Resource Scheduling and Energy Consumption Optimization Based on Lyapunov Optimization in Fog Computing

Chenbin Huang, Hui Wang *, Lingguo Zeng and Ting Li

School of Mathematics and Computer Science, Zhejiang Normal University, Jinhua 321000, China; rychen1@163.com (C.H.); jk63@zjnu.cn (L.Z.); lianz_lit@zjnu.edu.cn (T.L.)
* Correspondence: hwang@zjnu.cn

Abstract: Delay-sensitive tasks account for an increasing proportion of all tasks on the Internet of Things (IoT). How to solve such problems has become a hot research topic. Delay-sensitive tasks scenarios include intelligent vehicles, unmanned aerial vehicles, industrial IoT, intelligent transportation, etc. More and more scenarios have delay requirements for tasks and simply reducing the delay of tasks is not enough. However, speeding up the processing speed of a task means increasing energy consumption, so we try to find a way to complete tasks on time with the lowest energy consumption. Hence, we propose a heuristic particle swarm optimization (PSO) algorithm based on a Lyapunov framework (LPSO). Since task duration and queue stability are guaranteed, a balance is achieved between the computational energy consumption of the IoT nodes, the transmission energy consumption and the fog node computing energy consumption, so that tasks can be completed with minimum energy consumption. Compared with the original PSO algorithm and the greedy algorithm, the performance of our LPSO algorithm is significantly improved.

Keywords: IoT (Internet of Things); edge computing; fog computing; Lyapunov optimization

1. Introduction

With the evolution of technology, people’s lives are surrounded by a greater number of IoT devices. Data in IoT are also growing exponentially. If all this traffic is handed over to the cloud, the computing pressure on the cloud will be enormous [1]. Therefore, fog computing has arisen, first, to use the idle resources of IoT devices, second, to reduce the flow of cloud center network, and third, to reduce the delay of tasks. So far, fog network has been widely applied, including to the Internet of vehicles, smart buildings, smart cities, and industrial automation. A large number of application scenarios produce a large number of task requirements [2–4]. Among them, delay-sensitive tasks pose a great challenge to the computing capability of nodes. If the task cannot be completed within the specified time, it will be meaningless even if the task is completed. Therefore, the task needs to be offloaded into the fog network, and the nodes in the fog network can help the calculations [5].

The emergence of fog computing provides a solution to this kind of problem. Especially for delay-sensitive tasks, it can meet the needs of applications with high requirements on computing resources and delay, such as AR and online games. These are requirements that the cloud cannot solve, so we use fog networking to help cloud computing. A typical fog computing system consists of a set of fog nodes at different locations. These nodes are deployed on the periphery of the network and have flexible resource configurations such as storage, computing, and network bandwidth [6]. If the cloud is far from the edge node, the link connected to the cloud may not be reliable, which also causes a lot of delay. Therefore, in many cases, it does not make sense to put edge tasks in the cloud [7]. Hence, fog computing maximizes efficiency by offloading and distributing the workload of terminal equipment and fog nodes, while minimizing energy consumption and ensuring delay.

Generally speaking, the energy consumption required for processing a task is the sum of the transmission time, transmission power, and processing capability of computing
devices. In addition, the transmitted power combines channel state, such as the available bandwidth and data transfer rate. Thus, higher transmission rates can be achieved by increasing transmission power, which also reduces transmission time, but results in a greater energy consumption. Therefore, energy emissivity can be controlled by balancing transmission power and transmission time to achieve the lowest energy consumption while meeting the task delay requirement. The optimal task scheduling strategy should always control this balance and improve the transmission rate when the task delay requirement is high or reduce the transmission rate when the task delay requirement is low to achieve the energy-saving purpose [8].

In order to solve the above problems effectively, we use a Lyapunov optimization method and Lyapunov drift theory. The reason for using this method is that constraint optimization of the mean can be achieved in a stochastic system without the need to estimate random events that will occur in the future. The method first lists the constraints in the system and then builds a virtual queue. To optimize the objective function in each time slot, a drift-plus-penalty weight parameter is introduced. The reason for adding weights is to use this parameter to control the balance between drift and optimization.

Although Lyapunov optimization function can guarantee the stability of virtual queue, this method is not suitable for the bandwidth limitation and computing resource limitation of task offloading and processing in a fog network, because it has to be adapted to meet our needs. Therefore, we designed the PSO algorithm based on a Lyapunov optimization framework.

Our contributions to this paper are as follows:

1. The concept of pro-node is proposed and our proposed LPSO algorithm is used to optimize it.
2. A virtual queue is designed, and the optimization objective function is converted into the utility function under the Lyapunov framework. In addition, the balance between transmitting power and computing resource allocation can be achieved while ensuring queue stability and delay.
3. We evaluate our proposed strategy by comparing it with different algorithms. In the case of tasks with delay constraints, our proposed method performs better in various QoS parameters, including energy consumption, computing resource, and channel output coefficient.

The following parts of this paper are organized as follows: Section 2 introduces related work. Section 3 introduces our system model and the process of problem formulation. Section 4 mainly analyzes the problem, transforms it into a Lyapunov optimization problem and calculates its upper bound. Section 5 shows the algorithm we designed. Section 6 mainly compares the performance of LPSO with other algorithms. Finally, Section 7 is our conclusion and some expectations for future work.

2. Related Work

In recent years, many people have been interested in fog computing and edge computing and have also done a lot of related research. Ramadan et al. [9] proposed an energy-saving framework for fog networks based on multireceiver wireless sensor networks. They used two greedy algorithms to solve the multireceiver deployment problem of Harris hawks optimizer (Hho) and modified Harris hawks optimizer (Mhho), introducing the connection problem of wireless sensor networks (WSN) with fog network through a set of sink nodes (SNs). Arshed et al. proposed a resource-aware scheduler to assign incoming application modules to fog devices by placing application modules from the fog layer to the cloud instead of sending them directly to the cloud. Their placement strategy reduced the cost of using cloud resources with minimal execution time and minimal bandwidth usage [10]. Lin et al. [11] proposed a mathematical model for the multiterminal computing offloading problem, optimized the long-term average response delay objective under the constraints of long-term computing and energy consumption, and used the perturbation technique of the Lyapunov optimization method to convert the original problem into a
deterministic cap problem. Moreover, a distributed optimization algorithm was used to solve the upper bound problem per time slot, where each IoT device exchanges a limited amount of information with its associated edge nodes and decides on its own for resource allocation and computing offloading.

Liao et al. [12] used Directed Acyclic Graph (DAG) technology in the edge computing scheme to model the scheduling problem of application-dependent mobile tasks in the application of the allocation and scheduling of complex application-dependent applications and proposed a priority quantification method for application-dependent applications, considering the data volume, effort, deadlines, and dependencies. The estimated priorities were used as heuristics for task assignment and scheduling, effectively solving the NP-hard joint optimization problem.

Wang et al. [13] proposed a computing offloading architecture consisting of multiple fog nodes with heterogeneous communication and computational capabilities and a cloud center. They model dynamic communication and computing resources as two dynamic queues for recording resource availability and improving resource utilization. The whole problem was decomposed into channel power control and task assignment problems. A modified branch-and-bound method was used to solve the task assignment problem without considering resource availability. Then, they checked the resource availability to refine it with the knapsack problem.

Ren et al. [14] optimized virtual placeholders for all queues in a distributed fashion to create sufficient queue variance and activated Lyapunov optimizations under an infinite buffer setting. They proved that the optimal virtual placeholder was a new three-layer shortest-path problem and solved it by the Bellman–Ford algorithm.

Chang et al. [15] proposed a dynamic optimization scheme for an IoT fog computing system with multiple Mobile Devices (MDs) and utilized Lyapunov optimization to design an online dynamic algorithm to solve the joint optimization problem between a mobile device with harvesting energy and a fog node. Specifically, with the objective to minimize the system overhead related to delay, energy consumption, they proposed a joint computation offloading and channel resource allocation algorithm based on Lyapunov optimization, through minimizing the derived upper bound of the Lyapunov drift-plus-penalty function and dividing the main problem into several subproblems at each time slot and addressing them accordingly.

Mukherjee et al. [16] studied the optimum scheduling policy of the two-queue system in a fog node that allows a higher number of deadline-aware offloaded tasks to be processed while making both queues stable. The proposed approach was decoupled into two strategies: the priority-aware scheduling policy applying the Lyapunov drift-plus-penalty function and the fog nodes collaboration based on each fog node’s queueing status under the scheduling policy. The authors could effectively solve the problem of task completion rate.

Zhang et al. [17] studied resource scheduling for delay minimization in multiserver cellular edge computing systems. The traditional method defines queue-length-based Lyapunov functions and designs scheduling algorithms which solve the corresponding queue length control problem.

The experiments of Ren et al. were more ideal, as the buffer was not infinite, and they also reduced the delay but did not take into account the optimization of delay-sensitive tasks and the grade of completion of delay-sensitive tasks. In the research of Chang et al., only the energy consumption was optimized without considering the limit of task delay. Mukherjee et al. did not consider the energy consumption. In this paper, we study the combination of channel power and computing resource issues in edge computing networks, where multiple end-users (IoT devices) are considered to generate variable traffic loads. The goal is to minimize long-term system execution overhead when channel resources, computing resources, and task execution time are all limited. By extending the Lyapunov optimization, we transform the original problem into an upper-bound problem and design a scalable distributed algorithm. The symbols used in the paper are defined in Table 1.
Therefore, we adopt M/M/1 as the upload queue model for the tasks. In this section, we define the notations.

Table 1. Definitions of notations.

| Notations | Meanings |
|-----------|----------|
| $F_i$, $F_j$, $F_s$ | Computing power of devices $i$, $j$, $s$, respectively |
| $M$ | Set of IoT nodes |
| $N$ | Set of fog nodes |
| $S$ | Set of computing devices in pro-node |
| $F_{ij}(t)$ | Computing power allocated to $i$ by device $j$ in time slot $t$ |
| $F_{js}(t)$ | Computing power allocated to $i$ by device $s$ in time slot $t$ |
| $\tau_i$, $\tau_j$, $\tau_s$ | Node $i$, $j$, $s$, calculation delay |
| $\tau_c$ | Transmission delay |
| $\tau(t)$ | Delay at time slot $t$ |
| $\tau$ | Target delay vector |
| $P_c$ | Sent tasks |
| $\lambda_i(t)$ | Arrival rate of tasks for time slot $t$ |
| $\sigma$ | Device $i$ computes the standard deviation of the task size |
| $D_i$ | Mean value of task size for device $i$ |
| $C_j$ | Device $j$’s downstream bandwidth |
| $L_s$ | Local processing task delay |
| $P_a(t)$ | Total energy consumption of time slot $t$ |
| $P_i(t)$ | Energy consumed by fog node $i$ |
| $P_j(t)$ | Energy consumed by fog node $j$ |
| $P_s(t)$ | Energy consumed by fog node $s$ |
| $P_{ij}^c$ | Energy consumption of device $i$ to transfer tasks to $j$ |
| $P_j^c$ | Transmit power of device $i$ |
| $C_{ij}$ | Device $i$ to $j$ transfer rate |
| $\alpha$ | Offload rate |
| $p_i^b$ | Device $i$’s battery |

3. System Model

As shown in Figure 1, our system is modeled by a set $N = \{1, 2, \ldots, N_i\}$ consisting of $N$ IoT nodes, and a set $M = \{1, 2, \ldots, M_j\}$ consisting of $M$ fog nodes, and pro-nodes connected to the fog nodes. Tasks are randomly generated at each IoT node, and their arrivals are assumed to be independently and identically distributed. Then, in the long time case, the arrival of tasks can be considered as following the Poisson distribution. Therefore, we adopt M/M/1 as the upload queue model for the tasks. In this section, we propose a dynamic algorithm based on the Lyapunov optimization to solve the problem. The Lyapunov optimization is considered an efficient tool for designing online control algorithms without any prior knowledge [18–20]. However, since the power consumption is time-dependent, the decision sets in different time slots are not independently and identically distributed. Therefore, we advocate additional perturbation methods [19] to solve this problem.

Figure 1. System model diagram indicating whether the task that comes from the IoT device is processed locally or handed over to a pro-node for processing, mainly determined by the resources required by the task. Every pro-node has $S$ calculation subnodes.
3.1. Transmission Model

According to Shannon’s equation, the transmission rate of data transmitted from IoT nodes to fog nodes can be described as

\[ C_{i,j}(t) = W \log_2 \left( 1 + \frac{P_c(t)S_{i,j}(t)}{W N_0} \right) \] (1)

where \( P_c \) denote the transmission power, \( S_{i,j}(t) \) is the channel gain, \( W \) denotes the bandwidth, \( N_0 \) denotes the noise, and \( C_{i,j}(t) \) denotes the transmission rate from \( i \) to \( j \) at time \( t \). From the M/M/1 average delay formula [20], the uplink delay (including queuing and transmission time) can be given by the following equation

\[ \tau_c = \frac{1}{C_{i,j}(t)} - \alpha_i(t) \lambda_i(t) \] (2)

where \( \tau_c \) denotes the delay of transmission, \( \alpha \) is the ratio of offloading, and \( D_i \) denotes the average data size of task \( i \). Consider that if the total data size transmitted is constant and it is used to obtain the transmission rate, then it is not necessary to consider the length of an individual packet which does not affect the optimal decision and the optimal solution.

3.2. Energy Consumption Model

First is the energy consumption model for local processing, where the local task queue is in the M/M/1 model, and according to [20] the local computational delay \( \tau_i \) can be expressed as the following equation:

\[ \tau_i = \frac{\lambda_i(t)(\sigma^2 + D_i^2)}{2(\lambda_i(t)F_i - \lambda_i(t)F_iD_i) + D_i} \] (3)

\( \lambda_i \) denotes the arrival rate of task \( i \), \( \sigma \) denotes the standard deviation of the task size, \( F_i \) denotes the compute capability of \( i \), and \( D_i \) means the value of the task size for device \( i \).

Therefore, the total energy consumed, \( P_{l,i} \), by performing its tasks is

\[ P_{l,i} = \sum_{i=1}^{N} P_i(t) \] (4)

where \( P_i(t) = \tau_i \zeta(F_i(t)) \) means the energy consumed by IoT device \( i \), \( t \) is the time slot, \( \zeta \) means a parameter that depends on the deployed hardware and is measurable in practice [18].

We divide the latency tasks into two categories. The first category of delay-sensitive tasks requires a latency less than \( \tau_1 \), and just a single node can satisfy its computation requirements, such as the local process in Figure 1.

The second category of delay-sensitive tasks requires a latency less than \( \tau_2 \), and the compute capability of one node alone cannot meet its latency requirements, so it needs to be offloaded to the pro-node and use the pro-node for joint computation, such as the offload in Figure 1.

Let the vector \( \tau = [\tau_1, \tau_2] \).

For the first type of delay-sensitive tasks, its total energy consumption \( P_{a1}(t) \) can be expressed as

\[ P_{a1}(t) = \sum_{j=1}^{M} P_j(t) + \sum_{i=1}^{N} P_{i,j}^c(t) + \sum_{i=1}^{N} (1 - \alpha_i(t))P_i(t) \] (5)
where $P_j(t)$ means the energy consumed by fog node $j$ that can be calculated as $P_j(t) = \tau_j \xi(F_j(t))^3$. $P_{ij}(t)$ means the energy consumption of device $i$ to transfer tasks to $j$. $M$ denotes the set of IoT nodes, and $N$ denotes the set of fog nodes.

For the second type of delay-sensitive tasks, its total energy consumption $P_{a2}(t)$ can be expressed as

$$P_{a2}(t) = \sum_{s=1}^{S} P_s(t) + \sum_{i=1}^{N} P_{ij}^s(t) + \sum_{i=1}^{N} (1 - \alpha_i(t)) P_i(t)$$

where $P_s(t)$ means the energy consumed by subnode $s$ in the pro-node that can be expressed as $P_s(t) = \tau_s \xi(F_s(t))^3$.

### 3.3. Problem Formulation

In this section, we optimize the total energy consumption $P_a(t)$ using the Lyapunov method. Here, we only prove that $P_{a1}(t)$ and $P_{a2}(t)$ can be obtained in the same way. Our objective is to minimize the energy consumption $P_{a1}(t)$. Here, we use $P_a(t)$ to express $P_{a1}(t)$. Therefore, the whole optimization problem can be transformed into a Lyapunov optimization problem, i.e., to ensure that the energy consumption of the average time slot is minimum:

$$\text{P1: min } \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} P_a(t)$$

subject to

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{N} F_{ij}(t) < F_j$$

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{N} C_{ij}(t) < C_j$$

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \tau(t) < \tau$$

$$0 \leq F_{ij}(t) \leq F_j$$

$$0 \leq C_{ij}(t) \leq C_j$$

$$0 \leq P_i(t) \leq P_i^b$$

The objective of P1 is to minimize the long-term average energy consumption of all IoT devices. The decision variables are the offload fraction $\alpha_i(t)$, the computing resource allocation $F_i$, the transmit power per time slot $P_i(t)$, and $F_j$ (or $F_s$), where $i \in N$. In P1, the decision variables $s$ are influenced by the size of the tasks, the processing delay requirements, the channel bandwidth, the computational capability of $F_i$, $F_j$, and the transmit power of the IoT nodes. In this paper, we assume that the time slots are of the same length.

The constraint in P1 limits the computational capability allocated to the task from device $i$ (8) and the transmission speed of device $i$ (9) as well as the processing delay in the long term (10). In the long term, constrained by the computational capability of $F_i$, the computational capability allocated by $F_j$ to device $i$ should be less than the computational capability of $F_j$ itself. The size of the channel bandwidth occupied by the task should be smaller than the total channel bandwidth in the long term, due to the limitation of the device $i$’s own transmission speed. If the task can be processed locally, then only the energy consumption of $F_i$ needs to be calculated.

### 4. Upper-Bound Analysis Based on Lyapunov Optimization

To solve P1, we convert it into a Lyapunov optimization problem and solve it using our proposed PSO algorithm based on the Lyapunov framework.
### 4.1. Setting up Virtual Queues

In Lyapunov optimization, the satisfaction of the long-term average constraint is equal to the rate stability of the virtual queue. Specifically, a virtual queue is provided to replace the computing resource constraint at the edge nodes (8), and $A(t)$ denotes the random process of the length of the virtual queue $A$ at time slot $t$ [14]. $B(t)$ and $C(t)$, represent the channel constraint (9) and the computational delay constraint (10), respectively. Specific forms of $A(t)$, $B(t)$ and $C(t)$ can be expressed as the following equations:

$$A(t + 1) = \max \left( A(t) + \sum_{i=1}^{N} F_{ij}(t) - F_j, 0 \right)$$  \hspace{1cm} (14a)

$$B(t + 1) = \max \left( B(t) + \sum_{i=1}^{N} C_{ij}(t) - C_j, 0 \right)$$  \hspace{1cm} (14b)

$$C(t + 1) = \max (C(t) + \tau_i(t) + \tau_c(t) + \tau_j(t) - \tau, 0)$$  \hspace{1cm} (14c)

It is not difficult to find that in Equation (14a), if the request from $i$ is larger than the computational capability of $F_j$, then queue $A$ will get longer and longer and finally cause blocking, and vice versa, its queue will not increase. Queue $B(t + 1)$ (14b) represents the next time slot of the $B(t)$ state, i.e., the existing channel bandwidth occupation queue plus the channel bandwidth occupation of each node, then subtracting the total bandwidth yields the bandwidth occupation of the next time slot. Similarly to $B(t + 1)$, $C(t + 1)$ (14c) also represents the next time slot of the $C(t)$ state, i.e., adding the upcoming delay to the existing delay queue, and then subtracting the delay requirements yields the delay queue of the next time slot. The initialization of virtual queues $A$, $B$ and $C$ consists in setting them to 0.

If virtual queue $A$ is rate-stable, then by the definition of [19], only $\lim_{T \to \infty} A(T)/T = 0$ can satisfy the restriction (8). Virtual queue $B$ and virtual queue $C$ can be obtained in the same way. The proof is as follows:

First, we prove that virtual queue $A$ is stable:

$$A(t + 1) - A(t) = \max \left( \sum_{i=1}^{N} F_{ij}(t) - F_j, -A(t) \right)$$

For $t \in (0, 1, 2, \ldots, T - 1)$, there exists

$$\lim_{T \to \infty} \frac{A(T) - A(0)}{T} \geq \lim_{i \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{N} F_{ij}(t) - F_j$$

If $A(0) = 0$, then we can find that

$$\lim_{T \to \infty} \frac{A(T)}{T} = 0$$

Therefore, queue $A$ is stable, and similarly, queue $B$ and queue $C$ are stable, so virtual queues $A$, $B$, and $C$ can satisfy constraints (8), (9), and (10), respectively.

The proof is complete.

Using the above proof, we can convert P1 to P2:

$$\text{P2: } \min \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} P_s(t)$$  \hspace{1cm} (15)

\[ s.t. \ A(t) \text{ is rate stable} \]  \hspace{1cm} (16)

\[ B(t) \text{ is rate stable} \]  \hspace{1cm} (17)
C(t) is rate stable \hspace{1cm} (18)

4.2. Setting the Drift-Plus-Penalty

We use the Lyapunov drift-plus-penalty [21] to solve for P2. According to [21], we first set up the virtual queueing vector.

\[
\Theta(t) = [A(t), B(t), C(t)]
\]

\[
L(\Theta(t)) = \frac{1}{2} \sum_{i=1}^{N} Q_i(t)^2
\]

\[
= \frac{1}{2} (A(t)^2 + B(t)^2 + C(t)^2)
\]

\[
\Delta \Theta(t) = E(L(\Theta(t+1)) - L(\Theta(t))|\Theta(t))
\]

With stable rates across queues, P2 can then be converted to P3, because the original problem with a long-term average objective and constraints can be approximated to a problem with a drift-plus-penalty.

For convenience, here we assume that

\[
P(t) = \sum_{i=0}^{T-1} P_i(t)
\]

Hence, P3 can be expressed as:

\[
P3: \min \Delta \Theta(t) + VE(P(t)|\Theta(t))
\]

s.t. (11), (12), (13)

where \( V \) denotes the weight of the objective function, \( \Delta \Theta(t) \) denotes the drift of the queue, i.e., the stability performance of the queue.

Next, we show the specific form in preparation for P4 to find the upper bound. First, we show the proof of \( A(t+1)^2 - A(t)^2 \).

\[
A(t+1)^2 - A(t)^2 = \max(A(t) + \sum_{i=1}^{N} F_{ij}(t) - F_j, 0)^2 - A(t)^2
\]

\[
\leq 2A(t)(\sum_{i=1}^{N} F_{ij}(t) - F_j) + (\sum_{i=1}^{N} F_{ij}(t) - F_j)^2
\]

\[
\leq 2A(t)\sum_{i=1}^{N} F_{ij}(t) - 2A(t)F_j + (\sum_{i=1}^{N} F_{ij}(t))^2 - 2F_j \sum_{i=1}^{N} F_{ij}(t) + F_j^2
\]

\[
\leq 2(A(t) - F_j) \sum_{i=1}^{N} F_{ij}(t) + (\sum_{i=1}^{N} F_{ij}(t))^2 + F_j^2
\]

\[
\leq 2(A(t) - F_j) \sum_{i=1}^{N} F_{ij}(t) + (N + 1)F_j^2
\]

\[
= D_1 + 2(A(t) - F_j) \sum_{i=1}^{N} F_{ij}(t)
\]

Here, the value \( D_1 \), which is fixed, can be expressed as

\[
D_1 = (N + 1)F_j^2
\]

Similarly, we can obtain

\[
B(t+1)^2 - B(t)^2 = D_2 + 2(B(t) - C_j) \sum_{i=1}^{N} C_{ij}(t)
\]
Here, the value $D_2$, which is fixed, can be expressed as

$$D_2 = (N + 1)C^2$$

The proof of queue $C$’s drift is as follows.

$$C(t + 1)^2 - C(t)^2$$

Therefore, the $P_3$ problem can be expanded to generate the $P_4$ problem as follows:

$$\text{P4: } \min VP(t) + (A(t) - F_i) \times \sum_{i=1}^{N} F_{i,j}(t) + (B(t) - C_j) \sum_{i=1}^{N} C_{i,j}(t)$$

$$+ \frac{1}{2} (\tau_i(t) + \tau_c(t) + \tau_j(t)) \times (2C(t) + \tau_i(t) + \tau_c(t) + \tau_j(t))$$

$$\text{s.t. (13), (14), (15)}$$

Assume that $P_4$ has an upper bound, and its upper bound is the constant $C$. First, we have to prove that virtual queue $A$, virtual queue $B$, and virtual queue $C$ are stable. The proof is as follows.

Let $L(\Theta(0)) = 0$, then we can get

$$\frac{1}{2} (A(t)^2 + B(t)^2 + C(t)^2) \leq TC$$

$$A(t)^2 \leq 2TC$$

$$A(t) \leq \sqrt{2TC}$$

Therefore,

$$\lim_{t \to \infty} \frac{A(T)}{T} \leq \lim_{t \to \infty} \frac{\sqrt{2TC}}{T} = 0$$

Finally, we can get

$$\lim_{t \to \infty} \frac{A(T)}{T} = 0$$

Similarly,

$$\lim_{t \to \infty} \frac{B(T)}{T} = 0$$

$$\lim_{t \to \infty} \frac{C(T)}{T} = 0$$

Therefore, its queues are stable, so it is possible to use an approximate analysis to analyze the upper bound of $P_4$.

### 4.3. Proof of the Upper-Bound Calculation

Previously, we proved that $P_4$ was upper-bounded, so next, we try to find its specific value. First, substituting the computational drift into the formula, we prove that

$$\lim_{t \to \infty} \frac{1}{T} \sum_{i=0}^{T-1} P'(t) \leq P^* + \zeta$$  \hspace{1cm} (27)
where

\[
\zeta = \frac{(N^2 + 1)(F_r^2 + C_j^2) + 2r^2 + 2NA_{\text{max}}F_j + 2NB_{\text{max}}C_j}{2V}
\]  

(28)

is the upper bound of P4, and the Lyapunov drift-plus-penalty [21] enables the problem to be solved optimally within a certain range of deviations.

The proof is as follows.

\[
\Delta \Theta'(t) + VE(P'(t)\Theta(t)) \\
\leq \frac{D_1 + D_2 + D_3}{2V} + VP'(t) + (A(t) - F_j) \sum_{i=1}^{N} F'_i j(t) \\
+ (B(t) - C_j) \sum_{i=1}^{N} C'_i j(t) + \frac{1}{2}(\tau'_i(t) + \tau'_j(t)) \\
+ \frac{1}{2}(\tau'_i(t) + \tau'_j(t) + \tau'_j(t) + \tau'_i(t)) \\
\leq \frac{D_1 + D_2 + D_3}{2V} + VP* + N(A_{\text{max}}F_j + B_{\text{max}}C_j) \\
+ \frac{1}{2}(\tau'_i(t) + \tau'_j(t) + \tau'_j(t) + \tau'_i(t)) \\
\leq \frac{D_1 + D_2 + D_3}{2V} + VP* + 2C_{\text{max}}t + \tau^2 \\
+ N(A_{\text{max}}F_j + B_{\text{max}}C_j)
\]

By moving \(\Delta \Theta'(t)\) to the right and dividing by \(V\) at the same time, the original expression can be converted to the following inequality.

\[
\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} P'(t) \\
\leq \frac{D_1 + D_2 + D_3}{2V} + P* + \frac{2\tau C_{\text{max}}}{V} + \frac{\tau^2}{V} \\
+ \frac{N(A_{\text{max}}F_j + B_{\text{max}}C_j)}{V} - \frac{\Delta \Theta'(t)}{V}
\]

Next, we unfold \(\Delta \Theta'(t)\) to obtain the upper bound \(\zeta\)

\[
\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} P'(t) \\
\leq \frac{D_1 + D_2 + D_3}{2V} + P* + \frac{2\tau C_{\text{max}}}{V} + \frac{\tau^2}{V} \\
+ \frac{N(A_{\text{max}}F_j + B_{\text{max}}C_j)}{V} - \lim_{T \to \infty} \frac{L(\Theta'(T-1)) - L(\Theta'(0))}{V} \\
= P* + \frac{(N+1)^2F_r^2 + (N+1)^2C_j^2 + \tau^2}{V} \\
+ \frac{2C_{\text{max}}}{V} + \frac{N(A_{\text{max}}F_j + B_{\text{max}}C_j)}{V} + \frac{\tau^2}{2V}
\]

The proof is complete.

5. Algorithm Design

Based on the theoretical proof, we designed the PSO algorithm under the Lyapunov framework (LPSO) and used Algorithms 1 and 2 to solve P4.
Algorithm 1 LPSO for Task Type 1

INPUT: $F_i, F_j, Pr(max)$,  
OUTPUT $P^*(t)$

1 Initialization
   Initialize the model parameters
   iterations = 50, population size = 100

2 Foreach $t$ Do
   /* Obtaining required parameters and finding the best $X(t) = \{a(t), F_i(t), C_i(t)\}$ */
   3 for each particle $i$
      4 Initialize velocity $V_i$ and position $X_i$ for particle $i$
      5 $pBest = VP(t) + (A(t) - F)F_i(t)$
      6 $pBest = VP(t) + (A(t) - F)F_i(t)$
      7 Evaluate particle $i$ and set $pBest$
      8 $pBest = VP(t) + (A(t) - F)F_i(t)$
      9 parameters = $X_i$
   10 end for

while not stop
   11 for $i = 1$ to $N$
      12 Update the velocity and position of particle $i$
      13 Evaluate particle $i$ and set $pBest_i = X_i$
   14 end for

15 $gBest = \arg \min (pBest)$
16 return 0

Line 1: Initialize the required parameters in the IoT device, such as the controlling parameter $V$, iterations, population size.

Line 3–7: For all tasks, calculate the value of $P^*$ by node.

Line 8–16: Compare each calculated value and choose the minimum value to update the queue until it remains stable.
Algorithm 2 LPSO for Task Type 2

| Line | Description |
|------|-------------|
| 1    | **Initialization**<br>Initialize the model parameters<br>iterations = 50, population size = 100 |
| 2    | **Foreach t Do**<br>/* Obtaining required parameters and finding the best X(t)= {α(t), F(t), F_s(t), p^r (t)} */ |
| 3    | for each particle i |
| 4    | Initialize velocity V_i and position X_i for particle i |
| 5    | pBest = VP(t) + (A(t) - F_i)\sum_{i=1}^{S} F_j(t) + (B(t) - C_j)C_{s_j}(t) + \frac{1}{2}(\tau_i(t) + \tau_s(t)) + \sum_{i=1}^{S} \tau_i(t)(2C(t) + \tau_i(t) + \tau_s(t) + \sum_{i=1}^{S} \tau_i(t)) |
| 6    | Evaluate particle i and set pBest parameters = X_i |
| 7    | end for |
| 8    | while not stop |
| 9    | for i = 1 to N |
| 10   | Update the velocity and position of particle i |
| 11   | Evaluate particle i and set pBest_i = X_i |
| 12   | gBest = arg min (pBest) |
| 13   | end for |
| 14   | end while |
| 15   | print gBest |
| 16   | return 0 |

**Line 1:** Initialize the required parameters in the IoT device, such as the controlling parameter V_s, iterations, population size.

**Line 3–7:** For all tasks, calculate the value of P^r by the pro-node.

**Line 8–16:** Compare each calculated value and choose the minimum value to update the queue until it remains stable.

The difference between Algorithms 1 and 2 is that Task 2 requires scheduling more nodes to perform the computation.

### 6. Numerical Results

In this section we simulated the construction of a fog environment consisting of N = {1, 5, 10} and M = 2. The N IoT nodes were randomly distributed around the coverage area of the fog nodes. We used the experimental setup of [8] with a channel bandwidth of W = 5 MHz.
and background noise $N_0 = -100$ dm. The specific parameters in the experiments are shown in Table 2. The configuration of our simulation machine was an Intel Core i7-10200H CPU, RAM 32 GB system with Windows 10. The simulation platform used in this article was EdgeCloudSim.

Table 2. Parameter settings in the experiments.

| Parameter | Value |
|-----------|-------|
| $N$       | 1, 5, 10 |
| $W$       | 5 MHz |
| $N_0$     | $-100$ dBm |
| $\lambda_i(t)$ | 1, 1.1, 1.3 |
| $D_i$     | unif $[100, 300]$ KB |
| task_delay | unif $[8, 15]$ ms |
| $\sigma_i$ | $D_i, 5D_i, 7D_i$ |
| $F_i$     | 500 MHz |
| $F_j$     | 1 GHz |
| $F_s$     | 800 MHz |
| $p_i^c$   | 200 W |
| $V$       | 1, 5, 10, 20, 50 |
| $\varsigma$ | $10^{-10}$ |

We studied the relationship between the number of different IoT nodes and the energy consumption in Figure 2. Here, the LPSO parameters we chose to calculate are the task standard deviation $\sigma_i = D_i$, the weight $V = 50$, the arrival rate $\lambda_i(t) = \text{unif } [1, 1.1]$. The average energy consumption grows with the number of IoT nodes, because as the number of nodes increases, the number of arrivals of random tasks also increases, thus the average energy consumption of the three algorithms also increases. With the same number of nodes, we can see that LPSO has the lowest average energy consumption, PSO has the middle energy consumption, and the greedy algorithm has the highest energy consumption. Since in the same situation, compared to PSO, LPSO increases the control of virtual queues (including channel virtual queues, computing resource virtual queues), the realization of LPSO is better than that of PSO.

![Figure 2. Influence of IoT equipment quantity on energy consumption.](image)

We also studied the average energy consumption of LPSO for different weights $V$ in Figure 3. The performance varies at different weights, with the highest energy consumption at $V = 1$ and the lowest energy consumption at $V = 50$, where $V$ is the weight parameter for energy consumption, and a larger value of $V$ means that the energy consumption value has a greater impact on the objective value. This is because when the value of
$V$ increases, a small decrease in energy consumption leads to a large decrease in the value of the objective function, and then the optimization process tends to reduce the value of the energy consumption, which leads to a lower value of the objective function. In the experiments, we considered the standard deviation $\sigma_i = D_i$ and the arrival rate $\lambda_i(t) = \text{unif}[1, 1.1]$.

![Figure 3. Influence of different values of weight $V$ on average energy consumption.](image)

Figures 4 and 5 show the comparison between LPSO, PSO, and the greedy algorithm for the same number of nodes and other cases, including the total energy consumption and average energy consumption, where our proposed LPSO outperforms the other two. Because the measure taken by the greedy algorithm is to complete all task requests as much as possible, the power of all devices is set to the maximum based on the greedy algorithm. PSO is more reasonable for resource allocation than the greedy algorithm. Compared with the greedy algorithm, PSO pays more attention to the rationality of resource utilization. On the basis of PSO, LPSO increases the control of computing resources and channel resources, thus LPSO performs better than PSO and the greedy algorithm.

![Figure 4. Comparing the total energy consumption of different algorithms in the same case where the weight of LPSO is $V = 50$.](image)
Figure 4. Comparing the total energy consumption of different algorithms in the same case where the weight of LPSO is $V = 50$.

Figure 5. Comparing the average energy consumption of different algorithms in the same case where the weight of LPSO is $V = 50$.

In addition, we also compared the energy consumption of LPSO for different arrival rates, as shown in Figure 6, and as expected, the higher the value of the arrival rate $\lambda(t)$, the more energy is consumed. Here we set the weight $V = 50$ because LPSO performs best at $V = 50$. A higher arrival rate leads to more computational tasks competing for the limited network and computing resources, thus increasing the queuing time of the computing workload, which requires more computing resources to ensure that it can be completed within the specified time.

Figure 6. Influence of different arrival rates on the average energy consumption of the LPSO algorithm.

Finally, Figure 7 shows the completion degree of different algorithms under different arrival rates. When all conditions are the same, including the number of nodes, task size, and task delay, we can compare LPSO and the other algorithms in the number of tasks completed at different task arrival rates. LPSO performs better than PSO because PSO only considers the most ideal resource allocation scenario (i.e., each task just has to match its requirements), which causes the queue to block. This leads to an increase in queuing time, and LPSO combines the optimization of computing resources and channel resources in the case of a long time, where every time slot maximizes the use of resources while keeping the
queue stable. Moreover, the reason why the greedy algorithm does not complete tasks on time is because it completes as many tasks as possible; in other words, all the equipment power is turned on to the maximum, and the energy consumption is exchanged for the number of completed tasks.

Figure 6. Influence of different arrival rates on the average energy consumption of the LPSO algorithm.

Finally, Figure 7 shows the completion degree of different algorithms under different arrival rates. When all conditions are the same, including the number of nodes, task size, and task delay, we can compare LPSO and the other algorithms in the number of tasks completed at different task arrival rates. LPSO performs better than PSO because PSO only considers the most ideal resource allocation scenario (i.e., each task just has to match its requirements), which causes the queue to block. This leads to an increase in queuing time, and LPSO combines the optimization of computing resources and channel resources in the case of a long time, where every time slot maximizes the use of resources while keeping the queue stable. Moreover, the reason why the greedy algorithm does not complete tasks on time is because it completes as many tasks as possible; in other words, all the equipment power is turned on to the maximum, and the energy consumption is exchanged for the number of completed tasks.

Figure 7. Completion degree of different algorithms under different arrival rates.

7. Conclusions

The Lyapunov-based PSO algorithm we designed achieved a balanced distribution of transmission power and computing resources through the control of channel resources and computing resources and minimized the energy consumption value while ensuring that as many tasks as possible were completed. Comparing LPSO and PSO, our algorithm was better than the original PSO algorithm and the greedy algorithm in the performance index of energy consumption required to complete the task. The number of tasks completed was slightly lower than that of the greedy algorithm, because the greedy algorithm used a larger amount of energy than the LPSO algorithm to achieve its result, which obviously outweighed the gains. Regarding future work, we will set up nodes without mobility and fog nodes without energy constraints, as some tasks may not be completed if there are energy constraints. We hope to increase these constraints in future work, and further increase the number of tasks completed. Furthermore, regarding the task decomposition, since some subtasks might have a sequential relationship of backward and forward processing, we consider using a directed acyclic graph (DAG) in future work.

Author Contributions: Conceptualization, C.H. and H.W.; methodology, C.H. and H.W.; validation, C.H., L.Z. and T.L.; formal analysis, C.H. and H.W.; investigation, C.H. and H.W.; resources, C.H. and H.W.; data curation, C.H., L.Z. and T.L.; writing—original draft preparation, C.H.; writing—review and editing, C.H., H.W. and L.Z.; visualization, L.Z. and T.L.; supervision, H.W.; project administration, H.W.; funding acquisition, H.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by [National Natural Science Foundation of China under Grant] grant number [62171413] and the APC was funded by [Hui Wang].

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lei, L.; Xu, H.; Xiong, X.; Zheng, K.; Xiang, W. Joint computation offloading and multi-user scheduling using approximate dynamic programming in NB-IoT edge computing system. *IEEE Internet Things J.*, 2019, 8, 5345–5362. [CrossRef]

2. Chen, Y.; Zhang, N.; Zhang, Y.; Chen, X.; Wu, W.; Shen, X.S. Energy-efficient dynamic offloading in mobile edge computing for Internet of Things. *IEEE Trans. Cloud Comput.* 2020, 6, 4945–4962. [CrossRef]
3. Mao, Y.; Zhang, J.; Letaief, K.B. Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 3590–3605. [CrossRef]

4. Mukherjee, M.; Guo, M.; Lloret, J.; Iqbal, R.; Zhang, Q. Deadline-Aware Fair Scheduling for Offloaded Tasks in Fog Computing with Inter-Fog Dependency. *IEEE Commun. Lett.* **2020**, *24*, 307–311. [CrossRef]

5. Sun, Y.J.; Wang, H.; Zhang, C.X. Balanced Computing Offloading for Selfish IoT Devices in Fog Computing. *IEEE Access* **2022**, *10*, 30890–30898. [CrossRef]

6. Gao, X.; Huang, X.; Bian, S.; Shao, Z.; Yang, Y. PORA: Predictive offloading and resource allocation in dynamic fog computing systems. *IEEE Internet Things J.* **2019**, *7*, 72–87. [CrossRef]

7. Bononi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog computing and its role in the internet of things. In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, Helsinki, Finland, 17 August 2012; pp. 13–16. [CrossRef]

8. Hazra, A.; Adhikari, M.; Amgoth, T.; Srirama, S.N. Joint computation offloading and scheduling optimization of IoT applications in fog networks. *IEEE Trans. Netw. Sci. Eng.* **2020**, *7*, 3266–3278. [CrossRef]

9. Ramadan, R.A.; Sharif, H.; Alreshidi, E.J.; Sharif, H.U.; Ince, I.F.; Kamberaj, H.; Khedr, A.Y. Energy Coherent Fog Networks Using Multi-Sink Wireless Sensor Networks. *IEEE Access* **2021**, *9*, 167715–167735. [CrossRef]

10. Arshed, J.U.; Ahmed, M. RACE: Resource Aware Cost-Efficient Scheduler for Cloud Fog Environment. *IEEE Access* **2021**, *9*, 65688–65701. [CrossRef]

11. Lin, R.; Zhou, Z.; Luo, S.; Xiao, Y.; Wang, X.; Wang, S.; Zukerman, M. Distributed Optimization for Computation Offloading in Edge Computing. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 8179–8194. [CrossRef]

12. Liao, H.; Li, X.; Guo, D.; Kang, W.; Li, J. Dependency-Aware Application Assigning and Scheduling in Edge Computing. *IEEE Internet Things J.* **2021**, *9*, 4451–4463. [CrossRef]

13. Wang, J.; Liu, K.; Li, B.; Liu, T.; Li, R.; Han, Z. Delay-Sensitive Multi-Period Computation Offloading with Reliability Guarantees in Fog Networks. *IEEE Trans. Mob. Comput.* **2020**, *19*, 2062–2075. [CrossRef]

14. Ren, C.; Lyu, X.; Ni, W.; Tian, H.; Song, W.; Liu, R.P. Distributed Online Optimization of Fog Computing for Internet of Things Under Finite Device Buffers. *IEEE Internet Things J.* **2020**, *7*, 5434–5448. [CrossRef]

15. Chang, Z.; Liu, L.; Guo, X.; Sheng, Q. Dynamic Resource Allocation and Computation Offloading for IoT Fog Computing System. *IEEE Trans. Ind. Inf.* **2021**, *17*, 3348–3357. [CrossRef]

16. Kim, Y.; Kwak, J.; Chong, S. Dual-Side Optimization for Cost-Delay Tradeoff in Mobile Edge Computing. *IEEE Trans. Veh. Technol.* **2018**, *67*, 1765–1781. [CrossRef]

17. Zhang, Y.; Du, P.; Wang, J.; Ba, T.; Ding, R.; Xin, N. Resource Scheduling for Delay Minimization in Multi-Server Cellular Edge Computing Systems. *IEEE Access* **2019**, *7*, 86265–86273. [CrossRef]

18. Jiang, Z.; Mao, S. Energy Delay Tradeoff in Cloud Offloading for Multi-Core Mobile Devices. *IEEE Access* **2015**, *3*, 2306–2316. [CrossRef]

19. Wang, J.; Peng, J.; Wei, Y.; Liu, D.; Fu, J. Adaptive application offloading decision and transmission scheduling for mobile cloud computing. *China Commun.* **2017**, *14*, 169–181. [CrossRef]

20. Zukerman, M. Introduction to Queueing Theory and Stochastic Teletraffic Models. 2019. Available online: http://www.ee.cityu.edu.hk/~mzukerman/classnotes.pdf (accessed on 10 February 2020).

21. Neely, M.J.; Huang, L. Dynamic product assembly and inventory control for maximum profit. In Proceedings of the 49th IEEE Conference on Decision and Control (CDC), Atlanta, GA, USA, 15–17 December 2010; pp. 2805–2812.