

Article

Task Offloading and Resource Optimization Based on Predictive Decision Making in a VIoT System

Dan Lv ¹ , Peng Wang ², Qubeijian Wang ¹, Yu Ding ¹, Zeyang Han ¹ and Yadong Zhang ^{1,*}

¹ School of Cybersecurity, Northwestern Polytechnical University, Xi'an 710060, China; lvdan@mail.nwpu.edu.cn (D.L.); qubeijian.wang@nwpu.edu.cn (Q.W.); 2022100770@mail.nwpu.edu.cn (Y.D.); hzyokc@mail.nwpu.edu.cn (Z.H.)

² School of Cyber Engineering, Xidian University, Xi'an 710126, China; pengw@stu.xidian.edu.cn

* Correspondence: zhangyadong@mail.nwpu.edu.cn

Abstract: With the exploration of next-generation network technology, visual internet of things (VIoT) systems impose significant computational and transmission demands on mobile edge computing systems that handle large amounts of offloaded video data. Visual users offload specific tasks to cloud or edge computing platforms to meet strict real-time requirements. However, the available scheduling and computational resources for offloading tasks constantly destroy the system's reliability and efficiency. This paper proposes a mechanism for task offloading and resource optimization based on predictive perception. Firstly, we proposed two LSTM-based decision-making prediction methods. In resource-constrained scenarios, we improve resource utilization by encouraging edge devices to participate in task offloading, ensuring the completion of more latency-sensitive request tasks, and enabling predictive decision-making for task offloading. We propose a polynomial time optimal mechanism for pre-emptive decision task offloading in resource-abundant scenarios. We solve the 0–1 knapsack problem of offloading tasks to better meet the demands of low-latency tasks where the system's available resources are not constrained. Finally, we provide numerical results to demonstrate the effectiveness of our scheme.

Keywords: task offloading; resource assignment; multi-step time series forecasting; combinatorial optimization problem



Citation: Lv, D.; Wang, P.; Wang, Q.; Ding, Y.; Han, Z.; Zhang, Y. Task Offloading and Resource Optimization Based on Predictive Decision Making in a VIoT System. *Electronics* **2024**, *13*, 2332. <https://doi.org/10.3390/electronics13122332>

Academic Editor: Carlo Mastroianni

Received: 9 May 2024

Revised: 7 June 2024

Accepted: 9 June 2024

Published: 14 June 2024



Copyright: © 2024 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

With the rapid development of internet technology, simulated virtual technology, and sensor technology, an important milestone called the visual internet of things (VIoT) has emerged [1,2]. VIoT systems, which place high demands on computing and transmission resources, have been widely used in specific scenarios such as production monitoring [3], smart industry [4], and smart cities [5]. In order to further improve the reliability of the large-scale deployment of VIoT devices and applications, the area of task offloading has been given attention [6]. Visual task offloading means offloading part of the visual task to the cloud or edge computing platform, reducing computational burden and improving system performance and efficiency [7].

In a VIoT system, different visual users (for task offloading) have varying latency requirements, and the available resource status for offloading within the system is dynamic [8]. In order to address this issue, we face two challenges: how can we manage the uncertainty of resource supply and demand, minimize the latency loss caused by propagation, and handle visual task offloading under different resource states? Our focus is on planning and detection before visual task offloading as the first challenge. Figure 1 illustrates the generality of request task failures with delay requirements. The visual user makes a request at 12:00 as shown in step (1) of the figure. The base station selects the appropriate executor to receive the task and sends it to the edge device with excess computing resources as shown in step (2) of the figure. After executing the task, the calculated result is returned to the

requesting user as shown in steps (3) and (4), with delays in all three stages. This means we won't be able to select the appropriate edge device until 12:10. Even if an edge device that can perform offloading tasks is selected in a timely manner, when the edge device sends a request to the base station, the user must offload the task to the remote computing node (step 5 in Figure 1) and then return the requested user after the task is executed (step 6 in Figure 1). The task has expired, causing it to fail. From this, it can be seen that there is significant time loss in transmission during offloading tasks, particularly the uncontrollable selection process of edge devices capable of executing offloading tasks [8].

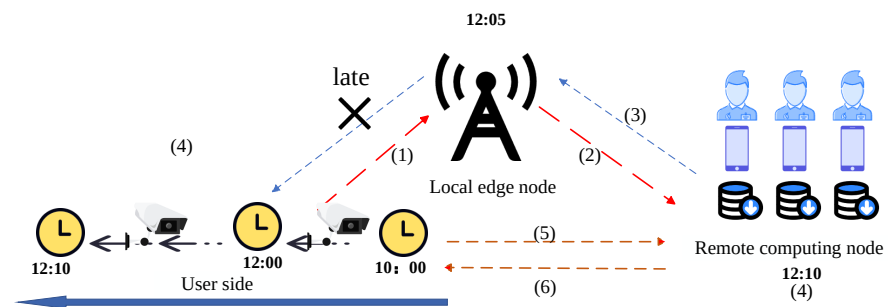


Figure 1. Examples of failed tasks include a lack of timely response or prolonged interaction between the base station and the responding device, which can prevent the task from being completed within a reasonable range.

We can reduce the time lost in task transmission by knowing the workload in advance through predictive methods. For example, suppose the system predicts that the video volume of a specific camera will increase within a certain time frame. Some processing tasks can be offloaded from the central server to edge devices for processing in advance. Furthermore, if the system predicts a major event or crowd gathering in a certain area in the future, the monitoring cameras in that area can also be offloaded in advance to ensure a timely response and reduce waiting time and delay time. Hence, the pre-emptive prediction task can cause propagation delays and selection delays [9]. Then, we still face the second challenge: how do we handle task offloading under different resource states? To the best of our knowledge, the heterogeneity of resource optimization is reflected in offloading methods. When the edge server has ample resources, it can directly participate in offloading tasks. The edge server can choose different participating methods based on its available computing resources. When the edge server has limited resources, it can incentivize edge nodes to collectively complete the request tasks. Otherwise, it can incentivize edge nodes to perform distributed offloading if the server involves privacy protection. On the other hand, the heterogeneity of resource allocation is reflected in the fact that when offloading low-latency tasks, the resource allocation method usually needs to prioritize task responsiveness and timeliness. In cases where high-latency requirements exist, the resource allocation techniques can be more flexible. Therefore, the amount of scheduled resources will determine the offloading methods and resource allocation.

In this article, we focus on addressing the aforementioned two challenges and propose a predictive task offloading decision mechanism. First, we utilize the long short-term memory (LSTM) model to predict the incoming task requests in VIoT. Then, we design two mechanisms for task offloading decision-making based on different task requirements and computing resource capacities.

For task prediction, based on the latency requirements of visual users, we consider two prediction results. Generally, edge servers can collect a large amount of data from onboard computing users. Based on these data, accurate predictions can be made for future user task workload and task arrival times. With the development of machine learning, numerous AI algorithms have, indeed, been used for predictive modeling. LSTM is a variant of recurrent neural networks (RNNs), which are used to handle sequential data. Compared to traditional RNNs, LSTM has more robust memory capabilities and can handle long-term dependencies

better. When predicting potential upcoming tasks using LSTM models, it is essential to consider that the time delay for task completion may vary depending on real scenarios. The LSTM model can predict a batch of potential incoming tasks for high-latency requests. As mobile nodes, visual users in the VIoT network can dynamically adjust resource allocation strategies based on the current network load, which helps to reduce resource wastage. For low-latency task offloading requests, edge servers need sufficient resources to utilize the LSTM model to accurately predict the arrival time and task volume of individual user task requests, enabling timely response to offloading tasks.

For resource allocation, we choose some edge devices capable of performing tasks based on task volume prediction. We have actually designed two different mechanisms for device selection based on different prediction results. For high-latency tasks, edge servers do not need to pay excessive attention to the offloading requirements of a single edge device. Instead, they can predict a batch of upcoming tasks that will require offloading. The edge server can perform task scheduling and resource allocation in a batch manner. Accordingly, we develop a two-stage task assignment approach by combining contract theory and matching theory. In the first stage, in order to motivate devices to share their resources, the edge server designs a contract. In the contract, each distinct performance-reward association is defined as a contract item. Then, the edge server broadcasts the contract, and each device chooses its desired contract item to maximize its payoff. In the second stage, devices that have signed a contract with the edge server serve as fog nodes. The task assignment problem is modeled as a task flow-based task assignment problem. For low-latency tasks, the batch processing approach used by edge servers may not meet the fast response to offloading tasks. The edge server needs to adopt a separate prediction, the individual selection of executable edge devices, and personal resource allocation methods. When modeling the task offloading problem as a variant of the 0–1 knapsack problem, where the computational resource requirement of the task is equivalent to the weight of the item, the value of the task is equivalent to the value of the item, and the capacity limitation of computational resources is comparable to the capacity limitation of the knapsack. We apply an improved algorithm to achieve our goal of task offloading. The main contributions of this paper are briefly summarized as follows:

- In the context of a VIoT network system model, we formulate energy-efficient task offloading to maximize the task completion rate while considering the latency constraint and energy consumption of computation and communication with multiple access points. We propose using the LSTM model to predict offloading tasks and reduce propagation delay. As part of preparing for the edge server to handle offloading tasks, it predicts the number of tasks to be offloaded for a batch of users or specific offloading tasks with high time requirements based on its available resources.
- We propose a mechanism for pre-emptive decision-making in task offloading. Due to the varying task latencies and available resources in offloading scenarios, we designed both resource-constrained pre-emptive decision-making methods and polynomially optimal pre-emptive decision-making methods to address task offloading and resource allocation under different resource constraints.
- We chose a smart city system with a specific base station coverage range, one of the classic application scenarios of VIoT, as the simulation scenario. We sampled the number of historical offloading tasks and evaluated our two proposed pre-emptive decision-making mechanisms. The results show that our mechanisms achieve better performance in terms of resource utilization and the visual task completion rate.

2. Related Works

2.1. Offloading in VIoT System

Recently, many researchers have participated in the study of edge offloading in a VIoT system. For task offloading architecture, Kochan et al. [10] developed a new VIoT platform that not only harmoniously integrates edge/cloud computing but also uses SDNs to overcome the challenges of the flexible management, control, and maintenance of VIoT

devices. Ji et al. [2] proposed a new visual internet of things architecture, namely A-VIoT, to improve the end-to-end performance of next-generation smart cities. From the perspective of scheduling and decision-making for task offloading, in [11], the authors studied the offloading of real-time video streams from visual odometry. They showed that efficient offloading can significantly reduce hardware costs in autonomous driving robots and vehicles. Zhu et al. [12] proposed a new vision-based assisted driving task offloading solution in the emerging visual-based driving assistance system, named “Chameleon”, which reduces the average service delay of task offloading. Existing research on task offloading in VIoT contexts rarely takes into account the varying task delay requirements and the limited computational resources available when handling high-concurrency offloading tasks.

2.2. Optimal Metrics in a VIoT System

For the indicators of energy consumption, latency, security, and other aspects of offloading computing tasks in VIoT applications, in [13], Wang et al. considered how to allocate the computing power of fog nodes to process monitoring data to achieve low latency while maintaining video quality. Li et al. [14] studied the edge offloading problem in autonomous detection and tracking tasks for unmanned aerial vehicles in marine environments. They proposed an edge-assisted unmanned aerial vehicle system with dynamic image resolution, achieved through the joint optimization of image resolution, offloading rate, transmission power, and local central processing unit (CPU) frequency under the constraint of task delay, to minimize energy consumption. There are other metrics, such as energy management and resource allocation. Trinh et al. [15] proposed a new algorithm to analyze the impact of decisions on energy consumption under different visual data consumption needs. Ji et al. [16] studied economic transmission to improve resource utilization and crowd co-ordination to enhance co-operation performance. Gao et al. [17] proposed a new algorithm called spatial attention-driven multi-domain network (SA MDNet), aimed at minimizing resources while maintaining acceptable performance. There is also research related to task offloading that aims to reduce the burden on terminal devices. For example, Zhang et al. [18] constructed a self-organizing D2D collaborative video content-sharing framework for VIoT to reduce the burden of controlling many VIoT devices and video data traffic. Although the above work can accelerate video task inference, it does not consider the task completion rate, latency constraints, and resource allocation jointly.

2.3. Optimization Algorithms in a VIoT System

From the perspective of optimization algorithms, various traditional optimizations have been utilized to address the offloading problem, such as game theory, heuristic optimization, and so on. In order to optimize the existing optimization of computing resource allocation in mobile vision applications that do not take into account the actual mobile environment in the design of learning models, a vision scaling optimization algorithm was proposed [19], which utilizes an online convex optimization framework to jointly optimize the design of the learning model, the size of the input layer, and the allocation strategy of computing resources to adapt to dynamic changes in the system. In order to efficiently allocate limited resources at edge devices, Hung et al. [20] proposed two auction frameworks, edge combined clock auction (ECCA) and stream combined clock auction (CCAS), to improve the QoE of real-time video streaming services in edge cellular systems. Sun et al. [21] proposed a commercial caching system consisting of video retailers (VRs) and multiple network service providers (NSPs). Each NSP rents its SBS to a VR at a certain price to generate profits, and they envision SBS as a specific type of resource within the framework of Stackelberg games. In order to utilize the most advanced computing and memory-demand deep learning methods at IoT nodes, the authors proposed a low-power and real-time deep learning-based approach as the solution for implementing a multi-object visual tracking Jetson TX2 development kit on NVIDIA [22]. A proposal for a multi-device uplink unencoded video transmission scheme (MDUcast) was proposed [23]. In MDUcast, an optimal power allocation strategy and subcarrier scheduling algorithm based on matching theory were proposed. Singh

et al. [24] introduced a blockchain-based licensing platform for decentralized computing task offloading between edge devices and proposed a solution to encourage resource-constrained devices to participate in partial task offloading. Most studies have not considered different offloading methods due to varying resource states and task diversity.

3. System Model and Problem Formulation

3.1. System Model

The VIoT system framework is shown in Figure 2. The VIoT system consists of the following entities: visual users, an MEC server, and a task executor. The essential functionality of the network is to schedule fixed MEC servers and dynamic devices to complete offloading tasks. Users who have offload requirements act as task requesters, for example, requesting driving assistance and other tasks. Considering privacy disclosure, MEC servers serve as relays for task offloading for users who do not directly participate in task execution. Edge devices with computing resources are responsible for offloading. We provide more details about the network entities as follows.

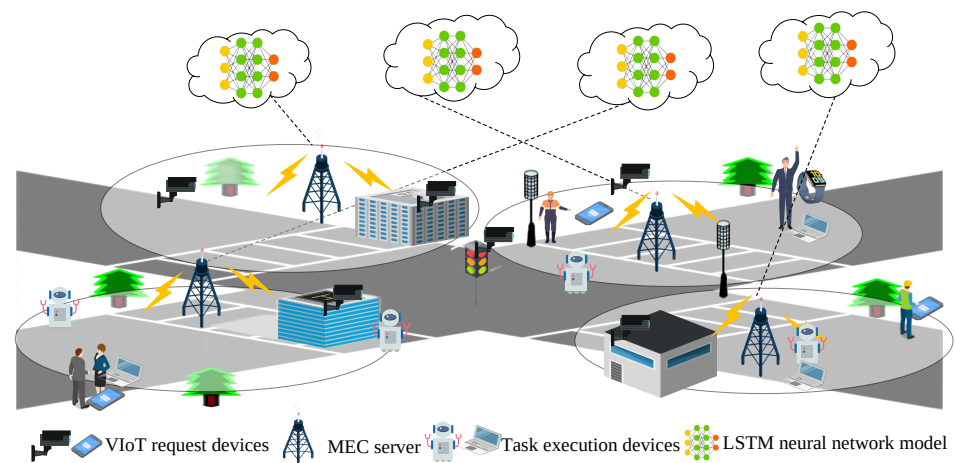


Figure 2. The VIoT system framework.

Let $\mathcal{N} = \{1, \dots, n, \dots, N\}$ be a task requester, and $|\mathcal{N}| = N$. In offloading computation, the requester refers to offload users, such as video conferencing applications and streaming service providers. They upload offloading requests to the subscribing devices with computing resources through a local base station. The offloading task request from user, n , can be represented as $[X_n, L_n]$, where X_n denotes the total workload of user n , and T_n represents the total latency requirement for a task request from user n . Finally, the output results are fed back to the base station and further delivered to the requester.

Let $\mathcal{M} = \{1, \dots, m, \dots, M\}$ be the task performers, and $|\mathcal{M}| = M$. We consider any edge device with computation hardware to process the offloading tasks. The edge devices allocate the required computing resources to accomplish the task. Then, the output result is delivered back, and the task is completed.

By analyzing historical data, such as task types, latency requirements, and quantities, MEC servers can use machine learning to predict task volume. Meanwhile, the amount of resources of MECs also affects task prediction, directly affecting the task's processing power and response time of the task. When MEC servers have sufficient resources, they can confidently handle larger task volumes, W , as a total task volume in a coverage area when predicting task volumes. When the resources of MEC servers are limited, the predicted task volume can only meet the task volume, W_i , of a certain user, i , to avoid overload or increased latency. In practical scenarios, users send tasks to the base station, and the base station selects the appropriate executor to receive the task and then sends it out, which is executed and returns the calculated result. The delay in these three stages may cause users to no longer need to complete the task, especially in the case of a large influx of tasks,

which is more likely to cause task failure. Based on resource demand prediction, the system can allocate and schedule resources in advance to meet the task execution needs, avoiding resource bottlenecks and wasting computing time.

3.2. Communication and Consumption Regarding Energy Models

The offloading process is described in the following steps. First, the task requester, N , transmits the input data to the preselected edge device, M , i.e., uplink transmission. Next, the edge device computes the received data, and finally, the edge device transmits the output data to the visual user, N , i.e., downlink transmission. We assume a frequency division duplex (FDD) in that equal bandwidth, B , is allocated for any uplink or downlink. Thus, there is no interference between uplink and downlink communication. The task of the visual user is described by the number, L_n , of input bits, the number, C_n , of CPU cycles per input bit for computation, and the number, D_n , of output bits produced by computation per input bit. In most cases, the size of the input data is much larger than the output data.

3.2.1. Computation Energy Model

When the CPU is operated at frequency, f^i , the computation energy consumption needed to execute the application of a visual user, N , with L input bits is obtained by

$$E_m^i(L) = \gamma^p C_m L (f^i)^2, \quad (1)$$

where $i = v$ for the visual user, and $i = z$ for the edge device, respectively. Here, $f^i \left[CPU_{s}^{cycles} \right]$ is the operating frequency of each processor, and γ^i denotes the effective switched capacitance of each processor, which is related to the chip architecture [25].

3.2.2. Communication Energy Model

When the visual user, N , transmits L_n^q bits during the slot duration δ , the following equation is obtained via Shannon theory:

$$B\delta \log_2 \left(1 + \frac{E_n^q(L_n^q) h_n}{N_0 B \delta} \right) = L_n^q, \quad (2)$$

where $q = u$ for uplinks and $q = d$ for downlinks. Hence, the communication energy consumption of the vehicle, n , is calculated as

$$E_n^q(L_n^q) = \frac{N_0 B \delta}{h_n} \left(2^{\frac{L_n^q}{B\delta}} - 1 \right). \quad (3)$$

From Equation (3), we can see that the communication energy consumption is related to the number of transmission bits and the channel condition affected by the communication distance.

Hence, the total energy consumption can be obtained as

$$E_{total} = E_m^i + E_n^q. \quad (4)$$

3.3. Problem Formulation

Improving task completion efficiency and on-time delivery for time-sensitive tasks in the VIoT system: Our optimization objective is to utilize existing resources to accomplish as many time-sensitive request tasks as possible, achieving overall system optimization. The optimization objective can be expressed as

$$\text{P0: } \underset{E_{m,n}}{\text{maximize}} \quad \frac{x_{nm}|S|}{E_{m,n}^{\text{total}}} \quad (5)$$

$$\text{s.t.} \quad \sum_{1 \leq n \leq N} C_n^{\text{comp}} \leq \sum_{1 \leq n \leq N} W_n \quad (5a)$$

$$\sum_{1 \leq i \leq N} x_{nm} C_{n,m}^{\text{comp}} \leq v_m \quad (5b)$$

$$\sum_{1 \leq n \leq N} x_{nm} C_{n,m}^{\text{comp}} \leq W_n \quad (5c)$$

$$\sum_{1 \leq n \leq N} x_{nm} \leq N \quad (5d)$$

$$\sum_{1 \leq m \leq M} x_{nm} \leq M \quad (5e)$$

$$x_{nm} \in \{0, 1\}. \quad (5f)$$

The objective function of Equation (5) is to maximize task completion quantity, $|S|$. For the constraints, (5a) indicates that the computational resources allocated to all task request users cannot exceed the available resources in the model; (5b) indicates that the computational resources allocated to service devices, m , cannot exceed the computational resources it can provide, where x_{nm} is an index representing whether a task request user, n , will offload their computational resources to a service device, j ; (5c) indicates that the computational resources allocated to a task request user, n , cannot exceed the computational resources they need, and (5d) and (5e) indicate that the maximum number of offload connections for n supported by device m and task request user n cannot exceed the maximum number limits N and M , respectively. The solved offloading problem has been proven to be an NP-complete combinatorial optimization problem with multiple knapsacks and constraints.

3.4. MEC Servers Leverage LSTM-Based Multi-Step Prediction Mechanisms

Predicting task workload is a typical time series forecasting problem, and the deep learning model LSTM (long-short-term memory) can effectively handle it [26]. An LSTM network consists of two data gates to capture the sequence information, i.e., the update gate and the forget gate. From data collection, the task information that needs to be offloaded for base station coverage over different time slots is recorded. We apply a sliding window approach to extract the input time series, $X_N = \{x_1, x_2, \dots, x_{\|L_n\|}\}$, of the long short-term memory model from the data source. When all the original data samples consist of y time slots, we obtain $\|L_n\| = y - z + 1$ input time series if the sliding window size is z . Let $H_n = \{h_1, h_2, \dots, h_{\|D_n\|}\}$ indicate the hidden states and $O_n = \{o_1, o_2, \dots, o_{\|L_n\|}\}$ represent the output states. We use $W^{\cdot h}$ and $W^{\cdot x}$ to denote the weights and use b to denote the bias. At the time step t , the value of forget gate Γ_t^f is calculated by

$$\Gamma_t^f = \sigma(W^{fh}h_{t-1} + W^{fx}x_t + b^f), \quad (6)$$

where x_t is the input vector at time step t , h_{t-1} is the hidden state at the previous time step, and $\sigma(\cdot)$ denotes a Sigmoid function that normalizes the value into the range $[0, 1]$. The forgotten fate determines what information will be eliminated from the cell state. When the value of the forget gate $\Gamma_t^f = 0$, it throws away all the information, and $\Gamma_t^f = 1$ means that it keeps all the information. As for the update gate, it decides what information can be stored in the cell state. The value of the update gate is calculated by

$$\Gamma_t^u = \sigma(W^{uh}h_{t-1} + W^{ux}x_t + b^u). \quad (7)$$

Next, we calculate the information of a candidate memory cell, $\tilde{c}_t$, at the time step t by

$$\tilde{c}_t = \tanh(W^{ch}h_{t-1} + W^{cx}x_t + b_c), \quad (8)$$

where $\tilde{c}_t$ represents the candidate information that should be stored in the cell state. Until now, we obtain $\{\Gamma_t^f, \Gamma_t^u, \tilde{c}_t\}$ to update the cell state by

$$c_t = \Gamma_t^f \odot c_{t-1} + \Gamma_t^u \odot \tilde{c}_t, \quad (9)$$

where $\odot$ is an operator of executing the point-wise multiplication of two vectors. The output and hidden states are computed by

$$\begin{cases} o_t = \sigma(W^{oh}h_{t-1} + W^{ox}x_t + b^o) \\ h_t = o_t \odot \tanh(c_t). \end{cases} \quad (10)$$

Finally, we take o_t as the input for the multiplayer perception and compute the time series prediction $\hat{x}_{t+1}$ of the next time step. After the local model training, the weights W^h and W^x and the bias, b , are shared to the parameter server. With the help of LSTM-based task prediction, the following incentive mechanism design and the task assignment can be measured well.

4. Proactive Task Offloading Mechanisms in Resource-Constrained Environments

We consider how MEC servers with limited resources can handle offloading tasks with lower real-time requirements in batches in this section. The MEC server predicts the total task volume for the next phase. Based on predicted task volume and budget value, the computing resources belonging to the edge devices can be purchased in advance by the MEC server to design a contract mechanism. Then, the resource allocation for batch processing can be handled through a two-side matching method named the “task flow based on task assignment” method.

4.1. Contract Mechanism for Purchasing Resources

Due to concerns about privacy leaks, resource owners are unwilling to share their paid computing resources with other users voluntarily. In this scenario, we need to design incentive measures. Our proposed contract-based incentive mechanism will motivate devices to share their computation resources for task offloading. First, the type of device model and the utility functions of the base station and devices are introduced, and the resource allocation problem is formulated. Second, we elaborate on how to derive the optimal contract under information asymmetry.

4.1.1. Device Type Modeling

In contract-based incentive mechanisms, we use the term “type”, represented as k , to describe private information that is unknown to others. Here, we focus on the privacy information cost for edge devices, M_j , where c_j represents the cost incurred from energy consumption and potential privacy leakage losses. We refer to an edge device using $\theta_k = (c_k, T_k)$ as a type- k device. Here, T_k is the tolerable delay of a type- k device. The base station selects delay-tolerant users based on predictions to motivate them to participate in offloading. In order not to lose generality, we rank them in ascending order: $\theta_1 < \dots < \theta_k < \dots < \theta_K$, $\{k \in 1, \dots, K\}$.

4.1.2. Contract Formulation

The MEC server designs the corresponding contracts and expresses the terms for k edge devices as (f_k, r_k) , where f_k denotes the computing resource contributed and r_k represents the corresponding reward. The specific representation of contracts is given by $\{C = (f_k, r_k), \forall k \in K\}$. Hence, the utility of the MEC server is represented by its satisfaction with the purchased resources minus the budget expenditure. The utility can be given as

$$U_B\{(f_k), (r_k)\} = \sum_{k=1}^K \lambda_k (\theta_k \log(f_k + 1)) - B^{t+\omega}, \quad (11)$$

where the first part on the right side of (11) is the satisfaction of the MEC with contributed resources, f_k . Here, the value of $|U_B|$ can be used to replace the value in the objective function $|S|$. Assuming that the probability of an edge device belonging to type θ_k has a probability of λ_k , which satisfies a range of conditions, as $\sum_{k=1}^K \lambda_k = 1$. In general, λ_k follows a Gaussian distribution. Considering that edge servers purchase resources in advance based on predicted workload, their expenditure budget is derived from the budgeted workload. This paper temporarily disregards price fluctuations caused by fluctuations in workload demand in the market and does not consider the impact of irrational decision-making. It assumes a linear relationship between unit computing resources and unit price. So we can obtain $B^{t+\omega} = b * W^{t+\omega}$, where b is the unit price of computing resources, and $W^{t+\omega}$ is the predictive workload; here, t represents the current time and ω is a constant, which can be changed according to the requirements.

The revenue minus the cost consumption represents the utility function of the task executor,

$$U(f_k, r_k) = \theta_k r_k - c_k f_k, \quad (12)$$

where θ_k characterizes the weight of r_k to type- k devices. A higher-type vehicle has a larger weight due to its higher preference for resource sharing. r_k denotes the edge device's reward received, and the sum of the revenues of all users is less than the total expenditure, $\sum_{k=1}^K r_k \leq B^{t+\omega}$.

The number of completed offloaded tasks will depend on the available reserved resources. The objective of the incentive mechanism for purchasing resources is to maximize the amount of computing resources purchased within the budget constraint. We can transform the original problem $\mathbf{P}_0$ into a subproblem $\mathbf{P}_1$ for purchasing resources and a subproblem for task allocation.

$$\mathbf{P}_1: \quad \max_{\{(f_k), (r_k)\}} U_B\{(f_k), (r_k)\} \quad (13)$$

$$\text{s.t.} \quad \theta_k r_k - c_k f_k \geq 0, \forall k \in K \quad (13a)$$

$$\theta_k r_k - c_k f_k \geq \theta_k r_{k'} - c_k f_{k'} \quad (13b)$$

$$\sum_{1 \leq k \leq K} r_k \leq B^{t+\omega} \quad (13c)$$

$$\sum_{1 \leq k \leq K} \delta_k - \theta_k \leq 0. \quad (13d)$$

The constraints (13a) and (13b) refer to the individual rationality (IR) and incentive compatibility (IC) constraints. In addition, (13c) indicates that the expenditure on purchasing resources is lower than the budget expected, based on predictions. In addition, (13d) represents the lower-limit value of θ_k .

Definition 1. The IR, IC, and monotonicity constraints are defined as follows:

- *Personal rationality:* The individual rationality constraint requires that the contract terms corresponding to each type of edge device have a non-negative utility value, i.e., $U(f_k, r_k) \geq 0, \forall k \in K$.
- *Incentive compatibility:* The maximum utility value is obtained only by selecting the contractual terms corresponding to the type- k edge devices of type (f_k, r_k) . The value of the utility function is greater when choosing other types of contract terms, i.e., $U(f_k, r_k) \geq U(f_{k'}, r_{k'})$.
- *Monotonicity of the contract:* The monotonicity ordering of contracts means that the reward obtained by edge type- k devices is higher than the reward obtained by edge devices of type $k - 1$ and lower than the reward obtained by edge devices of type $k + 1$.

The requirement for contract fulfillment to be feasible requires individual rationality (IR), incentive compatibility (IC), and monotonicity to be simultaneously satisfied.

4.1.3. Optimal Contract Design under Information Asymmetry

The optimization problem $\mathcal{P}_i^\infty$ involves M IR and $M(M - 1)$ IC constraints. Next, we simplify the model's parameters and solve for the optimal value based on the simplified model.

Lemma 1. Reducing the IR constraint: If the utility value of the task executor with the minimum type satisfies the individual rationality (IR) constraint, then other types of devices also satisfy the IR constraint.

Proof. From the compatibility of incentives (IC) constraint, it can be derived that $\theta_k r_k - c_k f_k \geq \theta_k r_1 - c_k f_1 \geq \theta_1 r_1 - c_1 f_1$. If the utility function of a type-1 task executor is non-negative, then the utility values of other types of task executors must also be non-negative. $\square$

Lemma 2. Reducing IC constraints: Define types k and k' IC constraints ($k' \in 1, \dots, k - 1$) are downward incentive constraints (DICs), and to define the type k and k'' , IC constraints ($k'' \in k + 1, \dots, K$) are upward incentive constraints (UICs); then, the number of DICs and UICs is reduced.

Proof. Consider three types of task performers: $\theta_{k-1} \leq \theta_k \leq \theta_{k+1}$, based on monotonicity and incentive compatibility constraints, as follows :

$$\begin{cases} \theta_{k+1} > \theta_k > \theta_{k-1}, r_{k+1} > r_k > r_{k-1} \\ \theta_k r_k - c_k f_k \geq \theta_k r_{k-1} - c_k f_{k-1} \\ \theta_{k+1} r_{k+1} - c_k f_{k+1} \geq \theta_{k+1} r_k - c_k f_k \\ \theta_{k-1} r_{k-1} - c_k f_{k-1} \geq \theta_{k-1} r_k - c_k f_k. \end{cases} \quad (14)$$

$\square$

We can obtain $\theta_{k+1}r_{k+1} - c_k f_{k+1} \geq \theta_{k+1}r_k - c_k f_{k+1} \geq \theta_{k+1}r_{k-1} - c_k f_{k-1}$. If the edge devices of type k satisfy the IC constraint, then the devices of type $k-1$ also satisfy the IC constraint. Similarly, type 1 simplifies the IC constraint, simplifying the downward incentive constraint. Similarly, we can obtain $\theta_{k-1}r_k - c_k f_k \geq \theta_{k-1}r_{k-1} - c_k f_{k-1}$, simplifying the upward incentive constraint. We simplify the number of feasible contract constraints by combining the M IR constraints into one and reducing $M(M-1)$ constraints to one IR constraint and $M-1$ constraints. With the constraints relaxed, we can rewrite the optimization problem as follows:

$$\mathbf{P}'_1: \max_{\{(f_k), (r_k)\}} U_B\{(f_k), (r_k)\} \quad (15)$$

$$\text{s.t. } 0 \leq f_1 \leq \dots \leq f_K \quad (15a)$$

$$0 \leq r_1 \leq \dots \leq r_K \quad (15b)$$

$$\theta_1 r_1 - c_1 f_1 \geq 0, \forall k \in K \quad (15c)$$

$$\begin{aligned} f_{k-1} + \theta_{k-1}(r_k - r_{k-1}) &\leq f_{k'} \\ &\leq f_{k-1} + \theta_k(r_k - r_{k-1}) \end{aligned} \quad (15d)$$

$$\sum_{1 \leq k \leq K} r_k \leq B^{t+\omega} \quad (15e)$$

$$\sum_{1 \leq k \leq K} \delta_k - \theta_k \leq 0, \quad (15f)$$

where the constraints in (15a) and (15b) denote the constraints on the monotonicity of the contract and the conditions (15c) and (15d) represent the IR constraints and IC constraints, respectively. (15e) denotes the budget constraint, and (15f) represents the lower-limit value of θ_k .

Theorem 1. From $\mathbf{P}'_1$, the derivation that leads to a set of known information sets in a feasible contract, $\theta_1 < \dots < \theta_k < \dots < \theta_K$, and the optimal reward is

$$r_k^* = \begin{cases} \frac{1}{\theta_k} c_k f_k & , \text{ if } k = 1, \\ r_{k-1} - \frac{1}{\theta_k} c_k (f_{k-1} - f_k) & , k = 2, \dots, K. \end{cases} \quad (16)$$

Proof. We use a contradiction-proof approach to verify this result. Assuming that the existence of a feasible contract in $r^{\tilde{c}}$ can bring more significant revenue to the base station, which is obtained from the local downward incentive constraint where $r_{k-1}^{\tilde{c}} \leq r_{k-1}^*$, this constraint result in $r_1^{\tilde{c}} \leq r_1^*$ does not satisfy the IR constraint. Therefore, no contract, $r^{\tilde{c}}$, is greater than the optimal value, r_k^* . By substituting the above equation into $\mathbf{P}'_1$, we have a convex programming problem. A set of optimal contract terms (f_k, r_k) can be obtained through optimization methods, such as that of Newton's or multiple methods. $\square$

After completing the theoretical design of the contract, edge devices sign contracts specifically designed for their types. Then, the specific type of information of the edge devices will be revealed to the base station. Based on the above analysis, we present Algorithm 1 Stage I for the contract-based incentive mechanism design. The algorithm starts by initializing a visual user set, $\mathcal{N}$, based on the predicted result and user types, θ_k , and type distribution, λ_k . Then, we incentivize an edge device to contribute its computation resource through the contract in step 5. The edge device, m_j , signs the contract item with the edge server and shares its idle resource, f_k . In summary, contract theory incentive design is necessary to provide the essential parameters for resource allocation.

Algorithm 1 Contract-Matching Algorithm for batch task processing

```

1: Input:  $\mathcal{N}, \mathcal{M}, \{\theta_k\} \lambda_k, E_m(L)$ .
2: Output: Optimal contract  $C^*$ , Perfect matching  $\mathcal{G}$ .
3: Stage I: Contract-based Incentive Mechanism Design
4: Sort the types of edge devices in ascending order;
5: Obtain the optimal contract  $C^*$  by solving (16);
6: Stage II: Task flow-based Task Assignment
7: Initialize  $G'$  as a zero matrix with  $M$  rows and  $M$  columns,  $label = 0$ , set  $l_x(j) = 0$ ,
    $l_y(i) = \max(e_{i,j}), i, j \in 1, 2, \dots, M$ 
8: while  $label=0$  do
9:   for ( $i = 1; i \leq M; i = i + 1$ ) do
10:    for ( $j = 1; j \leq M; j = j + 1$ ) do
11:       $e_{i,j} = k_x(i) + l_y(j) - e_{i,j}$ 
12:    end for
13:  end for
14:  if  $num \neq M$  then
15:    for  $i = 1; i \leq M; i = i + 1$  do
16:      for  $j = 1; j \leq M; j = j + 1$  do
17:         $slack(j) = l_x(i) + l_y(j) - e_{ij}$ 
18:      end for
19:      if (the row  $i$  is uncovered) then
20:         $l_x(i) = l_x(i) - slack(j)$ ;
21:      end if
22:      if the column  $i$  is covered then
23:         $l_y(i) = l_y(i) + slack(j)$ 
24:      end if
25:    end for
26:  else
27:     $label = 1$ 
28:  end if
29: end while
30: return  $G$ 

```

4.2. Resource Allocation for Batch Processing

In this subsection, we propose a new task flow-based matching algorithm for batch processing that is inspired by the widely used Kuhn-Munkre algorithm for optimal matching in weighted bipartite graphs [27].

(1) Weighted bipartite graph (WBG) construction: Define a bipartite graph as $G = (\mathcal{N}, \mathcal{M}, E)$, where $\mathcal{N}$ and $\mathcal{M}$ are the two non-empty subsets with $N \cap M = \emptyset$. E is the edge set between N and M with $E = \{\omega_{n,m}\}, n \in N, m \in M$. Note that ω_{nm} is the weight vector between vertex n and vertex m .

(2) Matching: We can solve this matching problem using a KM algorithm, which solves the maximum weighted-matching problem in a bipartite graph (BG) based on the Hungarian method. Note that the KM algorithm can solve the matching assignment problem when the number of visual users, N , equals the number of edge devices, M . In order to achieve the goal of maximizing the task completion rate and minimizing system energy consumption as much as possible, we improve the traditional KM algorithm and propose a task flow allocation algorithm.

We consider the situation that for $N \leq M$, we extend G to $G' = (\mathcal{N}, \mathcal{M}, E_m(L)_{M \times M})$. We set the computation consumption matrix $E_m(L)$ to represent the weights of a link. By considering the different numbers of N and M , we extend the matrix as follows:

$$E_m(L) = \begin{bmatrix} e_{11} & e_{1,2} & \cdots & e_{1,m-1} & e_{1,m} \\ e_{2,1} & e_{2,2} & \cdots & e_{2,m-1} & e_{2,m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ e_{m-1,1} & e_{m-1,2} & \cdots & e_{m-1,m-1} & e_{m-1,m} \\ e_{m,1} & e_{m,2} & \cdots & e_{m,m-1} & e_{m,m} \end{bmatrix}.$$

Based on the analysis above, we present Algorithm 1 for a Contract-Matching Algorithm for batch task processing in a resource-constrained environment. Contract theory incentive design in Algorithm 1 stage I can provide the essential parameters for resource allocation. Next, we show the task flow-based KM algorithm for task assignment in steps 7–13. We calculate the computational consumption of the edge devices being incentivized. Set the weight of the link to the computational cost, $E_m\{L\}$. Initialize the feasible vertex labels for m visual users and m task performers, which can be expressed as an array of $l_x(i)$ and $l_y(j)$. In order to simplify our discussion, we use i and j to express the vertex visual users and edge devices. The Hungarian algorithm uses an augmenting path to increase the number of matches in steps 14–24. If the augmenting path is not found, it passes through vertex i , and the feasible vertex labels are adjusted. Define the parameter $slack(j)$ as the adjustment scale in step 17. Repeat steps 9–25 until you find the M zeros that belong to different rows and different columns that relate to each other. Set the G' element in the same positions of M as 1. G is the first M rows of matrix G' . Hence, the perfect matching G is obtained.

5. A Polynomial-Time Optimal Pre-Emptive Task Offloading Mechanism

In this section, we consider a more common scenario where the response time to visual offloading task requests from VIoT users is particularly short, and batch processing offloading methods are no longer applicable. The edge server can utilize its own abundant computing resources to predict the requester, n 's, task workload, W_n , and the arrival time, t_n . We can see that the performer, m , has a budget of B_m , with the available costs incurred during offloading tasks, c_m , and the contribution of resources f_m . The objective is to acquire the required resources within the budget for purchasing computing resources. The classical 0–1 knapsack problem involves selecting items and a knapsack with known weights and values for each item [28]. Therefore, the task offloading problem can be modeled as a variant of the 0–1 knapsack problem.

5.1. The 0–1 Knapsack Problem with Pre-Emptive Decision-Making

By assuming the number of items is M , the capacity of the backpack is B_m , the weight of the M item is c_m , and the resources that can be contributed are f_m . The formal model for the 0–1 knapsack problem is **P2**, which selects items so that the total weight of the items in the knapsack does not exceed the budget, B_m , while maximizing the resource, f_{total} .

$$\begin{aligned} \textbf{P2:} \quad & \text{maximize} \quad \sum_{m=1}^M f_m \cdot x_m \\ \text{s.t.} \quad & \sum_{m=1}^M c_m \cdot x_m \leq B_m \\ & x_m \in \{0, 1\} \quad m \in \{1, 2, \dots, m\}, \end{aligned} \tag{17}$$

where x_m denotes the selection status of item m -th, and $x_m = 1$ indicates that the m -th item has been selected and placed in the backpack. Conversely, $x_m = 0$ denotes that the m -th item was not placed in the backpack.

In some cases, the greedy algorithm can be used to solve the 0–1 knapsack problem. By using the Greedy algorithm to solve the 0–1 Knapsack problem [29], we start with an initial solution and select the locally optimal solution at each step, attempting to find or approximate the globally optimal solution for the problem. However, the algorithm is highly

unpredictable and does not guarantee finding the optimal solution to the problem. The solution obtained using the greedy algorithm is usually only an approximate optimal solution. The present study proposes an improvement to the greedy algorithm to further enhance the solution's optimality. The newly improved polynomial-time approximation algorithm introduces a parameter, ε , which represents the desired error of the algorithm. Different error values indicate additional algorithm steps and additional running times. Therefore, we denote the improved greedy algorithm as a family of algorithms for $I - greedy_\varepsilon$. Introducing the expected error in the algorithm can improve the performance of greedy algorithms to a certain extent, making them closer to the global optimal solution. The I-greedy algorithm proposed for the computing offload in pre-emptive decision-making is described in detail in Algorithm 2.

Algorithm 2 I-greedy algorithm for polynomial-time optimal pre-emptive task offloading mechanism

- 1: **Initialize:** Input $\varepsilon > 0$, let $p = \left\lceil \frac{1}{\varepsilon} \right\rceil$.
 - 2: Sort so that $\frac{f_1}{c_1} \geq \frac{f_2}{c_2} \geq \dots \geq \frac{f_n}{c_n}$;
 - 3: Take $t = 1, 2, \dots, p$. There are t items, if the total weight of these t items does not exceed B , then load them and then use greedy to load the remaining items;
 - 4: Compare all installation methods and take the highest value as the solution.
-

5.2. I-Greedy Complexity Analysis Algorithm

We characterize the algorithm's effectiveness by validating its upper bound and analyzing its computational complexity.

Theorem 2. We evaluate the I-greedy algorithm by taking into account the upper bound, which is an effective approximation algorithm. For each $\varepsilon > 0$ and the 0–1 Knapsack problem, I , both have the objective function for the optimal solution, which is $OPT(I) < (1 + \varepsilon)PTAS_\varepsilon(I)$. The optimal solution does not exceed $1 + \varepsilon$ times the solution obtained by the I-greedy algorithm. The error is equal to $1 + \varepsilon$ times. When ε is very small, the error is tiny.

Proof. Set the optimal solution as Π^* . If $|\Pi^*| \leq p$, the algorithm necessarily obtains the optimal solution, Π^* . Set $|\Pi^*| > p$, considering the computation based on the item with the maximum value among the previous, p , items in S^* , where the result, S , is obtained by greedy algorithm. The algorithm's solution will not be worse than the solution in S' . Therefore, comparing the error between S and S^* can help determine the relationship between the optimal solution and the solution obtained by the algorithm. $\square$

As shown in Figure 3, Π is a feasible solution that the algorithm tries, and Π^* is the optimal solution. The algorithm has already tried the solution of putting the item with the maximum value among the previous, p , items in the optimal solution. Therefore, the solutions for the previous, p , items are the same in both graphs. Additionally, since item l may have already been included in S due to previous items, it cannot be included again in the optimal solution found by the algorithm. Assuming l is the first item in Π^* that is not in Π , the items before l have already been selected, and the algorithm prioritizes items with higher unit values. j is an item that was previously loaded and does not belong to S^* . If there is no possession of j , the greedy method will load l . In the algorithm, all values f_e divided by p are greater than the value of item l that was not selected. We can conclude that $f_l < \sum_{e \in \Pi} \frac{f_e}{p}$. Below is the derivation of the relationship between the optimal solution:

$$\begin{aligned}
 OPT(I) &< \sum_{e \in \Pi} f_e + f_l \\
 &< \sum_{e \in \Pi} f_e + \sum_{e \in \Pi} \frac{f_e}{p}
 \end{aligned} \tag{18}$$

$$\leq (1 + \frac{1}{p})PTAS_{\varepsilon}(I)$$

$$\leq (1 + \varepsilon)PTAS_{\varepsilon}(I).$$

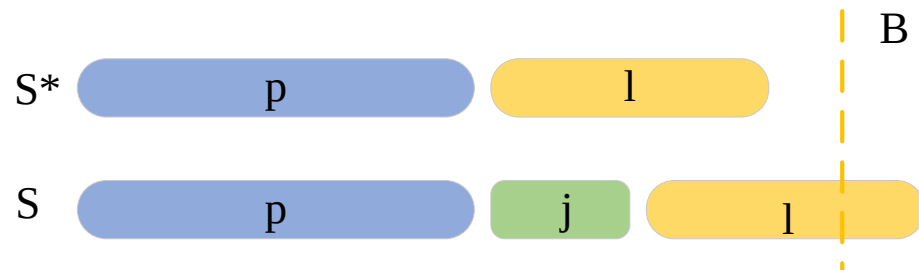


Figure 3. An improved greedy algorithm is used to solve the NP-hard problem of computing offload in pre-emptive decision-making.

Remark 1. (Computational complexity): I-greedy algorithm analysis of approximate schemes for polynomial time. The time complexity of the algorithm is $O(m^{\frac{1}{\varepsilon}+2})$.

Take t items from m items ($t = 1, 2, \dots, m$), where the number of possible alternatives is

$$C_m^1 + C_m^2 + \dots + C_m^p \leq p \times \frac{m^p}{p!} \leq m^p.$$

The running time of attempting to sequence is $O(m^2)$; therefore, the time complexity is $O(m^{m+2}) = O(m^{\frac{1}{\varepsilon}+2})$. When ε is a constant, the time complexity of the I-greedy algorithm is a polynomial function. Therefore, the PTAS algorithm is a polynomial approximation algorithm.

6. Performance Evaluation

In this section, we evaluate the proposed task offloading and resource optimization scheme based on predictive decision-making in a VIoT system. We consider a scenario where visual tasks are offloaded to a set of VIoT edge devices through an MEC server. Mainly, the tasks could be derived from utilizing technologies such as cameras and sensors to achieve more systematic traffic monitoring in intelligent transportation. We consider a single cell with one base station, 20 task requesters, and 30 edge devices to constitute the task of offloading samples. The dataset was partitioned using an 80/20 training/testing split. We use a time series dataset, which includes the amount of data recovered every 5 min and the number of requests. The dataset consists of two columns: one for timestamps and the other for hourly task offloading volume counts. We set up the experimental environment with Python 3.7.7, PyTorch 1.7.1, NumPy 1.21.5. Pandas 1.2.5, and Matplotlib 3.5.3 (Table 1).

Table 1. Parameters.

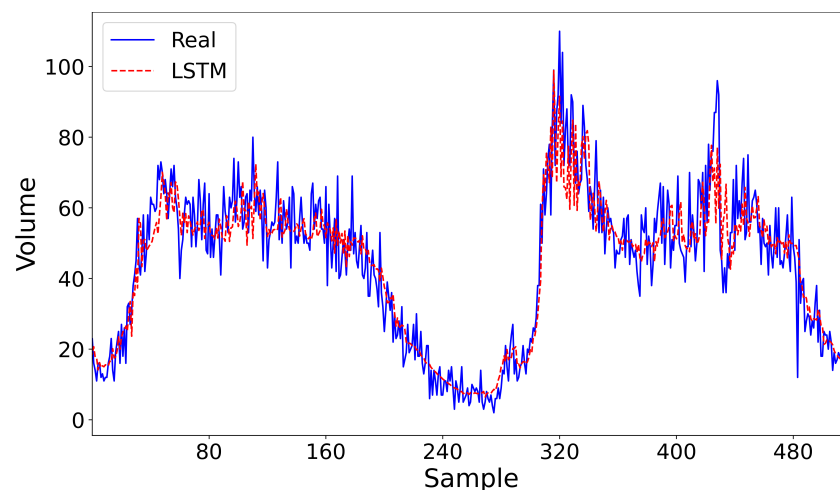
Parameter	Value
Size of the visual processing task	1–7 Mb its
Computation capacity of each visual task	1.5–2.5 Mb/s

Table 1. *Cont.*

Parameter	Value
Delay constraints	0.1–2 s
Cell radius	1000 m
Computation resources of the MEC server	5–8 GB/s
Maximum transmission power of each device	30 DBMS
Bandwidth of each sub channel	410 kHz
Noise power	−114 DBMS
Path loss exponent	−3.4

6.1. LSTM-Prediction

Figure 4 shows the task visual volume's prediction results, which are based on LSTM. In terms of network structure, the LSTM structure was built based on the PyTorch environment with an input size of 5, a hidden size of 6, and 2 layers. Afterwards, a linear layer was created to map the output of LSTM to a single node. Firstly, it is evident that the task volume demand changes frequently. When high-concurrency scene request tasks occur, it becomes more challenging to handle delay-sensitive tasks. Hence, discussing our work is of practical significance. Secondly, it can be seen that the LSTM model can accurately predict task demand and peak moments, providing strong evidence for our subsequent work. We successfully explored the possibility of using LSTM-based short-term task volume models. Although some new sequence prediction methods have performed well in specific fields or tasks, LSTM, as a classic and universal sequence model, still has representativeness and practicality.

**Figure 4.** Task volume prediction diagram based on LSTM.

6.2. Contract Feasibility and Efficiency

An MEC server acts as an offloading service broker and employs VIoT devices as edge computing nodes. When a VIoT device is effectively stimulated to share its idle computing resources, it continuously receives offloading tasks within a specific time frame, ranging from 5 to 10 min. Setting particular time frames can help devices better respond to unexpected situations and maintain system stability.

A series of simulations were conducted to verify the feasibility and efficiency of the contract-based incentive mechanism. Figure 5 compares the contract terms for different types of edge devices. The points indicate that the maximum utility of any type of edge device can only be achieved by selecting the appropriate contract terms. For example, the type-3 edge devices only choose a type-3 contract for the utility function to reach its

maximum value. This verifies the IC constraint under necessary conditions for contracts. Additionally, edge devices can only display their type to the MEC server to maximize utility after selecting a contract. This helps overcome the issue of information asymmetry. Furthermore, it can be seen that contracts of higher types also have larger utility function values, confirming the monotonicity constraint of contracts. The figure also verifies the IR constraint, which states that the utility values of all devices participating in offloading are more significant than zero.

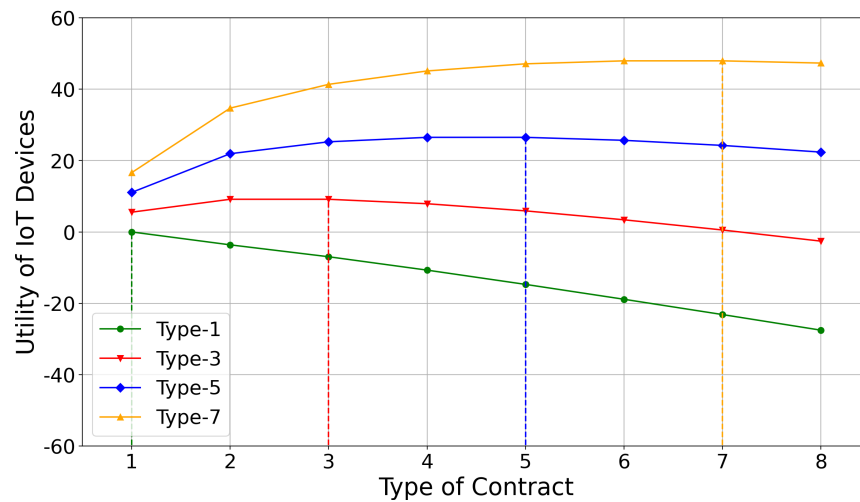


Figure 5. Utility of the edge devices versus contract item types.

Figure 6 compares the performance of edge devices under several different schemes. We choose two purchasing models for benchmarking: the full information contract model and the linear pricing model. In the full information contract model, where there is no information asymmetry in the contract design scenario, the utility value for any type of edge device is zero. In the incomplete information linear pricing model, the reward strategy for each edge device is defined as $r_k = \beta f_k$, where β is a unified reward value per unit of computing capability. Compared to incentives with the full information contract model, our method provides significant incentives for devices of higher types. As the edge device type increases, the utility value of edge devices decreases compared to the linear pricing model. The utility of edge devices is directly related to the amount of resources purchased. Thus, our method can better achieve the goal of purchasing sufficient resources.

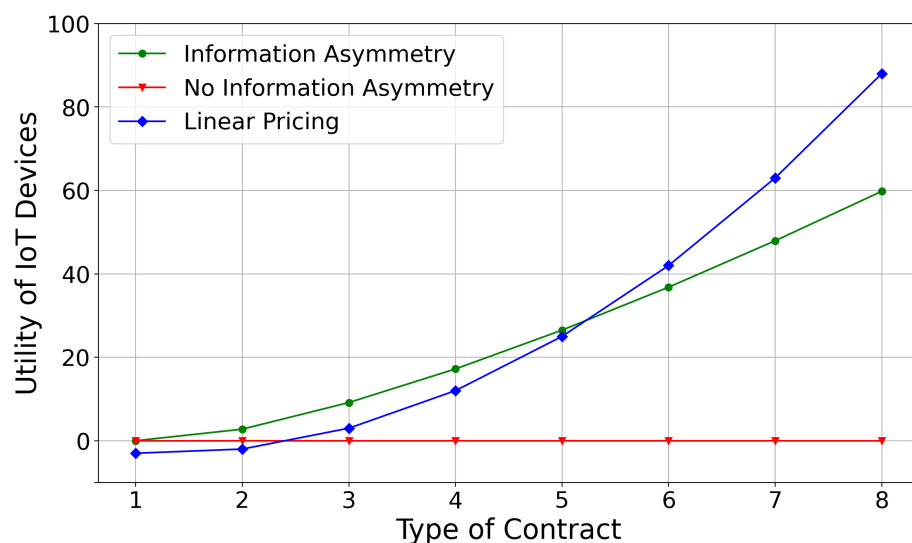


Figure 6. Comparison of the executor with respect to different schemes.

6.3. Task Offloading Evaluation Based on Predictive Decision-Making

In the following, we perform a numerical analysis of the performance of proactive task offloading in resource-constrained environmental systems and a polynomial-time optimal pre-emptive task offloading mechanism. Figure 7 shows the cost constraint versus the computing resources acquired by pre-emptive task offloading under resource constraints. The horizontal axis considers the specific value of latency cost constraints in different situations, which is the maximum delay time that the system can tolerate, while the vertical axis represents the resources acquired to complete offloaded tasks. Additionally, it can also regard the number of tasks completed. By observing the number of tasks completed under different delay constraints, the figure can assess the system's performance under different delay constraints and our proposed algorithm. The task completion rate of the matching algorithm for batch processing is always higher than that of the random and greedy algorithm under high utilization with high-delay requests. The delay loss limit is 35 s, which exceeds the system's maximum delay time. If the delay is excessive, the system will no longer handle offloading tasks with a response time exceeding 35 s. When the system no longer processes tasks with high latency, it can release computing and network resources, allowing these resources to be used to handle other, more urgent, or important tasks, thereby improving the system's overall efficiency and resource utilization. As a result, the proactive task offloading mechanism in resource-constrained environments can improve system resource utilization and exhibit good universality. It is suitable for various application scenarios with varying delay loss limitations.

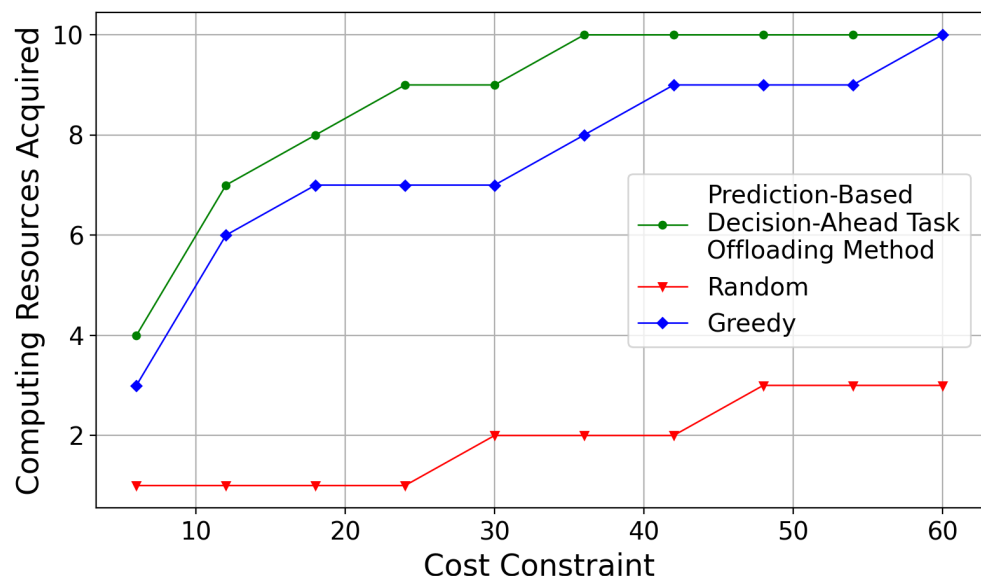


Figure 7. The cost constraint versus computing resources.

Figure 8 represents the number of participants versus the total cost. The average delay represents the system's loss and mainly refers to latency loss. If the total latency of the system always increases as the number of uninstalls increases, it usually means that the number of offloading tasks being processed by the system is also increasing. This is because the system's total delay is usually caused by an increase in the number of tasks. As the number of user requests increases and the system's complexity increases, the advantages of being able to complete matching in a relatively short amount of time due to the proactive task offloading algorithm become more pronounced. It is evident that the batch processing matching algorithm outperforms the other two benchmarks. The overall growth of delay loss in the system is not linear. When the number reaches 30, the system's resources may be partially utilized, resulting in high latency. As the number of tasks increased to 40, the system may have optimized resource allocation, resulting in more resource rational

utilization and reduced latency. However, as the number of tasks continues to increase, the system's resources may reach saturation, leading to an increase in latency.

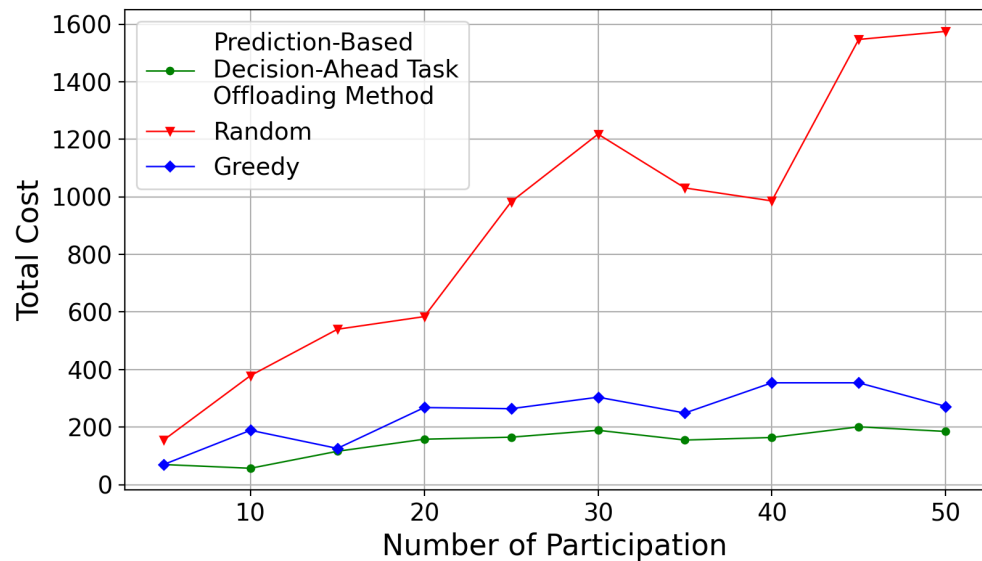


Figure 8. The number of participants versus the total cost.

Figure 9 demonstrates that the greedy algorithms can handle resource scheduling for requested tasks. By introducing the expected error, performance can be improved, and stability can be enhanced. As the number of eligible offloading executors increases, the improved greedy algorithm continues to exhibit superiority, effectively resolving the offloading problem we presented. This is due to the fact that the local optimal solution of greedy algorithms may not necessarily guarantee obtaining the global optimal solution. As the number of users participating in offloading increases, if the greedy algorithm can find a suitable strategy to fully utilize more user resources, the number of offloading will increase as the number of participating users increases. However, there is also a growing problem space for task offloading, and greedy algorithms may fall into the limitations of local optima when selecting local optimal solutions, resulting in the inability to obtain global optimal solutions and leading to a decrease in the number of offloads. Therefore, the number of offloads solved by greedy algorithms may show changes in growth and decrease rather than a stable growth trend.

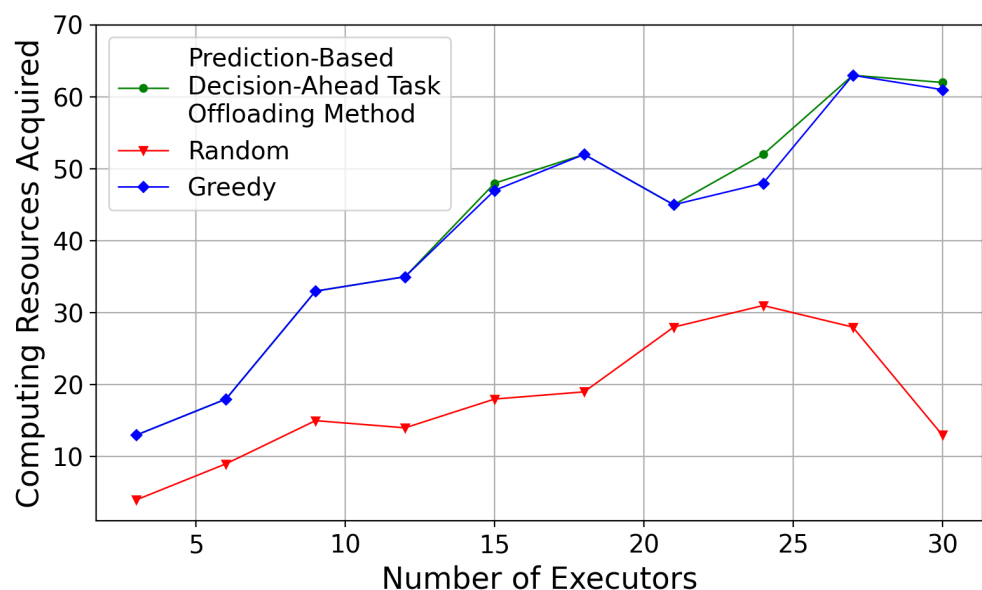


Figure 9. Implementation of improved G-kk algorithm for pre-emptive task decision-making.

7. Conclusions

In this paper, we proposed an energy-efficient task offload and resource allocation method for VIoT systems. In order to consider various resource availability and task latency requirements, we proposed two LSTM-based task volume prediction methods: batch prediction and individual prediction, which can provide more accurate task characteristics for delay sensitivity. We proposed implementing a proactive task offloading mechanism in resource-constrained environments and a polynomial-time optimal pre-emptive task offloading mechanism based on our prediction results. In a proactive task offloading mechanism in a resource-constrained environment, we designed and improved the system resource utilization rate from the perspective of a contract-matching integration algorithm. The simulation evaluated the superior performance of the algorithm in obtaining computing resources under different delay constraints of the system. Additionally, we extended the task offloading problem with lower latency requirements and proposed an improved greedy algorithm to solve the NP-complete problem of selecting task executors, which we named a polynomial-time optimal pre-emptive task offloading mechanism. We discussed the time complexity of the new algorithm as $O(\frac{1}{\epsilon}+1)$, and the simulation results showed its superiority in improving system resource efficiency. At the same time, the limitation of local optimal solutions also led to unstable growth in the number of unloading instances. The results demonstrate the effectiveness of our proposed prediction-based pre-emptive decision-making mechanism in meeting task latency requirements and improving resource utilization. Our future work will focus on enhancing the adaptability of prediction-based task offloading through dynamic configuration. By using more accurate predictions of user demand and customizing them based on user preferences and behavioral habits, we provide task-offloading solutions that align with user preferences.

Author Contributions: Conceptualization, D.L. and Y.Z.; Methodology, D.L., P.W., Q.W. and Y.D.; Software, Z.H.; Validation, Z.H.; Investigation, P.W.; Resources, Y.Z.; Writing—original draft, D.L. and Q.W.; Supervision, Y.D. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China under Grants 62272391; and in part by the Key Industry Innovation Chain of Shaanxi under Grant 2021ZDLGY05-08.

Data Availability Statement: The research data of this paper are available by contracting the corresponding author upon reasonable requests.

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

References

1. Chen, C.W. Internet of Video Things: Next-Generation IoT with Visual Sensors. *IEEE Internet Things J.* **2020**, *7*, 6676–6685. [\[CrossRef\]](#)
2. Ji, W.; Xu, J.; Qiao, H.; Zhou, M.; Liang, B. Visual IoT: Enabling Internet of Things Visualization in Smart Cities. *IEEE Netw.* **2019**, *33*, 102–110. [\[CrossRef\]](#)
3. Ma, J.; Liu, L.; Song, H.; Fan, P. On the Fundamental Tradeoffs Between Video Freshness and Video Quality in Real-Time Applications. *IEEE Internet Things J.* **2021**, *8*, 1492–1503. [\[CrossRef\]](#)
4. Li, L.; Ota, K.; Dong, M. Deep Learning for Smart Industry: Efficient Manufacture Inspection System with Fog Computing. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4665–4673. [\[CrossRef\]](#)
5. Duan, L.; Lou, Y.; Wang, S.; Gao, W.; Rui, Y. AI-Oriented Large-Scale Video Management for Smart City: Technologies, Standards, and Beyond. *IEEE MultiMedia* **2019**, *26*, 8–20. [\[CrossRef\]](#)
6. Chen, N.; Chen, Y.; You, Y.; Ling, H.; Liang, P.; Zimmermann, R. Dynamic Urban Surveillance Video Stream Processing Using Fog Computing. In Proceedings of the 2016 IEEE Second International Conference on Multimedia Big Data (BigMM), Taipei, Taiwan, 20–22 April 2016; pp. 105–112.
7. Yang, S.W.; Tickoo, O.; Chen, Y.K. A framework for visual fog computing. In Proceedings of the 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, USA, 28–31 May 2017; pp. 1–4.
8. Tran, T.X.; Pompili, D. Adaptive Bitrate Video Caching and Processing in Mobile-Edge Computing Networks. *IEEE Trans. Mob. Comput.* **2019**, *18*, 1965–1978. [\[CrossRef\]](#)
9. Zeng, F.; Wang, C.; Ge, S.S. A Survey on Visual Navigation for Artificial Agents With Deep Reinforcement Learning. *IEEE Access* **2020**, *8*, 135426–135442. [\[CrossRef\]](#)

10. Kochan, O.; Beshley, M.; Beshley, H.; Shkoropad, Y.; Ivanochko, I.; Seliuchenko, N. SDN-based Internet of Video Things Platform Enabling Real-Time Edge/Cloud Video Analytics. In Proceedings of the 2023 17th International Conference on the Experience of Designing and Application of CAD Systems (CADSM), Jaroslaw, Poland, 22–25 February 2023; pp. 1–5.
11. Qingqing, L.; Queralta, J.P.; Gia, T.N.; Tenhunen, H.; Zou, Z.; Westerlund, T. Visual Odometry Offloading in Internet of Vehicles with Compression at the Edge of the Network. In Proceedings of the 2019 Twelfth International Conference on Mobile Computing and Ubiquitous Network (ICMU), Kathmandu, Nepal, 4–6 November 2019; pp. 1–2.
12. Zhu, C.; Chiang, Y.H.; Mehrabi, A.; Xiao, Y.; Ylä-Jääski, A.; Ji, Y. Chameleon: Latency and Resolution Aware Task Offloading for Visual-Based Assisted Driving. *IEEE Trans. Veh. Technol.* **2019**, *68*, 9038–9048. [\[CrossRef\]](#)
13. Wang, Y.; Xu, J.; Ji, W. A Feature-based Video Transmission Framework for Visual IoT in Fog Computing Systems. In Proceedings of the 2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), Cambridge, UK, 24–25 September 2019; pp. 1–8.
14. Li, H.; Wu, S.; Li, D.; Jiao, J.; Zhang, N.; Zhang, Q. Optimal Offloading of Computing-intensive Tasks for Edge-aided Maritime UAV Systems. In Proceedings of the 2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring), Helsinki, Finland, 19–22 June 2022; pp. 1–6.
15. Trinh, H.; Callyam, P.; Chemodanov, D.; Yao, S.; Lei, Q.; Gao, F.; Palaniappan, K. Energy-Aware Mobile Edge Computing and Routing for Low-Latency Visual Data Processing. *IEEE Trans. Multimed.* **2018**, *20*, 2562–2577. [\[CrossRef\]](#)
16. Hussain, T.; Muhammad, K.; Del Ser, J.; Baik, S.W.; de Albuquerque, V.H.C. Intelligent Embedded Vision for Summarization of Multiview Videos in IIoT. *IEEE Trans. Ind. Inform.* **2020**, *16*, 2592–2602. [\[CrossRef\]](#)
17. Gao, H.; Yu, L.; Khan, I.A.; Wang, Y.; Yang, Y.; Shen, H. Visual Object Detection and Tracking for Internet of Things Devices Based on Spatial Attention Powered Multidomain Network. *IEEE Internet Things J.* **2023**, *10*, 2811–2820. [\[CrossRef\]](#)
18. Zhang, X.; Wei, X.; Zhou, L.; Qian, Y. Social-Content-Aware Scalable Video Streaming in Internet of Video Things. *IEEE Internet Things J.* **2022**, *9*, 830–843. [\[CrossRef\]](#)
19. Choi, P.; Ham, D.; Kim, Y.; Kwak, J. VisionScaling: Dynamic Deep Learning Model and Resource Scaling in Mobile Vision Applications. *IEEE Internet Things J.* **2024**, *11*, 15523–15539. [\[CrossRef\]](#)
20. Hung, Y.H.; Wang, C.Y.; Hwang, R.H. Optimizing Social Welfare of Live Video Streaming Services in Mobile Edge Computing. *IEEE Trans. Mob. Comput.* **2020**, *19*, 922–934. [\[CrossRef\]](#)
21. Li, J.; Sun, J.; Qian, Y.; Shu, F.; Xiao, M.; Xiang, W. A Commercial Video-Caching System for Small-Cell Cellular Networks Using Game Theory. *IEEE Access* **2016**, *4*, 7519–7531. [\[CrossRef\]](#)
22. Blanco-Filgueira, B.; Garcia-Lesta, D.; Fernández-Sanjurjo, M.; Brea, V.M.; López, P. Deep Learning-Based Multiple Object Visual Tracking on Embedded System for IoT and Mobile Edge Computing Applications. *IEEE Internet Things J.* **2019**, *6*, 5423–5431. [\[CrossRef\]](#)
23. Lu, Q.; Lu, H.; Yang, X.; Chen, F. MDUcast: Multi-Device Uplink Uncoded Video Transmission in Internet of Video Things. In Proceedings of the 2023 IEEE Wireless Communications and Networking Conference (WCNC), Glasgow, UK, 26–29 March 2023; pp. 1–6.
24. Singh, R.; Chowdhury, D.R.; Nandi, S.; Nandi, S.K. ATOM: A Decentralized Task Offloading Framework for Mobile Edge Computing through Blockchain and Smart Contracts. In Proceedings of the 2023 IEEE International Conference on Metaverse Computing, Networking and Applications (MetaCom), Kyoto, Japan, 26–28 June 2023; pp. 408–412.
25. Bai, T.; Heath, R.W. Location-Specific Coverage in Heterogeneous Networks. *IEEE Signal Process. Lett.* **2013**, *20*, 873–876.
26. Mahara, G.S.; Gangele, S. Fake news detection: A RNN-LSTM, Bi-LSTM based deep learning approach. In Proceedings of the 2022 IEEE 1st International Conference on Data, Decision and Systems (ICDDS), Bangalore, India, 2–3 December 2022; pp. 1–6.
27. Hernández, D.; Cecília, J.M.; Calafate, C.T.; Cano, J.C.; Manzoni, P. The Kuhn-Munkres algorithm for efficient vertical takeoff of UAV swarms. In Proceedings of the 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), Helsinki, Finland, 25–28 April 2021; pp. 1–5.
28. Hajarian, M.; Shahbahrani, A.; Hoseini, F. A parallel solution for the 0–1 knapsack problem using firefly algorithm. In Proceedings of the 2016 1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC), Bam, Iran, 9–11 March 2016; pp. 25–30.
29. Xu, L.; Lin, S.; Zeng, J.; Liu, X.; Fang, Y.; Xu, Z. Greedy Criterion in Orthogonal Greedy Learning. *IEEE Trans. Cybern.* **2018**, *48*, 955–966. [\[CrossRef\]](#) [\[PubMed\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.