC server $M_k$ and the mobile device for uploading and downloading data |
| $\delta$ | Channel bandwidth |
| $P$ | Optional set of transmission power levels |
| $T_{V_i}^{local}$ | Task running time of local process |
| $u_{V_i}^{local}$ | The assignment of the CPU frequency for locally processed task $V_i$ |
| $E_{V_i}^{local}$ | Local energy cost of task $V_i$ |
| $\psi$ | Effective switching capacitance |
| $T_{V_i,V_j}^{down}$ | Download time of intermediate data |
| $prec(V_i)$ | Set of preceding tasks of task $V_i$ |
| $T_{V_i,V_j}^{up}$ , $E_{V_i,V_j}^{up}$ | Time cost required and energy consumption for offloading task to the MEC server $M_k$ |
| $T_{V_i,M_k}^{Mec}$ | Time required for the MEC server $M_k$ to perform task $V_i$ |
| $Rt_{V_i,M_k}$ , $Ft_{V_i,M_k}$ | Ready time and completion time for executing task $V_i$ on the edge server |
| $Ft_{V_i}^{Mec}$ , $E_{V_i}^{Mec}$ | Total completion time and energy cost of task $V_i$ |

## B. Task model

It is assumed that the application can be divided into multiple dependent tasks, and each task can only be offloaded to an adjacent MEC server. The task set corresponding to the application on a mobile device is $V=\{V_s,V_1,\cdots,V_n,V_e\}$. Among them, the two virtual tasks $V_s$ and $V_e$, indicate the beginning and end of the entire task, respectively. $e_{V_i,V_j}$ represents the dependency relationship between tasks, which indicate that a task must be completed before it can be executed. Multiple tasks with dependencies can be constructed as a Direct Acyclic Graph (DAG) $G=(V,E)$, where $e_{V_i,V_j}\in E$. In addition, $d_{V_i}$ is the input data size of the task $V_i$, $c_{V_i}$ represents the CPU's calculation resources per bit for task processing. $d_{V_i,V_j}$ represents the amount of data to be transferred between the previous task and subsequent tasks. Thus, the CPU cycle $l_{V_i}$ required by the task $V_i$ can be expressed as:

$$l_{V_i}=\begin{cases}0, \textit{ If } V_i=V_s \textit{ or } V_e\\ d_{V_i}*c_{V_i}, \textit{ Otherwise}\end{cases} \tag{1}$$

The offloading strategy of a task is represented by vector $S_i=\{s_{V_i,M_1},s_{V_i,M_2},\cdots,s_{V_i,M_L}\}$, and the elements $s_{V_i,M_k}\in\{0,1\}$. The value of $s_{V_i,M_k}$ is set to 1, which indicates the task $V_i$ being offloaded to MEC server $M_k$. The vector element of all 0 indicates that the task will be executed locally. Since the tasks can only be processed locally or by MEC server, the sum of the vectors can be given by

$$\sum_{k=1}^{L}s_{V_i,M_k}=\begin{cases}0, \textit{If task } V_i \textit{ is executed locally},\\ 1, \textit{Otherwise}\end{cases} \tag{2}$$

The task processing includes two phases: preparation and execution. The ready time that a task can prepare to enter the

execution process after receiving all necessary input data can be defined as the ready time of the task $V_i$. Let $Rt_{V_i}^{local}$ and $Rt_{V_i,M_k}$ represent the ready time for executing tasks locally and on MEC servers, respectively. Besides, $Ft_{V_i}^{local}$ and $Ft_{V_i,M_k}$ indicate the completion time of the task $V_i$ being executed locally and on edge servers, respectively.

### C. Communication model

The communication system employs the 5G technology with orthogonal channels to allocate bandwidth for terminal devices, and all devices will share the entire bandwidth. The mobile devices can offload tasks to the MEC server for execution, and the required intermediate data will be transferred to the corresponding server. The data transmission also will result in certain delays and energy consumption. At this time, the mobile devices can choose different transmission powers to reduce task delay and energy cost. Let $p_s$ be the transmission power allocated by mobile users, the transmission rate of data uploaded by mobile devices to the MEC server can be obtained by Shannon theory [22]:

$$R_{up}=\delta \log_2\left|1+\frac{p_s h_{up,M_k}}{\sigma^2}\right| \tag{3}$$

where $\delta$ is the channel bandwidth. $p_s \in P$, and $P$ is an optional set of transmission power levels. Besides, $h_{up,M_k}$ is the channel gain between the MEC server $M_k$ and the mobile device for uploading data, and $\sigma^2$ is the noise power.

The transmission rate of the mobile device for downloading the results from corresponding server can be expressed as:

$$R_{down}=\delta \log_2\left|1+\frac{p_r h_{down,M_k}}{\sigma^2}\right| \tag{4}$$

where $p_r$ is the received power of the mobile device and $h_{down,M_k}$ is the channel gain between the mobile device and the MEC server $M_k$ for downloading data.

Since the edge servers are linked to the backbone network, the data transmission between them is set to a highly reliable bandwidth condition. Additionally, data transfer delays between edge servers will be ignored.

## IV. Formulated Problems and Solutions

### A. Problem definition

#### 1) Local computing

For locally processed task, the mobile device will allocate an appropriate CPU frequency to the computing task for energy saving. $u^{local} \in U_0$, and $U_0$ is the upper limit of mobile device's computing power. By using Dynamic Voltage and Frequency Scaling (DVFS) technology, CPU processor resources can be dynamically allocated as needed. For a task, the mobile device can assign the CPU frequency from a discrete level. Given the CPU frequency of the device and the CPU cycle required for the task, the task running time of local process can be calculated as follows:

$$T_{V_i}^{local}=\frac{l_{V_i}}{u_{V_i}^{local}} \tag{5}$$

where $u_{V_i}^{local}$ is the assignment of the CPU frequency for locally processed task $V_i$.

In addition, the local energy cost of task $V_i$ can be expressed as:

$$E_{V_i}^{local}=\psi l_{V_i}\left(u_{V_i}^{local}\right)^2 \tag{6}$$

where $\psi$ is the effective switching capacitance based on the hardware chip.

If the predecessor task $V_j$ of the task $V_i$ is executed on MEC server $M_k$, intermediate data $d_{V_i,V_j}$ should be downloaded before local processing. Then, the download time of intermediate data can be expressed as:

$$T_{V_i,V_j}^{down}=\frac{d_{V_i,V_j}}{R_{down}} \tag{7}$$

For task $V_j$, $s_{V_i,M_k}=1$ means that the task will be executed on MEC server $M_k$ and the ready time of task $V_i$ can be calculated by the sum of the completion time and the waiting time for intermediate results to be transmitted to the mobile device. Otherwise, $s_{V_i,M_k}=0$ indicates that the mobile device does not choose MEC server $M_k$ for offloading task $V_j$, and the ready time is equal to the completion time of the task. Thus, the ready time for locally executed task $V_i$ can be expressed as:

$$Rt_{V_i}^{local}=\max_{V_j\in prec(V_i)}\left\{\left(1-s_{V_i,M_k}\right)Ft_{V_j}^{local}+s_{V_i,M_k}\left(Ft_{V_j}^{M_k}+T_{V_i,V_j}^{down}\right)\right\} \tag{8}$$

where $prec\left(V_i\right)$ denote the set of preceding tasks of task $V_i$.

Therefore, the completion time of task $V_i$ being executed locally can be expressed as:

$$Ft_{V_i}^{local}=Rt_{V_i}^{local}+T_{V_i}^{local} \tag{9}$$

#### 2) Edge computing

If the mobile device offloads the task $V_i$ to MEC server for execution, the task latency should include the time delay for transferring the required intermediate data from the local to the server and the time delay for the edge server to execute the task. The time cost and energy consumption required for offloading task to the MEC server $M_k$ can be given by:

$$T_{V_i,V_j}^{up}=\frac{d_{V_i,V_j}+d_{V_i}}{R_{up}} \tag{10}$$

$$E_{V_i,V_j}^{up}=T_{V_i,V_j}^{up}p_{send} \tag{11}$$

In addition, the time required for the MEC server $M_k$ to perform task $V_i$ can be expressed as:

$$T_{V_i,M_k}^{Mec}=\frac{l_{V_i}}{u_{V_i,M_k}} \tag{12}$$

where $u_{V_i,M_k}$ is the CPU resources required for task $V_i$ being reserved by MEC server.

Next, the ready time and completion time for executing task $V_i$ on the edge server can be calculated by:

$$Rt_{V_i,M_k} = \max_{V_j \in prec(V_i)} \left\{ \left(1 - s_{V_j,M_k}\right) \left( Ft_{V_j}^{local} + T_{V_i,V_j}^{up} \right) + s_{V_j,M_k} T_{V_j,M_k}^{Mec} \right\} \quad (13)$$

$$Ft_{V_i,M_k} = Rt_{V_i,M_k}^{Mec} + + T_{V_i,M_k}^{Mec} \quad (14)$$

Therefore, the total completion time and energy cost of task $V_i$ can be expressed as:

$$Ft_{V_i}^{Mec} = \left(1 - s_{V_i,M_k}\right) Ft_{V_i}^{local} + s_{V_i,M_k} Ft_{V_i,M_k} \quad (15)$$

$$E_{V_i}^{Mec} = \left(1 - s_{V_i,M_k}\right) E_{V_i}^{local} + s_{V_i,M_k} E_{V_i,V_j}^{up} \quad (16)$$

*3)* Optimization objectives

According to the analysis, the completion time of all tasks is $\sum_{i=1}^{n} FT_{V_i}$ . In addition, the total energy consumption of task processing can be expressed as:

$$E = \sum_{i=1}^{n} \sum_{k=1}^{L} \left(1 - s_{V_i,M_k}\right) E_{V_i}^{local} + s_{V_i,M_k} E_{V_i,V_j}^{up} \quad (17)$$

This study aims to minimize the total completion time of tasks and the total energy consumption of mobile device by dealing with dependent task offloading and resource assignment. The optimization objectives can be defined as:

$$\min \ \theta E + (1-\theta) \sum_{i=1}^{n} Ft_{V_i} \quad (18)$$

$$s.t. \quad C1: Ft_{V_i} \le Delay(V_i),$$
$$C2: u_{V_i,M_k} < U_{M_k},$$
$$C3: u_{V_i}^{local} < U_0,$$
$$C4: s_{V_i,M_k} \in \{0,1\},$$
$$C5: 0 < \theta < 1.$$

Among them, $C1$ represents the delay tolerance that the task completion delay time must meet. $C2$ and $C3$ represent the computing power of edge servers and the CPU frequency limit of mobile devices, respectively. $C4$ indicates that the task can only be executed locally or offloaded to an edge server for execution. $C5$ is the proportion of factors of energy consumption or task completion time in objective function.

The problem includes the following optimization variables: offloading decision, MEC server's selection, and task scheduling. Obviously, the constraint $C4$ can be regarded as a multivariable integer programming problem, which is difficult to obtain a closed solution. For convenience of analysis, we can solve the problem by equivalently solving the sub problems: task scheduling, optimization of offload decisions and edge server's selection.

## B. Task scheduling

According to the previous derivation, the task being executed locally should require adequate CPU frequency of the mobile device in advance. Otherwise, it is necessary to select a suitable MEC server for task offloading, and meanwhile set appropriate transmission power to transmit the intermediate data required for the next task. Generally speaking, the CPU frequency of the mobile device is proportionate to the task execution time. Choosing a higher CPU frequency or transmit power can reduce latency but consume more energy. There will be a trade-off between the minimization of latency and energy consumption. According to analysis, energy consumption is positively correlated with the CPU frequency at which tasks are executed. Since the mobile device is usually energy constrained, DVFS technology can be applied to adjust the CPU frequency of mobile device to balance energy consumption and task latency. Based on the DAG of dependencies between the tasks, we use topology based sorting algorithm to obtain the solutions for task scheduling optimization.

Topological sorting can sort the vertices of a directed graph. It is concerned with the relationship of each vertex in the DAG graph rather than the position and distance of each vertex. The points with in-degree 0 should be chosen as the preferential scheduled tasks. After processing the tasks, the in-degree of all points associated with those points will be reduced by one, and then the remaining tasks can be rediscovered for processing. By analogy, the processing order of the tasks can be obtained. The specific algorithm of tasks processing order will be given as follows:

Algorithm 1: The processing order of the dependency-aware tasks.

Input: $G$, task delay constraint.

Output: tasks processing order.

1: initialize the set of tasks with dependency $Q$ and tasks processing queue $\Psi$;
2: the points with in-degree 0 are inserted into $Q$;
3: while $Q$
4: for each edge $e_{V_i,V_j}$
5: obtain the set of available processing nodes;
6: if $\sum_{k=1}^{L} s_{V_j,M_k} = 0$ then
7: calculate $T_{V_j}^{local}$ and $Rt_{V_i}^{local}$;
8 : else
9: calculate $Rt_{V_i,M_k}$, $Ft_{V_j,M_k}$ and $Ft_{V_i}^{Mec}$;
10: end if
11: if $Ft_{V_i} \le Delay(V_i)$
12: update the task scheduling queue $\Psi$;
13: end if
14: end for
15: delete the chosen point from $Q$;
16: end while

## C. Optimization of offload decisions

Particle Swarm Optimization (PSO) is a population based evolutionary method that originates from the study of the behavior of bird flocks during predation [23]. The basic idea is to utilize collaboration and information sharing among individuals in a group to seek the best solution. Different from other heuristic algorithms, PSO preserves the concepts of population and evolution, and obtains the fitness value as a criterion for evaluating the quality of optimization [24]. In the process of finding the global optimal solution, PSO can not only retain its own experience for self-learning, but also obtain experience from other particles for social learning [25]. PSO has good parallelism during optimization, and the algorithm has strong robustness and anti-interference ability. The optimization of offload decisions and edge server selection will

be solved by our proposed PSO algorithm. This study takes the latency and energy consumption generated by task execution on the MEC server as optimization objectives, and the fitness function is defined as follows:

$$f = \theta E + (1-\theta)\sum_{i=1}^{n} Ft_{V_i} \tag{19}$$

$N$ particles generates randomly in a 1-dimensional search space to form an original population. The original population is encoded according to the task's offloading strategy. Each particle contains two attributes: location and velocity. The location represents a solution in the search space, and the velocity represents the direction and speed at which the particle finds the optimal task processing strategy in the search space. The optimal location searched by the $i$ -th particle during the optimization process is denoted as $pbest_{iL}$ , and the optimal location searched by the particle swarm during the global search process is denoted as $gbest_L$ . Thus, the process for updating the position and velocity of the $i$ -th particle can be expressed as:

$$\begin{aligned} v_{iL}(t) = {} & w v_{iL}(t-1) + c_1 r_1 (pbest_{iL} - x_{iL}(t-1)) + \\ & c_2 r_2 (gbest_L - x_{iL}(t-1)) \end{aligned} \tag{20}$$

$$x_{iL}(t) = x_{iL}(t-1) + v_{iL}(t) \tag{21}$$

where $v_{iL}(t)$ indicates the velocity of $i$ -th particle during the iteration $t$ , and $x_{iL}(t)$ indicates the current position of the $i$ -th particle. $w$ is the inertia weight. Besides, $r_1$ and $r_2$ are random numbers between 0 and 1, $c_1$ and $c_2$ represent learning factors, respectively.

PSO tends to fall into local optimizations. Larger value of inertia weight is conducive to jumping out of the local solution and facilitating global search. Conversely, small inertia weight is beneficial for accurate local search of the current search area [26]. When the value of the inertia weight is constant, there is a lack of global search ability in the early stage of the iteration or lose of local search ability in the late stage of the iteration. Thus, it can easily cause the PSO method to fall into a local optimal solution or generate oscillations near the global optimal solution in the late stage. To balance global and local search capabilities, the inertia weight is designed to be adjusted dynamically as follows:

$$w_i = \begin{cases} w_{\max}, f_i > f_{\max} \\ w_{\max} - \dfrac{(w_{\max} - w_{\min})(f_{\max} - f_i)}{f_{\max} - f_{\min}}, f_{\min} \le f_i \le f_{\max} \\ w_{\min}, f_i < f_{\min} \end{cases} \tag{22}$$

where $f_i$ is the current fitness value of particle $i$ . $f_{\min}$ and $f_{\max}$ are the average fitness of the particles, whose fitness value is lower or greater than the mean value of the group, respectively. Besides, $w_{\min}$ and $w_{\max}$ are lower bound and upper bound of the inertia weight factor.

The specific process of the algorithm is as follows: First, the processing sequence can be obtained based on the task dependency relationship and algorithm 1. Secondly, initialize the requirement of task and resource of MEC servers, and determine whether the task processing sequence is empty. If it is not empty, offload the tasks in sequence to the MEC server with the lowest total cost. The improved PSO algorithm is used to solve the problem, updating the task completion time, total energy consumption, and the remaining resources of the MEC servers. The offloaded task can be deleted from the queue, and the cycle continues until the optimal solution is obtained. The details of the PSO algorithm are presented as follows:

Algorithm 2: The optimal offloading scheme of the dependency-aware tasks.

Input: The task set $V$ , the set of MEC servers $M$ , the task scheduling queue $\Psi$ .

Output: Minimum fitness function value.

1: initialize all particles;
2: while ( $t < T$ )
3: for each particle $i$
4: initialize the velocity $v_{iL}(t)$ and position $x_{iL}(t)$ ;
5: evaluate particle $i$ and set $pbest_{iL} = x_{iL}(t)$ ;
6: end for
7: for $i = 1$ to $N$
8: update the inertia weight using Eq. (22);
9: update the velocities of all particles using Eq. (20);
10: update the positions of all particles using Eq. (21);
11: evaluate particle $i$ ;
12: if $f(x_{iL}) < f(pbest_{iL})$ then
13: $pbest_{iL} = x_{iL}$
14: end if
15: if $f(pbest_{iL}) < f(gbest_L)$ then
16: $gbest_L = pbest_{iL}$
17: end if
18: end for
19: end while

The time complexity analysis of proposed algorithm is as follows: the number of tasks is $n$ , the number of edge servers is $L$ , the number of predecessor tasks for each task is $p$ , the particle population size is equal to $N$ and the maximum number of iterations is set to $T$ . The offloading order of the tasks can be solved from lines 1 to 6 in above pseudo code, and the initialization of the algorithm will be with a time complexity of $o(n^2)$ . Then, from lines 7 to 14, the offloading can be executed in the order of the tasks in optimal queue. During the offloading process, PSO algorithm needs to be used to solve the total completion time of tasks and the total energy consumption of mobile device by unloading with dependent task offloading and resource assignment. Since the PSO algorithm is one-dimensional, the time complexity is $o(N \times T + N)$ . Therefore, the time complexity of lines 7 to 14 in pseudo code is $o(nlp(N+1)T)$ . Therefore, the total time complexity of the algorithm will be $o(n^2 + nlp(N+1)T)$ .

## V. Experimental results

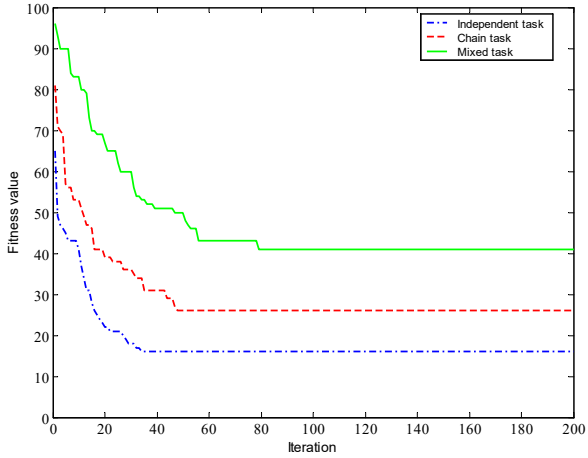
In this section, some numerical results are presented to evaluate the effectiveness of DTO-JODE. In the network scenario, the application can be divided into several tasks. According to the dependencies between tasks, the DAGs can be generated. The Google dataset [27] is used to simulate the

This article has been accepted for publication in IEEE Transactions on Consumer Electronics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCE.2023.3338620

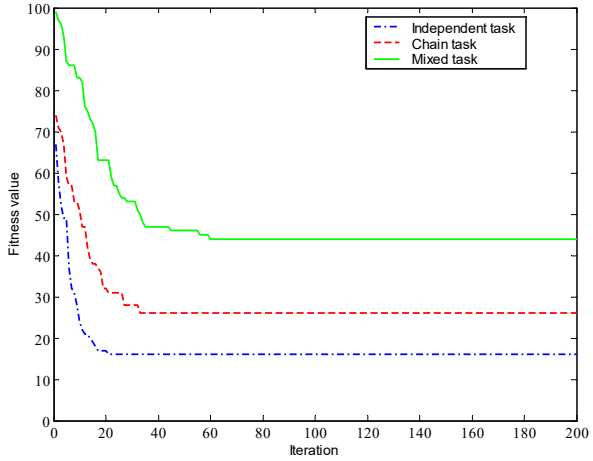


processing time and processing resources required for each task on the MEC servers. Two types of dependency-aware task are mainly considered: chain task and mixed task. For chain task, the relationship of dependency refers to that each task can have at most one successor task. For mixed task, any task can have multiple predecessors or successors. The number of MEC servers for caching a certain service is set to 50% of the total number of edge servers. The channel bandwidth and background noise environment between MEC servers are set according to [28], and the downlink channel $h_{down,M_k}$ is related to the uplink channel $h_{up,M_k}$ with a correlation coefficient of 0.7. Other specific parameter values are set as follows: $p_s = 0.1$W, $p_r = 0.05$W, $U^{local} = 500$MHz, $U_{M_k} = 3000$MHz, $\delta = 1$ GHz, $\sigma^2 = 10^{-10}$, $w_{\max} = 0.9$, $w_{\min} = 0.4$, $c_1 = c_2 = 2$.

Firstly, we analyze the convergence of the algorithm. The convergence under different task dependencies with different inertia weight values is shown in Figure 2. As can be observed from the experimental results, the independent tasks, chain tasks and mixed tasks will converge in the 37, 48, and 77 rounds with respect of $w = 0.5$, respectively. However, when the inertia weight is designed to be adjusted dynamically and $w \in [0.4, 0.9]$, the convergence speed of different types of tasks is better than fixed inertia weights, the convergence of proposed algorithm demonstrates a very significant improvement. Among them, independent tasks converge to 21 rounds, chain tasks converge to 34 rounds, and mixed tasks converge to 58 rounds. From the characteristics of PSO, the inertia weight value will affect the convergence speed of the algorithm. Figure 3 shows the convergence under different task dependencies for different population sizes. A larger group size can expand the search space, but the computational complexity also increases. From the experimental results, it can be seen that increasing the initial population can improve the convergence of the algorithm. However, to some extent, an excessively large population size can also affect the efficiency of the algorithm, as it is more sensitive to the complexity of the problem. The experimental results also verify that dynamically adjustments of the inertia weight value can significantly accelerate the convergence speed of the algorithm.

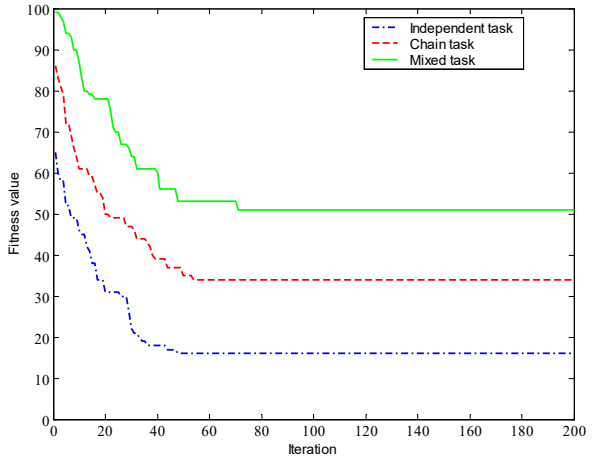



a) $w = 0.5$

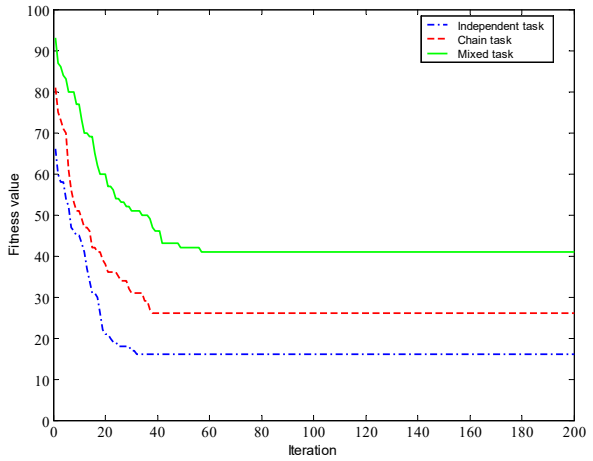



b) $w \in [0.4, 0.9]$

Fig. 2. Convergence under different task dependencies with different inertia weights.




a) N=10




b) N=50

Fig. 3. Convergence under different task dependencies for different population sizes.

Next, the impact of different weighting factors on task completion time and energy consumption is evaluated. Figures 4 and Figures 5 show the impact of weighting factors on task response time and energy consumption respectively. Based on

the previous theoretical analysis, that the larger of $\theta$ indicate the more attention mobile devices will pay to the energy consumption of task processing. At this point, the goal of the offloading strategy tends to optimize energy consumption. From the experimental results, when $\theta=0.8$, the energy consumption of task processing is smaller compared to other situations, and task response time is relatively high. While $\theta=0.2$, the goal of the offloading strategy prefers to optimize task processing time. At that time, the mobile device will consume higher energy consumption for task local processing. Overall, when $\theta$ is set to 0.6 or 0.4, the total cost of task processing energy consumption and time delay is relatively more balanced. In subsequent experiments, the weight factor of the method will be set to 0.6. In practical application scenarios, when the device has sufficient energy, mobile devices pay more attention to shortening the response time of the task. Conversely, while the energy is insufficient, mobile devices will pay more attention to minimizing energy consumption.

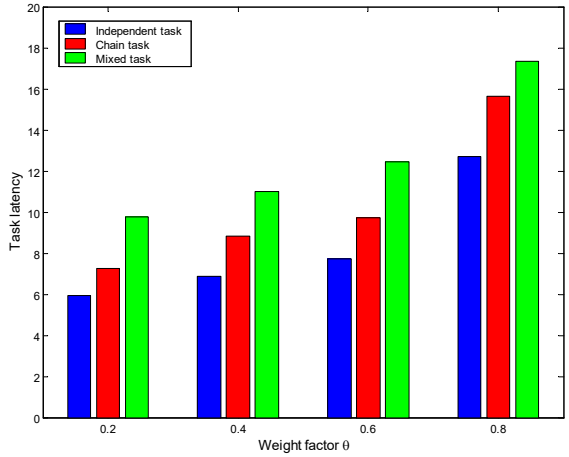



Fig. 4. Impact of weight factor on task latency.

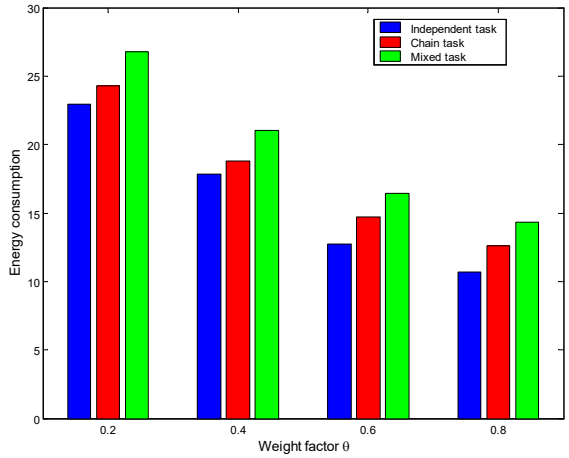
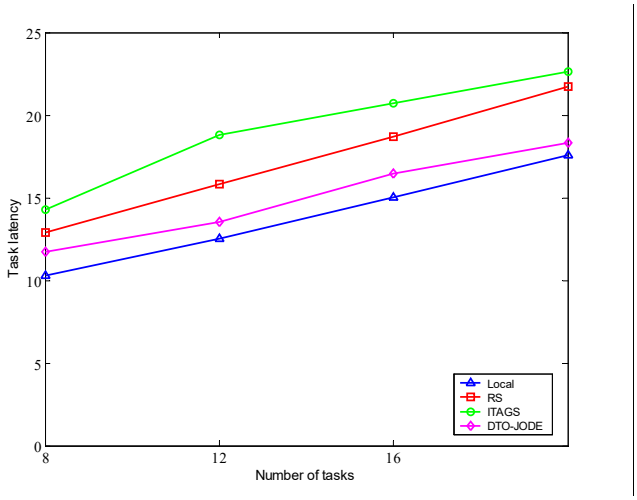
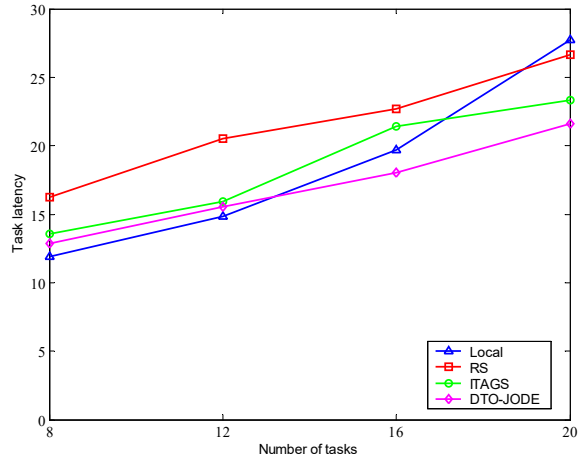



Fig. 5. Impact of weight factor on task energy consumption.

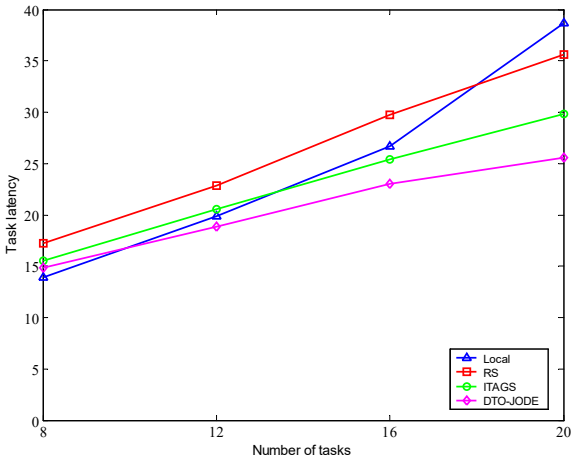
To evaluate the performance of the proposed method, several typical algorithms are compared. Among them, the local execution algorithm is to execute all tasks on the mobile device, and the Random Scheduling (RS) determines scheduling decisions for all tasks randomly. In the experiment, the number of tasks is set to 8 to 20, and the node density of the DAG graph is 0.5. Figure 6 illustrates the comparison of task offloading delays among various algorithms in different types of tasks. For independent tasks, local execution algorithm achieves the best results. This is due to the transmission time overhead of no task uploads or downloads, but on the other hand, local devices will consume too much energy for task execution. Local execution algorithm does not take into account of task size and dependencies between tasks. All tasks are executed locally, while mobile device's computing resources are limited, it will result in excessive task waiting time. On the one side, the tasks can eliminate the transmission time of intermediate data between tasks in local execution algorithm. When there are dependencies between tasks, DTO-JODE performs significantly better than other methods. This also indicates that as the dependency relationships between tasks become complex, the execution of one task depends on the execution results of other tasks, which may lead to an increase in task waiting time. Compared with ITAGS, DTO-JODE reduces task latency by about 13.5% to 17.8%. The reason is that DTO-JODE fully considers the dependency relationship between tasks and prioritizes offloading the task with the optimal path to the end task, which can reduce the waiting delay of the task at the MEC server.




a) Independent tasks

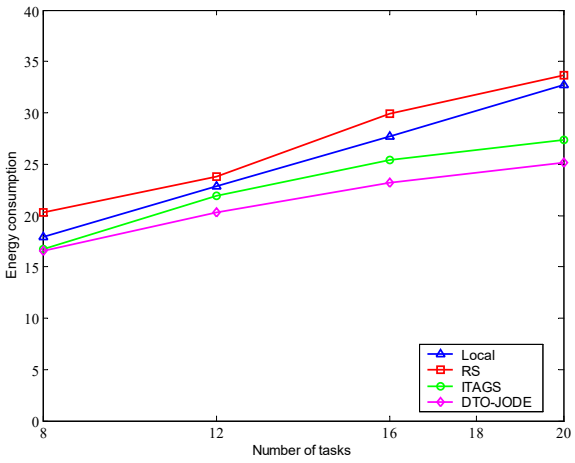



b) Chain tasks

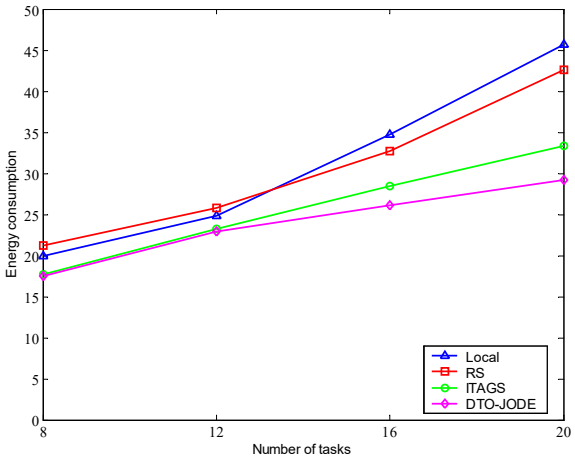



c) Mixed tasks

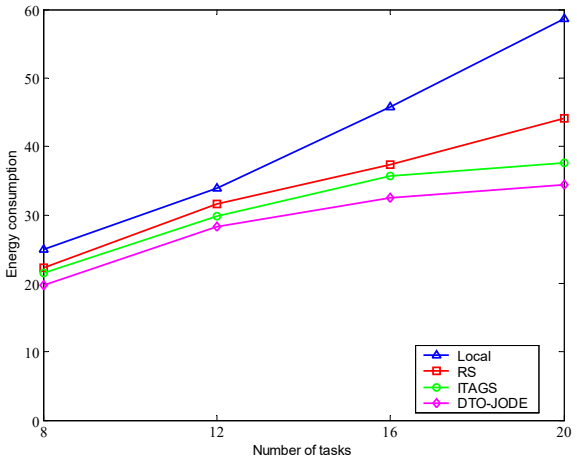
Figure 6. Task latency of different algorithms.




a) Independent tasks




b) Chain tasks




c) Mixed tasks

Figure 7. Energy consumption of different algorithms.

Figure 7 shows the comparison of energy consumption among various algorithms in different types of tasks. Local execution algorithm makes the worst performance in aspect of the energy cost, and there is a significant increase as the number of tasks increases and the complexity of task dependencies increases. RR obtains the lowest energy consumption among various algorithms in term of independent task. Since the task offloads to available MEC servers for execution, it can spend more communication time and consume a certain amount of the transmission energy of mobile devices. However, in the processing of the chain task and mixed task, random selection of RR cannot adapt well to the dependency relationship of the task and the resource allocation. Compared to local execution algorithm, the energy consumption of DTO-JODE is reduced by about 37% to 66%, Compared to ITAGS, it has decreased by approximately 19% to 59%. Especially in mixed task scenarios, the performance of the DTO-JODE method is more prominent. The dependency relationships between tasks can result in frequent data transmission between mobile devices and MEC servers, valid task scheduling decision can reduce energy consumption efficiently. By arranging the tasks appropriately, DTO-JODE takes the latency and energy consumption generated by task execution on the MEC server as optimization objectives. Thereby, it can achieve joint optimization of delay and energy consumption for dependency-aware tasks effectively.

In addition, the performance of our method is verified under the condition of changing the density of the DAG. Figure 8 and Figure 9 show the comparison in terms of task delay and energy consumption respectively. The number of tasks is set to 20, and the range of density variation in DAG is between 0.3 and 0.8. From the results, it can be observed that as the density of the graph increases, there is little change in local execution and RS methods. That is because the execution mode of each task in those two methods is fixed and not affected by changes in graph density. However, ITAGS is more significantly affected by changes in graph density. It demonstrates that the individual time allocation with greedy scheduling will discard a large amount of DAG graph structure information, resulting in inaccurate offloading decisions. In contrast, DTO-JODE method can maintain effective offloading scheduling performance under various graph density conditions.

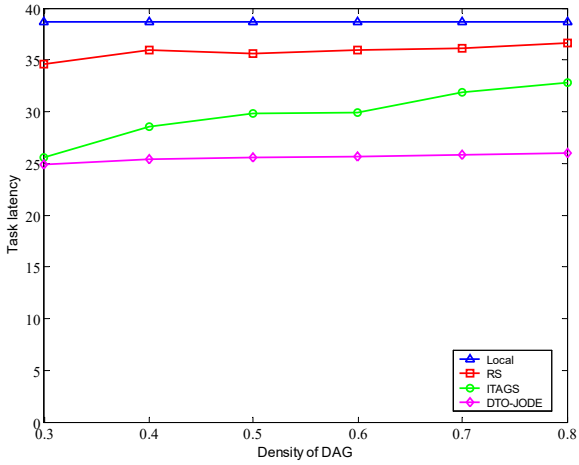



Figure 8. Task latency of different algorithms.

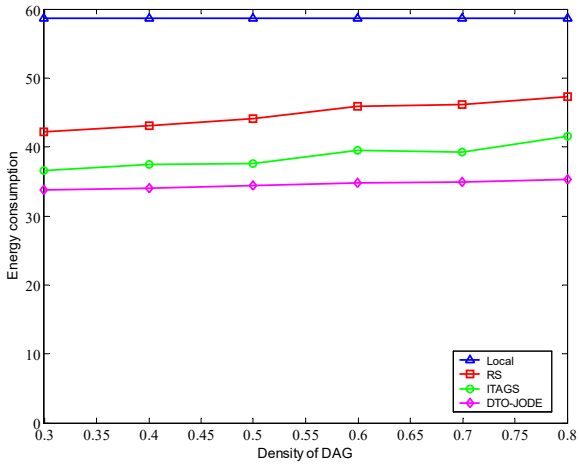



Figure 9. Energy consumption of different algorithms.

## VI. Conclusions

In this paper, we investigated the dynamic decisions and resource allocation for task offloading to minimize total computing time and energy consumption. To obtain the tradeoff between the latency and energy consumption under the constraints of limited computing resources, this paper proposes a dependency-aware task offloading for joint optimization of energy consumption and delay of task processing under the constraints of limited computing resources in MEC servers. First, a dependent task offloading model is established to minimize the energy consumption and task processing of mobile device. Secondly, we analyze the dependencies between tasks to construct a DAG, and propose an algorithm based on topological ordering to obtain a possible solution for task scheduling. Subsequently, to minimize the total cost, an improved PSO algorithm is used to obtain the optimal task offloading decision and MEC server selection optimization. Experimental results demonstrate that the proposed strategy can reduce the time cost and energy consumption compared to existing typical methods for tasks with different dependencies effectively.

This paper only investigates the offloading of a single application, and there exists collaborative offloading of multiple applications in practical scenarios. Therefore, our future work will extended to the multiuser case, investigate collaborative computation offloading with complex dependencies and experimentally verify the proposed strategy in real applications.

## Acknowledgment

This study was founded by Sichuan Science and Technology Program No. 23ZDYF0738, No. 2022YFG0343, and the National Natural Science Foundation of China with Grant 51975107 and Sichuan Science and Technology Major Project No. 2022ZDZX0039, No.2019ZDZX0020.

## References

[1] Khan W, Ahmed E, Hakak S, et al. “Edge computing: A survey,” Future Generation Computer Systems, 2019, 97: 219-235.

[2] Ma Hua-rong, Huang Peng, Zhou Zhi, et al. “GreenEdge:joint green energy scheduling and dynamic task offloading in multi-tier edge computing systems,” IEEE Transactions on Vehicular Technology, 2022, 71(4): 4322-4335.

[3] Parekh B, Amin K. “Edge intelligence: A robust reinforcement of edge computing and artificial intelligence,” In: Proc. of Innovations in Information and Communication Technologies (IICT-2020). Berlin: Springer, 2021, pp. 461-468.

[4] Nguyen D, Ding Ming, Pham Q, et al. “Federated learning meets blockchain in edge computing: Opportunities and challenges,” IEEE Internet of Things Journal, 2021, 8(1): 12806-12825.

[5] Ndikumana A, Tran N, Kim K, et al. “Deep learning based caching for self-driving cars in multi-access edge computing,” IEEE Transactions on Intelligent Transportation Systems, 2020, 22(5): 2862-2877.

[6] Munir A, Blasch E, Kwon J, et al. “Artificial intelligence and data fusion at the edge,” IEEE Aerospace and Electronic Systems Magazine, 2021, 36(7): 62-78.

[7] Khan L U, Yaqoob I, Tran N H, et al. “Edge-computing-enabled smart cities: A comprehensive survey,” IEEE Internet of Things Journal, 2020, 7(10): 10200-10232.

[8] Bahreini T, Badri H, Grosu D. “Mechanisms for resource allocation and pricing in mobile edge computing systems,” IEEE Transactions on Parallel and Distributed Systems, 2021, 33(3): 667-682.

[9] Ranadheera S, Maghsudi S, Hossain E. “Computation offloading and activation of mobile edge computing servers: a minority game,” IEEE Wireless Communications Letters, 2018, 7(5): 688-691.

[10] Liu L, Zhou Y Q, Garcia V, et al. “Load aware joint CoMP clustering and inter-cell resource scheduling in heterogeneous ultra-dense cellular networks,” IEEE Transactions on Vehicular Technology, 2018, 67(3): 2741-2755.

[11] Huang J W, Lan Y H, Xu M F. “A simulation-based approach of QoS-aware service selection in mobile edge computing,” Wireless Communications and Mobile Computing, 2018, 2018: 1-10.

[12] Pang S C, Wang S Y. “Joint wireless source management and task offloading in ultra-dense network,” IEEE Access, 2020, 8: 52917-52926.

[13] Zhang J, Hu X P, Ning Z L, et al. “Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks,” IEEE Internet of Things Journal, 2018, 5(4): 2633-2645.

[14] Wu D P, Yan J J, Wang H G, et al. “User-centric edge sharing mechanism in software-defined ultra-dense networks,” IEEE Journal on Selected Areas in Communications, 2020, 38(7): 1531-1541.

[15] Tran T, Pompili D. “Adaptive bitrate video caching and processing in mobile-edge computing networks,” IEEE Transactions on Mobile Computing, 2018, 18(9): 1965-1978.

[16] Wang Jiadai, Zhao Lei, Liu Jiajia, et al. “Smart resource allocation for mobile edge computing: A deep reinforcement learning approach,” IEEE Transactions on Emerging Topics in Computing, 2019, 9(3): 1529-1541.

[17] He Ying, Wang Yuhang, Qiu Chao, et al. “Blockchain-based edge computing resource allocation in IoT: A deep reinforcement learning approach,” IEEE Internet of Things Journal, 2020, 8(4): 2226-2237.

[18] Dai Yueyue, Zhang Ke, Maharjan S, et al. "Edge intelligence for energy-efficient computation offloading and resource allocation in 5G beyond," IEEE Transactions on Vehicular Technology, 2020, 69(10): 12175-12186.

[19] Lin Zehong, Bi Suzhi, Zhang Yingjun. "Optimizing AI service placement and resource allocation in mobile edge intelligence systems," IEEE Transactions on Wireless Communications, 2021, 20(11): 7257-7271.

[20] Shen Tao, Gao Chan, Xu Dawei. "The analysis of intelligent real-time image recognition technology based on mobile edge computing and deep learning," Journal of Real-Time Image Processing, 2021, 18(4): 1157-1166.

[21] Sundar S, Liang Ben. "Offloading dependent tasks with communication delay and deadline constraint," In: Proc of the 37th IEEE Conf on Computer Communications, Piscataway, NJ: IEEE, 2018, pp. 37-45.

[22] Shiang H P, Schaar M V D. "Queuing-Based Dynamic Channel Selection for Heterogeneous Multimedia Applications Over Cognitive Radio Networks," IEEE Transactions on Multimedia, 2008, 10(5): 896-909.

[23] Kennedy J, Eberhart R. "Particle Swarm Optimization," In: Proc. of IEEE International Conference on Neural Network, 1995, pp. 1942-1948.

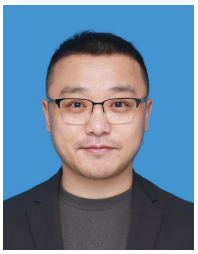
[24] Hsieh S T, Sun T Y, Liu C C, Tsai S J, "Efficient population utilization strategy for particle swarm optimizer," IEEE Trans Syst Man Cybern Part B Cybern, 2009, 39 (2): 444–456.

[25] Nickabadi A, Ebadzadeh MM, Safabakhsh R, "A novel particle swarm optimization algorithm with adaptive inertia weight," Appl Soft Comput, 2011, 11 (4): 3658–3670.

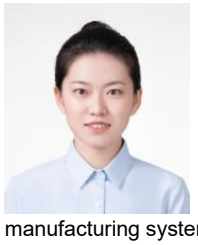
[26] Zhang W, Ma D, Wei J J, Liang H F, "A parameter selection strategy for particle swarm optimization based on particle positions," Expert Syst Appl, 2014, 41(7): 3576–3584.

[27] Zhang Jiao, Hu Xiping, Ning Zhaolong, et al. "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," IEEE Internet of Things Journal, 2017, 5(4): 2633-2645.

[28] Mazouzi H, Achir N, Boussetta K. "Elastic offloading of multitasking applications to mobile edge computing," In: Proc of the 22nd Int Conf on Modeling, Analysis and Simulation of Wireless and Mobile Systems, New York: ACM, 2019, pp. 307-314.



**Chen Xu** received the B.S. degree in natural polymer materials and engineering from Tianjin University of Science and Technology, Tianjin, China, in 2007, the M.S. degree in transportation planning and management from Lanzhou Jiaotong University, Lanzhou, China, in 2010, and the Ph.D. degree in traffic information engineering and control from Tongji University, Shanghai, China, in 2021. From 2010 to 2013, he was an Engineer with the Tongji Architectural Design(Group) Co., Ltd., Shanghai, China. From 2014 to 2021, he worked as a lecturer and master student supervisor at the School of Computer Science and Information Engineering, Shanghai Institute of Technology, Shanghai, China. Since 2022, he has been a post-doctoral with the School of Mechanical and Electrical Engineering, University of Electronic Science and Technology of China, Chengdu, China. His research interests include sensor network, underwater unmanned system, UUV database, intelligent computing.



**Mengzhuo Lv** received the B.S. degree in mechanical and electronic engineering from Beijing Institute of Technology, Beijing, China, in 2020.She is currently a Master's degree student with the School of Mechanical and Electronic Engineering,University of Electronic Science and Technology of China, Chengdu, China. Her research interests mainly include intelligent manufacturing systems, robotics, and its applications.

**Kun Zhang** received the B.S. degree in electronic and information engineering from Hainan University, Haikou, China, in 2018. He is currently working toward the Ph.D. degree in mechanical engineering with the Mechanical Engineering of the University of Electronic Science and Technology of China, Chengdu, China. His research interests include intelligent manufacturing systems, robotics, and its applications.

**Kui Cao** received the B.S. degree in electrical engineering and the automatization specialty from Aeronautical University, Yantai, China, in 2016. He is currently working toward the M.S. degree in mechanical engineering with the University of Electronic Science and Technology of China, Chengdu, China. His research interests include intelligent manufacturing systems, robotics, and its applications.

**Gang Wang** received the B.E. degree in Communication Engineering and the Ph.D. degree in Biomedical Engineering from University of Electronic Science and Technology of China, Chengdu, China, in 1999 and 2008, respectively. In 2009, he joined the School of Information and Communication Engineering, University of Electronic Science and Technology of China, China, where he is currently an Associate Professor. His current research interests include signal processing and intelligent systems. Since 2017, she has been an Assistant Professor with the School of Aeronautics and Astronautics, University of Electronic Science and Technology of China, Chengdu, China. Her research interests include multi robot cooperation, weak target detection and tracking from complex environment.

**Mingzhu Wei** received the B.S. degree in instrument science and technology from Yanshan University, Qinhuangdao, China, in 2006, the M.S. degree in instrument science and technology from the Harbin Institute of Technology, Harbin, China, in 2008, and the Ph.D. degree in information and system engineering from Politecnico di Torino, Turin, Italy, in 2012. From 2012 to 2015, She was an Engineer with the China Electronics Technology Group Corporation No.38 Research Institute, Hefei, China. Since 2017, she has been an Assistant Professor with the School of Aeronautics and Astronautics, University of Electronic Science and Technology of China, Chengdu, China. Her research interests include multi robot cooperation, weak target detection and tracking from complex environment.

**Bei Peng** received the B.S. degree in mechanical engineering from Beihang University, Beijing, China, in 1999, and the M.S. and Ph.D. degrees in mechanical engineering from Northwestern University, Evanston, IL, USA, in 2003 and 2008, respectively. He is currently a Full Professor of Mechanical Engineering with the University of Electronic Science and Technology of China,

Chengdu, China. He holds 30 authorized patents. He has served as a PI or a CoPI for more than ten research projects, including the National Science Foundation of China. His research interests mainly include intelligent manufacturing systems, robotics, and its applications.