Research Article

Multiple Local-Edge-Cloud Collaboration Strategies in Industrial Internet of Things: A Hybrid Genetic-Based Approach

Xiaokai Liu, 1 Fangmin Xu, 2 Ye Xiao, 2 Xiaoming Zhou, 3 Zhao Li, 3 Chenglin Zhao, 2 and Min Zhang 4

1 School of Mechanical and Electrical Engineering, Beijing Information Science and Technology University, Beijing, China
2 School of Information and Communication Engineering, Beijing University of Posts and Telecommunications (BUPT), Beijing, China
3 State Grid Liaoning Electronic Power Co., Ltd, Beijing, China
4 School of Electronic Engineering, Beijing University of Posts and Telecommunications (BUPT), Beijing, China

Correspondence should be addressed to Fangmin Xu; xufm@bupt.edu.cn

Received 1 February 2022; Revised 5 May 2022; Accepted 7 September 2022; Published 24 September 2022

Academic Editor: B. Rajanarayan Prusty

Copyright © 2022 Xiaokai Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

To cope with the challenge of successful edge offloading brought by the mobility of mobile devices in intelligent factories, this paper studies the optimization problem of the edge offloading strategy of mobile devices based on mobility. Considering the decision task flow executed by priority, the unique offloading mode of a single task, the communication range of the edge server, and the delay constraint of the offloading of a single task, appropriate computing resources are selected according to the real-time location of the mobile device to offload the computing task. Based on the edge computing architecture of an intelligent factory, this paper puts forward five different computation offloading methods. From a global perspective, the energy consumption and delay of tasks offloading in local, edge, cloud center, local-edge collaboration, and local-edge-cloud collaboration are considered. In this paper, the algorithm based on the genetic algorithm and particle swarm optimization is used to design and obtain the decision task offloading strategy with the lowest energy consumption and delay. Simulation results show that the proposed algorithm can reduce the computation offloading energy consumption and delay of mobile devices.

1. Introduction

As a new computing model in the 21st century, edge computing has occupied the research hotspot of academia and industry [1]. By offloading computation-intensive tasks to nearby edge servers, edge computing can provide low-latency services for the automation of intelligent factories and assist intelligent factories to realize intelligent management and manufacturing. By combining the original industrial network with the industrial Internet based on the edge computing architecture, the intelligent factory responds to the call of intelligent manufacturing and accelerates the process of enterprise intelligence [2]. However, the limited computing and communication edge resources become the main obstacle to improve the efficiency of edge computing paradigm. Thus, more and more research studies focus on the computational offloading technique to rationally schedule edge resources for improving the edge computing efficiency.

Computational offloading is a particularly important technology in edge computing that could effectively improve resource utilization efficiency in edge computing. However, the mobility of devices threatens the continuity of communication access and offloading services. Therefore, many scholars have studied the mobility management of terminal devices in practical application scenarios. In the medical wisdom scenario, Qi Ping [3] studied the optimal moving path and task scheduling scheme of the intelligent medical bed. For a given starting point and endpoint of movement, the paper constructed the optimal path selection model based on energy consumption and the task unloading decision model based on the optimal path of workflow under
the constraints of calculating the offloading time delay. After receiving the computing task request and basic information of the intelligent wearable device, the mobile net Internet of Vehicles (IoV) scenario, Xiang and Liu [4] aimed at the mobile edge computing offload system of IoV. They studied the optimal computing offload strategy of roadside units in the background of IoV when a load of the mobile edge computing server is unbalanced. Work operators select multiple edge sites for each intelligent wearable device to offload by implementing the cross-edge computing offloading algorithm based on Lyapunov optimization technology [5].

In the above study, the researchers only considered offloading computing tasks to local, edge, and cloud-centric computing resources for processing. Inspired by them, this paper studies the optimization of the offloading decision of mobile devices in the process of moving computing tasks in intelligent factories to minimize the energy consumption and delay of the offloading decision task flow of mobile devices. In this paper, we consider the combination of multiple computing resources, which can effectively reduce the energy consumption and time delay of computing tasks. Within the maximum communication distance, under the constraint of each task only choosing a discharge mode and the biggest offloading time delay of a single task, a hybrid algorithm based on the genetic algorithm and particle swarm optimization is adopted. This method jointly optimizes the energy consumption and time delay of the decision task flow in the intelligent factory, that is, the optimization of the joint cost of computation offloading.

The main contributions of this paper are listed as follows.

(i) A local-edge-cloud collaboration industrial Internet architecture is presented which considers not only the multiple resource characteristics but also tasks moving characteristics

(ii) The collaborative local-edge-cloud multiple resources allocation problem is constructed to minimize the energy consumption and time delay of computing tasks

(iii) A hybrid algorithm based on the genetic algorithm and particle swarm optimization is adopted to jointly optimize the energy consumption and time delay of the decision task flow in the intelligent factory

The rest of this paper is organized as follows. Section 2 introduces the related works, Section 3 introduces the system model and problem formulation, Section 4 presents the proposed algorithm, Section 5 shows and discusses the numerical results, and finally, conclusions are given in Section 6.

2. Related Work

In this section, related works about edge-cloud collaboration and mobility computing offloading are studied. Usually, content caching and content delivery are studied separately.

2.1. Edge-Cloud Collaboration Computing in the Industrial Internet. The industrial Internet links all types of industrial devices which collect, transmit, and analyze production data. Meanwhile, the industrial Internet optimizes production processes to reduce cost and improve efficiency in smart factories. The introduction of edge computing in industrial Internet can significantly reduce the decision-making latency, save bandwidth resources, and protect privacy [6–8]. Chen modeled the dynamic resource management problem of joint power control and computing resource allocation for MEC in the industrial Internet as the Markov decision process and use a deep reinforcement learning algorithm to solve the problem [9]. Multiple quality of service (QoS) and quality of experience (QoE) scenarios in edge computing resource allocation are considered in [10, 11]. Multiaccess and multitasks scenarios are considered in [12].

Edge computing paradigm is suitable for processing and analyzing local, real-time data [13, 14]. However, edge servers are not suitable for large-scale and long-term data processing and analyzing [15, 16, 17]. Therefore, more and more research studies focus on edge-cloud collaboration technology, and the complementary characteristics of edge-cloud resources are utilized to realize efficient large-scale task processing and analysis [18, 19]. To be specific, the combination of edge and cloud computing has been extensively studied [20], including architecture design [21], energy management [22, 23], and resource optimization [24]. Considering the heterogeneous and finite characteristics of edge servers, the author designs a device-edge-cloud collaborative computing offload method for mobile devices [25]. In the Internet of Vehicles environment, the load of edge computing devices is not an ideal stable load state most of the time and may be overloaded or underloaded. Xiaolong et al. [26] designed a vehicle communication route acquisition algorithm based on load balancing and joint optimization of delay and energy consumption. Besides, there are some gaps in the research on the implementation cost of operators and the fluency of user service experience in the research on the Internet of Vehicles. Therefore, researchers in [27] aimed to jointly realize a low-cost and low blocking probability vehicle cloud system resource allocation scheme from the perspectives of operators and users.

To date, most works focused on optimization of energy consumption, quality of service, and so on. However, it lacks multiresource offloading decision optimization in practical application scenarios. This work aims at the optimization of computing task offloading decision-making for mobile devices in intelligent factories.

2.2. Mobility Management in Industrial Internet Computing Offloading. In real industrial Internet, the location of terminal equipment such as automated guided vehicle (AGV) is time-varying [28, 29]. The mobility of terminal equipment affects the continuity of communication access and computing task offloading services. There are three main research methods for user mobility management: route prediction, power control, and virtual machine migration [30]. The mobility prediction-based mobile edge computing
A resource management method is proposed in [31, 32]. The proposed approach uses the Kalman filter to predict the user’s mobile location, and this method can ensure that the user can select the edge server with the most stable current state during task request and task collection. When the user chooses to offload the application to the mobile edge computing server for execution, the base station dynamically adjusts the reception power and transmission power. When the user needs to access another base station due to location change, the virtual machine on the computing node of the edge server needs to be migrated to establish a new application service request path. Mach and Becvar proposed a power control algorithm which aims to ensure that the application service request of the end mobile user can be returned within a given time limit after being processed by the small cell cloud [33]. Ksentini et al. proposed an online virtual machine migration method to realize mobile user service migration when a mobile user switches from the communication range of one base station or edge server to the communication range of another base station or edge server [34].

In the above research, the researchers only considered offloading computing tasks to local, edge, and cloud-centric computing resources for processing. Inspired by them, the optimization of the offloading decision of mobile devices in the process of moving computing tasks in intelligent factories needs to be further studied to minimize the energy consumption and delay of the offloading decision task flow of mobile devices.

### 3. System Model and Problem Formulation

#### 3.1. System Model

The edge computing architecture based on the intelligent factory in this paper is shown in Figure 1. It is mainly composed of the cloud center, that is, the centralized unit(CU), the distributed unit(DU), the radio unit(RU), and the sensing equipment from the top to the bottom. Each RU device is connected to an edge computing server. The sensing equipment is designed to realize the sensing of the volume and weight of the goods in the factory, the recognition of the QR code information, and electricity information transmission of the AGV.

AGV is an important mobile carrier to complete intelligent sorting and delivery in an intelligent factory. The machine makes self-judgment on the transportation route of goods and whether to supplement its electricity. To save the energy consumption of the AGV and speed up the working efficiency of the AGV, it is particularly important to select the appropriate computing resources for the AGV due to a large number of decision-making tasks to complete the commodity dispatch. Figure 2 shows the AGV work simplification plane diagram with grid processing. RU-MEC server in the figure represents the RU device connected to the edge server. Taking a single AGV as an example, after receiving the parcel to be sorted on the left side of Figure 2, the AGV delivers the parcel to the 9 sorting ports of A–I. The dotted line represents the movement path of the AGV.

The decision task flow of AGV l often consists of l tasks with priority. Assuming the AGV decision task in each decision task flow is detachable, it can be represented with a binary group $Z: Z = \{ z_{ij} | z_{ij} = (C_{ij}, D_{ij}), \quad i \in A, l \in X \}$, in which $C_{ij}$ and $D_{ij}$ represent the task computation and task data. $A$ represents the task set and $X$ represents the AGV set. AGV sorting path can also be defined as a binary group: $H = \{ p_{ij} | p_{ij} = (m_{ij}, n_{ij}), j \in B, l \in X \}$, where $p_{ij}$ represents the position information of the $j$ grid in the sorting path of AGV $l$ and $B$ represents the AGV path set. Assume that there are $k$ RU devices in the AGV sorting site, and the set of RU devices is $E = \{ e_1, e_2, \ldots, e_k \}$, with reference to the communication model between road side unit and the vehicle [35]. The instantaneous data transmission rate of uplink between the AGV and RU equipment is $R^{RU} = B_{\text{uplink}} \log_2 (1 + P_{\text{up}}^l d_{l}^{RU})^2/\delta$, in which $B_{\text{uplink}}$ represents the uplink bandwidth allocated by RU to AGV $l$, $P_{\text{up}}^l$ is the transmitting power of AGV $l$, $d_{l}^{RU}$ refers to the path loss between the $l$th AGV and the RU device, $\chi$ and $l$ represent the path loss factor and channel fading factor.
between AGV $l$ and Ru, respectively, and parameter $\delta$ refers to white noise power level. Similarly, the instantaneous data transmission rate of the downlink between the AGV $l$ and RU equipment is $R_{l}^{\text{down}} = B_{\text{downlink}} \log_2 (1 + P_{l}^{\text{down}} d_{l}^{\text{RU}}/\delta)$. Since the instantaneous data transmission rate between the AGV $l$ and RU devices is closely related to distance $d_{l}^{\text{RU}}$, this paper assumes that when there is no edge computing unloading migration of the AGV in the same grid, its instantaneous data transmission rate in the same grid is constant. Set the rate of the AGV as $\epsilon_l$, and the time it takes for AGV to transmit task in uplink is related to the grid side length $L$, and the following inequality is satisfied.

$$\sum_{j=1}^{\epsilon_l} R_{l}^{\text{RU}} L \geq D_{l,i},$$  \hspace{1cm} (1)$$

where $\epsilon_l$ represents the number of grids that need to be moved through in order to successfully send task $i$ of AGV $l$; then, the meaning of this inequality is as follows: the sign that task $i$ of AGV $l$ sends successfully is that the sum of the data amount sent in $\epsilon_l$ grids is greater than or equal to the task data amount $D_{l,i}$. Similarly, the time of downlink needs to meet the following inequality:

$$\sum_{j=1}^{\epsilon_l} R_{l}^{\text{down}} L \geq \phi D_{l,i},$$  \hspace{1cm} (2)$$

Moreover, the computed task data amount is set to be $\phi$ times of the task data amount.

The tasks in this article can be offloaded to three locations: an AGV with computing ability, an edge server connected to an RU device, and a cloud-centric device with computing ability. To optimize the offloading energy consumption and delay of the AGV decision task flow, this paper provides five computing task offloading methods for each AGV decision task flow: local offloading, RU edge offloading, cloud center offloading, local-edge cooperative offloading, and local-edge-cloud cooperative offloading which is shown in Figure 3. The following is a detailed explanation of five computation offloading method for task $i$ of AGV $l$:

3.1.1. Local Offloading. Assuming that the computing capacity of the AGV is $f_{AGV}$ and the power of decision task execution is $P_{AGV}$, the energy consumption and delay of local computing offload on AGV are as follows:

$$E_{AGV}^{i,l} = P_{AGV} T_{AGV}^{i,l},$$

$$T_{AGV}^{i,l} = C_{l,D_{l,i}} f_{AGV}.$$

3.1.2. RU Edge Offloading. If the task $z_{i,l}$ is offloaded to the edge server, it involves the offloading of the task data and the return of the computed task data, which should be considered in three cases. Note that, since the route of the AGV is fixed, the case could be determined by the current position and speed of moving the AGV.

(1) The task is not offloaded successfully. When the AGV deviates from the communication range of the optimal edge server in the process of sending or receiving task data, the task offloading to the edge
server cannot be carried out, and the local offloading mode needs to be selected.

(2) The computed task data cannot be transmitted back. If the edge server finds that the AGV is no longer in its communication range when sending back the computed task data, it needs to migrate the computed task data to the edge server nearest to the AGV to complete the data transmission. Assuming that the data transmission rate between edge servers is $R'$, then the total energy consumption and delay of the \( i \)-th task of the \( i \)-th AGV for edge offloading are as follows:

\[
\begin{align*}
E_{i,RU}^i &= P_{i,AGV,up}^i + P_{\text{agg}}^i \tau_{\text{comp}}^i + P_{\text{agg}}^i \tau_{\text{mig}}^i + P_{i,\text{AGV,down}}^i \tau_{\text{comp}}^i + E_{\text{AGV,down}}^i, \\
\tau_{\text{comp}}^i &= \frac{D_{i,j}}{R_{i}} + \frac{C_{i,j}D_{i,j}}{f_{\text{RU}}} + \frac{\phi D_{i,j}}{R_{i}^{\text{RU}}}.
\end{align*}
\]

Where $E_{i,RU}^i$ and $E_{\text{AGV,down}}^i$, respectively, represent the data transmission delay and energy consumption, $P_{\text{agg}}^i$, $\tau_{\text{comp}}^i$, and $f_{\text{RU}}$, respectively, represent the power, delay, and energy consumption of waiting for edge server to process computing task. Besides, $f_{\text{RU}}$ represents the computing ability of the edge server. Furthermore, $E_{i,\text{AGV,down}}^i$, and $\tau_{\text{comp}}^i$, respectively, represent the delay and energy consumption of data migration of the AGV waiting for data migration to complete.

3.1.3. Cloud Center Offloading. As can be seen from Figure 1, if the decision-making task of the AGV is offloaded to the cloud center for computing, it needs to complete the communication with the cloud center through the RU equipment. If the AGV is far away from the coverage range of the RU device in the process of data return, it can be migrated to the RU device that can communicate with the AGV to ensure the successful completion of task offloading. Assume that the data transmission rate between the RU equipment and the cloud center is $f_{\text{CU}}$, and the computing capacity of tasks executed by the cloud center is $R'_{\text{CU}}$. In this case, the total energy consumption and delay of task offloading are as follows:

\[
\begin{align*}
E_{i,\text{CU}}^i &= (P_{i,AGV,up}^i + P_{\text{agg}}^i \tau_{\text{RU,up}}^i) + P_{\text{agg}}^i \tau_{\text{CU,comp}}^i + P_{\text{agg}}^i \tau_{\text{mig}}^i + (P_{i,\text{AGV,down}}^i + P_{\text{agg}}^i \tau_{\text{CU,down}}^i) \\
\tau_{\text{comp}}^i &= \frac{D_{i,j}}{R_{i}} + \frac{D_{i,j}}{R_{\text{ru-cu}}} + \frac{C_{i,j}D_{i,j}}{f_{\text{CU}}} + \frac{\phi D_{i,j}}{R_{i}^{\text{CU}}} + \frac{\phi C_{i,j}D_{i,j}}{R_{\text{ru-cu}}} + \frac{\phi D_{i,j}}{R_{i}^{\text{RU}}}.
\end{align*}
\]

(3) The computing task is offloaded completely. In this case, the computing task can complete the offloading of the computing task on the same edge server. Therefore, the total energy consumption and delay of the AGV for edge offloading are as follows:
where $T_{RU_{up}}^{ij}$ represents the delay when task is transmitted from the RU device to the cloud center and $T_{RU_{down}}^{ij}$ represents the delay when the task data are returned from the cloud center to the RU device; $E_{CU_{comp}}^{ij}$ and $E_{RU_{comp}}^{ij}$, respectively, represent the delay and energy consumption of waiting for task to be calculated in the cloud center.

3.1.4. Local-Edge Cooperative Offloading. When computing resources of edge servers on local and RU devices are considered to coprocess a decision task, the edge offloading time of this task should be the larger value of the offloading delay of the two tasks split equally according to the amount of data, and the edge offloading energy consumption of this task is the sum of the energy consumption of the two tasks.

$$
\begin{align*}
T_{A-R}^{ij} &= \max(T_{AGV}^{ij}, T_{RU_2}^{ij}). \\
E_{A-R}^{ij} &= E_{AGV}^{ij} + E_{RU_2}^{ij},
\end{align*}
$$

(7)

3.1.5. Local-Edge-Cloud Collaborative Offloading. When local, edge, and cloud center computing resources are considered to coprocess a decision task, the edge offloading time of the task should be the larger value of the offloading delay of the three tasks divided equally according to the amount of data, and the edge offloading energy consumption of the task is the sum of the energy consumption of the three tasks.

$$
\begin{align*}
T_{A-R-C}^{ij} &= \max(T_{AGV}^{ij}, T_{RU_2}^{ij}, T_{CU_3}^{ij}), \\
E_{A-R-C}^{ij} &= E_{AGV}^{ij} + E_{RU_2}^{ij} + E_{CU_3}^{ij},
\end{align*}
$$

(8)

When multiple AGVs offload computing tasks at the same time, if multiple computing tasks are offloaded on the same computing resource, multiple computing tasks will divide the computing resource equally.

3.2. Problem Formulation. Considering the collaborative processing of decision-making tasks by computing resource, this paper provides five computing task offloading methods for each AGV: local offloading, RU edge offloading, cloud center offloading, local-edge collaborative offloading, and local-edge-cloud collaborative offloading, corresponding to decision variables $x_{AGV}^{ij}$, $x_{RU}^{ij}$, $x_{CU}^{ij}$, $x_{A-R}^{ij}$, and $x_{A-R-C}^{ij}$, respectively. For the latter two compute offloading modes, computing tasks can be divided into two independent data with the same size and three independent data with the same size.

The objective function and constraints of the joint optimization problem of edge offloading energy consumption and delay in this paper are given. Among them, constraint C1 ensures that each task in the decision task flow of the AGV can only choose one offloading mode for task offloading and constraint C2 ensures that variables are 0-1 binary variables. Constraints C3–C7 ensure that each computing task is processed within the execution time $T_{\text{max}}$, and the number of computing tasks executed on the same computing resource needs to be constrained. Besides, $s$ represents the weight coefficient of energy consumption and delay with a value between 0 and 1.

$$
\begin{align*}
\min s \times \left( \sum_i \sum_l \left( x_{AGV}^{ij} E_{AGV}^{ij} + x_{RU}^{ij} E_{RU}^{ij} + x_{CU}^{ij} E_{CU}^{ij} + x_{A-R}^{ij} E_{A-R}^{ij} + x_{A-R-C}^{ij} E_{A-R-C}^{ij} \right) + (1 - s) \times \left( \sum_i \sum_l \left( x_{AGV}^{ij} T_{AGV}^{ij} + x_{RU}^{ij} T_{RU}^{ij} + x_{CU}^{ij} T_{CU}^{ij} + x_{A-R}^{ij} T_{A-R}^{ij} + x_{A-R-C}^{ij} T_{A-R-C}^{ij} \right) \right) \right),
\end{align*}
$$

(9)

s.t.C1: $x_{AGV}^{ij} + x_{RU}^{ij} + x_{CU}^{ij} + x_{A-R}^{ij} + x_{A-R-C}^{ij} = 1, \quad i \in A, \quad l \in X,$

C2: $x_{AGV}^{ij}, x_{RU}^{ij}, x_{CU}^{ij}, x_{A-R}^{ij}, x_{A-R-C}^{ij} \in \{0, 1\},$

C3: $T_{AGV}^{ij} \leq T_{\text{max}},$

C4: $T_{RU}^{ij} \leq T_{\text{max}},$

C5: $T_{CU}^{ij} \leq T_{\text{max}},$

C6: $T_{A-R}^{ij} \leq T_{\text{max}},$

C7: $T_{A-R-C}^{ij} \leq T_{\text{max}}.$

(10)

4. Proposed Algorithm

To solve the joint optimization problem of edge offloading energy consumption and delay, this paper combines the iterative searching ability of genetic algorithm crossover and mutation operation and the characteristic of the particle swarm optimization algorithm to retain the results of each iteration of the population [36, 37]. In this paper, a hybrid
algorithm based on the genetic algorithm and particle swarm optimization algorithm is proposed, that is, the edge offloading algorithm for mobile devices based on energy consumption and delay (EOAMDBEC). Specific operations are described in detail as follows.

4.1. Genetic Algorithm Coding and Initialization. According to formula (9), the joint optimization problem of energy consumption and delay in this work consists of five variables, which can be coded as a binary number of five bits, representing \( x_{AGV}^i \), \( x_{RU}^i \), \( x_{CU}^i \), \( x_{AR}^i \), and \( x_{CAR}^i \) from high to low. As shown in Figure 4, the code 01000 of Task 1 represents \( x_{RU}^i = 1 \), indicating that Task 1 is offloaded to the edge server of the RU device for execution. After the coding is completed, the next step is to initialize the crossover probability and mutation probability of the genetic algorithm. Meanwhile, the number of computation tasks is the number of chromosomes.

4.2. Initialization of the Particle Swarm Optimization Algorithm. Since the binary code in operation (1) is five bits,

\[
\text{Fitness}(x) = s \times \left( \sum_{i} \sum_{l} x_{AGV}^i r_{AGV}^i + x_{RU}^i r_{RU}^i + x_{CU}^i r_{CU}^i + x_{AR}^i r_{AR}^i + x_{CAR}^i r_{CAR}^i \right) + (1 - s) \times \left( \sum_{i} \sum_{l} x_{AGV}^i r_{AGV}^i + x_{RU}^i r_{RU}^i + x_{CU}^i r_{CU}^i + x_{AR}^i r_{AR}^i + x_{CAR}^i r_{CAR}^i \right).
\]  

In this case, the initialized particle population needs to be transformed into the corresponding chromosome binary code. It can be seen from constraint CI that only one bit in the five-bit binary encoding of a correct task offloading decision can be valued as 1. Before calculating the fitness value, it is necessary to check whether the initialized binary code meets constraint CI. At the same time, it is necessary to judge whether the number of computing tasks on the same computing resource does not exceed the maximum number of computing tasks. For particles that do not meet one of the above two conditions, its fitness value is infinity. Therefore, the value of the current particle and the corresponding fitness value are saved in Gbest and Gbest-fitness, and then, the particle with the smallest fitness value in the current particle population is obtained by comparison, while its value and fitness value are saved in Pbest and Pbest-fitness.

4.4. Updating of Particle Value and Searching Speed. Update the position value and velocity of the particle according to formula (12), where \( \omega \) is defined as the inertia weight value, \( c_1 \) and \( c_2 \) are the learning factors greater than or equal to 0, \( r_1 \) and \( r_2 \) are the two randomly generated numbers between 0 and 1, and \( p_i \) represents the current position value of particle \( i \).

\[
\begin{cases}
V_i(q + 1) = \omega V_i(q) + c_1 r_1 (Pbest_i - p_i), \\
+c_2 r_2 (Gbest - p_i), \\
p_i(q + 1) = p_i + V_i(q + 1).
\end{cases}
\]  

4.5. Crossover. The value of the current particle population is transformed into the corresponding chromosome binary code, and two chromosomes are selected to complete the crossover operation with the given crossover probability \( p_c \).

4.6. Mutation. Based on the given mutation probability \( p_m \), the value of the randomly generated mutation bit is changed from 1 to 0 or from 0 to 1.

4.7. Terminate Algorithm Iteration. When the maximum number of iterations or the accuracy of the proposed algorithm is satisfied, the algorithm iteration is ended and the particle with the smallest fitness value in the global range is obtained. The solution method is elaborated in Algorithm 1.

5. Numerical Result

In this paper, the simulation is achieved by MATLAB. The scale of the AGV mobile scene is 200 m \( \times \) 200 m. The position of the RU equipment obeys Poisson distribution, and
the side length of square grid $L = 0.5\text{ m}$. After the AGV starts from the left side of the site, the given AGV movement path goes straight along the path parallel to $y = 100$. When it reaches the same position as the AGV’s sorting destination, it goes straight along the path perpendicular to $y = 100$ until it reaches the final sorting port to complete the parcel delivery. The path selected in this paper is the path from the AGV with the starting point below $(0, 100)$ to the sorting port. Figure 5 shows the coordinate position and path schematic diagram of the sorting port, respectively.

The parameter values used in the simulation are set by referring to the previous research [9, 10], and the specific parameter values are shown in Table 1:

First, under the condition of different number of the AGV, this paper will study the influence of different computation offloading methods on the joint cost of computation offloading, computation offloading energy consumption, and delay of the AGV. Meanwhile, the effectiveness of the proposed algorithm is verified. Then, the influence of the weight coefficient, the number of computing tasks, and the number of the RU devices connected to the edge server to the joint cost of computation offloading under different offloading methods is further investigated.

5.1. Algorithm Performance under Different Numbers of AGVs. In the case of different numbers of AGVs, the number of the RU devices is 16. Meanwhile, each AGV needs to process 7 decision tasks, and the weight coefficient $\alpha$ is 0.6. As can be seen from Figure 6, when the number of AGVs is 1, the joint cost of computation offloading with the local offloading method is the largest. With the increasing number of AGVs, the joint cost of computation offloading with each offloading method increases. However, the joint cost of computation offloading obtained by the EOAMDBECD algorithm proposed in this paper is always the minimum. As shown in Figures 7 and 8, when the decision task of the AGV is processed locally, the computation offloading energy consumption is the largest. Nevertheless, the computation offloading delay is very low. When all the computing tasks of the AGV are offloaded to the edge server on the RU device, the computation offloading delay is higher than that with other offloading methods.

5.2. Algorithm Performance under Different Weight Parameters. In this experiment, the number of AGVs is 6, the number of computing tasks of each AGV is 7, and the number
of the RU devices is 16. As can be seen from Figure 8, with the increase of $s$, the joint cost of computation offloading with the local offloading method also increases, while the joint cost of computation offloading with other offloading methods decreases. This is because in the case of joint optimization computation offloading energy consumption and delay, computation offloading energy consumption is greater than delay when the AGV offloads the task to local computation resource. However, energy consumption of computation offloading with the other four offloading methods and the algorithm proposed in this work is less than delay of computation offloading. In addition, the EOAMDBECD algorithm can get the minimum joint cost of computation offloading.

5.3. Algorithm Performance under Different Numbers of Computing Tasks. In this comparative experiment, the weight parameter $s$ is 0.6, the number of RU devices is 16, and the number of AGVs is 6. As can be seen from Figure 9, when $s = 0.6$, the joint cost of computation offloading with the edge offloading method is the highest. Therefore, it can be predicted that in the case of different numbers of computing tasks, the joint cost of computation offloading with each computing offloading method will increase, while the number of tasks increase and joint cost of computation offloading with the edge offloading method will be the
It can be seen from Figure 10 that the variation trend of joint cost of computation offloading is the same as the variation trend of the number of decision tasks. Nevertheless, the joint cost of computation offloading obtained with the edge unloading method and the EOAMDBECD algorithm is the maximum and the minimum, respectively.

5.4. Algorithm Performance under Different Numbers of RU Devices. In this comparative experiment, the weight parameter $s$ is 0.6 and the number of AGVs is 6. Besides, the number of computing tasks for each AGV is 7. As shown in Figure 11, when the number of the RU devices increases, the joint cost of computation offloading of AGVs decreases. It is because while the number of RU devices increases, the distance between the AGV and RU devices becomes smaller, and the data transmission rate of uplink and downlink increases accordingly. In addition, it can be seen from the figure that the joint cost of computation offloading with the edge offloading method is the highest, while the joint cost of computation offloading obtained by the EOAMDBECD algorithm is the lowest.

Through the experiment results analysis above, it can be inferred that the joint cost of computation offloading of the proposed EOAMDBECD algorithm is minimal compared with the local offloading, edge offloading, cloud center offloading, local-edges collaborative offloading, and local-edges-cloud collaborative offloading in the cases of different numbers of the AGV, weight coefficient, the number of computing tasks, and the RU equipment number. Therefore, it can be concluded that the EOAMDBECD algorithm can effectively reduce the joint cost of computation offloading of moving the AGV. In the research of this paper, the moving path of the AGVs is known, but in fact, the moving path of the end user is updated in real time, such as the running trajectory of the vehicle in the Internet of Vehicles and the AGV of the intelligent factory. The bagging sites for delivery of goods are determined based on real-time feedback information. Therefore, there are still points to be further studied in this study.

6. Conclusion

In this paper, a four-tier intelligent factory edge computing architecture is proposed. In order to jointly optimize the energy consumption and delay of the AGV edge offloading, five computing task offloading methods are proposed: local offloading, RU edge offloading, cloud center offloading, local-edge collaborative offloading, and local-edge-cloud collaborative offloading. First, a plan diagram of the AGV sorting delivery path is constructed and the computation offloading of the decision task flow is carried out on the given path. Besides, the problem of energy consumption and delay minimization of decision task flow is modeled as an integer linear programming problem under
multiconstraints. The problem is solved by the hybrid algorithm based on the genetic algorithm and particle swarm optimization algorithm. Simulation results show that the EOAMDBEC algorithm can significantly reduce the joint cost of computation offloading of the decision task flow of the AGV in the process of moving.

For unknown dispatch tasks, the mobility of the AGVs actually needs to be predicted. In future studies, it can be considered to predict the position of the AGV in the next phase. Offloading the decision-making task to the closest edge server to the next time position for task offloading can reduce the time cost of the AGV trial and error and also improve the success rate of task offloading.

Data Availability

The data used to support the findings of this study cannot be shared at this time as the data also form part of an ongoing study.

Disclosure

The articles published in the Science Paper Online are only online preprint (only preliminary review of the articles) not official publication. The authors still retain the copyright and can be invested in other publications. Please refer to the paper [38].

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was funded by the National Natural Science Foundation of China (61971050) and Beijing Natural Science Foundation-Haidian Frontier Project Research on Key Technologies of wireless edge intelligent collaboration for industrial Internet scenarios (L202017).

References


