processes

MDPI

Article

Task-Offloading and Resource Allocation Strategy in Multidomain Cooperation for IIoT

Zuojun Dai *, Ying Zhou, Hui Tian and Nan Ma

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
* Correspondence: george.dai@bupt.edu.cn

Abstract: This study proposes a task-offloading and resource allocation strategy in multidomain cooperation (TARMC) for the industrial Internet of Things (IIoT) to resolve the problem of the non-uniform distribution of task computation among various cluster domain networks in the IIoT and the solidification of traditional industrial wireless network architecture, which produces low efficiency of static resource allocation and high delay in closed-loop data processing. Based on the closed-loop process of task interaction of intelligent terminals in wireless networks, the proposed strategy constructs a network model of multidomain collaborative task-offloading and resource allocation in IIoT for flexible and dynamic resource allocation among intelligent terminals, edge servers, and cluster networks. Considering the partial offloading mechanism, various tasks were segmented into multiple subtasks marked at bit-level per demand, which enabled local and edge servers to process all subtasks in parallel. Moreover, this study established a utility function for the closed-loop delay and terminal energy consumption of task processing, which transformed the process of multidomain collaborative task-offloading and resource allocation into the problem of task computing revenue. Furthermore, an improved Cuckoo Search algorithm was developed to derive the optimal offloading position and resource allocation decision through an alternating iterative method. The simulation results revealed that TARMC performed better than strategies.

Keywords: cross-domain; MEC; resource allocation; IIOT

check for updates

Citation: Dai, Z.; Zhou, Y.; Tian, H.; Ma, N. Task-Offloading and Resource Allocation Strategy in Multidomain Cooperation for IIoT. *Processes* **2023**, *11*, 132. https://doi.org/10.3390/pr11010132

Academic Editor: Xiong Luo

Received: 4 December 2022
Revised: 21 December 2022
Accepted: 23 December 2022
Published: 2 January 2023

Copyright: © 2023 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 advanced industrial manufacturing modes such as Industry 4.0, intelligent factory, and flexible manufacturing, communication technology and the manufacturing industry have become deeply integrated in recent years. Consequently, the digital and integrated industrial Internet of Things (IIoT) has emerged as a research hotspot [1–5]. However, the advantages of the three major application scenarios of 5G are incompatible and cannot coexist simultaneously. Moreover, the synchronous realization of ultralow closed-loop delay transmission, large connection, and large-bandwidth transmission is challenging for large-scale machine differentiated services. Therefore, further research is required to transform existing industrial wireless network architectures and resource allocation methods.

Currently, the mainstream wireless edge network adopts a centralized network mode on the terminal side, [6,7], and the information is transmitted between the nodes via the access edge server and cloud platform, which considerably increases the delay of data transmission between the nodes. For the large-scale wireless edge network, this rigid and centralized network mode poses problems such as high network-computation cost, serious resource conflict, and reduced network performance, which yields inferior scalability of the edge networks. This further raises the difficulty of supporting flexible, adaptive resource scheduling and ubiquitous computing functions of industrial edge networks. To support a large IIoT, the cluster domain network [8,9] contains numerous terminal nodes distributed in a certain region

for constructing hierarchical networks using the clustering algorithm [10,11]. In addition, mobile edge computing (MEC) [12,13] extends cloud services to network edges that effectively coordinate the distributed edge resources. With the increasing performance requirements for computation-intensive and delay-sensitive loads in IIoT, cluster domain networks and MEC have emerged as potential solutions to accommodate complex IIoT applications for the integrated management of IIoT communications, sensing, and computing resources [14].

The traditional industrial wireless network based on the IEEE 802 protocol [15] can achieve only a 10 ms level delay in one-way communication transmission, which is vastly inadequate to achieve the performance requirements of IIoT. On one hand, the hierarchical network control mode relying on the core network yields low resource-management efficiency and does not fulfill the communication requirements of differentiated machine services. On the other hand, considering the extremely low delay requirements of closed-loop management for several IIoT applications in the manufacturing workshop, computing tasks require more fine-grained offloading strategies. To fulfill the demand for efficient interaction of information flow between massive machines in IIoT, the edge network should allow a portion of the computing tasks in mobile devices to be processed locally, and the remaining computing tasks are offloaded onto the server. According to the strict requirements for reliable real-time performance in industrial scenarios, computing tasks should be processed for real-time stored industrial data. Therefore, low-delay computing task-offloading methods and resource allocation strategies must be studied based on cluster domain network structure.

2. Related Work

In recent years, the study of computational offloading strategies has emerged as a research hotspot in the field of edge computing [16,17]. With various objectives, researchers have proposed several computational offloading strategies. To minimize task execution delay, Yang et al. [18] formulated a selection strategy for optimal offloading node as a Markov decision process and minimized the offloading delay by adopting a value iteration algorithm. Li et al. [19] studied a paradigm for dual computational offloading and proposes a hierarchical, cell-based distributed algorithm to obtain the optimal dual offloading scheme for implementing the overall delay minimization of task-offloading. Zhu et.al [20] studied the single-user multi-edge-server MEC system based on downlink NOMA to minimize task computation delay by jointly optimizing the NOMA-based transmission duration (TD) and workload offloading allocation (WOA) among edge computing servers. Luo et al. [21] proposed a self-taught-based distributed computational offloading algorithm to minimize its delay and information cost. Huang et al. [22] proposed an efficient multidevice and multi-BSs task-offloading scheme to minimize the delay of computational tasks. Gao et al. [23] proposed a two-stage computational offloading scheme to minimize task processing delay. Liao et al. [24] proposed a novel UAV-assisted edge computational framework that provided edge computational offloading based on the user distribution of time-varying hotspots to minimize average user delay. The current research primarily focuses on the simple network structure model, which is limited to the optimization of the one-way empty port-time delay that contradicts the closed-loop interaction characteristics of the information flow of the industrial intelligent machine network.

To minimize total energy consumption, Fang et al. [25] proposed a content-aware multi-subtask-offloading problem based on the individual features of subtasks with various delay requirements, under which the offloading decision and channel allocation were optimized. Aiming to obtain resource allocation and offloading decisions, Wu et al. [26] developed an efficient two-layer optimization algorithm for resolving the residual energy maximization problem. Chen et al. [27] formulated the offloading task as a stochastic optimization problem and proposed an energy-efficient dynamic offloading algorithm that minimized the energy consumption of task-offloading. In addition, Bozorgchenani et al. [28] modeled task-offloading in MEC as a constrained multi-objective optimization problem that minimizes both the energy consumption and task processing delay of the mobile

devices. More recently, Zhang et.al [29] proposed an energy-saving algorithm based on deep reinforcement learning to optimize the overall energy cost in real-time multi-user MEC systems. However, the delay and energy consumption performance may bear distinct weight coefficients, for instance, the system focuses on the delay performance by increasing the delay weight, which consequently places higher requirements for optimizing and offloading the system task.

To reduce service response delay and energy expenditure, the user can opt to offload the task to the edge server or the cloud server for execution. Therefore, offloading schemes that combine delay demand and energy consumption demand should be considered. Lu et al. [30] designed a multitask-offloading policy that could handle dense offloading requests from various mobile devices to optimize the overall execution delay and energy consumption. Wang et al. [31] developed an efficient multi-objective evolutionary algorithm to solve the problems of minimizing the response time, minimizing the energy consumption, and minimizing the cost. Guan et al. [32] proposed a novel MEC-based mobility-aware offloading model to address the in-cloud offloading scheduling problem and the load between cloud sensing problems by offloading execution efficiency, task processing delay, and energy efficiency. Aiming to reduce system energy consumption as well as computational task delay, Chen et al. [33] proposed a robust computational offloading strategy with fault recovery capabilities in intermittently connected small cloud systems. Fang et al. [34] investigated the multi-user computational task offload problem in device-enhanced MEC based on the perspective of joint optimization of channel allocation, device pairing, and offload modes, considering the significance of delay and energy consumption to maximize the total offload benefit of all computationally-intensive users in the network.

The existing work mainly studies the offloading scheme of minimizing time delay, energy consumption or synthesis under the traditional industrial wireless network architecture, and solves some optimization problems of one-way air port delay and energy consumption under the simple network structure. However, the joint optimization of task-offloading delay and energy consumption can be realized only with the traditional network architecture. In case of multidomain network collaboration and extremely low closed-loop delay demand of IIoT, the utility aspect of network closed-loop delay and terminal energy consumption cannot still be effectively balanced. So, we developed an improved multidomain collaborative task-offloading mechanism to deeply analyze the impact of the multidomain resource linkage collaboration mode on the task-offloading process of intelligent terminals in the workshop in order to solve the problem of non-uniform distribution of task computation in the traditional hierarchical network and improve the utilization of idle resources in the full-domain network in the workshop. On the other hand, we established a utility function on task processing closed-loop time delay and terminal energy consumption to transform the multidomain collaborative task-offloading and resource allocation process into a task computation gain problem in order to solve the problems of low static resource allocation efficiency and high data processing closed-loop time delay in traditional industrial wireless networks. An improved Cuckoo Search algorithm is proposed to calculate the optimal offloading location and resource allocation decision to effectively weigh the network closed-loop delay and terminal energy consumption to improve the network communication performance of a flexible manufacturing workshop.

The primary contributions of this paper are summarized as follows:

(1) This study proposes a task-offloading and resource allocation strategy in multidomain cooperation (TARMC), to investigate the closed-loop process of intelligent terminal task interaction in wireless networks and the partial offloading mechanism of edge network for IIoT.
(2) A utility function was established for the closed-loop delay and terminal energy consumption of task processing to transform the multidomain collaborative task-offloading and resource allocation process into a task computing revenue problem.
(3) An improved Cuckoo Search algorithm was developed to compute the optimal offloading location and resource allocation decisions. In addition to strengthening

the network load balance, this algorithm effectively reduced the delay and energy consumption of task processing.

(4) An experiment was designed to compare TARMC, a genetic algorithm (GA) and simulated annealing algorithm (SA), to validate the optimization of delay and energy consumption in the multidomain collaboration method based on a real IIoT environment.

3. System Model

3.1. System Model

The TARMC network model comprises the terminal layer, network layer and resource management layer, as depicted in Figure 1. The terminal layer includes industrial intelligent terminals (e.g., robotic arms, AGV) with varying computing needs. In addition, it covers the corresponding application scenarios (e.g., industrial vision, AGV collaboration, and industrial detection), responsible for real-time local processing of computing tasks, establishing communication links with network layer equipment, and requesting collaborative task-offloading services. The set of industrial intelligent terminals is defined as $D = \{D_1, D_2, \ldots, D_i, \ldots, D_N\}$, which is responsible for offloading the task to the edge server on the router end. The network layer contains multiple routers with edge servers deployed around it, which is responsible for communicating and exchanging data with other cluster domain networks. This feature enables high real-time information interaction and multidomain collaborative task-offloading services for the terminal layer. The set of edge servers can be expressed as $S = \{S_1, S_2, \ldots, S_j, \ldots, S_J\}$ and is responsible for providing computing resources to complete the computing task. Furthermore, the resource management layer hosts a 5G intelligent control base station and cloud platform, which analyzes the computing requirements of various industrial terminals and offers resource allocation strategies to edge servers.

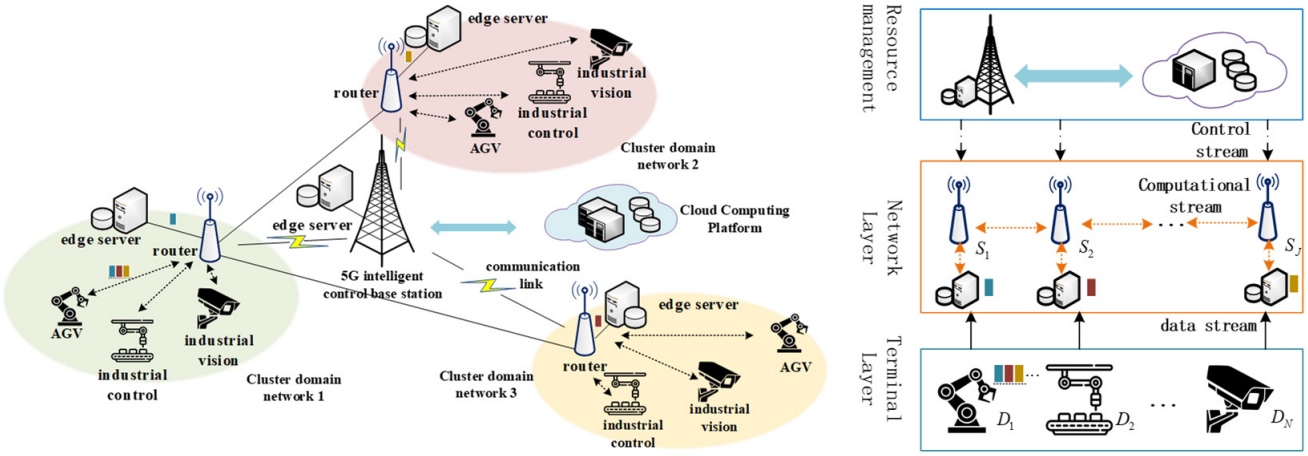

Figure 1. System model for task-offloading and resource allocation.

Upon assigning a computing task to the industrial intelligent terminal, a computing task service request will be uploaded, and the service demand will be reported by the 5G intelligent control base station through the access network. Thereafter, the edge server on the base station end receives the task request and sends a service response. The task is computed in the local processing if the industrial intelligent terminal can compute tasks within the scope of local computing power. In case the local computing power is insufficient, the 5G intelligent control base station offloads the task to an edge server installed in the remaining cluster domain networks within the communication range for processing. After processing, the calculation results will be fed back to each intelligent terminal. For non-real-time service requirements, the 5G intelligent control base station offloads the tasks to the cloud platform under a more accurate resource scheduling strategy and accessed from the cloud network.

The total frequency spectrum of the network system is segmented into orthogonal sub-channels, each with a bandwidth of B Hz. Under normal operation of the system, each industrial intelligent terminal provides a computing task for processing. Let us assume that the computing task quantity of the industrial intelligent terminal D_i is l_i, measured in Mbits; η_i denotes the ratio of output and input data for task D_i, i.e., the feedback received after task calculation is $\eta_i l_i$. The computing tasks of the industrial intelligent terminal D_i can be either locally processed or offloaded to the edge server for processing. b_i represents the local offloading ratio for an industrial intelligent terminal computing task, where ($b_i \in [0, 1]$).

Subsequently, the industrial intelligent terminal forwards the computing task to the edge server on the 5G intelligent control base station for task-offloading, and the data require to be transmitted through the subchannel corresponding to the 5G intelligent control base station. According to [35], the uplink transmission rate from the industrial intelligent terminal D_i to the edge server S_j is expressed using Shannon's formula as:

$$r_{i,j} = B \log_2 \left(1 + \frac{p_i h_{i,j}^k}{N_0} \right) \tag{1}$$

where p_i denotes the transmission power of the industrial intelligent terminal D_i, $h_{i,j}^k$ represents the channel gain of the industrial intelligent terminal D_i and the edge server S_j on the sub-channel k.$h_{i,j} = \xi_0 d_{i,j}^{-\lambda} \sigma^2$, where ξ_0 is the path loss at a distance of one meter; $d_{i,j}^{-\lambda}$ symbolizes the propagation loss, d indicates the propagation distance, λ denotes the path loss exponent; σ^2 represents the Rayleigh fading parameters; N_0 indicates the noise power of Gaussian channel.

The computing resource allocated by the industrial intelligent terminal D_i for local processing of the computing task is denoted as (CPU cycles/second), and the CPU cycles required for processing 1 bit data is expressed as δ_{Local}. Thus, the delay in local task processing can be derived as

$$t_i^{local} = \frac{l_i b_i \delta_{Local}}{f_i} \tag{2}$$

If the power of the industrial intelligent terminal D_i at idle is P_i^{local}, the energy consumption of processing the calculation task can be locally evaluated as follows:

$$E_i^{local} = P_i^{local} t_i^{local} + \rho f_i^3 t_i^{local} \tag{3}$$

where $\rho f_i^3 t_i^{local}$ denotes the additional energy consumption during the terminal computing task, ρ indicates the CPU architecture constant and $\rho = 10^{-27}$. The industrial intelligent terminal D_i can process the computational tasks through collaboration between local computing and edge computing. If the industrial intelligent terminal D_i is processing a portion of the local computing tasks, all the remaining computing tasks are transmitted to the edge server S_j most proximate to the terminal, which schedules and assigns the remaining computing tasks to the other edge servers and jointly completes the task offload. As the edge servers communicate through a wired network, the transmission time is negligible, and further potential delays (e.g., packet preprocessing and queuing delays) can be neglected. The transmission delay generated when the smart terminal D_i transmits all the remaining computing tasks to the nearest edge server S_j can be derived as follows:

$$t_i^s = \frac{l_i (1 - b_i)}{r_{i,j}} \tag{4}$$

Therefore, the transmission energy consumption of the computing task offloaded by D_i to S_j can be derived as follows:

$$E_{D_i S_j}^e = p_i t_i^s \tag{5}$$

The computing resource allocated by the edge server S_j to the industrial intelligent terminal D_i is f_j^i, the CPU of the edge server to process 1 bit data is δ_{MEC}, and the power required to perform the computing task is P_j^i. After offloading the task, multiple edge servers can perform the computing task, and the processing time of the computing task on the edge server S_j can be evaluated as follows:

$$t_{i,j}^m = \frac{\omega_{i,j} l_i (1 - b_i) \delta_{MEC}}{f_j^i} \tag{6}$$

where $\omega_{i,j}$ denotes the ratio of computing tasks on edge server S_j to total computing task of D_i offloaded to servers. Let us assume that the transmitting power of the edge server S_j to p_j. After processing the task, the multiple collaborative edge servers will feedback the results of the remaining tasks to the edge server S_j located near the intelligent terminal. Thus, the transmission delay occurring when S_j feeds back all the remaining task results to the terminal D_i can be expressed as follows:

$$t_i^c = \frac{(1 - b_i) l_i \eta_i}{r_{i,j}} \tag{7}$$

Therefore, if the intelligent terminal D_i receives the edge server S_j feedback result, the corresponding transmission energy consumption will be:

$$E_i^{offload} = p_i \cdot t_i^c \tag{8}$$

Considering the simultaneous processing of the computing tasks on multiple edge servers, the industrial intelligent terminal D_i records the total processing duration of offloading the remaining tasks to the edge server for auxiliary computing process as T_i^m, and $T_i^m = \max_{j=1,2,\ldots,\mathrm{M}} t_{i,j}^m$. Therefore, in case the remaining computing task is offloaded to the edge server and the feedback data is received, the multidomain collaborative task-processing delay can be expressed as follows:

$$t_i^{offload} = t_i^s + T_i^m + t_i^c \tag{9}$$

In case the industrial intelligent terminal D_i completes the corresponding computing task in local processing, it waits for the edge server to process together and provides feedback to D_i for the final result. In another case, if the industrial intelligent terminal D_i has not yet completed the local processing of the corresponding computing task, whereas the edge server has completed collaborative computing, the industrial intelligent terminal D_i receives the feedback of the final result after processing the local computing task. Therefore, the closed-loop time delay of the task processing at the industrial intelligent terminal D_i can be stated as follows:

$$t_i = \begin{cases} t_i^{offload}, t_i^{local} < t_i^{offload} \\ t_i^{local}, t_i^{local} \geq t_i^{offload} \end{cases} \tag{10}$$

For the industrial intelligent terminal D_i, the total energy consumption is stated as follows

$$E_i = \begin{cases} E_i^{local} + E_{D_i S_j}^e + E_i^{offload}, t_i^s + T_i^m < t_i^{local} \\ E_i^{local} + E_{D_i S_j}^e + E_i^{offload} + P_i^{local}(t_i^s + T_i^m - t_i^{local}), t_i^s + T_i^m \geq t_i^{local} \end{cases} \tag{11}$$

The total closed-loop time for task processing of all industrial intelligent terminals in the system can be expressed as:

$$T_{total} = \sum_{i=1}^{N} t_i \tag{12}$$

The total power consumption of all industrial intelligent terminals to complete the task-offloading can be stated as

$$E_{total} = \sum_{i=1}^{N} (E_i) \tag{13}$$

3.2. Optimization Objective

In practice, the delay and energy consumption performance can exhibit varying weight coefficients, for instance, the system improves the delay weight to focus on the delay performance when the AGV is included in route planning. Let us assume that the two weight coefficients are denotes as ω_T and ω_E, respectively. The impact of such delay and energy consumption on the performance of industrial intelligent terminals can be adjusted through ω_T and ω_E, and such a design can expand the applicability of the model. This study considers the premise of ensuring the energy consumption of all industrial intelligent terminals to complete task-offloading, minimizes the closed-loop delay of task processing, and obtains the optimal resource allocation strategy along with the computing task-offloading scheme of the edge server. The utility function is defined as follows:

$$f(b, f) = \frac{1}{N} \left(\sum_{i=1}^{N} \omega_T t_i + \sum_{i=1}^{N} \omega_E E_i \right) \tag{14}$$

The optimization objective can be expressed as:

$$\begin{aligned} &\min_{b,p,f} f(b, f) \\ &s.t. C1 : b_i \in [0, 1], \forall i \\ &\quad C2 : 0 \le f_i \le f_{i,\max}, \forall i \\ &\quad C3 : 0 \le f_j^i \le f_{j,\max}, \forall i \\ &\quad C4 : t_i \le t_{i,\max}, \forall i, j \end{aligned} \tag{15}$$

where constraint C_1 represents the range of offloading ratio for the industrial intelligent terminal computing task; C_2 indicates that the industrial intelligent terminals do not exceed the maximum allocated local computing resources; C_3 represents that the computing resources allocated to the edge server for industrial intelligent terminal tasks do not exceed the maximum allocated computing resources of the edge server; C_4 indicates the maximum value of the task calculation delay for the industrial intelligent terminal.

4. Improved Cuckoo Search Algorithm

Unlike the NP difficult problem under complete offloading mechanism, Equation (15) can be reduced to the ordinary combination optimization problem under partial offloading mechanism. To this end, an improved Cuckoo Search algorithm is proposed in this paper. In principle, the algorithm can weigh the number of local and global searches through adaptive discovery probability and step size, as well as conduct a local fine search representing the global optimal solution to improve the operation accuracy and search efficiency. The idea of a differential evolution algorithm is introduced to adjust, cross, and select the process of nest update position. By inheriting the optimal solution genetic information, the algorithm avoids intersection with the local optimal and converges speedily to provide the optimal resource allocation results.

Let us assume the nest location x_i^t of the i-th nest in generation t and M denotes the dimensionality, where $x_i^t = \{x_{i1}^t, x_{i2}^t, \ldots, x_{im}^t, \ldots, x_{iM}^t\}$. According to [36], the cuckoo's updated expression of the path and location for deriving a parasitic nest follows:

$$x_{i,m}^{t+1} = x_{i,m}^{t} + \alpha \cdot Rand \cdot Levy(\beta_1),\ i = 0, 1, 2, \ldots, J \tag{16}$$

where x_i^{t+1} denotes the new position of the i-th nest at nest position x_i^t in generation t after a global update. $x_{i,m}^t$ denotes the value of the i-th nest in the m-dimension in the nest position at generation t. α indicates a step factor and $Rand$ symbolizes a uniform distribution between $(0, 1)$. $Levy(\beta_1)$ represents a random wandering process formed by the flight of cuckoo $Levy$, $Levy(\beta_1) \sim u = t^{-\beta_1},\ (1 < \beta_1 \leq 3)$, where β_1 represents the impact factor, typically, $\beta_1 = 1.5$; According to the [36], the expression for the $Levy$ distribution is stated as follows.

$$Levy(\beta_1) = 0.01 \cdot \frac{u}{|v|^{1/\beta_1}} \cdot \left(x_{j,m}^t - b_{g,m}^t\right),\ j, g = 0, 1, 2, \ldots, J \tag{17}$$

where u and v both follow a normal distribution, i.e., $u \sim N(0, \sigma_u^2)$, $v \sim N(0, \sigma_v^2)$, $\sigma_u = \left\{ \frac{\Gamma(1+\beta_1)\cdot\sin(\pi\cdot\beta_1/2)}{\Gamma[(1+\beta_1)/2]\cdot\beta_1\cdot 2^{(\beta_1-1)/2}} \right\}^{1/\beta_3}$, $\sigma_v = 1$.

$b_g^t = \left\{ b_{g,1}^t, b_{g,2}^t, \ldots, b_{g,m}^t \ldots, b_{g,M}^t \right\}$ denotes the current optimal solution space covered by the algorithm in the current search state, and if the current nest location corresponds with the optimal solution space, the magnitude of step adjustment is 0, i.e., $Levy(\beta_1) = 0$. In addition, the host bird of a parasitized nest will abandon the nest with an P_α probability of recognizing an egg parasitized by a cuckoo.

4.1. Adaptive Adjustment of Discovery Probabilities

In the original cuckoo algorithm, a given discovery probability is generally used to control the global search and preference random wandering process, which is conducive toward the balance between global and local search as the number of iterations increases. Thus, to improve the algorithm's search performance, this study applies dynamic discovery probability instead of fixed discovery probability P_α

$$P_\alpha = \begin{cases} \frac{1}{1+\frac{t_{it}}{T_{it}}\cdot e}, 0 < t_{it} \leq \beta_2 T_{it} \\ 1 - e^{\frac{t_{it}}{T_{it}}-1} + \gamma \cdot \frac{t_{it}}{T_{it}}, \beta_2 T_{it} < t_{it} \leq T_{it} \end{cases} \tag{18}$$

where γ denotes the correction factor, generally, $\gamma = 0.1$, and β_2 indicates the trade-off factor, mostly, $0.5 \leq \beta_2 \leq 1$. This segmentation function represents a progressive decline in the probability of discovery P_α till the number of iterations t_{it} reaches $\beta_2 T_{it}$, and it is suitable for global searches over a large area with improved search efficiency. As the number of iterations t_{it} exceeds that of $\beta_2 T_{it}$, the probability of discovery decreases significantly, thereby enabling local search in a small area to improve the search accuracy.

4.2. Adaptive Adjustment of Step Size

Similarly, the step size of the Levy flight can be continuously decreased with the adaptive adjustment of the step-size factor in each iteration. Specifically, a larger step-size factor in the early iterations of the algorithm is conducive toward improving the global search capability of the algorithm and ensuring speedy convergence in the early stages of the algorithm. In the later stages of the algorithm iteration, the scope of the local search is narrowed by decreasing the step size for enhancing the local search performance of the algorithm.

$$\alpha = \beta_3 e^{\left(-\frac{t_{it}}{T_{it}}+1\right)} \tag{19}$$

where β_3 denotes the correction factor, typically, $\beta_3 = 0.5$. In addition, after a global search in the Levy flight, certain solutions will further perform a local search to update the location, thereby retaining a more accurate set of solutions. During this local search, differential evolution of $x_{i,m}^t$ is performed by analyzing the differences between current and excellent

individuals in the population to ensure that a great amount of genetic information from the excellent individuals is inherited by their offspring. The specific process is stated as follows.

4.3. Differential Evolution

First, the genetic information of multiple individuals can be obtained by mutating the individuals through a differential strategy, wherein the mutated individual $u_{i,m}^t$ is expressed as

$$u_{i,m}^t = x_{i,m}^t + \kappa \cdot \left(x_{p,m}^t - x_{q,m}^t \right),\ p, q = 0, 1, 2, \ldots, J \tag{20}$$

where κ denotes the scaling factor, $x_{p,m}^t$ represents the value in the m-th dimension of the nest position in generation t. After variation, individual $u_{i,m}^t$ retains an amount of information regarding maternal $x_{i,m}^t$, and, simultaneously, inherits information from individuals $x_{p,m}^t$ and $x_{q,m}^t$ to realize the transmission of the information between individuals. Thereafter, the candidate individual $v_{i,m}^t$ is generated by crossing over the maternal and intervariant information, thus ensuring that at least one set of individual information in the succeeding generation is contributed by the variant individual. The $v_{i,m}^t$ can be expressed as:

$$v_{i,m}^t = \begin{cases} u_{i,m}^t, & \alpha_2 \le CR \ or \ m = \beta_4 \\ x_{i,m}^t, & otherwise \end{cases} \tag{21}$$

where $CR \in [0, 1]$ denotes the cross probability, $\alpha_2 = rand(0, 1)$ represents the random number generated in $[0, 1]$ interval, and $\beta_4 = unidrnd(M)$ indicates the random positive integer generated in $[1, M]$ interval.

Finally, the dominant relationship between individual $v_{i,m}^t$ and parent $x_{i,m}^t$ is determined by comparing the optimized objective function size, and a new generation of individual $x_{i,m}^{t+1}$ is generated to inherit the traits of excellent individuals in the succeeding generation.

$$x_{i,m}^{t+1} = \begin{cases} v_{i,m}^t, & T(v_{i,m}^t) < T(x_{i,m}^t) \\ x_{i,m}^t, & otherwise \end{cases} \tag{22}$$

In summary, the specific flow of the improved Cuckoo Search algorithm is stated as follows in Algorithm 1.

Algorithm 1 Improved Cuckoo Search Algorithm

InpuT: System parameters include set D of industrial intelligent terminal, set S of edge server, calculation task amount l_i of industrial intelligent terminal D_i and other indicators;
The cuckoo algorithm parameters include the nest position set $x_i^t = \{x_{i1}^t, x_{i2}^t, \ldots, x_{im}^t, \ldots, x_{iM}^t\}$, the maximum number of iterations t_{max}, etc.
Initialization: Initialize the nest position and other parameters to record the current optimal solution.
Output: begin

1. **for** $t < t_{max}$ **do**
2. Update Bird's Nest location according to Equation (16)
3. Calculate and obtain the optimal solution according to Equation (15), and preserving the optimal solution space.
4. Adaptive adjustment of discovery probability and step size according to Equation (18) and Equation (19)
5. **if** $rand(0, 1) > P_\alpha$ **then**
6. Differential evolution according to Equations (20) and (21) and Equation (21) to locally update the nest position and obtain the optimal solution.
7. **end if**
8. **end for**
9. Outputs optimal task-offloading and resource allocation results.
10. **end**

Due to the combinatorial nature of the optimization problem, this paper first analyzes the time complexity of each embedded subprocess in the improved Cuckoo Search algorithm, and finally performs an overall analysis. Assume that the time to generate distributed random numbers is ξ_1, the population size is N, and the dimensionality of the problem is M. In the first iteration, the complexity of updating the positions of all nests is $O(\xi_1 + NM)$, the complexity of computing the optimal solution is $O(NM)$, the complexity of adaptively adjusting the discovery probability and step size is $O(1)$, and the complexity of the global search is $O(N)$. Because the discovery probability P_α changes adaptively, the number of populations in the differential evolution process also changes dynamically, the number of evolving populations is set to $P_\alpha \cdot N$, and the complexity of differential evolution is $O(P_\alpha NM)$, where $P_\alpha NM \in (0, NM)$. Therefore, the complexity of the improved Cuckoo Search algorithm after one iteration is $O(2NM) = O(NM)$ in the worst case. Moreover, we considered that the number of iterations of the algorithm cannot give a closed-form solution, this paper assumes that the maximum number of iterations is t_{max}. The complexity of the improved Cuckoo Search algorithm in the worst case is $O(t_{max}NM)$ when it iterating to the last convergence at same time. In addition, because the processes of global search and local search in this algorithm are jointly optimization, its convergence speed is fast, and we will verify the algorithm convergence by simulation in the next section.

5. Simulation and Results

An industrial manufacturing scenario within 300 m × 300 m area was simulated in MATLAB to cross-sectional to compare the performance of the TARMC strategy with those of GA algorithm [37] and SA algorithm [38]. The variations in closed-loop delay and endpoint energy consumption in processing tasks were obtained from simulations of non-cross-domain and cross-domain collaborative network architectures. In addition, we analyzed the impact of various resource allocation algorithms on the closed-loop delay and terminal energy consumption of task processing in case of interaction with multiple orders of magnitude of industrial smart terminals. The specific simulation parameters are summarized in Table 1.

Table 1. Simulation parameters.

Parameters	Numerical Values
Sub-channel bandwidth, B	1 MHz
Number of CPU cycles in industrial intelligent terminals, δ_{Local}	[500, 2000] cycles/bit
Number of CPU cycles for edge servers, δ_{MEC}	[500, 2000] cycles/bit
Industrial intelligent terminal idle operating power, P_i^{local}	0.3 W
Industrial intelligent terminal transmission power, p_i	1.3 W
Industrial Smart Terminal Computing Resources, $f_{i,\max}$	4 GHz
Edge server computing resources, $f_{j,\max}$	20 GHz
Channel gain, $h_{i,j}$	10^{-5}
Gaussian white noise, N_0	10^{-13} W

The trend of variations in the average closed-loop delay during processing tasks under the simulated resource allocation strategies is presented in Figure 2, where the horizontal axis represents the number of iterations and the vertical axis represents the average closed-loop delay of task processing. Furthermore, the plots with diamond, circular, star, vertical, triangular, and rectangular curves represent the TARMC strategy, TARMC strategy under non-cross-domain networks, GA strategy, and GA strategy under non-cross-domain networks, SA strategy, and SA strategy under non-cross-domain networks, respectively. As observed, the average closed-loop delay for processing tasks with TARMC, GA skimming, and SA strategies under cross-domain networks was much less than that of TARMC, GA and SA strategies under non-cross-domain networks. This is because the number and types of industrial terminals vary for each cluster domain network in the actual

industrial production, and the corresponding task calculation quantity varies as well. If the task is calculated only in the current cluster domain network, the computational resource allocation of the global network in this industrial area is non-uniform, thereby resulting in low resource utilization and increased average closed-loop delay of task processing. A multidomain collaborative network enables cross-domain computing processing in performing tasks, which allocates numerous real-time tasks to cluster domain networks with more idle resources for collaborative computing. Thus, this feature reduces the task processing delay and diminishes the corresponding average closed-loop delay for task processing. In addition, the average closed-loop delay of processing tasks with the TARMC strategy is substantially less than that of the GA and SA strategies. This is because the TARMC strategy considers the task between large-scale industrial intelligent terminals, edge servers, and the cluster domain network communication interaction process according to the intelligent manufacturing workshop closed-loop control business requirements. In addition, it formulates the average closed-loop delay of task processing as an optimization function to ensure the communication performance of new IIoT networks, which effectively reduces the average closed-loop delay of task processing. However, the GA and SA strategies are based on a simple network model, considering the one-way empty port delay as the optimization goal and ignoring the impact of the task interaction between edge servers and cluster domain network on the overall delay. Thus, they are unable to satisfy the network requirements of intelligent manufacturing workshop, which increases the average closed-loop delay of task processing. As observed in Figure 2, the average closed-loop delay for task processing with TARMC strategy under multidomain synergy was 1.8401 s, whereas that for TARMC strategy under non-cross-domain networks was 2.6372 s, demonstrating an improvement of 30.2%. In addition, the average closed-loop delay in processing tasks with GA and SA strategies under multidomain collaboration was 3.2867 s and 2.0797 s, respectively, which corresponds to a performance improvement of 44.0% and 11.5% in comparison with the latter.

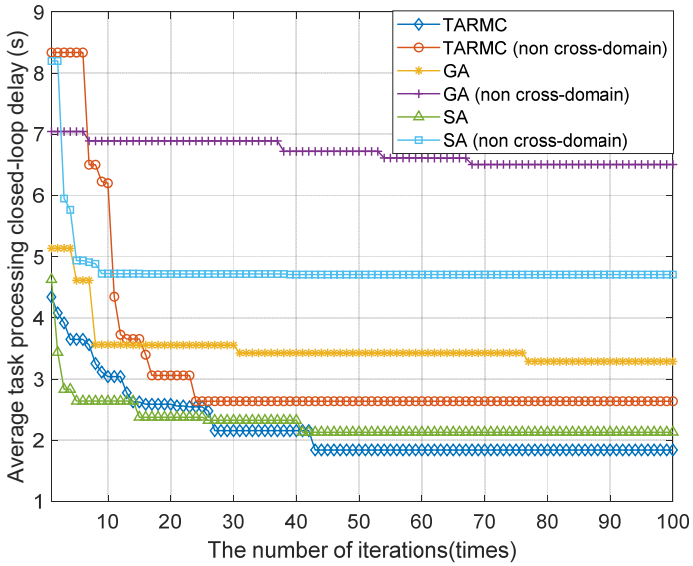

Figure 2. Variations in closed-loop delay of task processing under various resource allocation policies.

The trend of variations in average energy consumption between multiple resource allocation strategies is discussed herein. In Figure 3, the horizontal axis represents the number of iterations, and the vertical axis denotes the average energy consumption. Furthermore, the plots with diamond, circular, star, vertical, triangular, and rectangular curves represent the TARMC strategy, TARMC strategy under non-cross-domain networks, GA strategy, and GA strategy under non-cross-domain networks, SA strategy, and SA strategy under non-cross-domain networks, respectively. As observed, the average energy consumption of the TARMC, GA skimming and SA strategies under cross-domain networks is considerably

less than the TARMC, GA strategy, and SA strategy under non-cross-domain networks. This is because the multidomain collaborative network can completely schedule the computing resources in the entire region, reduce the delay of industrial intelligent terminals in the local processing tasks, decrease the computing energy consumption of industrial intelligent terminals, and eventually, improve the average energy consumption of industrial intelligent terminals compared to the non-cross-domain network mode. In addition, the average energy consumption of the TARMC strategy is much less than that of the GA and SA strategies, as indicated in the figure. This is because the TARMC strategy can adaptively adjust the weight of the closed-loop delay and average energy consumption of task processing according to the requirements of the industrial intelligent terminal business, evaluate the priority of optimizing the average energy consumption, and thus, effectively reduce the average energy consumption. However, the GA and SA strategies only consider the impact of one-way air interface delay on the data processing process of industrial intelligent terminals, which inevitably sacrifices energy consumption in the optimization process, resulting in high average energy consumption. Moreover, as noted from Figure 3, the average energy consumption of the TARMC strategy under multidomain collaboration was 0.5091 J, whereas that under non-cross-domain networks was 1.2462 J, corresponding to an improvement of 59.2%. In addition, the average energy consumption of GA and SA strategies under multidomain collaboration was 1.3090 J and 0.9193 J, respectively, which improved by 61.1% and 44.6% in comparison to the latter.

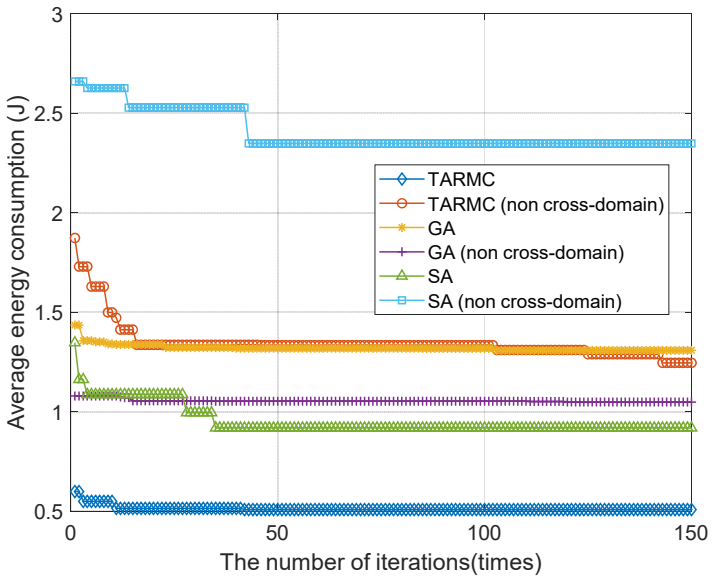

Figure 3. Average energy consumption of industrial intelligent terminals varies with resource allocation strategies.

The variations in delay of processing various tasks in multiple stages with several data sizes are plotted in Figure 4, wherein the horizontal axis represents the average size of the task data volume, and the vertical axis represents the delay. In the bar graph, the delay in task processing in various processes is stated as follows (from left to right): closed-loop processing, multidomain collaborative computing process, and local computing process. As observed from the figure, compared to the delay in the first two processes, the task exhibits the lowest processing delay during the local calculation process, and it gradually increases with the amount of task data, before eventually its convergence. This is because the local limited resources fail to execute the task of higher data volume. If the amount of data of the pending task attains the maximum limit of local computing processing, it will request multidomain collaborative processing instead of allocating new tasks to local processing. During this process, the delay in processing the task of local calculation gradually increases before stabilizing. Moreover, in the process of the multidomain collaboration task, the

processing delay increases with the amount of data related to the task under process, and accordingly, the delay of the closed-loop process increases. Therefore, the processing delay in the calculation process is reasonably less than that of the other two processes, with a slow growth rate.

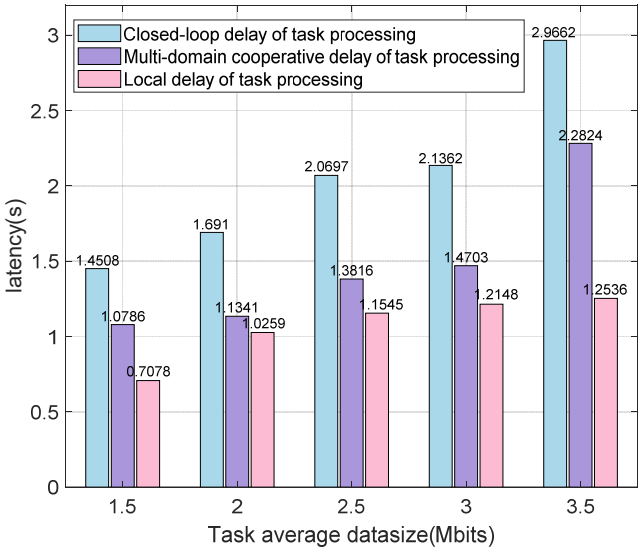

Figure 4. Variations in delay under various task processing stages for multiple task sizes.

This paper considered the energy consumption of industrial intelligent terminals at various stages of several task data sizes. In Figure 5, the horizontal axis represents the average size of task data volume, and the vertical axis represents energy consumption. In the bar chart, the energy consumption of industrial intelligent terminals in the closed-loop process of task processing, energy consumption in the process of multidomain collaborative computing, and energy consumption in the local calculation process are presented. As observed, the industrial intelligent terminal consumes the least energy in the multidomain collaborative computing process compared with that in the previous two processes, and the energy consumption of the industrial intelligent terminal in all processes increased with the task data volume. This is because the energy consumption of the industrial intelligent terminal in the process of multidomain collaborative computing primarily includes industrial terminal local computing after the end of standby energy consumption, transmit-and-accept-task data energy consumption, and local computing energy consumption. In the process of task computing and CPU computing energy consumption, task closed-loop process includes the energy consumption of the above-mentioned two processes. Owing to the extremely short time span of transmitting and receiving data, the process exhibits the lowest energy consumption, which often results in the lowest energy consumption in the domain collaborative computing process. Simultaneously, an increase in the amount of task data increases the time to transmit and receive the data, increases CPU computing time, the increases computing energy consumption, and eventually, increases energy consumption in all the processes.

The variations in delay of processing tasks at various stages for multiple industrial intelligent terminals is presented in Figure 6, where the horizontal axis indicates the number of industrial intelligent terminals, and the vertical axis represents the delay. In the bar graph, from left to right: delays of processing task in closed-loop process, the task during multidomain collaborative computing process, and the processing delay of task during local computing process. As depicted in Figure 6, the delay in processing tasks at various stages increases with the number of terminals. This is because the network task data doubles upon increasing the number of industrial intelligent terminals. In global network resources allocation, considering each task provides maximum delay and the network load should be balanced, the collaborative computing resources are allocated with several

terminals to ensure the normal operation of various intelligent terminals, which increases the network delay.

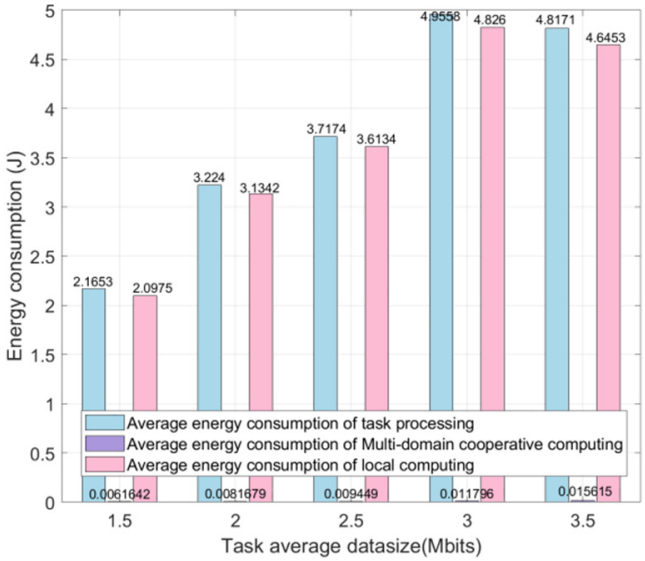

Figure 5. Variations in energy consumption of multiple task processing stages for various task sizes.

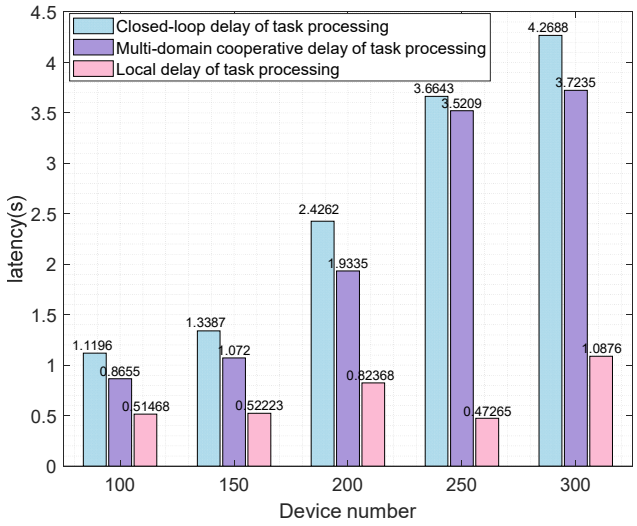

Figure 6. Delay variations in various task processing stages for multiple industrial intelligent terminals.

In this study, the energy consumption varied with the number of industrial intelligent terminals and the task processing was considered in multiple stages. In Figure 7, the horizontal axis represents the number of industrial smart terminals and the vertical axis represents energy consumption. In the bars graph, the energy consumption for various cases is stated as follows (from left to right): industrial intelligent terminal in the closed-loop process of task processing, the process of multidomain collaborative computing, and the process of local computing. As observed from the graph, the energy consumption of industrial intelligent terminals in the closed-loop process of task processing is not easily affected by the variations in the number of intelligent terminals and is minimized if the number of terminals reaches 150. This is because the proposed TARMC strategy can categorize various tasks into several subtasks with an extremely small volume of data. As the number of intelligent terminals increases, the amount of task data in the global network increases. However, the data amount of the segmented subtasks for multidomain collaborative computing remains negligibly small, and the impact of subtask data growth can be minimized by flexibly scheduling the global network computing resources. At this

instant, the delays in processing and data of the industrial terminal transmission process are low, and the corresponding energy consumption is not easily affected by the growth of the number of intelligent terminals. In addition, as the improved Cuckoo Search algorithm of this strategy obtained a global approximate optimal solution through continuous iteration, the energy consumption fluctuated as the number of intelligent terminals increased, and it was minimized as the number of terminals reached 150.

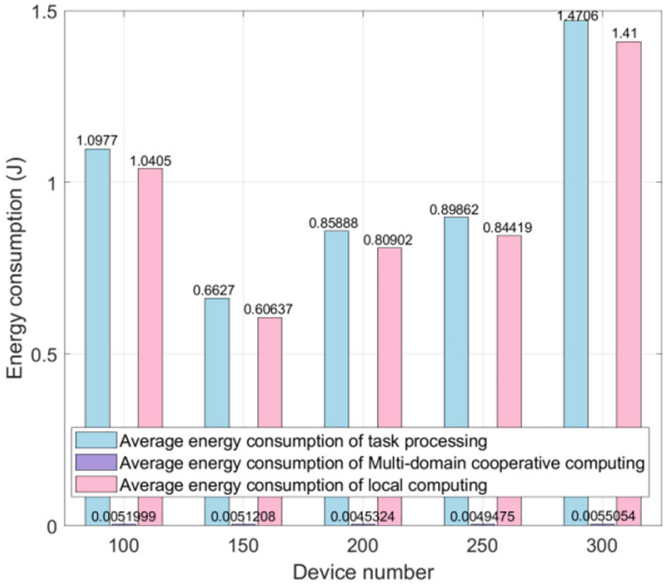

Figure 7. Variations in energy consumption across multiple task processing stages under various industrial intelligent terminals.

6. Conclusions

This study proposed a task-offloading and resource allocation strategy in multidomain cooperation for the IIoT. First, this strategy deeply examines the closed-loop process of information flow interaction between various layers of intelligent terminals in the wireless network, constructs a multidomain collaborative task-offloading and resource allocation network model for the IIoT, and efficiently allocates the resources between intelligent terminals, edge servers, and cluster domain networks according to the dynamic changes of the network load. Subsequently, various tasks are segmented and identified, enabling local and edge servers to process all subtasks in parallel. Simultaneously, the joint task-processing closed-loop delay and terminal energy consumption utility function of the intelligent terminal are developed around the machine, transforming the multidomain collaborative task-offloading and resource allocation process into the problem of task calculation revenue. Moreover, a modified Cuckoo Search algorithm was developed through the iterative alternating solution, which calculated the optimal offloading location and resource allocation decisions. The simulation results revealed that the TARMC strategy effectively improved the closed-loop delay and energy consumption of task processing compared with the GA- and SA-based resource allocation strategies. Furthermore, it verified that the delay and energy consumption optimization performance of multidomain collaborative methods is much higher than that of non-cross-domain methods. In future, we will continue to develop flexible manufacturing scenarios of the wireless-network resource-scheduling scheme, considering the workshop-level massive heterogeneous data information fusion method. In addition, the coupling correlation between the variations in physical environment and digital space should be further explored to improve the efficiency of multidimensional resource scheduling, enhance the overall real-time data interaction from factory production lines (businesses, control instructions, etc.), and increase production decision-making efficiency.

Author Contributions: Conceptualization, Z.D. and Y.Z.; methodology, Z.D. and Y.Z.; formal analysis, N.M. and H.T.; writing—original draft preparation, Z.D. and Y.Z.; writing—review and editing, Z.D. and Y.Z.; funding acquisition, N.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Wu, Y.; Dai, H.-N.; Wang, H. Convergence of blockchain and edge computing for secure and scalable IIoT critical infrastructures in industry 4.0. *IEEE Int. Things J.* **2020**, *8*, 2300–2317. [CrossRef]
2. Chen, Q.; Xu, X.; You, Z.; Jiang, H.; Zhang, J.; Wang, F. Communication-Efficient Federated Edge Learning for NR-U based IIoT Networks. *IEEE Int. Things J.* **2021**. [CrossRef]
3. Jiang, T.; Zhang, J.; Tang, P.; Tian, L.; Zheng, Y.; Dou, J.; Asplund, H.; Raschkowski, L.; D'Errico, R.; Jamsa, T. 3GPP standardized 5G channel model for IIoT scenarios: A survey. *IEEE IEEE Int. Things J.* **2021**, *8*, 8799–8815. [CrossRef]
4. Goswami, P.; Mukherjee, A.; Maiti, M.; Tyagi, S.K.S.; Yang, L. A neural-network-based optimal resource allocation method for secure IIoT network. *IEEE Int. Things J.* **2021**, *9*, 2538–2544. [CrossRef]
5. Guo, B.; Liu, J.; Liu, S.; Wang, J.; Li, M.; Wang, C.; Yu, Z. CrowdIM: Crowd-Inspired Intelligent Manufacturing Space Design. *IEEE Int. Things J.* **2022**, *9*, 19387–19397. [CrossRef]
6. Chen, N.; Qiu, T.; Zhao, L.; Zhou, X.; Ning, H. Edge intelligent networking optimization for Internet of things in smart city. *IEEE Wireless Commun.* **2021**, *28*, 26–31. [CrossRef]
7. Wu, D.; Xu, Z.; Chen, B.; Zhang, Y.; Han, Z. Enforcing access control in information-centric edge networking. *IEEE Trans. Commun.* **2020**, *69*, 353–364. [CrossRef]
8. Ali, H.; Tariq, U.U.; Hussain, M.; Lu, L.; Panneerselvam, J.; Zhai, X. ARSH-FATI: A novel metaheuristic for cluster head selection in wireless sensor networks. *IEEE Syst. J.* **2020**, *15*, 2386–2397. [CrossRef]
9. Singh, J.; Yadav, S.S.; Kanungo, V.; Pal, V.; Yogita, Y. A node overhaul scheme for energy efficient clustering in wireless sensor networks. *IEEE Sens. Lett.* **2021**, *5*, 1–4. [CrossRef]
10. Rahman, G.M.; Wahid, K.A. LDCA: Lightweight dynamic clustering algorithm for IoTconnected wide-area WSN and mobile data sink using LoRa. *IEEE Int. Things J.* **2021**, *9*, 1313–1325. [CrossRef]
11. Al Khafaf, N.; Jalili, M.; Sokolowski, P. A novel clustering index to find optimal clusters size with application to segmentation of energy consumers. *IEEE Trans. Ind. Inf.* **2020**, *17*, 346–355. [CrossRef]
12. Abbas, N.; Zhang, Y.; Taherkordi, A.; Skeie, T. Mobile edge computing: A survey. *IEEE Int. Things J.* **2017**, *5*, 450–465. [CrossRef]
13. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A survey on mobile edge computing: The communication perspective. *IEEE Commun. Surv. Tutorials.* **2017**, *19*, 2322–2358. [CrossRef]
14. Liao, Y.; Shou, L.; Yu, Q.; Ai, Q.; Liu, Q. An intelligent computation demand response framework for IIoT-MEC interactive networks. *IEEE Networking Lett.* **2020**, *2*, 154–158. [CrossRef]
15. Liang, W.; Zhang, J.; Shi, H.; Wang, K.; Wang, Q.; Zheng, M.; Yu, H. An experimental evaluation of WIA-FA and IEEE 802.11 networks for discrete manufacturing. *IEEE Trans. Ind. Inf.* **2021**, *17*, 6260–6271. [CrossRef]
16. Liu, Y.; Peng, M.; Shou, G.; Chen, Y.; Chen, S. Toward edge intelligence: Multiaccess edge computing for 5G and Internet of Things. *IEEE Int. Things J.* **2020**, *7*, 6722–6747. [CrossRef]
17. Zhou, F.; Hu, R.Q.; Li, Z.; Wang, Y. Mobile edge computing in unmanned aerial vehicle networks. *IEEE Wireless Commun.* **2020**, *27*, 140–146. [CrossRef]
18. Yang, G.; Hou, L.; He, X.; He, D.; Chan, S.; Guizani, M. Offloading time optimization via Markov decision process in mobile-edge computing. *IEEE Int. Things J.* **2020**, *8*, 2483–2493. [CrossRef]
19. Li, Y.; Wu, Y.; Dai, M.; Lin, B.; Jia, W.; Shen, X. Hybrid NOMA-FDMA Assisted Dual Computation Offloading: A Latency Minimization Approach. *IEEE Trans. Network Sci. Eng.* **2022**, *9*, 3345–3360. [CrossRef]
20. Zhu, B.; Chi, K.; Liu, J.; Yu, K.; Mumtaz, S. Efficient Offloading for Minimizing Task Computation Delay of NOMA-Based Multiaccess Edge Computing. *IEEE Trans Commun.* **2022**, *70*, 3186–3203. [CrossRef]
21. Luo, Q.; Li, C.; Luan, T.H.; Shi, W.; Wu, W. Self-learning based computation offloading for internet of vehicles: Model and algorithm. *IEEE Trans. Wireless Commun.* **2021**, *20*, 5913–5925. [CrossRef]
22. Huang, J.; Wang, M.; Wu, Y.; Chen, Y.; Shen, X. Distributed Offloading in Overlapping Areas of Mobile Edge Computing for Internet of Things. *IEEE Int. Things J.* **2022**, *9*, 13837–13847. [CrossRef]
23. Gao, M.; Shen, R.; Li, J.; Yan, S.; Li, Y.; Shi, J.; Han, Z.; Zhuo, L. Computation offloading with instantaneous load billing for mobile edge computing. *IEEE Trans. Serv. Comput.* **2022**, *15*, 1473–1485. [CrossRef]

24. Liao, Z.; Ma, Y.; Huang, J.; Wang, J.; Wang, J. HOTSPOT: A UAV-assisted dynamic mobilityaware offloading for mobile-edge computing in 3-D space. *IEEE Int. Things J.* **2021**, *8*, 10940–10952. [CrossRef]
25. Fang, T.; Chen, J.; Zhang, Y. Content-Aware Multi-Subtask Offloading: A Coalition Formation Game-Theoretic Approach. *IEEE Commun. Lett.* **2021**, *25*, 2664–2668. [CrossRef]
26. Wu, M.; Qi, W.; Park, J.; Lin, P.; Guo, L.; Lee, I. Residual Energy Maximization for Wireless Powered Mobile Edge Computing Systems With Mixed-Offloading. *IEEE Trans. Veh. Technol.* **2022**, *71*, 4523–4528. [CrossRef]
27. Chen, Y.; Zhang, N.; Zhang, Y.; Chen, X.; Wu, W.; Shen, X. Energy efficient dynamic offloading in mobile edge computing for internet of things. *IEEE Trans. Cloud Comput.* **2021**, *9*, 1050–1060. [CrossRef]
28. Bozorgchenani, A.; Mashhadi, F.; Tarchi, D.; Monroy, S. Multi-Objective Computation Sharing in Energy and Delay Constrained Mobile Edge Computing Environments. *IEEE Trans. Mob. Comput.* **2021**, *20*, 2992–3005. [CrossRef]
29. Zhang, X.; Zhang, X.; Yang, W. Joint Offloading and Resource Allocation Using Deep Reinforcement Learning in Mobile Edge Computing. *IEEE Trans. Network Sci. Eng.* **2022**, *9*, 3454–3466. [CrossRef]
30. Lu, J.; Li, Q.; Guo, B.; Li, J.; Shen, Y.; Li, G.; Su, H. A Multi-task Oriented Framework for Mobile Computation Offloading. *IEEE Trans. Cloud Comput.* **2022**, *9*, 3454–3466. [CrossRef]
31. Wang, P.; Li, K.; Xiao, B.; Li, K. Multi-objective optimization for joint task offloading, power assignment, and resource allocation in mobile edge computing. *IEEE Int. Things J.* **2021**, *9*, 11737–11748. [CrossRef]
32. Guan, S.; Boukerche, A. A novel mobility-aware offloading management scheme in sustainable multi-access edge computing. *IEEE Trans. Sustainable Comput.* **2021**, *7*, 1–13. [CrossRef]
33. Chen, M.; Guo, S.; Liu, K.; Liao, X.; Xiao, B. Robust computation offloading and resource scheduling in cloudlet-based mobile cloud computing. *IEEE Trans. Mob. Comput.* **2020**, *20*, 2025–2040. [CrossRef]
34. Fang, T.; Yuan, F.; Ao, L.; Chen, J. Joint Task Offloading, D2D Pairing, and Resource Allocation in Device-Enhanced MEC: A Potential Game Approach. *IEEE Int. Things J.* **2021**, *9*, 3226–3237. [CrossRef]
35. Liu, X.; Zheng, J.; Zhang, M.; Li, Y.; Wang, R.; He, Y. A novel D2D–MEC method for enhanced computation capability in cellular networks. *Sci. Rep.* **2021**, *11*, 16918. [CrossRef]
36. Yang, X.S.; Deb, S. Cuckoo Search via Levey Flights. In *World Congress on Nature and Biologically Inspired Computing*; DEC 09-12: Coimbatore, India, 2009.
37. Aburukba, R.O.; AliKarrar, M.; Landolsi, T.; El-Fakih, K. Scheduling Internet of Things requests to minimize latency in hybrid Fog-Cloud computing. *Future Gener. Comput. Syst.* **2020**, *111*, 539–551. [CrossRef]
38. Liu, W.; Huang, G.; Zheng, A.; Liu, J. Research on the optimization of IIoT data processing latency. *Comput. Commun.* **2020**, *151*, 290–298. [CrossRef]

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.