




Article

# Energy-Aware Cloud-Edge Collaborative Task Offloading with Adjustable Base Station Radii in Smart Cities

Qian Su , Qinghui Zhang  and Xuejie Zhang \* 

School of Information Science &amp; Engineering, Yunnan University, Kunming 650500, China

\* Correspondence: xjzhang@ynu.edu.cn

**Abstract:** In smart cities, the computing power and battery life of terminal devices (TDs) can be effectively enhanced by offloading tasks to nearby base stations (BSs) with richer resources. With the goal of TDs being fully served and achieving low-carbon energy savings for the system, this paper investigates task offloading in cloud-edge collaborative heterogeneous scenarios with multiple BSs and TDs. According to the proportional relationship between the energy and coverage radii of BSs, a complete coverage task offloading model with adjustable BS radii is proposed. The task offloading problem is formulated as an integer linear program with multidimensional resource constraints to minimize the sum of energy consumption of BS coverage, offloading tasks to BSs and the cloud data center (CC). Since this task offloading problem is NP-hard, two approximate algorithms with polynomial time complexity are designed based on the greedy strategy of seeking the most energy-effective disk and the primal–dual method of constructing primal feasible solutions according to dual feasible solutions. Experimental results show that both the greedy and primal–dual algorithms can achieve good approximation performance, but each of them has its own advantages due to different design principles. The former is superior in execution time and energy consumption, while the latter has advantages in balancing loads among BSs and alleviating core network bandwidth pressure.

**Keywords:** task offloading; energy-aware; greedy algorithm; primal–dual algorithm; cloud-edge collaboration

**MSC:** 90-10; 90C59



**Citation:** Su, Q.; Zhang, Q.; Zhang, X. Energy-Aware Cloud-Edge Collaborative Task Offloading with Adjustable Base Station Radii in Smart Cities. *Mathematics* **2022**, *10*, 3992. <https://doi.org/10.3390/math10213992>

Academic Editor: Ripon Kumar Chakraborty

Received: 28 September 2022

Accepted: 24 October 2022

Published: 27 October 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Smart cities [1,2] have prospered and developed with the advancement of new-generation information technologies such as cloud computing, the Internet of Things (IoT), and wireless communications. Terminal devices (TDs) (such as sensors, cameras, smartphones, and tablets) are widely distributed throughout cities and collect various kinds of static and dynamic data, transmitting these data to cloud data centers (CCs) for processing. According to GSMA [3], more than 25 billion IoT and mobile devices will be connected to the Internet by 2025. Considering these diverse processing demands, edge computing [4–6] has been proposed as an effective solution for addressing CC service quality, bandwidth, and privacy issues in smart cities.

The base station (BS) is an important part of smart city infrastructure construction, and the uplink and downlink traffic of wireless TDs must pass through BSs. Therefore, BSs can be regarded as or integrated with edge servers, which has been investigated in several studies [4]. As a result, TDs with low computing power or battery power can offload tasks to nearby BSs that have advantages over TDs in terms of resources and computing power. However, there are many challenges in completely offloading the tasks of all TDs in smart cities to BSs.

All TDs in smart cities must be served, and the premise is that each TD must be located in the coverage area of one or more BSs. Therefore, task offloading must first solve

the problem of TDs being fully covered. Although some researchers have theoretically proposed partial coverage based on penalty functions [7] to reduce costs, these penalty methods are difficult for telecom operators to implement in practical applications. If the coverage area supported by each BS is fixed, the TDs can only be fully covered by increasing the deployment density of BSs, which requires expensive BS construction costs. If the coverage area of each BS is adjustable, the most intuitive way to achieve the full coverage of TDs is to expand the coverage area of BSs without building new BSs. However, the energy consumption of BSs has long been a key issue for smart cities in realizing low carbon emissions [8]. According to [9], when the coverage area of a BS is regarded as an approximately circular surface, the energy is proportional to the coverage radius, that is,

$$E(b) = c \cdot r(b)^\theta, \quad (1)$$

where  $c > 0$  is the energy coefficient and  $\theta \geq 1$  is called the attenuation factor. In other words, the larger the coverage area is, the more energy BSs need to provide, and the higher the corresponding operating costs. Therefore, it is necessary to reasonably adjust and set the radii of BSs to achieve the complete coverage of TDs and minimize the waste of coverage energy. In this paper, the BS radii are adjusted by mapping each BS to a series of disks with different radii, and at least one TD is required to be located at the boundary of each disk.

Since TDs are generally battery-powered, existing studies [10–12] on the energy consumption of edge computing have mainly focused on TD energy efficiency. However, green and low carbon initiatives emphasize reducing the end-to-end energy consumption of the entire system. Compared with the higher BS coverage energy consumption, the offloading energy consumption of each task may be only a few to tens of joules; however, the total offloading energy consumption of all tasks in a system is significant and cannot be ignored, potentially accounting for a third or more of the energy consumed by the BS. Therefore, the communication and execution energy costs of tasks that are offloaded should also be investigated [13,14]. The task offloading problem proposed in this paper attempts to minimize the total energy consumed by the BSs and the offloaded tasks from the perspective of the overall system.

The multi-BS and multi-TD scenario in smart cities is inherently heterogeneous. On the one hand, there are various types of BSs, such as macro BSs, micro BSs and pico BSs. These BSs can also be divided into 5G and 4G BSs according to the type of communication technology. Each type of BS supports a different coverage and provides distinct bandwidth and computing power; for example, the coverage of a 5G BS is only approximately 1/4 of that of a 4G BS, while the network speed of a 5G BS is more than 10 times that of a 4G BS. On the other hand, different TD tasks have distinct computing and communication resource requirements, and the resource requirements have multidimensional attributes [15,16]. For example, online games require more advanced CPUs or GPUs than bandwidth, while augmented reality requires more bandwidth and storage than computing resources. Therefore, not all TD tasks covered by a BS can be offloaded to the BS, and CCs need to be involved. Some famous service providers, such as Google, Amazon and Alibaba, have proposed cloud-edge collaboration solutions in their application scenarios. However, the coordination and complementarity between CCs and BSs, as well as the reasonable allocation of multidimensional resources, are directly related to several factors, such as application objectives and scene constraints; thus, the task offloading problem in this environment has high computational complexity and is difficult to solve. We later prove that the considered task offloading problem is NP-hard. Motivated by this, it is necessary to design polynomial-time algorithms to obtain approximate optimal solutions to the problem. In recent years, some novel methods summarized from natural laws have been put into practice in dealing with NP-hard problems. For example, refs. [17–19] effectively solved some difficult practical problems such as the disaster evacuation and security routing of wireless sensor networks, by using the artificial bee colony, adaptive genetic algorithm, and giza pyramids construction algorithm of metaheuristic methods. These methods can

quickly find feasible solutions of the problems by searching the solution space, which is worthy of further research and exploration.

In this paper, we model the task offloading problem in a cloud-edge collaborative environment as a multiconstrained integer linear program and design approximation algorithms based on greedy strategy and primal duality theory, which are most commonly used in combinatorial optimization [20]. The greedy algorithm is simple, intuitive and fast. It can obtain the approximate solution of the problem at a lower time cost, but sometimes its solution easily falls into the local optimum. Based on rigorous combinatorial optimization theory, the primal–dual algorithm can obtain an approximate optimal solution with high quality, but its time cost is larger than that of the greedy algorithm. Both of them have their own characteristics in algorithm design.

In summary, we propose an energy-aware cloud-edge collaborative task offloading model based on multi-BS and multi-TD scenarios in smart cities. To the best of our knowledge, our work is the first to combine multi-BS complete coverage and cloud-edge collaboration to address the task offloading problem in smart cities. Our contributions can be summarized as follows:

- Inspired by Equation (1), the concept of a disk is introduced, and a complete coverage task offloading model with adjustable BS radii is proposed for cloud-edge collaborative environments. Considering the total energy consumption of BS coverage, offloading to BSs and offloading to CC, the task offloading problem is formulated as an integer linear program under CPU and bandwidth constraints, and the proposed problem is proven to be NP-hard.
- Based on the “most energy-effective” greedy strategy, a greedy algorithm with polynomial time complexity is designed. The experimental results show that our greedy algorithm can obtain an objective function value close to the optimal solution obtained by IBM CPLEX [21], and has a significant advantage in terms of the execution time.
- According to the method of constructing primal feasible solutions by dual feasible solutions, a primal–dual algorithm with polynomial time complexity is designed. The experimental results show that although the primal–dual algorithm is inferior to the greedy algorithm in energy consumption and execution time, it is better than the greedy algorithm in balancing the load among BSs and relieving the bandwidth pressure of the CC.

The remainder of this paper is organized as follows. Section 2 describes related work. Section 3 presents the system model, disk construction, energy consumption calculations, and a formal description of the task offloading problem. In Sections 4 and 5, two approximate algorithms are designed according to different ideas. In Section 6, extensive comparative experiments are performed to evaluate algorithm performance. Finally, Section 7 summarizes the work in this paper.

## 2. Related Work

Task offloading is the main way that wireless TDs prolong their battery life and expand their processing power in various applications in smart cities. Moreover, BSs are located at network edges near TDs and are thus ideal carriers for task offloading. Various researchers have investigated task offloading using mobile edge computing-enabled BSs (MEC-BSs) [4]. Wang et al. [22] investigated partial computation offloading by jointly optimizing the computational speed, transmit power, and offloading ratio of a single smart mobile device and single BS. You et al. [23] considered a single-user/single-BS system in which the BSs can either transfer power to the user or move computations from the user to a CC. Furthermore, they minimized the weighted sum of the energy consumption of mobile devices under delay constraints and studied the computation offloading problem of a multi-mobile user single-edge cloud (BS acts) based on time-division and orthogonal frequency-division multiple access [10]. Cao et al. [24] proposed a user cooperation approach that considered both computation and communication in an MEC system consisting of a user node, helper node, and small BS node to improve the energy efficiency of computing tasks under latency

constraints. Du et al. [25] considered computation offloading in a scenario including an LTE-A small BS with an MEC server and multiple smart mobile devices to maximize the system capacity and transmit rate. Most early works considered one BS with one or multiple users; however, scenarios with multiple BSs and multiple users are more consistent with practical applications.

Zhang et al. [11] considered the insufficient computing power of the IoT under delay constraints and used time-division multiple access and frequency-division multiple access to offload data to multiple BSs for execution. Yang et al. [12] modeled the computation offloading problem as a mixed integer program in a multi-BS system with an EMC server to minimize the energy consumption of mobile devices. Ma et al. [26] divided BS groups in an MEC network, with small-cell BSs in the same BS group collaboratively processing tasks. For each BS group, the authors designed a task offloading strategy based on dynamic voltage scaling technology that effectively reduced the power consumption of the BS group. Rahman et al. [27] regarded the BSs as edge clouds and proposed an optimized solution for HTTP adaptive streaming by shifting the adaptation intelligence from the client to the edge cloud to jointly optimize the quality of experience for the streaming clients. The above multi-BS computation offloading strategies involve optimization objectives such as low energy consumption, low latency, and high user experience. However, the performance differences among multiple BSs are not emphasized, and the multiresource requirements of users are not discussed. Bahreini et al. [28] proposed a distributed vehicular edge computing system that balanced quality of service (QoS) requirements with energy costs and constructed a road-side unit with a small number of BSs or wireless access points. Under multidimensional resource constraints, the problem was decomposed into two subproblems, and a resource selector algorithm and energy manager algorithm were proposed. Liu et al. [29] considered wireless channels and CPU cycles to minimize the local execution energy and transmission energy consumption between BSs and mobile users to provide task offloading services to multiple mobile devices. Similar to [28], multidevice computation offloading strategies with multidimensional resource constraints have difficulty obtaining optimal solutions in polynomial time. Therefore, the authors divided the problem into two subproblems, resource allocation and offloading decisions, and designed heuristic and approximate algorithms to address the two subproblems.

Compared with the large amount of resources and computing power of CCs, BSs have limited resource capacity and capability. When a large number of tasks are transferred to BSs, the BS resources may be consumed quickly, which is not considered in the above studies. At present, there are three methods for addressing BS overload: multi-BS cooperation, device-to-device (D2D) communication, and cloud-edge collaboration. Kuang et al. [30] established a multi-user system model with structured tasks and multiple offloading points (BSs combined with edge servers) to minimize the weighted sum of the delay and energy costs. In this system, the offloading points were connected through high-speed communication media. When the computing resources of one offloading point were insufficient, the tasks of the users connected to this point were offloaded to other resource-rich offloading points. Fan et al. [31] transferred tasks between MEC-BSs to reduce the pressure on high-load BSs and designed a game theory-based task offloading scheme between the MEC-BSs. Although these studies have achieved load balancing among BSs, when most BSs are overloaded, systems reject user requests, and task offloading cannot continue, greatly reducing user experiences. To maximize the number of devices supported by cellular networks, D2D communications were proposed to improve the performance of MEC systems [32]. In these systems, the tasks of each device can be offloaded to an MEC-BS or a nearby D2D device. Tong et al. [33] considered indivisible tasks, balanced the number of mobile devices to be offloaded and the energy consumption of task offloading, and used D2D technology to reduce the backhaul load of the BSs. Both works discussed D2D communication in terms of a single BS; however, in the case of large-scale task offloading, D2D communication is not always feasible because device resources are inherently limited. Compared with multi-BS collaboration and D2D communication, cloud-side collaboration

is a more recognized method for enhancing MEC performance in industry and academia. A collaborative utility maximization scheme was proposed in [34]. An overall system utility maximization problem was formulated to minimize the time and energy consumption of all mobile devices. In the cooperation strategy, the available free computing resources of neighboring MEC-BSs are exploited in the first layer, and additional workloads are processed in the second layer by CCs. Ren et al. [35] studied joint communication and computation resource allocation to minimize the weighted-sum delay costs of all devices in a cloud-edge collaboration system using combinatorial optimization theory. Zhao et al. [36] presented a cloud-assisted MEC computation offloading problem in vehicular networks as a constrained optimization problem by jointly optimizing computation offloading decisions and computing resource allocation to maximize system utility.

The above cloud-edge collaborative architectures effectively addressed the deficiencies of single-edge computing. While these works assumed that the TDs are covered by corresponding BSs, the BS coverage is not explicitly discussed. In addition, while the considerable energy consumption of BSs has been highlighted, few studies have devised methods for reducing energy costs by adjusting BS coverage. The minimum power partial coverage problem in wireless networks is discussed in detail in [9,37], with sensors or wireless access points having their coverage scaled by changing the transmit power. Similarly, multi-BS systems with self-regulating power can reasonably adjust BS coverage according to the TD distribution and task requirements. Compared with BSs with constant coverage, this method effectively reduces the BS coverage energy costs. In this paper, to ensure that all tasks in smart cities are served, in contrast to partial power coverage, we propose a complete coverage task offloading scheme with adjustable BS coverage radii and multiresource constraints to minimize the total energy consumption of the system. Alnoman et al. [38,39] studied sleep modes that effectively reduce the energy consumption of BSs, which is essentially equivalent to the special case where the BS coverage radii are set to 0 in this paper. Table 1 compares our method with those proposed in previous studies.

**Table 1.** Comparison of related works.

Reference	Adjustable BS Radius	Cloud-Edge Collaboration	Multi-BS	Multidimensional Resources	Energy Consumption
You et al. (2017) [10]	×	×	×	✓	User side
Cao et al. (2019) [24]	×	×	×	✓	User side
Zhang et al. (2019) [11]	×	×	✓	×	User side
Yang et al. (2019) [12]	×	×	✓	✓	User side
Ma et al. (2020) [26]	×	×	✓	×	BS side
Bahreini et al. (2021) [28]	×	×	✓	✓	User side
Liu et al. (2021) [29]	×	×	×	✓	User side
Kuang et al. (2020) [30]	×	×	✓	×	User side
Khan et al. (2019) [34]	×	✓	✓	✓	User side
Ren et al. (2019) [35]	×	✓	✓	✓	Not involved
Zhao et al. (2019) [36]	×	✓	✓	×	Not involved
Our approach	✓	✓	✓	✓	System

### 3. System Modeling and Preparation

#### 3.1. System Model

We consider a specific scenario in a smart city. The system has two types of resources: computing resources (such as CPUs or virtual machines) and bandwidth resources.  $n$  TDs with limited computing power and battery capacity are randomly distributed in a certain geographical area, which is expressed as  $U = \{1, 2, \dots, n\}$ . Each TD  $i \in U$  generates an indivisible computing task  $t_i = (q_i, d_{iCPU}, d_{iBW})$ , where  $q_i$  represents the size of  $t_i$ 's input data, and  $d_{iCPU}$  and  $d_{iBW}$  represent  $t_i$ 's demands for computing and bandwidth resources, respectively. In addition,  $m$  BSs are randomly distributed near the  $n$  TDs, which is expressed



as  $B = \{1, 2, \dots, m\}$ . Each BS  $b \in B$  is regarded as an edge server with a different capacity in terms of computing and bandwidth resources  $C_b = \{C_{bCPU}, C_{bBW}\}$ , which provides task offloading services to TDs within its coverage range through wireless networks. The core network includes a CC with a large resource pool far from the TDs. The CC works with the BSs to provide task offloading services to TDs through the Internet. The system model is shown in Figure 1. The solid lightning symbol indicates that the TD’s tasks are offloaded to a BS, while the hollow lightning symbol indicates that tasks are offloaded to a CC through a BS. All TDs are located within the coverage region of the BSs. Although each TD may be covered by multiple BSs, its tasks can be offloaded to exactly one BS or CC.



Figure 1. Cloud-edge collaborative task offloading model.

Assuming that the computing power and battery capacity of the TDs are insufficient to support the execution of their own computing tasks, all computing tasks need to be offloaded to BSs at the edge of the network or the CC for processing. The model proposed in this paper aims to achieve minimum energy consumption while ensuring that all tasks are offloaded. We discuss two types of energy consumption: the BS coverage energy consumption related to Equation (1) and the execution and transmission energy consumption caused by task offloading. Table 2 lists the notations used in this paper.

Table 2. Notations.

Notation	Description
$m$ ( $n$ )	Number of BSs (TDs)
$B$ ( $U$ )	Set of all BSs (TDs)
$t_i$	Task of TD $i$
$q_i$	Input data size of $t_i$
$d_{iCPU}$ ( $d_{iBW}$ )	CPU (bandwidth) demand of $t_i$
$C_{bCPU}$ ( $C_{DCPU}$ ), $C_{bBW}$ ( $C_{DBW}$ )	CPU (bandwidth) capacity of BS $b$ (disk $D$ )
$\tilde{C}_{bCPU}$ ( $\tilde{C}_{DCPU}$ ), $\tilde{C}_{bBW}$ ( $\tilde{C}_{DBW}$ )	Remaining CPU (bandwidth) capacity of BS $b$ (disk $D$ )

Table 2. Cont.

Notation	Description
$r(b)$ ( $r(D)$ )	Coverage radius of BS $b$ (disk $D$ )
$E(b)$ ( $E(D)$ )	Coverage energy consumption of BS $b$ (disk $D$ )
$c$ ( $\theta$ )	Energy coefficient (attenuation factor) in Equation (1)
$b(D)$	BS serves as the center of disk $D$
$E_{ib}^{bs}$ ( $E_{ib}^{CC}$ )	Energy consumption of offloading $t_i$ to a BS (CC)
$U^{unse}$	Set of unserved TDs
$U_b^{dise}$ ( $U_D^{dise}$ ), $U_b^{indise}$ ( $U_D^{indise}$ )	TDs set directly (indirectly) served by BS $b$ (disk $D$ )
$\mathcal{D}$ ( $\mathcal{D}(b)$ )	Set of all disks (centered at BS $b$ )
$\mathcal{D}^{sele}$	Set of selected disks
$\tilde{\mathcal{D}}^{sele}$	Set of temporarily selected disks
$\mathcal{D}^{full}$	Set of full disks
$D^{mee}$	Most energy-effective disk
$f_b$ ( $f_{vm}^{CC}$ )	CPU frequency of BS $b$ (virtual machines in the CC)
$p_b$ ( $p_{vm}^{CC}$ )	CPU power of BS $b$ (virtual machines in the CC)
$e_{i1}$	Energy consumption per bit of data sent by $i$ 's transmitter
$e_{i2}$	Energy consumption per bit of data transmitted per square meter by $i$ 's transmitting amplifier
$e_{wired}$	Wired transmission energy consumption coefficient of the Internet
$k$	Transmission attenuation factor of the TDs
$x_{iD}$ ( $z_{iD}$ )	Primal variable indicating whether $t_i$ is directly (indirectly) served by disk $D$
$y_D$	Primal variable indicating whether disk $D$ is selected
$\alpha_i$	Dual variable indicating the cost budget of TD $i$
$\beta_{iD}$ ( $\gamma_{iD}$ )	Dual variable indicating the disk coverage energy cost shared by $i$ when it is directly (indirectly) served by $D$
$\delta_D$ ( $\epsilon_D$ )	Dual variable indicating the unit price of the CPU (bandwidth) of disk $D$
$\mu_b$	Dual variable indicating the total resource cost of BS $b$
$l$	Step size for updating the dual variables

### 3.2. Disk Structure of the BSs

Based on set coverage theory, similar to the literature [9], we introduce the concept of disks to reflect the coverage of the BSs when they have different energy. According to Equation (1), each BS  $b \in B$  corresponds to a number of disks, where  $b$  is the center and  $r(b)$  is the radius supported by energy  $E(b)$ . The set of these disks is denoted as  $\mathcal{D}(b)$ . The greater the energy that the BS can provide, the greater the radius of the corresponding disk. Here, we assume that the maximum energy provided by each BS is enough to cover the TD farthest away from it to ensure that all TDs in the system are fully covered and that the maximum disk mapped to each BS is valid. In Section 6, we verify through experiments that the coverage radius of each booted BS is far less than the radius of its maximum disk.

To ensure the minimum energy consumption, we construct the set of all disks in the system as follows. The BS-TD pair  $(b, i)$  is used to determine the disk  $D(b, i)$ , that is,  $D(b, i)$  is a disk with  $b$  as the center and the Euclidean distance  $dist(b, i)$  between  $b$  and  $i$  as the radius. Therefore, for each disk, at least one TD should be located on its boundary; otherwise, the radius of the disk can be reduced while keeping the set of covered TDs unchanged, thereby reducing the disk coverage energy. Thus, there are at most  $mn$  disks in the system.

For  $\forall i \in U$ , if  $i$  is within the range of disk  $D(b, i')$ , that is, if  $dist(b, i) \leq r(D)$ ,  $i$  is covered by disk  $D(b, i')$  and is denoted as  $i \in D(b, i')$ , where  $r(D)$  represents the radius of disk  $D(b, i')$ . To simplify the notations,  $D$  is used to represent both a disk and the set of TDs covered by disk  $D$ , denoted as  $i \in D \in \mathcal{D}$ , where  $\mathcal{D}$  denotes the set of all disks in the system.  $E(D)$  represents the coverage energy of disk  $D$  (that is, the energy required by BS  $b(D)$  to maintain a coverage region with a radius of  $r(D)$ , where  $b(D)$  represents the BS corresponding to  $D$ ). According to Equation (1),  $E(D)$  can be formulated as

$$E(D) = c \cdot r(D)^\theta. \tag{2}$$

Note that we transform the BS coverage energy consumption into the disk coverage energy consumption by introducing the concept of disks.

### 3.3. Disk Coverage Requirements

As shown in Figure 1, regardless of whether the TD computing task is offloaded to some BS or CC, the task must first be transmitted to a BS. Therefore, the TD must be within the coverage region of the energy supported by the BS. Next, we define directly served and indirectly served.

**Definition 1** (directly/indirectly served).  $\forall i \in U$ , if the computing resources and bandwidth of computing task  $t_i$  are satisfied by BS  $b$  and  $t_i$  is offloaded to  $b$  for execution, then  $i$  is directly served by  $b$ ,  $b$  is the direct service provider of  $i$ , and the set of TDs directly served by  $b$  is  $U_b^{\text{dise}}$ . In this case, if  $\forall i \in D \in \mathcal{D}(b)$ ,  $i$  is directly served by disk  $D \in \mathcal{D}(b)$ , and the set of TDs directly served by  $D \in \mathcal{D}(b)$  is  $U_D^{\text{dise}}$ .

Similarly, if the bandwidth demand of  $t_i$  is satisfied by  $b$  and  $t_i$  is transferred to the CC for execution after being relayed by  $b$ , then  $i$  is indirectly served by  $b$ ,  $b$  is the indirect service provider of  $i$ , and the set of TDs indirectly served by  $b$  is denoted as  $U_b^{\text{indise}}$ . In this case, if  $\forall i \in D \in \mathcal{D}(b)$ ,  $i$  is indirectly served by  $D \in \mathcal{D}(b)$ , and the set of TDs indirectly served by  $D \in \mathcal{D}(b)$  is  $U_D^{\text{indise}}$ .

If  $i$  is served directly or indirectly by  $b$  (or  $D \in \mathcal{D}(b)$ ),  $i$  is served by  $b$  (or  $D \in \mathcal{D}(b)$ ).

Definition 1 shows that  $\forall b \in B, D \in \mathcal{D}(b), U_b^{\text{dise}} = \bigcup_{D \in \mathcal{D}(b)} U_D^{\text{dise}}$  and  $U_b^{\text{indise}} = \bigcup_{D \in \mathcal{D}(b)} U_D^{\text{indise}}$ .

To minimize energy consumption, based on the above discussion, the disk coverage must satisfy the following two requirements:

- $\forall b \in B$ , at most one disk  $D \in \mathcal{D}(b)$  is selected because each BS is either not selected or can only be activated with a specific energy during each period in practice.
- $\forall i \in D \in \mathcal{D}$ ,  $i$  must be served by exactly one selected disk. First, this is a requirement for completely offloading all TD tasks; second, if  $i$  is repeatedly served by multiple disks with different centers,  $i$  is served by multiple BSs, thereby occupying multiple BS resources and wasting system resources. Furthermore, in general models considering the total energy consumption of the system, repeatedly servicing  $i$  increases the energy consumption of the system.

### 3.4. Energy Consumption of Offloading Tasks to BSs

Here, we analyze the computing energy consumption and uplink (from TDs to BSs) transmission energy consumption when a TD is directly served by a BS because the downlink (transmitting results from the BSs to the TDs) energy consumption is included in the BS coverage energy.

When task  $t_i$  is transferred to BS  $b$  for execution, the computing energy is  $E_{ib}^{\text{bsex}} = p_b \cdot \frac{d_i^{\text{CPU}}}{f_b}$ , where  $f_b$  represents the computing power of BS  $b$  (i.e., the CPU frequency of  $b$ ), and  $p_b$  represents the CPU power of  $b$  corresponding to computing power  $f_b$ .

The transmission energy consumption of task  $t_i$  from TD  $i$  to BS  $b$  is related to the size of the input data  $q_i$  and the transmission distance  $\text{dist}(b, i)$ , that is,  $E_{ib}^{\text{bstr}} = e_{i1} \cdot q_i + e_{i2} \cdot q_i \cdot \text{dist}(b, i)^k$ , where  $e_{i1}$  represents the energy consumption per bit of data sent by TD  $i$ 's transmitter,  $e_{i2}$  represents the energy consumption per bit of data transmitted per square meter by  $i$ 's transmitting amplifier, and  $k$  is the transmission attenuation factor. The value of  $k$  depends on the environment, and  $2 \leq k \leq 5$ . If the physical environment is flat and barrier-free,  $k$  is set to 2; if there are buildings, dense vegetation, or bad weather for long-distance transmission,  $k$  ranges from 3 to 5.

Therefore, the total energy consumed by offloading tasks to the BSs is  $E_{ib}^{\text{bs}} = E_{ib}^{\text{bsex}} + E_{ib}^{\text{bstr}}$ .

Let  $E_{iD}^{\text{bs}}$  denote the energy consumption of TD  $i$  covered by disk  $D$  offloaded to BS  $b(D)$ , which is equal to the offloading energy consumption of  $i$  to  $b$ , i.e.,  $E_{iD}^{\text{bs}} = E_{ib}^{\text{bs}}, \forall i \in D \in \mathcal{D}(b), b \in B$ .



### 3.5. Energy Consumption of Offloading Tasks to CC

When the TDs are indirectly served by the BSs, the energy includes the computing energy of the CC after the TD tasks are relayed by the BSs and the uplink transmission energy from the TDs to the BSs and the BSs to the CC. The downlink process is divided into two stages: the result is first transmitted from the CC to the BSs and then from the BSs to the TDs. Since the output is usually considerably smaller than the input [29], the energy consumption of the first stage can be ignored. The energy consumption of the second stage is similar to that described in Section 3.4 and is thus included in the BS coverage energy.

When task  $t_i$  is transferred to the CC for processing, the computing energy is  $E_i^{ccex} = p_{vm}^{cc} \cdot \frac{d_{iCPU}^{cc}}{f_{vm}^{cc}}$ , where  $f_{vm}^{cc}$  indicates the computing power (i.e., CPU frequency) of virtual machines in the CC, and  $p_{vm}^{cc}$  indicates the CPU power of virtual machines with computing power  $f_{vm}^{cc}$ .

In the uplink process, the BSs and CC are connected through a wired medium (such as an optical cable). According to the literature [40], the energy consumption of wired transmission is proportional to the size of the transmitted data, that is,  $E_i^{ccctr} = e_{wired} \cdot q_i$ , where  $e_{wired}$  is the wired transmission energy consumption coefficient of the Internet. Therefore, the uplink transmission energy of the tasks offloaded to the CC after being relayed by BS  $b$  can be expressed as  $E_{ib}^{ccctr} = E_{ib}^{bstr} + E_i^{ccctr} = e_{i1} \cdot q_i + e_{i2} \cdot q_i \cdot dist(b, i)^k + e_{wired} \cdot q_i$ . Thus, the total energy consumed by offloading tasks to the CC is  $E_{ib}^{cc} = E_i^{ccex} + E_{ib}^{ccctr}$ .

Let  $E_{iD}^{cc}$  denote the energy consumed by TD  $i$ , which is covered by disk  $D$ , being offloaded to the CC, which is equal to the offloading energy consumed by  $i$  being relayed to the CC via  $b(D)$ , i.e.,  $E_{iD}^{cc} = E_{ib}^{cc}, \forall i \in D \in \mathcal{D}(b), b \in B$ .

### 3.6. Problem Formulation

For  $\forall i \in U, \forall D \in \mathcal{D}$ , we define three 0–1 variables:  $x_{iD}, y_D$  and  $z_{iD}$ , as shown in Equations (3)–(5).

$$x_{iD} = \begin{cases} 1, & i \text{ is directly served by } D \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$$y_D = \begin{cases} 1, & D \text{ is selected} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

$$z_{iD} = \begin{cases} 1, & i \text{ is indirectly served by } D \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Thus, the task offloading problem with minimum system energy consumption can be formulated as the following integer linear program (LP1):

$$\min_{x,y,z} \sum_{D \in \mathcal{D}} E(D) \cdot y_D + \sum_{i \in U} \sum_{D: i \in D \in \mathcal{D}} (E_{iD}^{bs} \cdot x_{iD} + E_{iD}^{cc} \cdot z_{iD}) \tag{6}$$

$$\text{s.t.} \quad \sum_{D: i \in D \in \mathcal{D}} (x_{iD} + z_{iD}) \geq 1, \forall i \in U \tag{6a}$$

$$y_D - x_{iD} \geq 0, \forall i \in D \in \mathcal{D} \tag{6b}$$

$$y_D - z_{iD} \geq 0, \forall i \in D \in \mathcal{D} \tag{6c}$$

$$C_{DCPU} \cdot y_D - \sum_{i \in D} d_{iCPU} \cdot x_{iD} \geq 0, \forall D \in \mathcal{D} \tag{6d}$$

$$C_{DBW} \cdot y_D - \sum_{i \in D} d_{iBW} \cdot (x_{iD} + z_{iD}) \geq 0, \forall D \in \mathcal{D} \tag{6e}$$

$$- \sum_{D \in \mathcal{D}(b)} y_D \geq -1, \forall b \in B \tag{6f}$$

$$x_{iD} \in \{0, 1\}, z_{iD} \in \{0, 1\}, \forall i \in D \in \mathcal{D} \tag{6g}$$

$$y_D \in \{0, 1\}, \forall D \in \mathcal{D} \tag{6h}$$

Constraint (6a) guarantees the second disk coverage requirement discussed in Section 3.3 due to the limitation of minimizing the system energy consumption. Constraints (6b) and (6c) indicate that TD  $i$  can be served directly or indirectly by disk  $D$  only if  $D$  is selected. Constraints (6d) and (6e) represent the resource constraints of disk  $D$ , that is, the sum of the CPU demands of the TDs directly served by  $D$  should not exceed the CPU capacity of  $D$ , while the sum of the bandwidth demands of the TDs directly and indirectly served by  $D$  should not exceed the bandwidth capacity of  $D$ . Constraint (6f) guarantees the first disk coverage requirement discussed in Section 3.3. The CPU and bandwidth capacities of disk  $D$  mentioned here are actually those of  $D$ 's center  $b(D)$ , that is,  $\forall D \in \mathcal{D}(b), C_{DCPU} = C_{bCPU}, C_{DBW} = C_{bBW}$ . We assume that the CC has a considerable resource pool and is connected to the BSs via a wired network (such as fiber optic cables); thus, the resource constraints of the CC are not considered in this paper.

Unfortunately, due to the multidimensional resource constraints of the BSs and the requirement of complete TD coverage, it is intractable to obtain the optimal solution to the task offloading problem shown in LP1 in polynomial time. Next, we infer that the problem is NP-hard by proving that a special case is NP-hard.

**Theorem 1.** *The task offloading problem in LP1 for minimizing system energy consumption is NP-hard.*

**Proof.** Assuming that the tasks of all TDs are directly served by the BSs, no tasks need to be offloaded to the CC for processing, which leads to a special case of our proposed problem. We denote this special problem as LP1-S and present its integer program as follows:

$$\min_{x,y} \sum_{D \in \mathcal{D}} E(D) \cdot y_D + \sum_{i \in U} \sum_{D: i \in \mathcal{D}(D)} E_{iD}^{bs} \cdot x_{iD} \tag{7}$$

$$\text{s.t.} \quad \sum_{D: i \in \mathcal{D}(D)} x_{iD} \geq 1, \forall i \in U \tag{7a}$$

$$y_D - x_{iD} \geq 0, \forall i \in D \in \mathcal{D} \tag{7b}$$

$$C_{DCPU} \cdot y_D - \sum_{i \in D} d_{iCPU} \cdot x_{iD} \geq 0, \forall D \in \mathcal{D} \tag{7c}$$

$$C_{DBW} \cdot y_D - \sum_{i \in D} d_{iBW} \cdot x_{iD} \geq 0, \forall D \in \mathcal{D} \tag{7d}$$

$$- \sum_{D \in \mathcal{D}(b)} y_D \geq -1, \forall b \in B \tag{7e}$$

$$x_{iD} \in \{0, 1\}, \forall i \in D \in \mathcal{D} \tag{7f}$$

$$y_D \in \{0, 1\}, \forall D \in \mathcal{D} \tag{7g}$$

The key step in the above proof is to find a well-known NP-hard problem and reduce it to LP1-S to show that LP1-S is also an NP-hard problem. We observe that LP1-S is a general case of the capacitated facility location problem (CFLP) [41]. The CFLP can be described as follows: consider a set of facilities  $F$  and a set of clients  $C$  requiring service. Opening each facility  $i \in F$  incurs a fixed cost  $f_i$ . Each client  $j \in C$  must be assigned to a single open facility  $i$  (Constraints (7a) and (7b)) and incur a service cost  $c_{ij}$ . Each facility  $i \in F$  provides a capacity of  $u_i$ , each client  $j \in C$  has a demand of  $d_j$ , and the sum of the demands of multiple clients served by a facility cannot exceed the capacity of the facility (Constraint (7c)). The problem objective is to minimize the facilities' opening costs and service costs (Expression (7)). Note that the CFLP is a special case of LP1-S. The CFLP is a famous NP-hard problem; thus, LP1-S is also NP-hard. Furthermore, it can be deduced that the task offloading problem considered in this paper is NP-hard. □

Theorem 1 motivates us to develop efficient approximation algorithms to obtain approximate optimal solutions of LP1 in polynomial time. The task offloading studied in this paper is a typical combinatorial optimization problem. The greedy strategy and

primal–dual technique are the two most commonly used methods to solve combinatorial optimization problems. Thus, we designed two polynomial-time approximation algorithms based on these two different strategies. Note that although CPLEX is used to calculate the optimal solution to our problem in the experimental evaluation, CPLEX does not work when the number of TDs exceeds a certain size.

#### 4. Greedy Algorithm for Cloud-Edge Collaborative Task Offloading

In this section, referring to the most cost-effective greedy strategy [42], we design a cloud-edge collaborative greedy algorithm for the task offloading problem that focuses on the total energy consumption of the system.

##### 4.1. Design of the Greedy Algorithm

We use the concept of “energy effectiveness” to design a greedy algorithm. The algorithm includes three key steps: constructing the full disk, selecting the most energy-effective disk, and updating the disk coverage energy and BS resource capacity. First, we define a full disk and its energy effectiveness as follows:

**Definition 2** (full disk). *A disk and the set of TDs served directly and indirectly by it form the full disk  $(D, U_D^{\text{dise}}, U_D^{\text{indise}})$ , where  $U_D^{\text{dise}} \cap U_D^{\text{indise}} = \phi$  and  $U_D^{\text{dise}} \cup U_D^{\text{indise}} \neq \phi$ .*

According to Definitions 1 and 2, the resource capacity of a full disk must meet the resource demands of the computing tasks of the TDs served by it, that is,  $C_{DCPU} \geq \sum_{i \in U_D^{\text{dise}}} d_{iCPU}$ ,  $C_{DBW} \geq \sum_{i \in U_D^{\text{dise}} \cup U_D^{\text{indise}}} d_{iBW}$ .

**Definition 3** (energy effectiveness). *The energy effectiveness of a full disk is the ratio of its energy consumption to the number of TDs directly and indirectly served by it, that is, the average energy consumption apportioned by each TD.*

Note that for the system energy consumption minimization problem, the energy consumed by a full disk is the sum of its coverage energy and the energy consumed by TD tasks served by the full disk that are offloaded to BSs and the CC. Thus, the energy consumption effectiveness corresponds to  $\frac{E(D) + \sum_{i \in U_D^{\text{dise}}} E_{iD}^{\text{bs}} + \sum_{i \in U_D^{\text{indise}}} E_{iD}^{\text{cc}}}{|U_D^{\text{dise}} \cup U_D^{\text{indise}}|}$ . Compared with intuitively selecting the disk with the lowest energy consumption, the use of the energy effectiveness concept can alleviate the situation in which the algorithm falls into a local optimal solution to a certain extent.

Next, we provide a detailed description of the greedy algorithm. In the initialization of Algorithm 1, all TDs are not served, and all disks are not selected. Since service providers tend to serve customers with high resource requirements in practice, all unserved TDs are first sorted in descending order of their tasks’ CPU demands (line 2). The algorithm then iteratively performs three important steps until all TDs are served (lines 3–15). Finally, for each BS  $b$ , the algorithm identifies the disk with the largest radius from the temporarily selected disks as the final selected disk and updates the TDs directly and indirectly served by disk  $D_b^{\text{max}}$  with the TDs served by  $b$  (lines 16–20).

**Algorithm 1** Greedy algorithm for cloud-edge collaborative task offloading.

**Input:** TD set  $U$  and their tasks  $\{t_i\}$ , disk set  $\mathcal{D}$ , BS set  $B$  and their resource capacity  $\{C_b\}$   
**Output:** Set of final selected disks  $\mathcal{D}^{sele}$ , sets of directly and indirectly served TDs  $\{U_D^{dise}\}$  and  $\{U_D^{indise}\}$

- 1: Initialize:  $U^{unse} = U, \mathcal{D}^{sele} = \phi, \tilde{\mathcal{D}}^{sele} = \phi, U_D^{dise} = \phi, U_D^{indise} = \phi, \tilde{C}_{bCPU} = C_{bCPU}, \tilde{C}_{bBW} = C_{bBW}, U_b^{dise} = \phi, U_b^{indise} = \phi, \forall D \in \mathcal{D}, \forall b \in B;$
- 2: Sort all TDs in  $U^{unse}$  in descending order of their tasks' CPU demands;
- 3: **while**  $U^{unse} \neq \phi$  **do**
- 4:   Call Algorithm 2 to construct the set of full disks  $\mathcal{D}^{full};$
- 5:    $D^{mee} = \arg \min_{D \in \mathcal{D}^{full}} \left\{ \frac{E(D) + \sum_{i \in U_D^{dise}} E_{iD}^{bs} + \sum_{i \in U_D^{indise}} E_{iD}^{cc}}{|U_D^{dise} \cup U_D^{indise}|} \right\};$
- 6:   **if**  $D^{mee} \notin \tilde{\mathcal{D}}^{sele}$  **then**
- 7:      $\tilde{\mathcal{D}}^{sele} = \tilde{\mathcal{D}}^{sele} \cup \{D^{mee}\};$
- 8:     **for each**  $D \in \mathcal{D}(b(D^{mee}))$  **do**
- 9:       **if**  $r(D) > r(D^{mee})$  **then**
- 10:          $E(D) = E(D) - E(D^{mee});$
- 11:       **else**
- 12:          $E(D) = 0;$
- 13:      $U^{unse} = U^{unse} \setminus (U_{D^{mee}}^{dise} \cup U_{D^{mee}}^{indise});$
- 14:      $U_{b(D^{mee})}^{dise} = U_{b(D^{mee})}^{dise} \cup U_{D^{mee}}^{dise}, U_{b(D^{mee})}^{indise} = U_{b(D^{mee})}^{indise} \cup U_{D^{mee}}^{indise};$
- 15:      $\tilde{C}_{b(D^{mee})CPU} = \tilde{C}_{b(D^{mee})CPU} - \sum_{i \in U_{D^{mee}}^{dise}} d_{iCPU}, \tilde{C}_{b(D^{mee})BW} = \tilde{C}_{b(D^{mee})BW} - \sum_{i \in U_{D^{mee}}^{dise} \cup U_{D^{mee}}^{indise}} d_{iBW};$
- 16: **for each**  $b \in B$  **do**
- 17:   **if**  $(\mathcal{D}(b) \cap \tilde{\mathcal{D}}^{sele}) \neq \phi$  **then**
- 18:      $D_b^{max} = \arg \max_{D \in \mathcal{D}(b) \cap \tilde{\mathcal{D}}^{sele}} \{r(D)\};$
- 19:      $\mathcal{D}^{sele} = \mathcal{D}^{sele} \cup \{D_b^{max}\};$
- 20:      $U_{D_b^{max}}^{dise} = U_b^{dise}, U_{D_b^{max}}^{indise} = U_b^{indise};$
- 21: **return**  $\mathcal{D}^{sele}, \{U_D^{dise}\}, \{U_D^{indise}\}, \forall D \in \mathcal{D}^{sele}.$

In the *while* loop, lines 4, 5, and 7 call Algorithm 2 to construct the full disk set  $\mathcal{D}^{full}$  for this round, identify the most energy-effective disk  $D^{mee}$ , and add this disk to the temporarily selected disk set  $\tilde{\mathcal{D}}^{sele}$ . Then, lines 8–15 update the relevant data of the disks and BS related to  $D^{mee}$ . When calculating the energy effectiveness of the full disks, to ensure that the coverage energy of the selected disks is not repeatedly included, lines 8–12 use the following formula to update the energy consumption of the disks centered at BS  $b(D^{mee})$ :

$$\begin{cases} E(D) = E(D) - E(D^{mee}), & r(D) > r(D^{mee}) \text{ and } b(D) = b(D^{mee}) \\ E(D) = 0, & r(D) \leq r(D^{mee}) \text{ and } b(D) = b(D^{mee}) \end{cases}$$

Line 13 removes TDs that are directly or indirectly served by  $D^{mee}$  from  $U^{unse}$  to update the *while* loop end condition. Lines 14 and 15 update the sets of TDs served by BS  $b(D^{mee})$ , as well as the remaining CPU and bandwidth capacities. These operations ensure that the greedy strategy has no aftereffect, that is, the subsequent most energy-effective disk will not affect the previous most energy-effective disk and the TDs it serves.

---

**Algorithm 2** Algorithm for constructing full disks

---

**Input:** Ordered  $U^{\text{unse}}$  and tasks  $\{t_i\}$ ,  $\mathcal{D}$ ,  $B$  and the remaining resource capacity  $\{\tilde{C}_b\}$   
**Output:** Full disk set  $\mathcal{D}^{\text{full}}$

- 1: Initialize:  $\mathcal{D}^{\text{full}} = \phi$ ,  $\tilde{C}_{\text{DCPU}} = \tilde{C}_{b(D)\text{CPU}}$ ,  $\tilde{C}_{\text{DBW}} = \tilde{C}_{b(D)\text{BW}}$ ,  $U_D^{\text{dise}} = \phi$ ,  $U_D^{\text{indise}} = \phi$ ,  
 $\forall b \in B, \forall D \in \mathcal{D}(b)$ ;
- 2: **for** each  $b \in B$  **do**
- 3:     **for** each  $D \in \mathcal{D}(b)$  **do**
- 4:         **for** each  $i \in D \cap U^{\text{unse}}$  **do**
- 5:             **if**  $\tilde{C}_{\text{DBW}} \geq d_{i\text{BW}}$  **then**
- 6:                 **if**  $\tilde{C}_{\text{DCPU}} \geq d_{i\text{CPU}}$  **then**
- 7:                      $U_D^{\text{dise}} = U_D^{\text{dise}} \cup \{i\}$ ,  $\tilde{C}_{\text{DCPU}} = \tilde{C}_{\text{DCPU}} - d_{i\text{CPU}}$ ,  $\tilde{C}_{\text{DBW}} = \tilde{C}_{\text{DBW}} - d_{i\text{BW}}$ ;
- 8:                 **else**
- 9:                      $U_D^{\text{indise}} = U_D^{\text{indise}} \cup \{i\}$ ,  $\tilde{C}_{\text{DBW}} = \tilde{C}_{\text{DBW}} - d_{i\text{BW}}$ ;
- 10:             **if**  $U_D^{\text{dise}} \cup U_D^{\text{indise}} \neq \phi$  **then**
- 11:                  $\mathcal{D}^{\text{full}} = \mathcal{D}^{\text{full}} \cup \{D\}$ ;
- 12: **return**  $\mathcal{D}^{\text{full}}$ .

---

Algorithm 2 shows how a full disk is constructed. First, the sets of TDs that are directly or indirectly served by the disks in the current call are reset, and the resource capacities of the disks are initialized according to the remaining resource capacity of the corresponding BS (line 1). For each disk, lines 2–9 determine whether the resource demands of TDs covered but not served by the disk in this round can be satisfied by the disk. If the CPU and bandwidth requirements are met, the TD is directly served by the disk; if only the bandwidth requirements are met, the TD is indirectly served by the disk. Finally, Algorithm 2 returns the set of disks with service objects to Algorithm 1 as the full disk set.

4.2. An Illustrative Example of the Greedy Algorithm

We provide an example of how our greedy algorithm works. Consider a 100 m × 100 m area with 4 BSs and 10 TDs. Tables 3 and 4 list the parameters. The greedy algorithm executes six rounds of the *while* loop.  $D(b, 0)$ ,  $D(a, 7)$ ,  $D(c, 6)$ ,  $D(b, 8)$ ,  $D(d, 1)$ , and  $D(b, 5)$  are the most energy-effective disks found during each loop, and the TDs that they serve are  $U_{D(b,0)}^{\text{dise}} = \{0, 3\}$ ,  $U_{D(a,7)}^{\text{dise}} = \{7\}$ ,  $U_{D(c,6)}^{\text{dise}} = \{4, 6\}$ ,  $U_{D(b,8)}^{\text{indise}} = \{8\}$ ,  $U_{D(d,1)}^{\text{dise}} = \{1\}$ , and  $U_{D(b,5)}^{\text{dise}} = \{2, 5, 9\}$ . The radii of these five disks are 12.81 m, 12.81 m, 23.02 m, 26.68 m, 21.38 m, and 49.50 m, indicating that the most energy-effective disks are ordered from approximately smaller to larger radii. A total of three disks centered on BS b are temporarily selected. When the *while* loop ends,  $D(b, 5)$ , which has the largest radius, is selected as the final disk, and the TDs served by the other two disks become its service objects. Only one of the disks centered at BSs a, c, and d is temporarily selected, and all of these disks are selected as final disks. Therefore, the selected disks include  $D(a, 7)$ ,  $D(b, 5)$ ,  $D(c, 6)$ , and  $D(d, 1)$ ; that is, BSs a, b, c, and d maintain circular coverage areas with radii of 12.81 m, 49.50 m, 23.02 m, and 21.38 m, respectively. Figure 2 shows the BS coverage and offloaded TD tasks after the greedy algorithm is executed. The solid red dots indicate that the TDs are transferred to the BSs for processing, and the hollow red dots indicate that the TDs are relayed by the BSs and offloaded to the CC. As a comparison, we show the optimal solution obtained by CPLEX in Figure 3. The difference between the two methods is that in the optimal solution, BS a is not chosen, and TD 7 is indirectly served by  $D(b, 5)$  and then moved to the CC, allowing the optimal solution to consume less energy. The total energy consumption of the optimal solution and greedy algorithm is 4764.6 J and 4854.8 J, respectively, and in both methods, the CPU of BS b is fully allocated.

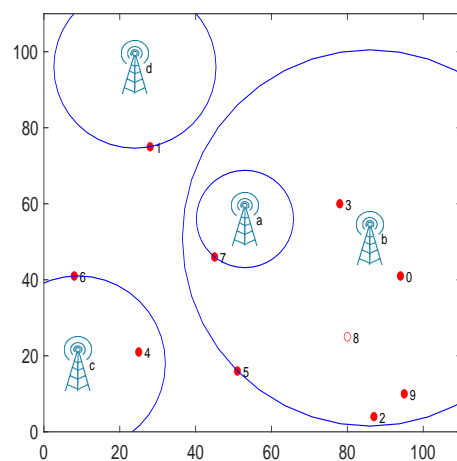


**Table 3.** BS parameters in the example.

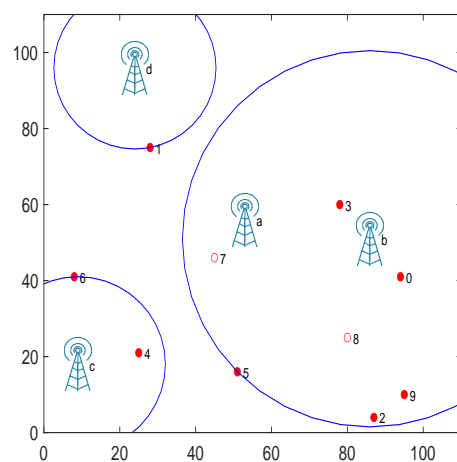
BS	$x$ (m)	$y$ (m)	$C_{bCPU}$ (Gcycle)	$C_{bBW}$ (MHz)	$f_b$ (GHz)	$p_b$ (W)
a	53	56	25	15.4	1.8	45
b	86	51	20	15.7	1.9	85
c	9	18	30	15.2	1.9	45
d	24	96	40	18.7	1.8	55

**Table 4.** TD parameters in the example.

TD	$x$ (m)	$y$ (m)	$q_i$ (MB)	$d_{iCPU}$ (Gcycle)	$d_{iBW}$ (MHz)	$e_{i1}$ (nJ/bit)	$e_{i1}$ (nJ/bit·m <sup>2</sup> )
0	94	41	3.67	5	2.38	40	9
1	28	75	4.78	7	3.66	40	9
2	87	4	3.83	5	5.98	40	11
3	78	60	4.58	7	1.05	40	9
4	25	21	0.49	3	0.47	50	10
5	51	16	0.3	2	1.15	50	8
6	8	41	0.19	3	0.92	40	11
7	45	46	0.25	1	0.71	50	10
8	80	25	0.17	3	1.58	50	8
9	95	10	0.66	1	0.25	40	9



**Figure 2.** Results of the greedy algorithm.



**Figure 3.** Results of the optimal solution solved by CPLEX.

#### 4.3. Analysis of Algorithm Correctness and Time Complexity

**Theorem 2.** The greedy algorithm can obtain a feasible solution for LP1 in polynomial time.

**Proof.** (1) *Correctness.* In the *while* loop in Algorithm 1, after the most energy-effective full disk  $D^{\text{mee}}$  is identified in a round, the TDs directly and indirectly served by  $D^{\text{mee}}$  must be removed from the set of unserved TDs  $U^{\text{unse}}$  (line 13), and the served TDs are added to the service object set of the corresponding BS (line 14). Therefore, Constraint (6a) holds. In lines 16–20 in Algorithm 1, when disk  $D_b^{\text{max}}$  with the largest BS radius is finally selected, the sets of TDs directly and indirectly served by BS  $b$  are regarded as the service objects of  $D_b^{\text{max}}$ , thereby ensuring that Constraints (6b) and (6c) hold. Lines 16–20 also show that for  $\forall b \in B$ , if there are multiple temporarily selected disks, only the disk with the largest radius is selected. If  $b$  does not have any temporarily selected disks,  $b$  is not enabled, thus guaranteeing that Constraint (6f) holds. The full disk construction process in Algorithm 2 ensures that the resource capacities of a full disk satisfy the resource demands of the TDs it serves and that Constraints (6d) and (6e) hold.

(2) *Polynomial time complexity.* At the beginning of Algorithm 1, line 1 takes  $O(mn)$  time to initialize, and line 2 takes at most  $O(n^2)$  time to sort the TDs in descending order according to their CPU demands. Next, the *while* loop executes at most  $n$  times. In the *while* loop, line 4 calls Algorithm 2, which spends at most  $O(mn^2)$  time constructing the set of full disks, line 5 takes  $O(mn)$  time to identify the most energy-effective full disk, and lines 6–15 take at most  $O(n^2)$  time to update the coverage energy of the disks centered at  $b(D^{\text{mee}})$ , the TDs that have not been served,  $b(D^{\text{mee}})$ 's service objects, and the remaining resource capacities. After the *while* loop, for  $\forall b \in B$ , Algorithm 1 needs at most  $O(mn)$  time to identify the disk with the largest radius from the set of temporarily selected disks and to update the TDs that are directly or indirectly served by that disk. Therefore, the time complexity of the greedy algorithm is  $O(mn^3)$ .  $\square$

### 5. Primal–Dual Algorithm for Cloud-Edge Collaborative Task Offloading

In this section, based on the primal–dual method devised in [9,43] and the disk structure proposed in this paper, we design a primal–dual cloud-edge collaborative task offloading algorithm for minimizing system energy consumption.

#### 5.1. Dual Programming

First, we relax the integer constraints (6g) and (6h) of the three primal variables in LP1, i.e.,

$$\begin{cases} x_{iD} \geq 0, z_{iD} \geq 0, \forall i \in D \in \mathcal{D} \\ y_D \geq 0, \quad \forall D \in \mathcal{D} \end{cases}$$

We introduce six dual variables  $\alpha_i, \beta_{iD}, \gamma_{iD}, \delta_D, \epsilon_D$ , and  $\mu_b$  for the six inequality constraints (6a)–(6f) in LP1 and three inequality constraints for the three primal variables  $x_{iD}, y_D$ , and  $z_{iD}$ . Thus, the dual program DP1 of LP1 can be obtained:

$$\max_{\alpha, \beta, \gamma, \delta, \epsilon, \mu} \sum_{i \in U} \alpha_i - \sum_{b \in B} \mu_b \tag{8}$$

$$\text{s.t. } \alpha_i - \beta_{iD} - d_{i\text{CPU}} \cdot \delta_D - d_{i\text{BW}} \cdot \epsilon_D \leq E_{iD}^{\text{bs}}, \forall i \in D \in \mathcal{D} \tag{8a}$$

$$\sum_{i \in D} \beta_{iD} + \sum_{i \in D} \gamma_{iD} + C_{\text{DCPU}} \cdot \delta_D + C_{\text{DBW}} \cdot \epsilon_D - \mu_b \leq E(D), \forall D \in \mathcal{D}(b), b \in B \tag{8b}$$

$$\alpha_i - \gamma_{iD} - d_{i\text{BW}} \cdot \epsilon_D \leq E_{iD}^{\text{cc}}, \forall i \in D \in \mathcal{D} \tag{8c}$$

$$\alpha_i \geq 0, \forall i \in U \tag{8d}$$

$$\beta_{iD} \geq 0, \gamma_{iD} \geq 0, \forall i \in D \in \mathcal{D} \tag{8e}$$

$$\delta_D \geq 0, \epsilon_D \geq 0, \forall D \in \mathcal{D} \tag{8f}$$

$$\mu_b \geq 0, \forall b \in B \tag{8g}$$

Transforming constraints (8a)–(8c) in DP1, we obtain:

$$\alpha_i \leq E_{iD}^{bs} + \beta_{iD} + d_{iCPU} \cdot \delta_D + d_{iBW} \cdot \epsilon_D, \forall i \in D \in \mathcal{D} \tag{8h}$$

$$\sum_{i \in D} \beta_{iD} + \sum_{i \in D} \gamma_{iD} + C_{DCPU} \cdot \delta_D + C_{DBW} \cdot \epsilon_D \leq E(D) + \mu_b, \forall D \in \mathcal{D}(b), b \in B \tag{8i}$$

$$\alpha_i \leq E_{iD}^{cc} + \gamma_{iD} + d_{iBW} \cdot \epsilon_D, \forall i \in D \in \mathcal{D} \tag{8j}$$

According to inequalities (8h)–(8j), we give the economic meanings of dual variables as follows.  $\alpha_i$  can be regarded as the cost budget paid by TD  $i$  to be served.  $\beta_{iD}$  and  $\gamma_{iD}$  represent the disk coverage energy costs shared by  $i$  when  $i$  is directly or indirectly served by disk  $D$ , respectively.  $\delta_D$  and  $\epsilon_D$  correspond to the unit prices of the CPU and bandwidth of disk  $D$ , respectively.  $\mu_b$  represents the total resource cost of BS  $b$ . Inequalities (8h) and (8j) indicate that  $\alpha_i$  covers the expenses of three aspects, namely, the energy cost of  $i$  being directly or indirectly served by the disk, the energy cost of the disk coverage, and the resource cost of  $i$ . Next, we construct the approximate optimal solution for the primal problem by determining a feasible solution for the dual problem.

### 5.2. Design of Primal–Dual Algorithm

The key idea of the primal–dual algorithm proposed in this paper is updating the dual variables to gradually tighten the corresponding dual constraints.

A new instance is constructed from the disk set  $\mathcal{D}$  and TD set  $U$  before calculating the feasible solution. First, we guess the disk  $D^{\max}$  with the largest radius in the optimal solution and use the full disk construction method in Algorithm 2 to obtain the sets  $U_{D^{\max}}^{\text{dise}}$  and  $U_{D^{\max}}^{\text{indise}}$  of TDs directly and indirectly served by  $D^{\max}$ . Then, all disks with radii larger than  $D^{\max}$  and all disks concentric with  $D^{\max}$  (including  $D^{\max}$  itself) are removed from  $\mathcal{D}$ , i.e.,  $\mathcal{D} \setminus (\{D | r(D) > r(D^{\max})\} \cup \{D | b(D) = b(D^{\max})\})$ . Moreover,  $U_{D^{\max}}^{\text{dise}}$  and  $U_{D^{\max}}^{\text{indise}}$  are removed from the set  $U$  of TDs that are not served, i.e.,  $U \setminus (U_{D^{\max}}^{\text{dise}} \cup U_{D^{\max}}^{\text{indise}})$ , and  $b(D^{\max})$  is removed from  $B$ , i.e.,  $B \setminus \{b(D^{\max})\}$ . Thus, we obtain a new instance. We need to determine whether the new instance has a feasible solution (namely, whether all disks can cover all TDs in the instance and whether the total BS bandwidth capacity satisfies the task demands of all TDs in the instance). If a feasible solution exists, we perform Algorithm 3 on the new instance. For convenience, the TD set, BS set, and disk set in the new instance are still denoted by  $U$ ,  $B$ , and  $\mathcal{D}$ , respectively.

In the process of constructing a feasible solution to dual program DP1, Algorithm 3 obtains the temporarily selected disk set  $\tilde{\mathcal{D}}^{\text{sele}}$  and the set of TDs served by these disks, allowing the primal–dual algorithm to construct a feasible solution to the primal program LP1. Initially, all TDs are not serviced, all disks are not selected, all dual variables are set to 0, and all flag variables are set to *false* and used to determine updates to the dual variables. In each round of the *while* loop, lines 3–15 update the dual variables. For each unserved TD  $i$ ,  $\alpha_i$  needs to be updated with a step length of  $l$  during each loop. With the repeated execution of the *while* loop and the continuous updating of  $\alpha_i$ ,  $\beta_{iD}$ , and  $\gamma_{iD}$ , three types of events corresponding to the three conditional statements occur (lines 16–21), namely, the BS offloading energy consumption in Constraint (8h), the disk coverage energy consumption in Constraint (8i), and the CC offloading energy consumption in Constraint (8j) are tightened. For different events, Algorithm 3 invokes the corresponding procedures. Note that in each round of the *while* loop, the three types of events may or may not occur at the same time.

---

**Algorithm 3** Algorithm for constructing a feasible dual solution of an instance.

---

**Input:** TD set  $U$  and their tasks  $\{t_i\}$ , disk set  $\mathcal{D}$ , BS set  $B$  and their resources capacity  $\{C_b\}$  in the instance

**Output:** Temporarily selected disks set  $\tilde{\mathcal{D}}^{\text{sele}}$  and set of TDs  $\{U_D^{\text{dise}}\}$  and  $\{U_D^{\text{indise}}\}$  they serve directly or indirectly

- 1: Initialize:  $U^{\text{unse}} = U, \tilde{\mathcal{D}}^{\text{sele}} = \phi; \forall i \in U, \alpha_i = 0; \forall i \in D \in \mathcal{D}, \beta_{iD} = 0, \gamma_{iD} = 0, \text{flag}_{iD}^\beta = \text{false}, \text{flag}_{iD}^\gamma = \text{false}; \forall D \in \mathcal{D}, U_D^{\text{dise}} = \phi, U_D^{\text{indise}} = \phi, \delta_D = 0, \epsilon_D = 0, \text{flag}_D^\delta = \text{false}, \text{flag}_D^\epsilon = \text{false}; \forall b \in B, \mu_b = 0, \tilde{C}_{b\text{CPU}} = C_{b\text{CPU}}, \tilde{C}_{b\text{BW}} = C_{b\text{BW}};$
  - 2: **while**  $U^{\text{unse}} \neq \phi$  **do**
  - 3:     **for each**  $i \in U^{\text{unse}}$  **do**
  - 4:          $\alpha_i = \alpha_i + l;$  //  $l$  is the step length
  - 5:         **for each**  $D : i \in D \in \mathcal{D}$  **do**
  - 6:             **if**  $\text{flag}_{iD}^\beta$  **then**
  - 7:                  $\beta_{iD} = \beta_{iD} + l;$
  - 8:             **if**  $\text{flag}_{iD}^\gamma$  **then**
  - 9:                  $\gamma_{iD} = \gamma_{iD} + l;$
  - 10:            **if**  $\text{flag}_D^\delta$  **then**
  - 11:                 $\delta_D = \delta_D + l;$
  - 12:            **if**  $\text{flag}_D^\epsilon$  **then**
  - 13:                 $\epsilon_D = \epsilon_D + l;$
  - 14:     **for each**  $b \in B$  **do**
  - 15:          $\mu_b = \max_{D \in \mathcal{D}(b)} \{C_{D\text{CPU}} \cdot \delta_D + C_{D\text{BW}} \cdot \epsilon_D\};$
  - 16:     **if**  $\exists i \in U^{\text{unse}}$  and  $\exists b \in B$  such that  $\alpha_i = E_{ib}^{\text{bs}}$  **then**
  - 17:         call Algorithm 4;
  - 18:     **if**  $\exists D \in \mathcal{D} \setminus \tilde{\mathcal{D}}^{\text{sele}}$  such that  $\sum_{i \in D} \beta_{iD} + \sum_{i \in D} \gamma_{iD} = E(D)$  **then**
  - 19:         call Algorithm 5;
  - 20:     **if**  $\exists i \in U^{\text{unse}}$  and  $\exists b \in B$  such that  $\alpha_i = E_{ib}^{\text{cc}}$  **then**
  - 21:         call Algorithm 6;
  - 22: **return**  $\tilde{\mathcal{D}}^{\text{sele}}$  and their  $\{U_D^{\text{dise}}\}$  and  $\{U_D^{\text{indise}}\}.$
- 

Algorithms 4–6 represent a series of operations corresponding to the three types of events. When the BS offloading energy consumption in Constraint (8h) is tightened, Algorithm 4 follows two cases. (1) If the temporarily selected disk set include a disk centered at  $b$  that covers  $i$ , the algorithm determines whether the remaining resources of  $b$  can satisfy the resource demands of  $i$ . If the CPU and bandwidth requirements of  $i$  are both met,  $i$  is served directly by that temporarily selected disk,  $\alpha_i$  and all  $\{\beta_{iD}\}_{i \in D \in \mathcal{D}}$  and  $\{\gamma_{iD}\}_{i \in D \in \mathcal{D}}$  stop increasing, and the remaining resource capacity of  $b$  is updated. If one of the resource requirements is not satisfied,  $\{\delta_D\}$  or  $\{\epsilon_D\}$  of the temporarily selected disks centered at  $b$  that cover  $i$  is increased accordingly. (2) If the temporarily selected disk set does not include a disk centered on  $b$  that covers  $i$  and satisfies  $i$ 's resource requirements, the algorithm attempts to find the disk with the smallest radius centered at  $b$  that covers  $i$  among the unselected disks, and determine whether the remaining resources of  $b$  satisfy the resource requirements of the unserved TDs covered by that disk. If both resource requirements are met,  $\{\beta_{iD}\}$  of the unselected disks centered on  $b$ , which covers  $i$ , increases. If one of the resource requirements is not satisfied,  $\{\delta_D\}$  or  $\{\epsilon_D\}$  of the unselected disks centered on  $b$  covering  $i$  is increased accordingly.

---

**Algorithm 4** Algorithm for event 1

---

```

1: if  $\exists D : i \in D \in \mathcal{D}(b) \cap \tilde{\mathcal{D}}^{\text{sele}}$  then
2:   if  $\tilde{C}_{b\text{CPU}} \geq d_{i\text{CPU}}$  and  $\tilde{C}_{b\text{BW}} \geq d_{i\text{BW}}$  then
3:      $U_D^{\text{dise}} = U_D^{\text{dise}} \cup \{i\}, \forall D : i \in D \in \mathcal{D}(b) \cap \tilde{\mathcal{D}}^{\text{sele}};$ 
4:      $U^{\text{unse}} = U^{\text{unse}} \setminus \{i\};$ 
5:      $flag_{iD}^{\beta} = false, flag_{iD}^{\gamma} = false, \forall D : i \in D \in \mathcal{D};$ 
6:      $\tilde{C}_{b\text{CPU}} = \tilde{C}_{b\text{CPU}} - d_{i\text{CPU}}, \tilde{C}_{b\text{BW}} = \tilde{C}_{b\text{BW}} - d_{i\text{BW}};$ 
7:   if  $\tilde{C}_{b\text{CPU}} < d_{i\text{CPU}}$  then
8:      $flag_D^{\delta} = true, \forall D : i \in D \in \mathcal{D}(b) \cap \tilde{\mathcal{D}}^{\text{sele}};$ 
9:   if  $\tilde{C}_{b\text{BW}} < d_{i\text{BW}}$  then
10:     $flag_D^{\epsilon} = true, \forall D : i \in D \in \mathcal{D}(b) \cap \tilde{\mathcal{D}}^{\text{sele}};$ 
11: if  $i \in U^{\text{unse}}$  and  $\exists D_b^{\text{min}} = \arg \min_{i \in D \in \mathcal{D}(b) \cap (\mathcal{D} \setminus \tilde{\mathcal{D}}^{\text{sele}})} \{r(D)\}$  then
12:   if  $\tilde{C}_{b\text{CPU}} \geq \sum_{j \in D_b^{\text{min}} \cap U^{\text{unse}}} d_{j\text{CPU}}$  and  $\tilde{C}_{b\text{BW}} \geq \sum_{j \in D_b^{\text{min}} \cap U^{\text{unse}}} d_{j\text{BW}}$  then
13:      $flag_{iD}^{\beta} = true, \forall D : i \in D \in \mathcal{D}(b) \cap (\mathcal{D} \setminus \tilde{\mathcal{D}}^{\text{sele}});$ 
14:   if  $\tilde{C}_{b\text{CPU}} < \sum_{j \in D_b^{\text{min}} \cap U^{\text{unse}}} d_{j\text{CPU}}$  then
15:      $flag_D^{\delta} = true, \forall D : i \in D \in \mathcal{D}(b) \cap (\mathcal{D} \setminus \tilde{\mathcal{D}}^{\text{sele}});$ 
16:   if  $\tilde{C}_{b\text{BW}} < \sum_{j \in D_b^{\text{min}} \cap U^{\text{unse}}} d_{j\text{BW}}$  then
17:      $flag_D^{\epsilon} = true, \forall D : i \in D \in \mathcal{D}(b) \cap (\mathcal{D} \setminus \tilde{\mathcal{D}}^{\text{sele}});$ 

```

---



---

**Algorithm 5** Algorithm for event 2

---

```

1:  $\tilde{\mathcal{D}}^{\text{sele}} = \tilde{\mathcal{D}}^{\text{sele}} \cup \{D\};$ 
2:  $U_D^{\text{dise}} = U^{\text{unse}} \cap \{i | \beta_{iD} > 0\}, U_D^{\text{indise}} = U^{\text{unse}} \cap \{i | \gamma_{iD} > 0\} \setminus U_D^{\text{dise}};$ 
3:  $U^{\text{unse}} = U^{\text{unse}} \setminus \{U_D^{\text{dise}} \cup U_D^{\text{indise}}\};$ 
4:  $flag_{iD'}^{\beta} = false, flag_{iD'}^{\gamma} = false, \forall i \in \{U_D^{\text{dise}} \cup U_D^{\text{indise}}\}, \forall D' : i \in D' \in \mathcal{D};$ 
5:  $\tilde{C}_{b(D)\text{CPU}} = \tilde{C}_{b(D)\text{CPU}} - \sum_{i \in U_D^{\text{dise}}} d_{i\text{CPU}}, \tilde{C}_{b(D)\text{BW}} = \tilde{C}_{b(D)\text{BW}} - \sum_{i \in U_D^{\text{dise}} \cup U_D^{\text{indise}}} d_{i\text{BW}};$ 

```

---



---

**Algorithm 6** Algorithm for event 3

---

```

1: if  $\exists D : i \in D \in \mathcal{D}(b) \cap \tilde{\mathcal{D}}^{\text{sele}}$  then
2:   if  $\tilde{C}_{b\text{BW}} \geq d_{i\text{BW}}$  then
3:      $U_D^{\text{indise}} = U_D^{\text{indise}} \cup \{i\}, \forall D : i \in D \in \mathcal{D}(b) \cap \tilde{\mathcal{D}}^{\text{sele}};$ 
4:      $U^{\text{unse}} = U^{\text{unse}} \setminus \{i\};$ 
5:      $flag_{iD}^{\beta} = false, flag_{iD}^{\gamma} = false, \forall i \in D \in \mathcal{D};$ 
6:      $\tilde{C}_{b\text{BW}} = \tilde{C}_{b\text{BW}} - d_{i\text{BW}};$ 
7:   else
8:      $flag_D^{\epsilon} = true, \forall D : i \in D \in \mathcal{D}(b) \cap \tilde{\mathcal{D}}^{\text{sele}};$ 
9: if  $i \in U^{\text{unse}}$  and  $\exists D_b^{\text{min}} = \arg \min_{i \in D \in \mathcal{D}(b) \cap (\mathcal{D} \setminus \tilde{\mathcal{D}}^{\text{sele}})} \{r(D)\}$  then
10:  if  $\tilde{C}_{b\text{BW}} \geq \sum_{j \in D_b^{\text{min}} \cap U^{\text{unse}}} d_{j\text{BW}}$  then
11:     $flag_{iD}^{\gamma} = true, \forall D : i \in D \in \mathcal{D}(b) \cap (\mathcal{D} \setminus \tilde{\mathcal{D}}^{\text{sele}});$ 
12:  else
13:     $flag_D^{\epsilon} = true, \forall D : i \in D \in \mathcal{D}(b) \cap (\mathcal{D} \setminus \tilde{\mathcal{D}}^{\text{sele}});$ 

```

---

When the coverage energy of the non-temporarily selected disk  $D$  in Constraint (8i) is tightened, Algorithm 5 sets  $D$  as the temporarily selected disk (line 1). Lines 2–5 classify the TDs contributing to  $E(D)$  into sets served directly or indirectly by  $D$  according to the values of  $\{\beta_{iD}\}$  and  $\{\gamma_{iD}\}$ , stop increasing the  $\alpha_i$  of each  $i$  in  $U_D^{\text{dise}} \cup U_D^{\text{indise}}$  and all  $\{\beta_{iD'}\}_{i \in D' \in \mathcal{D}}$  and  $\{\gamma_{iD'}\}_{i \in D' \in \mathcal{D}}$ , and update the remaining resource capacity of BS  $b(D)$  corresponding to the temporarily selected disk  $D$ .



Algorithm 6 also considers two cases when the CC offloading energy consumption in Constraint (8j) is tightened. In this case, only the bandwidth limitation of the BSs is involved, and the algorithm process is similar to that of Algorithm 4; thus, the explanation is not repeated.

When all TDs are provisionally served, Algorithm 7 constructs the final selected disk set  $\mathcal{D}^{\text{sele}}$  according to the provisionally selected disk set  $\tilde{\mathcal{D}}^{\text{sele}}$ . First, the temporarily selected disks are sorted in decreasing order according to their radii, and each temporarily selected disk is examined in turn. If there is a disk  $D$  in the final selected disk set that is concentric with the temporarily selected disk  $\tilde{D}$  that is being examined,  $D$  acts as the service provider for the TDs served by  $\tilde{D}$ , and the algorithm examines the next temporarily selected disk; otherwise,  $\tilde{D}$  is marked as the final selected disk.

---

**Algorithm 7** Algorithm for constructing a feasible primal solution of an instance

---

**Input:** Temporarily selected disk set  $\tilde{\mathcal{D}}^{\text{sele}}$  and their  $\{U_{\tilde{D}}^{\text{dise}}\}$  and  $\{U_{\tilde{D}}^{\text{indise}}\}$

**Output:** Final selected disk set  $\mathcal{D}^{\text{sele}}$  and their  $\{U_D^{\text{dise}}\}$  and  $\{U_D^{\text{indise}}\}$

- 1: Initialize:  $\mathcal{D}^{\text{sele}} = \phi$ ;
  - 2: Sort all disks in  $\tilde{\mathcal{D}}^{\text{sele}}$  in descending order according to their radii;
  - 3: **for** each  $\tilde{D} \in \tilde{\mathcal{D}}^{\text{sele}}$  **do**
  - 4:     **if**  $\exists D \in \mathcal{D}^{\text{sele}}$  and  $b(\tilde{D}) = b(D)$  **then**
  - 5:          $U_D^{\text{dise}} = U_D^{\text{dise}} \cup U_{\tilde{D}}^{\text{dise}}, U_D^{\text{indise}} = U_D^{\text{indise}} \cup U_{\tilde{D}}^{\text{indise}};$
  - 6:     **else**
  - 7:          $\mathcal{D}^{\text{sele}} = \mathcal{D}^{\text{sele}} \cup \{\tilde{D}\};$
  - 8: **return**  $\mathcal{D}^{\text{sele}}$  and their  $\{U_D^{\text{dise}}\}$  and  $\{U_D^{\text{indise}}\}$ .
- 

Finally, we obtain a feasible solution to the primal problem LP1 by merging  $\mathcal{D}^{\text{sele}}$  of the instance and  $D^{\text{max}}$ . After at most  $mn$  guesses, the scheme with the smallest objective function value is selected, which is the approximate optimal solution obtained by the primal–dual algorithm for LP1.

### 5.3. An Illustrative Example of the Primal–Dual Algorithm

We use the same data as in the greedy algorithm example to explain the execution of the primal–dual algorithm. In this example, at most  $4 \times 10$  guesses are required to find the  $D^{\text{max}}$  that minimizes energy consumption. According to the experimental results presented in Section 6, we use only a disk with a radius between the average radii of the disks selected by the optimal solution (31.3 m) and its value plus 25 m (56.3 m) as  $D^{\text{max}}$ . Therefore, 13 instances are constructed. When  $D(b, 2)$  is taken as  $D^{\text{max}}$ , the total energy consumption is the lowest, with a value of 5894.5 J. The following specifically describes the algorithm process when  $D(b, 2)$  is selected as  $D^{\text{max}}$ . First, the TDs that are directly or indirectly served by  $D(b, 2)$  are obtained by constructing the full disks, namely,  $U_{D(b,2)}^{\text{dise}} = \{0, 2, 3, 8\}$  and  $U_{D(b,2)}^{\text{indise}} = \{7, 9\}$ . Then, all disks centered on BS b and disks with radii greater than  $D(b, 2)$  are removed from the set of 40 disks, and  $\{0, 2, 3, 7, 8, 9\}$  are removed from the 10 TDs, yielding a new instance:  $U = \{1, 4, 5, 6\}$ ,  $\mathcal{D} = \{D(a, 7), D(a, 3), D(a, 1), D(a, 5), D(a, 8), D(a, 0), D(a, 4), D(c, 4), D(c, 6), D(c, 5), D(c, 7), D(d, 1)\}$ . This instance has a feasible solution. In the execution of the primal–dual algorithm, events 1 and 3 occur first; thus, the dual variables  $\beta_{4D(c,4)}, \beta_{5D(a,5)}, \beta_{5D(c,5)}, \gamma_{6D(c,6)}, \beta_{4D(a,4)}, \gamma_{5D(a,5)}, \gamma_{5D(c,5)}, \gamma_{4D(c,4)}, \beta_{1D(d,1)}$ , and  $\gamma_{4D(a,4)}$  start to gradually increase. Then, event 2 occurs, and the coverage energy consumption of  $D(c, 4)$ ,  $D(c, 6)$ ,  $D(c, 5)$ , and  $D(d, 1)$  are tightened successively. The four disks are provisionally selected and serve the TDs in  $U$ , namely,  $U_{D(c,4)}^{\text{dise}} = \{4\}$ ,  $U_{D(c,6)}^{\text{indise}} = \{6\}$ ,  $U_{D(c,5)}^{\text{dise}} = \{5\}$ , and  $U_{D(d,1)}^{\text{dise}} = \{1\}$ . Among the three temporarily selected disks centered on BS c, we finally choose  $D(c, 5)$ , which has the largest radius. The coverage of the selected three disks  $D(b, 2)$ ,  $D(c, 5)$ , and  $D(d, 1)$  and the offloading of the TDs are shown in Figure 4. Compared with the optimal solution and the greedy algorithm, the coverage radius of BS b is reduced, the coverage radius of BS c is increased, and TD 5’s task

is directly served by BS c. Although the average radius of the selected BS increases, the coverage of the TDs is more balanced.

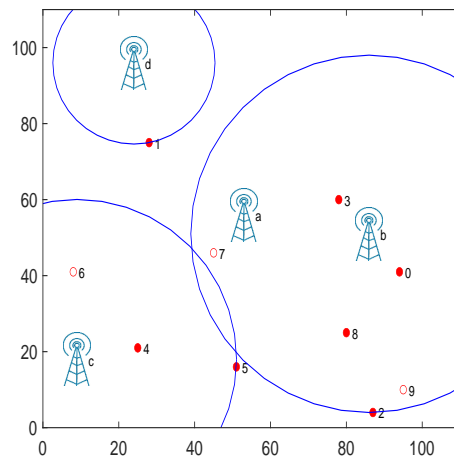


Figure 4. Results of the primal–dual algorithm.

5.4. Analysis of Algorithm Correctness and Time Complexity

**Theorem 3.** The primal–dual algorithm can obtain a feasible solution for LP1 in polynomial time.

**Proof.** (1) *Correctness.* In the three events in the primal–dual algorithm, the TDs can be served directly or indirectly only when their resource demands are satisfied by BSs corresponding to the temporarily selected disk covering the TDs (lines 1–4 in Algorithm 4, lines 1–3 in Algorithm 5, and lines 1–4 in Algorithm 6). Lines 3–5 in Algorithm 7 indicate that when a disk is finally selected, the service objects of other temporarily selected disks that are concentric with this disk and have smaller radii are served by that disk. Therefore, Constraints (6b)–(6e) of LP1 hold. The feasibility judgment before Algorithm 3 and the while loop ensure that Constraint (6a) holds. Lines 3–7 in Algorithm 7 ensure that at most one concentric disk is selected; thus, Constraint (6f) holds. Therefore, the primal–dual algorithm guarantees that the constraints of LP1 are valid in this instance. In addition, the construction process of the instance shows that  $D^{\max}$  and the set of TDs served by it, as well as the deleted disks, also satisfy the constraints of LP1. Therefore, the primal–dual algorithm can obtain a feasible solution for LP1.

(2) *Polynomial time complexity.* In Algorithm 7, line 1 takes  $O(1)$  time for initialization. Line 2 takes at most  $O(m^2n^2)$  time to sort all temporarily selected disks in descending order according to their radii. Lines 3–7 take at most  $O(m^2n)$  time to obtain the final selected disk set. Therefore, the time complexity of Algorithm 7 is at most  $O(m^2n^2)$ .

Next, we investigate whether Algorithm 3 can run in polynomial time. The TDs are served in the following three cases. In the first case, when event 1 occurs, there is a temporarily selected disk centered at  $b$  that covers  $i$ , and the two types of remaining resources for this disk satisfy  $i$ 's task demands (lines 1–4 in Algorithm 4). In the second case, when event 2 occurs, there are unserved TDs that contribute to the selection of  $D$ , i.e.,  $\beta_{iD} > 0$  or  $\gamma_{iD} > 0$  and  $i \in U^{\text{unse}}$  (lines 1–3 in Algorithm 5). In the third case, when event 3 occurs, there is a temporarily selected disk centered at  $b$  that covers  $i$ , and its remaining bandwidth satisfies  $i$ 's task requirements (lines 1–4 in Algorithm 6). Thus, we investigate the while loop rounds during which these three cases occur. Suppose that one of the above three cases is executed during the  $\hat{i}$ -th while loop, and let  $\hat{U}$  and  $\hat{\mathcal{D}}$  be the set of TDs served and the set of temporarily selected disks, respectively, at the end of the  $\hat{i}$ -th while loop. Then, we examine the next while loop  $\hat{i}^{\text{next}}$ , during which  $i \in U \setminus \hat{U}$  is served, and we consider three cases.

Case 1 In event 1, some TD is serviced in the next *while* loop, denoted as  $\hat{t}_1^{\text{next}}$ . For  $i \in U \setminus \hat{U}$  and  $b \in B$ , if  $D : i \in D \in \mathcal{D}(b) \cap \hat{\mathcal{D}}$  exists and  $\tilde{C}_{b\text{CPU}} \geq d_{i\text{CPU}}$  and  $\tilde{C}_{b\text{BW}} \geq d_{i\text{BW}}$ , then  $\hat{t}_1^{\text{next}} = \min_{i \in U \setminus \hat{U}, b \in B} \{\lfloor \frac{E_{ib}^{\text{bs}}}{T} \rfloor\}$ .

Case 2 In event 2, some TD is serviced in the next *while* loop, denoted as  $\hat{t}_2^{\text{next}}$ . We solve the following equation related to  $t$ -th loop:  $\sum_{i \in \hat{U} \cap D} \beta_{iD} + \sum_{i \in \hat{U} \cap D} \gamma_{iD} + \sum_{i \in (U \setminus \hat{U}) \cap D} \max\{t \cdot l - E_{iD}^{\text{bs}}, 0\} + \sum_{i \in (U \setminus \hat{U}) \cap D} \max\{t \cdot l - E_{iD}^{\text{CC}}, 0\} = E(D), \forall D \in \mathcal{D} \setminus \hat{\mathcal{D}}$ . For each  $D \in \mathcal{D} \setminus \hat{\mathcal{D}}$ , the solution  $\hat{t}_{2D}^{\text{next}}$  to the above equation can be obtained in polynomial time when  $\tilde{C}_{b(D)\text{CPU}} \geq \sum_{i \in D_{b(D)}^{\text{min}} \cap (U \setminus \hat{U})} d_{i\text{CPU}}$  and  $\tilde{C}_{b(D)\text{BW}} \geq \sum_{i \in D_{b(D)}^{\text{min}} \cap (U \setminus \hat{U})} d_{i\text{BW}}$  or  $\tilde{C}_{b(D)\text{BW}} \geq \sum_{i \in D_{b(D)}^{\text{min}} \cap (U \setminus \hat{U})} d_{i\text{BW}}$ , where  $D_{b(D)}^{\text{min}}$  represents the disk with the smallest radius among the temporarily unselected disks centered at  $b(D)$ . Thus,  $\hat{t}_2^{\text{next}} = \min_{D \in \mathcal{D} \setminus \hat{\mathcal{D}}} \{\hat{t}_{2D}^{\text{next}}\}$  can be obtained.

Case 3 In event 3, some TD is serviced in the next *while* loop, denoted as  $\hat{t}_3^{\text{next}}$ . For  $i \in U \setminus \hat{U}$  and  $b \in B$ , if  $D : i \in D \in \mathcal{D}(b) \cap \hat{\mathcal{D}}$  exists and  $\tilde{C}_{b\text{BW}} \geq d_{i\text{BW}}$ , then  $\hat{t}_3^{\text{next}} = \min_{i \in U \setminus \hat{U}, b \in B} \{\lfloor \frac{E_{ib}^{\text{CC}}}{T} \rfloor\}$ . Thus, the above analysis shows that the next time  $i \in U \setminus \hat{U}$  is served occurs during the  $\hat{t}^{\text{next}} = \min\{\hat{t}_1^{\text{next}}, \hat{t}_2^{\text{next}}, \hat{t}_3^{\text{next}}\}$ -th *while* loop.  $\hat{t}_1^{\text{next}}$ ,  $\hat{t}_2^{\text{next}}$  and  $\hat{t}_3^{\text{next}}$  can be obtained in polynomial time; therefore, the *while* loop can be completed in polynomial time.

In addition, the construction of a new instance takes at most  $O(mn)$  time. Thus, the primal–dual algorithm has polynomial time complexity. □

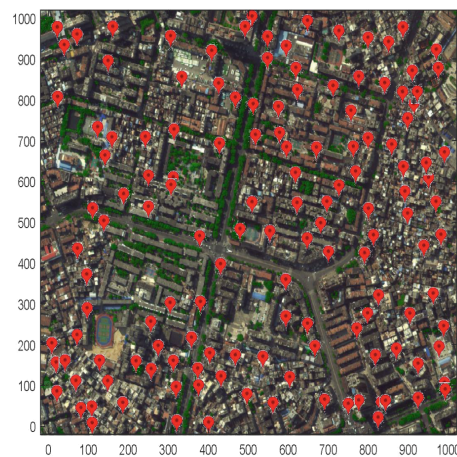
### 6. Performance Evaluation and Discussion

We implement the greedy algorithm and primal–dual algorithm using C++ in Microsoft Visual Studio 2019 and conduct extensive experiments to evaluate their performance. The relevant experimental parameters are shown in Table 5, and the specific experimental settings are described below.

- The hardware configuration of the experimental platform is a 48-core Intel Xeon E7-4850 v2 2.3 GHz CPU with 512 GB memory and 557 GB disk storage.
- The BS and TD coordinates used in the experiments are derived from live network site coordinates and the coverage grid data in question D of the 2022 MathorCup University Mathematical Modeling Challenge [44]. Figure 5 shows the distribution of 139 BSs within a 1.0 km × 1.0 km region of the data source.
- The CPU and frequency ranges of the virtual machines in the CC and the BSs refer to the instance specification family of Alibaba ECS [45]. The CPU requirements of the TDs are derived from 2018 Alibaba cluster data [46]. The bandwidth requirements of the TDs are similar to those in the literature [47] and are generated according to the formula  $d_{i\text{CPU}} \cdot \bar{f}_{\text{bs}} = X \cdot d_{i\text{BW}}$ , where  $\bar{f}_{\text{bs}}$  is the average computing power of the BSs, and  $X$  is a random variable conforming to the  $\gamma$  distribution.
- Each group of experiments was performed 30 times with randomly selected samples, and the results were averaged to reduce the influence of randomness.
- We compare the approximate solutions obtained by the two methods proposed in this paper with the optimal solutions obtained using IBM ILOG CPLEX Optimization Studio 20.1.0 to illustrate the approximate performance of the algorithms.

**Table 5.** Experimental parameters.

Parameters	Description	Value
$C_{bCPU}$	CPU capacity of BS $b$	[121, 243] Gcycles
$C_{bBW}$	Bandwidth capacity of BS $b$	[100, 200] MHz
$f_b$	CPU frequency of BS $b$	[1.8, 2.8] GHz
$f_{vm}^{CC}$	CPU frequency of virtual machines in the CC	[2.5, 3.8] GHz
$p_b$	CPU power of BS $b$	[35, 135] W
$p_{vm}^{CC}$	CPU power of virtual machines in the CC	[85, 150] W
$q_i$	Input data size of $t_i$	[0.1, 5] MB
$d_{iCPU}$	CPU demand of $t_i$	[1, 10] Gcycle
$e_{i1}$	Energy consumption per bit sent by $i$	[40, 60] nJ/bit
$e_{i2}$	Energy consumption per bit transmitted per square meter by $i$	[8, 12] nJ/bit·m <sup>2</sup>
$e_{wired}$	Wired transmission energy consumption coefficient of the Internet	0.06 kWh/GB
$l$	Step size for updating the dual variables	1
$x$	Abscissa of BS and TD locations	[0, 1000] m
$y$	Ordinate of BS and TD locations	[0, 1000] m



**Figure 5.** BS distribution within the 1.0 km × 1.0 km region.

6.1. Performance with Different Numbers of TDs

In this experiment, we examine the performance of the algorithms with different numbers of TDs in terms of the total energy consumption, execution time, BS radii, resource utilization, and approximation ratio in a 500 m × 500 m region.

Figure 6 illustrates that as the number of TDs increases, the total energy consumption of the system increases continuously with all three methods. This is because the increase in the number of TDs causes the increase in the computing and transmission energy costs of task offloading. Moreover, to serve more TDs, the coverage regions of some BSs expand, which consumes more coverage energy. The energy consumption of the optimal solution increases the slowest, followed by the greedy algorithm and the primal–dual algorithm. When the number of TDs reaches 500, CPLEX cannot obtain the optimal solution within 2 h, so the energy consumption results are omitted for this case.

Figure 7 shows that CPLEX takes considerably longer to calculate the optimal solution than our two approximation algorithms. For the same number of TDs, the greedy algorithm obtains an approximate solution faster than the primal–dual algorithm, which fully reflects the advantage of the greedy algorithm in solving efficiency. The dual variables are gradually updated in the unit of step size, and the corresponding event occurs only when the dual variables reach certain values. The step size affects the solving speed, the accuracy of the results and whether the solution can be obtained. In this paper, the step size is set to 1.

Figure 8 shows the proportion of TDs directly served by the BSs, that is, the ratio of the number of TDs offloaded to the BSs for processing to the total number of TDs. In the first three groups, the optimal solution completely offloads the tasks to the BSs, and

when the number of TDs is 200, the ratio still reaches 99.1%. The BS offloading ratio of the primal–dual algorithm exceeds 92.5%; thus, this algorithm not only makes full use of BS resources but also meets the needs of delay-sensitive tasks. The BS offloading ratio of the greedy algorithm decreases rapidly as the number of TDs increases from 50 to 500 because the greedy algorithm fully utilizes the resources of each most energy-effective disk. Thus, even if its CPU is insufficient, as long as the bandwidth is sufficient, the TDs covered by that disk should be offloaded to the CC instead of considering other disks with sufficient resources. As a result, as the number of TDs increases, many TDs cannot be moved to BSs for execution.

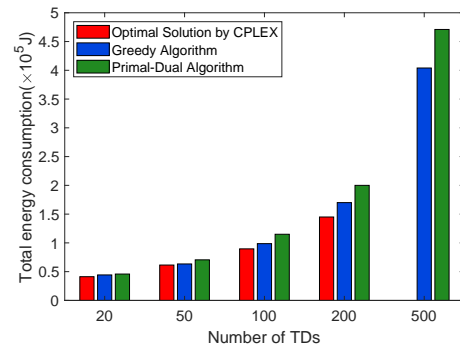


Figure 6. Total energy consumption under different numbers of TDs.

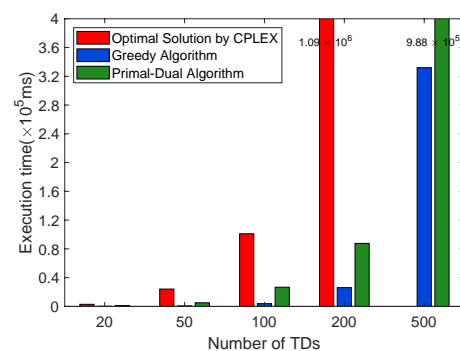


Figure 7. Execution time under different numbers of TDs.

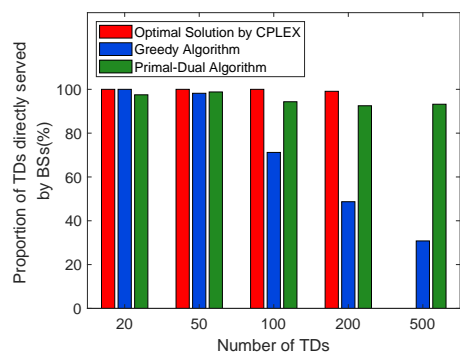


Figure 8. Proportion of TDs directly served by BSs under different numbers of TDs.

In this experiment, 25 BSs were available in the system. Figure 9 shows the average number of BSs launched with different numbers of TDs for the three methods. To handle more tasks, the number of BSs selected by the three methods increases. Among them, the greedy algorithm selects the most BSs, followed by the primal–dual algorithm. To cover more TDs, the average coverage radius of the BSs selected by the primal–dual algorithm is larger than that of the greedy algorithm, as shown in Figure 10. The optimal solution not only selects a small number of BSs but also has a smaller average coverage radius, resulting in reduced coverage energy consumption.



Recall from Section 3.2 that to ensure that the disk with the largest radius corresponding to each BS is valid, we assume that each BS should be able to supply energy that can cover the TD farthest from it. However, in fact, BSs can completely cover all TDs without providing so much energy. From Figure 11, even if the number of TDs reaches 500, the primal–dual algorithm enables each selected BS to boot with a coverage radius of 200 m at most, and it only needs to provide energy of no more than  $c \cdot 200^{\theta}$ . The maximum coverage radii of the BSs calculated by CPLEX and the greedy algorithm are smaller. In this group of experiments, the distance between the BS and TD is at most  $500\sqrt{2}$  m.

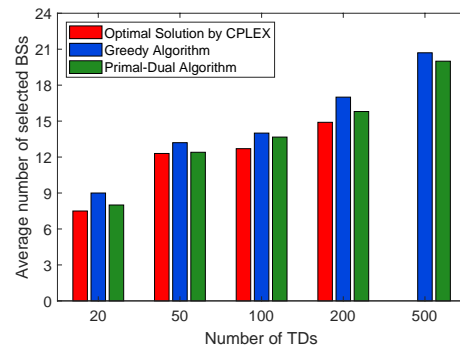


Figure 9. Average number of selected BSs under different numbers of TDs.

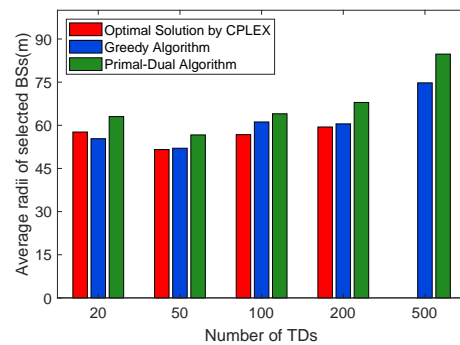


Figure 10. Average coverage radii of selected BSs under different numbers of TDs.

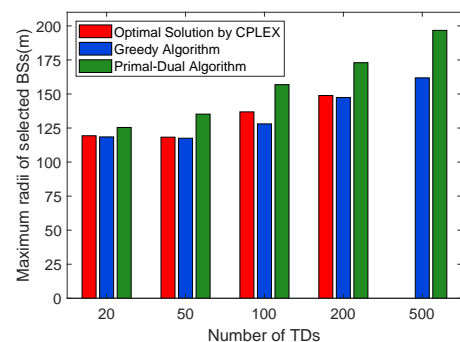


Figure 11. Maximum coverage radii of selected BSs under different numbers of TDs.

Figure 12 shows the effect of changing the number of TDs on CPU and bandwidth utilization. The resource utilization ratio of a single BS is defined as the ratio of its allocated resources to its resource capacity. Therefore, the resource utilization ratio of a system is the ratio of the sum of the resource utilization ratios of all selected BSs to the number of selected BSs. The CPU utilization trends of the three methods are similar. When the number of TDs reaches 500, more than 50% of the BS computing resources in the system are still available. Compared with the other two methods, the bandwidth utilization changes considerably with the greedy algorithm, especially when the number of TDs increases from 200 to 500,

with the bandwidth utilization increasing sharply. This increase occurs because when a full disk is constructed, the algorithm tends to make that disk serve all the TDs it covers, thereby maximizing the allocation of resources for each disk, especially the bandwidth. Thus, the most energy-effective disk selected in each *while* loop has a relatively high load, and this phenomenon becomes more apparent as the number of TDs increases.

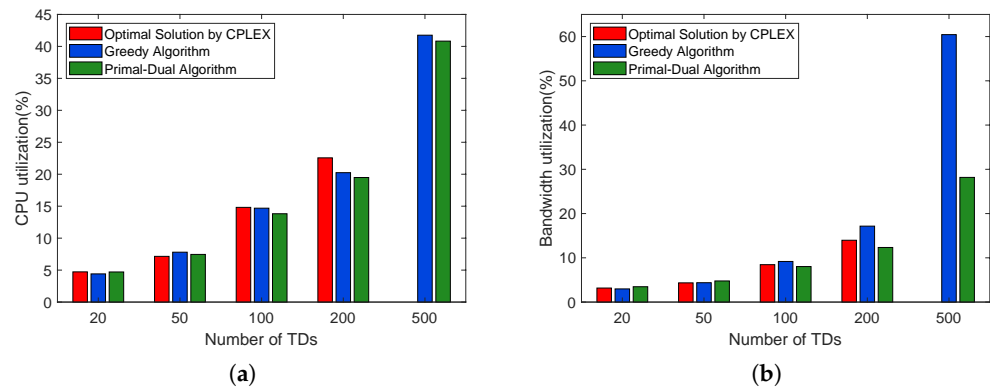


Figure 12. Resource utilization of BSs under different numbers of TDs: (a) CPU and (b) bandwidth.

The approximation ratio is an important performance metric to evaluate the approximation algorithms, which represents the gap between the approximation solution and the optimal solution of the problem. Here, we verify the approximation ratio of the two algorithms through experiments. For minimization of the energy consumption problem, the approximation ratio is the upper bound of the ratio of the energy consumption obtained by the greedy algorithm or the primal–dual algorithm to the energy consumption obtained by the optimal solution of LP1. In Figure 13, since the greedy algorithm produces less energy consumption, its approximation ratio is lower than that of the primal–dual algorithm. With the increase in TDs, the approximation ratio curves of the two algorithms show a slow upward trend, and the two curves gradually become parallel and approach a certain value, respectively.

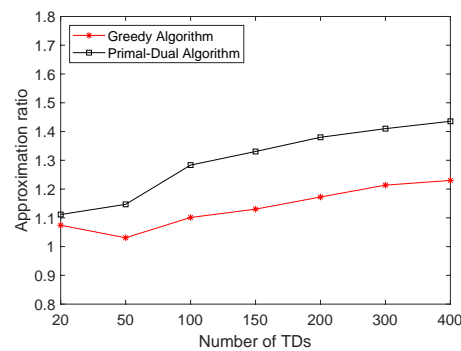


Figure 13. Approximation ratio under different numbers of TDs.

### 6.2. Performance with Different Numbers of BSs

In this experiment, we examine the performance of the three methods under different numbers of BSs in a 500 m × 500 m area, with the number of TDs set to 100.

Figure 14 shows that as the number of BSs in the region increases, the total energy consumption of the three methods decreases rapidly. As the number of BSs increases, the resource capacity of the system increases, and the demands of the TDs can be met by nearby disks with smaller radii; thus, the average radius of the selected BS decreases. When the number of BSs reaches 15, the energy consumption of our two methods is close to that of the optimal solution.

Figure 15 shows that CPLEX takes the longest time to calculate the optimal solution. We find that the change in the number BSs has no obvious effect on the two approximation algorithms we designed.

Figure 16 shows that in the optimal solution, almost all TDs are directly served by BSs, resulting in low energy consumption. In the greedy algorithm and primal–dual algorithm, the proportion of TDs offloaded to BSs increases slowly as the number of BSs increases. Since the greedy algorithm makes full use of the bandwidth of each energy-effective disk, more than 20% of the tasks are still offloaded to the CC, even if other BS resources remain abundant. When the number of BSs reaches 8, 90% or more of the TDs in the primal–dual algorithm are directly served by BSs, and the energy consumption decreases rapidly at this time.

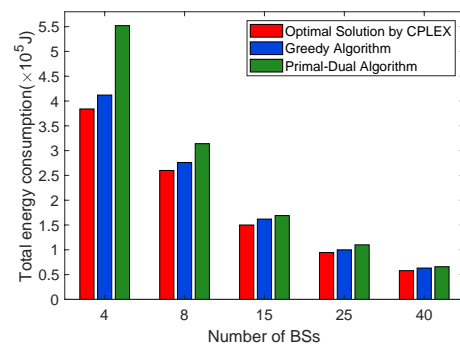


Figure 14. Total energy consumption under different numbers of BSs.

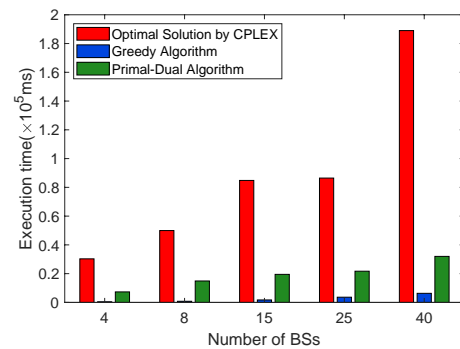


Figure 15. Execution time under different numbers of BSs.

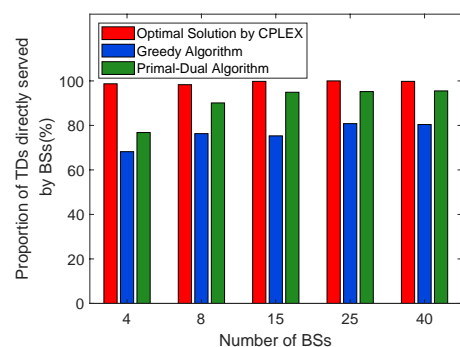


Figure 16. Proportion of TDs directly served by BSs under different numbers of BSs.

Figure 17 illustrates that the average number of activated BSs increases with the number of BSs in all three methods, which is the opposite of the results shown in Figure 18. The radii of the selected BSs decrease, but the number of TDs to be served remains unchanged; thus, more small-radius BSs must be selected. Figures 14 and 18 show similar change trends, which reflects that the coverage radii of BSs has a significant impact on energy

consumption. Figure 19 shows that the maximum coverage area supported by the booted BSs obtained by the three algorithms has little difference. When fewer BSs are deployed, the maximum radii of BSs are less than half of the maximum distance between BSs and TDs.

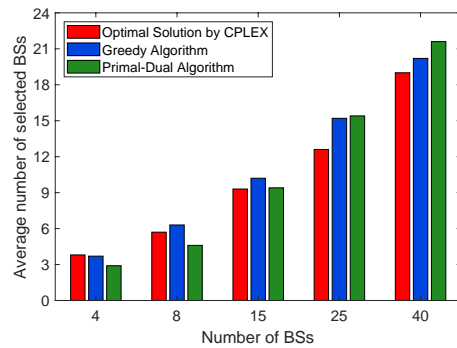


Figure 17. Average number of selected BSs under different numbers of BSs.

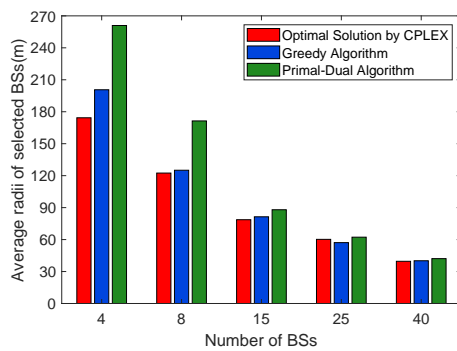


Figure 18. Average coverage radii of the selected BSs under different numbers of BSs.

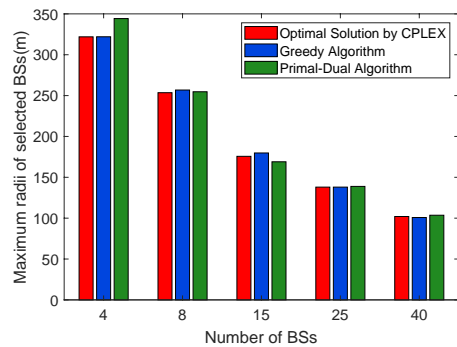


Figure 19. Maximum coverage radii of selected BSs under different numbers of BSs.

Figure 20 shows the impact of the number of BSs on CPU and bandwidth utilization. As the number of BSs increases, the number of TDs served by a single BS decreases, and the resources allocated to a single BS also decrease. The utilization of the two types of resources presents a decreasing trend in all three methods, and their resource utilization trends are similar when the number of BSs exceeds 15.

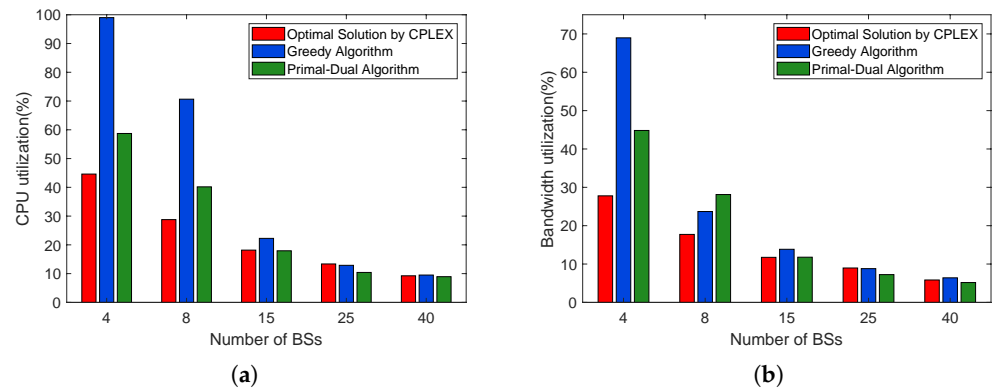


Figure 20. Resource utilization of the BSs under different numbers of BSs: (a) CPU and (b) bandwidth.

### 6.3. Performance with Different Areas

In this experiment, we analyze the effect of area size on the performance of the three methods, with the numbers of TDs and BSs set to 100 and 25, respectively. As shown in Figure 21, the TDs and BSs are more dispersed as the area increases, and more large-radius BSs need to be activated to fully cover all the TDs. Therefore, the energy consumption of the three methods increases sharply with increasing area. In particular, when the area changes from  $500\text{ m} \times 500\text{ m}$  to  $800\text{ m} \times 800\text{ m}$ , the average radius of the selected BS increases rapidly, and the energy consumption changes substantially.

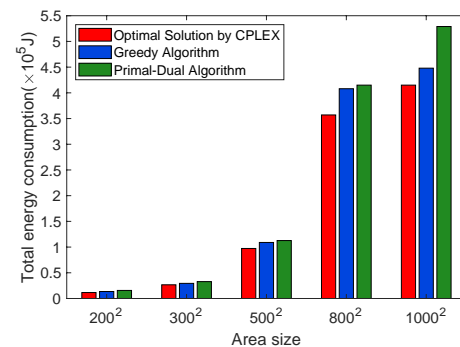


Figure 21. Total energy consumption under different areas.

Figure 22 shows that increasing the region has a considerable impact on the execution times of the optimal solution and primal–dual algorithm; however, this increase has little effect on the execution time of the greedy algorithm. To guarantee event 2, which involves a disk with a larger radius, the *while* loop needs to iterate more times, thereby increasing the execution time of the primal–dual algorithm.

Figure 23 illustrates that regardless of how the area changes, as long as the BS resources are sufficient, the optimal solution obtained by CPLEX ensures that all TDs are directly served by BSs. As the region becomes larger, the number of TDs directly served by BSs in the primal–dual algorithm gradually increases. When the area is small, for a small number of TDs, disks with bandwidths and CPUs that can meet their requirements are selected later than disks with only bandwidths that satisfy their requirements; thus, these TDs can only be served indirectly by the disks. When the area gradually expands and more BSs are selected (Figure 24), disks that satisfy both resource requirements of the TDs are selected first; thus, these TDs are served directly by the disks. As the region expands, less TDs are directly served by BSs in the greedy algorithm, which may occur because the bandwidths of the BSs satisfy the demands of some TDs.

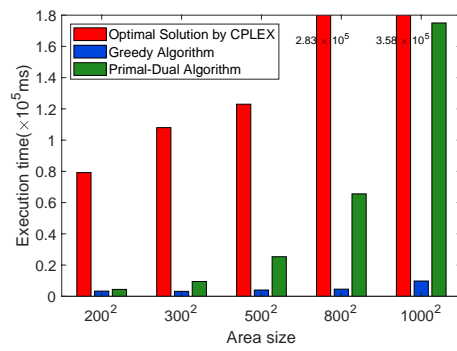


Figure 22. Execution time under different areas.

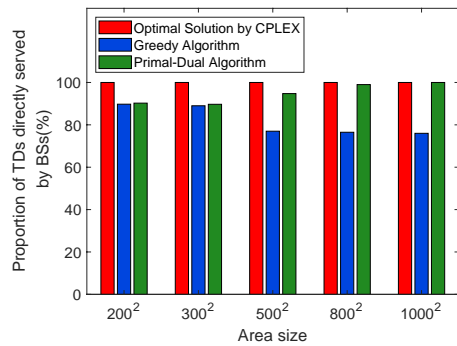


Figure 23. Proportion of TDs directly served by BSs under different areas.

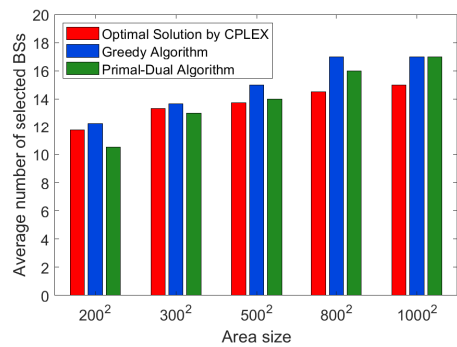


Figure 24. Average number of selected BSs under different areas.

Figures 24 and 25 show that to serve TDs distributed over a wider area, more BSs should be activated, and wider coverage radii should be supported by the BSs. Figure 21 is similar to Figures 25 and 26, demonstrating that changes in the coverage radii of the BSs have a substantial impact on the total energy consumption. In Figure 26, limited by the objective function, even if the considered range extends to 1 km<sup>2</sup>, the selected BSs only need to provide energy at most  $c \cdot 300^{\theta}$ , which is enough to completely cover all TDs. It is unnecessary for the BSs to guarantee the maximum energy we assumed above to cover the farthest TDs from them, because these TDs are already covered and served by other BSs closer to them.



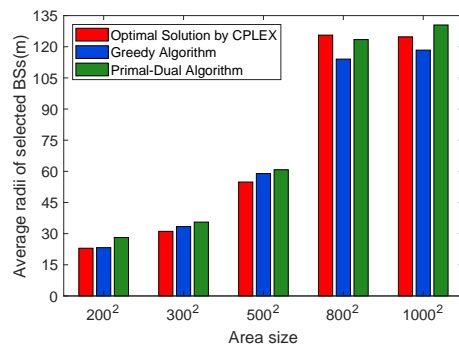


Figure 25. Average coverage radii of the selected BSs under different areas.

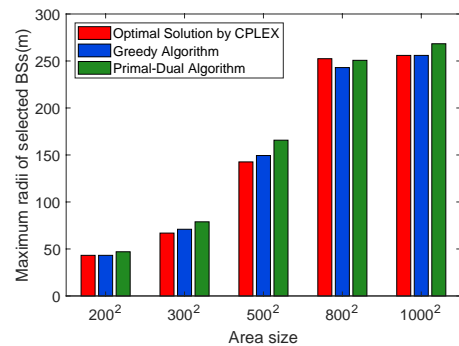


Figure 26. Maximum coverage radii of selected BSs under different areas.

As the number of activated BSs increases, the average number of TDs directly served by a single BS decreases. Therefore, the CPU and bandwidth utilization of the BSs decreases with increasing area, as shown in Figure 27. Additionally, as the region becomes larger, the selected disks do not cover as many TDs; thus, the load imbalance among the selected BSs in the greedy algorithm is alleviated.

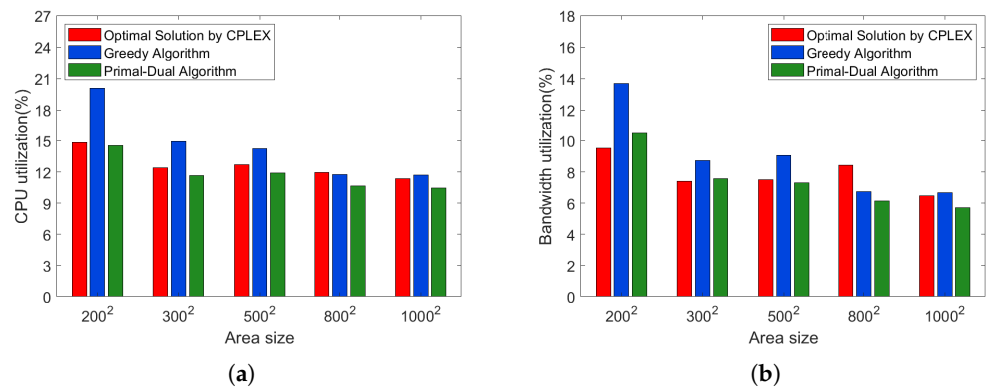


Figure 27. Resource utilization of BSs under different areas: (a) CPU. (b) bandwidth.

In the above experiments, compared with the optimal solution obtained by CPLEX, the greedy algorithm and the primal–dual algorithm achieve good approximate performance in terms of energy consumption. In addition, the two algorithms have distinct characteristics due to their different design principles. The greedy algorithm obtains a lower total energy consumption than the primal–dual algorithm in a shorter time. However, each most energy-effective disk selected by the greedy strategy must serve as many TDs in the coverage region as possible, and disks that are selected earlier have heavier loads, resulting in unbalanced loads among the selected BSs with the greedy algorithm. In the primal–dual algorithm, due to the high proportion of TDs served directly by BSs, few TDs are offloaded to the CC, which effectively relieves bandwidth pressure in the core network. Moreover,

the primal–dual algorithm selects disks when the dual constraints are tightened, and the task load is dispersed among the selected BSs; thus, in general, there is no load imbalance except that of  $D^{\max}$ .

## 7. Conclusions

This work focuses on the energy consumption of BS coverage and the computing and transmission energy costs associated with offloading tasks to BSs and CCs, and discusses cloud-edge collaborative task offloading in smart city applications with multiple BSs and TDs. First, inspired by Equation (1), a complete coverage model with adjustable BS radii that maps each BS to a series of disks with different radii is proposed, and the task offloading problem is formulated as an integer linear program. Then, since the optimal solution to the task offloading problem with complete coverage and multidimensional resource constraints is difficult to obtain, a greedy algorithm and a primal–dual algorithm are designed, both of which can obtain approximate solutions to the problem in polynomial time. Finally, we perform experiments to examine the effect of the number of TDs, the number of BSs, and the region size on the performance of the algorithms. In contrast to previous research on BS energy consumption, this paper proposes the novel concept of adjustable BS radii. Although some methods have proposed BSs that sleep or deactivate, these approaches are limited to cases in which the BSs are not activated or are activated with fixed supporting energy, and changes in the BS radius after the BSs are activated have not been discussed.

However, our model discusses task offloading in short time intervals when TDs remain relatively stationary, and does not consider the mobility of TDs in real-world scenarios. Thus, our future work will consider expanding the model to multiple time intervals, in which TDs remain stationary within one time interval or move between different time intervals. Unfortunately, the mobility of TDs may cause TDs to frequently switch between BSs, which affects task offloading and results in additional costs. One solution is to break the restriction that TDs are only served by exactly one BS in this paper and allow TDs to be served by multiple BSs simultaneously. In this case, when a TD moves from the coverage region of one BS to that of another BS, task offloading can still be achieved, thereby guaranteeing the quality of service. However, this approach will inevitably lead to an increase in energy consumption and a waste of resources, which requires a trade-off between service quality and cost.

In addition, unlike our work on low-carbon energy conservation in cloud-edge collaboration from the perspective of consumption, scholars have carried out extensive studies on the sustainable utilization of energy from the perspective of generation and have achieved fruitful results. For example, new advancements in energy harvesting technologies can facilitate the energy supply and transfer of entities at different end-edge-cloud levels, forming the so-called renewable-energy-powered cloud-edge system [4]. On the one hand, the TDs in such systems can eliminate the dilemma of battery power to a certain extent through the support of renewable energy. On the other hand, it is reasonable and feasible to flexibly adjust BS coverage by combining energy harvesting with the model proposed in this paper. This opens up another avenue for future energy-aware research.

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

**Funding:** This research was supported in part by the National Natural Science Foundation of China (Nos. 12071417, 62266051, and 62062065), the Scientific Research Fund Project of Yunnan Education Department (No. 2022J0002) and the 12th Postgraduate Innovation Project of Yunnan University (No. 2020294).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The code and data involved in the experiments can be obtained by emailing the first author.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Hajam, S.S.; Sofi, S.A. IoT-Fog architectures in smart city applications: A survey. *China Commun.* **2021**, *18*, 117–140. [CrossRef]
2. Kirmat, A.; Krejcar, O.; Kertesz, A.; Tasgetiren, M.F. Future trends and current state of smart city concepts: A survey. *IEEE Access* **2020**, *8*, 86448–86467. [CrossRef]
3. GSMA. The Mobile Economy 2020. Available online: [https://www.gsma.com/mobileeconomy/wp-content/uploads/2020/03/GSMA\\_MobileEconomy2020\\_Global.pdf](https://www.gsma.com/mobileeconomy/wp-content/uploads/2020/03/GSMA_MobileEconomy2020_Global.pdf) (accessed on 3 September 2022).
4. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A survey on mobile edge computing: The communication perspective. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2322–2358. [CrossRef]
5. Abbas, N.; Zhang, Y.; Taherkordi, A.; Skeie, T. Mobile edge computing: A survey. *IEEE Internet Things J.* **2018**, *5*, 450–465. [CrossRef]
6. Khan, W.Z.; Ahmed, E.; Hakak, S.; Yaqoob, I.; Ahmed, A. Edge computing: A survey. *Future Gener. Comput. Syst.* **2019**, *97*, 219–235. [CrossRef]
7. Liu, X.; Li, W.; Dai, H. Approximation algorithms for the minimum power cover problem with submodular/linear penalties. *Theor. Comput. Sci.* **2022**, *923*, 256–270. [CrossRef]
8. Chang, K.C.; Chu, K.C.; Wang, H.C.; Lin, Y.C.; Pan, J.S. Energy saving technology of 5G base station based on internet of things collaborative control. *IEEE Access* **2020**, *8*, 32935–32946. [CrossRef]
9. Li, M.; Ran, Y.; Zhang, Z. A primal-dual algorithm for the minimum power partial cover problem. *J. Comb. Optim.* **2020**, *44*, 1913–1923. [CrossRef]
10. You, C.; Huang, K.; Chae, H.; Kim, B.H. Energy-efficient resource allocation for mobile-edge computation offloading. *IEEE Trans. Wirel. Commun.* **2017**, *16*, 1397–1411. [CrossRef]
11. Zhang, P.; Yang, J.; Fan, R. Energy-efficient mobile edge computation offloading with multiple base stations. In Proceedings of the 15th International Wireless Communications and Mobile Computing Conference (IWCMC 2019), Tangier, Morocco, 24–28 June 2019; pp. 255–259. [CrossRef]
12. Yang, X.; Yu, X.; Huang, H.; Zhu, H. Energy efficiency based joint computation offloading and resource allocation in multi-access MEC systems. *IEEE Access* **2019**, *7*, 117054–117062. [CrossRef]
13. Zhang, K.Y.; Gui, X.L.; Ren, D.W.; Li, J.; Wu, J.; Ren, D.S. Survey on computation offloading and content caching in mobile edge networks. *J. Softw.* **2019**, *30*, 2491–2516. [CrossRef]
14. Jiang, C.; Fan, T.; Gao, H.; Shi, W.; Liu, L.; Cérin, C.; Wan, J. Energy aware edge computing: A survey. *Comput. Commun.* **2020**, *151*, 556–580. [CrossRef]
15. Zhang, J.; Yang, X.; Xie, N.; Zhang, X.; Li, W. An online auction mechanism for time-varying multidimensional resource allocation in clouds. *Future Gener. Comput. Syst.* **2020**, *111*, 27–38. [CrossRef]
16. Zhang, X.; Li, J.; Li, G.; Li, W. Generalized asset fairness mechanism for multi-resource fair allocation mechanism with two different types of resources. *Clust. Comput.* **2022**, *25*, 3389–3403. [CrossRef]
17. Zhao, Y.; Liu, H.; Gao, K. An evacuation simulation method based on an improved artificial bee colony algorithm and a social force model. *Appl. Intell.* **2021**, *51*, 100–123. [CrossRef]
18. Ebrahimnejad, S.; Harifi, S. An optimized evacuation model with compatibility constraints in the context of disability: An ancient-inspired Giza Pyramids Construction metaheuristic approach. *Appl. Intell.* **2022**, *52*, 15040–15073. [CrossRef]
19. Han, Y.; Hu, H.; Guo, Y. Energy-aware and trust-based secure routing protocol for wireless sensor networks using adaptive genetic algorithm. *IEEE Access* **2022**, *10*, 11538–11550. [CrossRef]
20. Vazirani, Vijay V. *Approximation Algorithms*; Springer: Berlin/Heidelberg, Germany, 2001.
21. IBM. IBM ILOG CPLEX Optimizer. Available online: <https://www.ibm.com/analytics/cplex-optimizer> (accessed on 3 September 2022).
22. Wang, Y.; Sheng, M.; Wang, X.; Wang, L.; Li, J. Mobile-edge computing: Partial computation offloading using dynamic voltage scaling. *IEEE Trans. Commun.* **2016**, *64*, 4268–4282. [CrossRef]
23. You, C.; Huang, K.; Chae, H. Energy efficient mobile cloud computing powered by wireless energy transfer. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 1757–1771. [CrossRef]
24. Cao, X.; Wang, F.; Xu, J.; Zhang, R.; Cui, S. Joint computation and communication cooperation for energy-efficient mobile edge computing. *IEEE Internet Things J.* **2019**, *6*, 4188–4200. [CrossRef]
25. Du, J.; Lu, G.; Jiang, J.; Zhao, Q. Capacity and rate maximization in MEC systems. In Proceedings of the 11th International Conference on Wireless Communications and Signal Processing (WCSP 2019), Xi’an, China, 23–25 October 2019. [CrossRef]
26. Ma, Z.; Wei, Z.; Zhang, W.; Lyu, Z.; Xu, J.; Zhang, B. Energy-saving strategy for edge computing by collaborative processing tasks on base stations. In Proceedings of the 16th International Conference on Mobility, Sensing and Networking (MSN 2020), Tokyo, Japan, 17–19 December 2020; pp. 198–205. [CrossRef]

27. Rahman, W.U.; Hong, C.S.; Huh, E.N. Edge computing assisted joint quality adaptation for mobile video streaming. *IEEE Access* **2019**, *7*, 129082–129094. [[CrossRef](#)]
28. Bahreini, T.; Member, S.; Brocanelli, M.; Grosu, D. VECMAN: A framework for energy-aware resource management in vehicular edge computing systems. *IEEE Trans. Mob. Comput.* **2021**, *1*, 1–15. [[CrossRef](#)]
29. Liu, X.; Liu, J.; Wu, H. Energy-efficient task allocation of heterogeneous resources in mobile edge computing. *IEEE Access* **2021**, *9*, 119700–119711. [[CrossRef](#)]
30. Kuang, L.; Gong, T.; OuYang, S.; Gao, H.; Deng, S. Offloading decision methods for multiple users with structured tasks in edge computing for smart cities. *Future Gener. Comput. Syst.* **2020**, *105*, 717–729. [[CrossRef](#)]
31. Fan, W.; Liu, Y.; Tang, B.; Wu, F.; Wang, Z. Computation offloading based on cooperations of mobile edge computing-enabled base stations. *IEEE Access* **2017**, *6*, 22622–22633. [[CrossRef](#)]
32. He, Y.; Ren, J.; Yu, G.; Cai, Y. D2D communications meet mobile edge computing for enhanced computation capacity in cellular networks. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 1750–1763. [[CrossRef](#)]
33. Tong, M.; Wang, X.; Wang, Y.; Lan, Y. Computation offloading scheme with D2D for MEC-enabled cellular networks. In Proceedings of the 2020 IEEE/CIC International Conference on Communications in China (ICCC Workshops 2020), Chongqing, China, 9–11 August 2020; pp. 111–116. [[CrossRef](#)]
34. Khan, M.B.; Manzoor, S.; Manzoor, H.; Islam, M.S. A collaborative computation offloading scheme for 5G heterogeneous networks. In Proceedings of the 2019 International Conference on Frontiers of Information Technology (FIT 2019), Islamabad, Pakistan, 16–18 December 2019; pp. 19–24. [[CrossRef](#)]
35. Ren, J.; Yu, G.; He, Y.; Li, G.Y. Collaborative cloud and edge computing for latency minimization. *IEEE Trans. Veh. Technol.* **2019**, *68*, 5031–5044. [[CrossRef](#)]
36. Zhao, J.; Li, Q.; Gong, Y.; Zhang, K. Computation offloading and resource allocation For cloud assisted mobile edge computing in vehicular networks. *IEEE Trans. Veh. Technol.* **2019**, *68*, 7944–7956. [[CrossRef](#)]
37. Dai, H.; Deng, B.; Li, W.; Liu, X. A note on the minimum power partial cover problem on the plane. *J. Comb. Optim.* **2022**, *44*, 970–978. [[CrossRef](#)]
38. Yang, J.; Zhang, X.; Wang, W. Traffic adaptive base station sleeping control in inhomogeneous network. In Proceedings of the 2018 Annual IEEE International Systems Conference, Vancouver, BC, Canada, 23–26 April 2018; pp. 1–5. [[CrossRef](#)]
39. Alnoman, A.; Anpalagan, A. Computing-aware base station sleeping mechanism in H-CRAN-Cloud-Edge networks. *IEEE Trans. Cloud Comput.* **2021**, *9*, 958–967. [[CrossRef](#)]
40. International Energy Agency. Data Centres and Data Transmission Networks. Available online: <https://www.iea.org/reports/data-centres-and-data-transmission-networks> (accessed on 3 September 2022).
41. Chudak, F.A.; Williamson, D.P. Improved approximation algorithms for capacitated facility location problems. *Math. Program.* **2005**, *102*, 207–222. [[CrossRef](#)]
42. Jain, K.; Mahdian, M.; Markakis, E.; Saberi, A.; Vazirani, V.V. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *J. ACM* **2003**, *50*, 795–824. [[CrossRef](#)]
43. Liu, X.; Li, W.; Xie, R. A primal-dual approximation algorithm for the k-prize-collecting minimum power cover problem. *Optim. Lett.* **2022**, *16*, 2373–2385. [[CrossRef](#)]
44. Saikr. 2022 MathorCup University Mathematical Modeling Challenge. Available online: <https://www.saikr.com/c/nd/8805> (accessed on 3 September 2022).
45. Aliyun. Instance Specification Family of ECS. Available online: [https://help.aliyun.com/document\\_detail/25378.html](https://help.aliyun.com/document_detail/25378.html) (accessed on 3 September 2022).
46. Alibaba/Clusterdata. Available online: [https://github.com/alibaba/clusterdata/blob/v2018/cluster-trace-v2018/trace\\_2018.md](https://github.com/alibaba/clusterdata/blob/v2018/cluster-trace-v2018/trace_2018.md) (accessed on 3 September 2022).
47. Meskar, E.; Liang, B. Fair multi-resource allocation with external resource for mobile edge computing. In Proceedings of the 2018 IEEE Conference on Computer Communications Workshops, Honolulu, HI, USA, 15–19 April 2018; pp. 184–189. [[CrossRef](#)]