Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

Reinforcement learning empowered multi-AGV offloading scheduling in edge-cloud IIoT

Peng Liu[1], Zhe Liu[1], Ji Wang[2*], Zifu Wu[3], Peng Li[4] and Huijuan Lu[5]

Abstract

The edge-cloud computing architecture has been introduced to industrial circles to ensure the time constraints for industrial computing tasks. Besides the central cloud, various numbers of edge servers (ESes) are deployed in a distributed manner. In the meantime, most large factories currently use auto guided vehicles (AGVs). They usually travel along a given route and can help offload tasks to ESes. An ES maybe accessed by multiple AGVs, thus incurring offloading and processing delays due to resource competition. In this paper, we investigate the offloading scheduling issue for cyclical tasks and put forth the Multi-AGV Cyclical Offloading Optimization (MCOO) algorithm to reduce conflicts. The solution divides the offloading optimization problem into two parts. Firstly, the load balancing algorithm and greedy algorithm are utilized to find the optimal allocation of tasks for a single AGV under limited conditions. Then, multiple AGVs are asynchronously trained by applying the Reinforcement Learning-based A3C algorithm to optimize the offloading scheme. The simulation results show that the MCOO algorithm improves the global offloading performance both in task volume and adaptability compared with the baseline algorithms.

Keywords: Mobile Edge Computing, Task Offloading Optimization, Reinforcement Learning, Q-learning

Introduction

The modern industrial Internet of things (IIoT) has been integrating more and more tasks. In this situation, efficient task offloading to the cloud server plays a very important role in coping with the problems of insufficient processing capacity and limited resources of smart devices [1, 2]. However, the singular cloud center mode can no longer meet the increasing needs. Multi-clouds [3], such as multiple fog nodes [4] and multiple edge servers (ESes) [5] are applied to distribute data computing tasks closer to data sources and users. This enables the emerging paradigm of edge-cloud IIoT, which performance relies heavily on the support of the network. However, in some highly dynamic manufacturing site, wired industrial ethernet may not be present and wireless networks are also not available due to low reliability. To this end, other approaches to the problem-solving process are therefore necessary.

Auto Guided Vehicles (AGVs) are becoming more and more popular in smart factories [6]. They have abundant digital storage spaces and usually travel along fixed routes throughout the factory. Therefore, they can help offload tasks generated by various manufacturing devices to edge servers/local clouds during drop-in visits. In addition to easing the strain on the central cloud server's transmission and processing capabilities, this can also satisfy the low delay requirements of data tasks. Mobile node-assisted task offloading is proven to be practical in the existing literature, such as satellites [7], vehicles [8], UAVs [9], and vessels [10].

According to the proposed mobile task offloading scenario, producing units generate tasks regularly, and AGVs carry out their daily transporting tasks along cyclical routes. AGVs collect tasks from producing units and

*Correspondence: wangjiok@126.com

[2] Yuxiang Technology (Hangzhou) Co Ltd, Hangzhou, China
Full list of author information is available at the end of the article

offload them to edge servers when they visit them on the way. Then, time can be categorized into rounds and in each round, the traveling patterns of AGVs and the number of tasks remain the same. Overlapped routes and mutual edge servers are common in this case. How to schedule the task offloading plan for individual AGVs as well as multiple AGVs across rounds is the challenge, as doing so will prevent overload at some edge servers and reduce the overall task-offloading delay. The fact is, any AGV at an ES cannot offload more than it can handle. However, to strictly enforce task offloading, an AGV must offload all remaining tasks when it reaches the final edge server before returning to the beginning for the subsequent round, regardless the queueing time at it. Figure 1 illustrates ESes and path segments that make up the routes which AGV_1 and AGV_2 follow as they move along them at a speed of v. Two routes have an overlapped edge server (ES_4) so that multiple AGVs may offload tasks to the same ES and put a heavy burden on it. Therefore, to balance the performance, if the AGVs arriving first offload more tasks, the AGVs behind can only offload less. This may cause the later AGVs to pay a lot more and even lead to Butterfly Effect. Centralized control is a traditional solution, which is, however, costly. If the central control system neither summarizes all information nor are AGVs able to share information in a real-time manner, a pheromone can be used as the intermediate medium of AGVs leaving at the ES. These practical factors need to be considered in the optimization solution.

To increase the effectiveness of global task offloading, we propose the Multi-AGV Cyclical Offloading Optimization (MCOO) algorithm in this paper. The solution first applies load balancing and the greedy algorithm to find the optimal allocation of tasks for a single AGV under given constraints, then, the Reinforcement Learning-based A3C algorithm is adopted to optimize the offloading scheme for multiple AGVs. The main contributions of this paper are summarized as follows:

- This work is based on our previous work which to our best knowledge is the first one to utilize AGVs to help offload computing tasks in the multi-clouds enabled industrial Internet of things. The solution is extended from once-only scheduling to a practical cyclical working model.
- An algorithm called the Multi-AGV Cyclical Offloading Optimization Algorithm (MCOO), which combines game theory and the A3C algorithm, is proposed to solve the resource conflict problem of multiple AGVs in multiple cycles.

The rest of this paper is arranged as follows. In Section 2, relevant work is summarized, and then the system model is described in detail in Section 3. In Section 4, the problem is modeled and the MCOO algorithm is proposed to optimize the offloading performance. A comprehensive test data set is designed on which experiments have been done to evaluate the performance of the algorithm in Section 5. Finally, the research work of this paper is summarized in Section 6.

Related work

Many scholars have proposed different methods to help optimize the strategy of computing offloading and improve the efficiency of the offloading process [11]. The current research

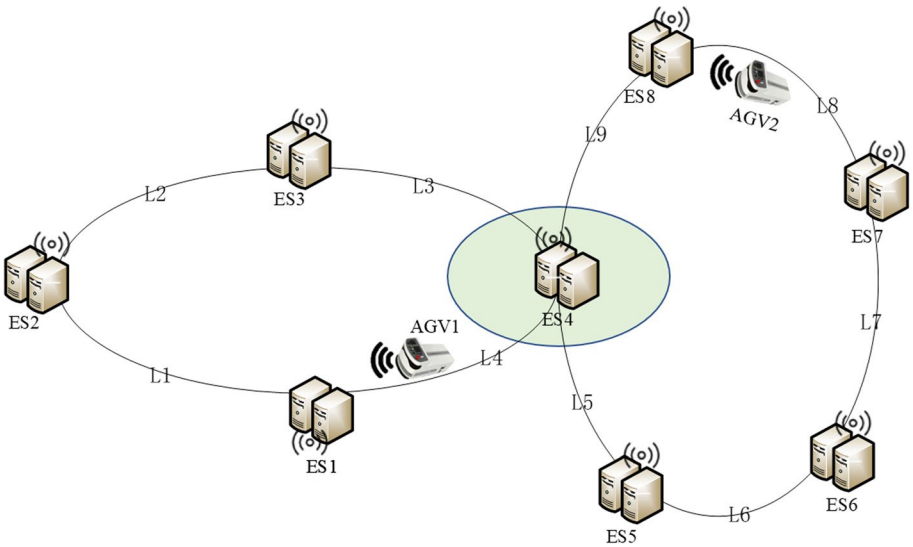

Fig. 1 Conflicts of multi-AGV cyclical task offloading

mostly focuses on the mobile offloading scenario of multi-user and multi-server, in which the objectives are usually to optimize the system delay [12, 13], energy consumption [14–16], and computing efficiency [17, 18]. Besides efficient static schemes [19], more work focus on the use of distributed methods to solve the task offloading scheduling problem.

Because game theory has a good performance in solving multi-objective optimization problems, ref. [20] introduced a multi-objective optimization scheme based on game theory to solve the problem of minimizing cost and improve reliability. Ref. [21] created an incentive mechanism based on game theory to more effectively allocate distributed resources to realize the dynamic allocation of resources to tasks. In general, scheduling requires centralized control to achieve the best overall performance, but it is impractical to force all users to work according to the arrangement of centralized control. Thus, ref. [22] proposed a distributed game-theoretic task scheduling model for edge computing servers. When selecting edge servers, the computing resource allocation of servers is considered, and an acceleration method is proposed to achieve Nash equilibrium faster.

Vehicular Edge Computing (VEC) has been studied recently, Wan et al. proposed the video segmentation algorithm with the support of edge computing in the Internet of vehicles [23]. In [24], the partial task offloading problem in vehicular edge computing in an urban scenario had been studied, where the vehicle computed part of a task locally, and offloaded the remaining task to a nearby vehicle and to VEC server subject to the maximum tolerable delay and vehicle's stay time. Furthermore, a UAV-assisted multi-clouds system was considered in [25]. The objective was to minimize the power consumption of UAVs with the constraint of queue stability, and the problem was further decomposed into three sub-problems using stochastic optimization techniques. Ref. [26] studied the relative delay optimization problem in the MEC assisted UAV group. Considering the great influence of the scheduling method on delay and the coupling relationship between scheduling and resource allocation, the joint optimization of computing offload and channel access was done, and a distributed game-theoretic learning approach was designed.

Nowadays, machine learning has been utilized to solve the complex offloading problem, in which reinforcement learning shows strong adaptability [27]. In [28], the authors integrated two conflicting offloading goals, i.e., maximizing the task-finish ratio with tolerable delay and minimizing the power consumption of devices. They designed a Deep Reinforcement Learning (DRL)-based dynamic task offloading (DDTO) algorithm to achieve the objective. Considering age of information-aware computation offloading, Markovian queueing models were constructed to capture the dynamics of IoT devices and edge servers in [29]. The authors applied DRL techniques for adapting to large-scale dynamic IIoT environments, and designed an intelligent Energy Control and Computation Offloading (ECCO) algorithm. To deal with the problem of insufficient computing resources, ref. [30] and [31] proposed a task offloading and resource allocation scheme based on game theory and reinforcement learning. Ref. [32] proposed a deep random online scheduling algorithm based on actor critical to optimize data transmission scheduling and to minimize the energy of the UAV auxiliary communication network.

Although these studies have used different methods to solve the task offloading scheduling problem in different aspects, none of them can efficiently solve the problem in the scenario of periodic AGV offloading scheduling. Based on the distributed idea, this paper combines the game theory and reinforcement learning to optimize the offloading decision and task allocation scheme of multiple AGVs, which also meets the goal of maximizing the total amount of AGV offloading tasks in a period while improving the utilization of system resources.

System model

Figure 1 shows the cyclical task offloading architecture of multi-AGVs, which is composed of AGVs and edge servers/cloudlets/multi-cloud. Here we regard any of them as edge servers (ESes). This section assumes that M AGVs are operating along their routes and a total of N edge servers out in the field. The set of AGVs is expressed as $M = \{AGV_1, AGV_2, ..., AGV_m, ..., AGV_M\}$, and the set of ESes is expressed as $N = \{ES_1, ES_2, ..., ES_n, ..., ES_N\}$. For the planned route, $L_{i,j}$ is the distance between ES_i and ES_j. The route of each AGV is composed of paths between multiple sections of ESes, and is expressed as $P = \{P_1, P_2, ..., P_K\}$. For ease of reference, Table 1 shows the main symbols and notations used in this paper.

This paper assumes that each AGV carries the same amount of tasks from the start point in each cycle and drives along the planned route at a constant speed v. When passing an ES on the route, it can offload the tasks to the ES for processing. Before the AGV leaves the last ES, full offloading of the completed tasks is required. Then, the AGV returns to the starting point to receive new tasks and start the next round of transportation. Because the computing power and storage capacity of each ES are limited, it should carefully plan how many tasks to offload on which edge server. In the proposed model, $Task_i^m$ represents the number of tasks offloaded by the m-th AGV at the i-th ES. Since the starting point of each AGV is different, the time of encountering each ES will also be different. The tasks offloaded by the first arriving AGV will also be processed first at the ES.

Table 1 Some symbolic variables used in the model

Notation	Description
M, m	Total number of AGVs, $m \in \{1, 2, 3, ..., M\}$
N, n	Total number of ESes, $n \in \{1, 2, 3, ..., N\}$
v	Driving speed of AGVs
$L^m_{total}, L^m_{i,j}$	The distance traveled by the m-th AGV in one cycle and the distance from point i to point j
$Task^m$	Total number of tasks carried by the m-th AGV in one cycle
T_m	Total time spent by the m-th AGV in one cycle including loading/offloading tasks and traveling
t^{on}_m	The time spent by the m-th AGV to load tasks at the starting point
t^{mov}_m	Travel time of the m-th AGV
t^{off}_m	Task offloading time of the m-th AGV
I_m	Starting point of the m-th AGV
D_m	End point of the m-th AGV
$\|D_m - I_m\|$	Number of ESes that the m-th AGV passes from the beginning to the end
C_n	The capacity of n-th ES
E_n	Processing efficiency of n-th ES
$Task^m_i$	Number of tasks offloaded by the m-th AGV at the i-th ES
C^m_i	Available capacity of the i-th ES when the m-th AGV passes through the i-th ES
RT^m_i	Number of tasks allocated on the i-th ES for the remaining tasks of the m-th AGV
RC^m_i	Remaining available resources of the i-th ES in the m-th AGV path

We now explain the task offloading and executing process of an individual AGV. Specifically, the time for the AGV to complete a round of schedule includes three parts: the time t^{on}_m for the AGV to receive the tasks, the moving time t^{mov}_m of the AGV, and the offloading time t^{off}_m of the tasks. The amount of time needed by an AGV to gather tasks at the starting point is t^{on}_m. t^{mov}_m is the length of time needed for an AGV to move from its starting point to its destination and back again after finishing its tasks. The amount of time t^{off}_m is needed for an AGV to send the tasks to the ESes. Since the transmission rate and the number of tasks received and offloaded by AGVs are the same, t^{on}_m is equal to t^{off}_m. For tasks, it is also necessary to calculate the processing time at ESes. This time will also be one of the key factors to judge the utilization rate of ES resources and the quality of the scheduling scheme.

Moving time: AGV_m moves uniformly along the planned route from the starting point I_m at a fixed speed v. $t^{mov}_{m,i}$ represents the moving time of an AGV, that is, the time taken by AGV_m from the starting point to the offloading range of the i-th ES in a round of operation. The specific calculation formula is as follows:

$$t^{mov}_{m,i} = \sum_{L_{j,k} \in I_m \to \{ES_i\}} L^m_{j,k}/v, I_m \to \{ES_i\} \subseteq P_m \tag{1}$$

Offloading time: the task offloading time is the time required for an AGV to transmit data to the ES. We express $t^{off}_{m,i}$ as the time spent by the AGV_m offloading to the i-th ES and s as the offloading bandwidth. Please be advised, s is identical among ESes and equal to task receiving bandwidth. Suppose the number of tasks is $Task^m_i$, then we have:

$$t^{off}_{m,i} = Task^m_i/s \tag{2}$$

Similarly, it can be obtained that the time t^{on}_m required for the AGV_m to receive the tasks at the starting point is:

$$t^{on}_m = t^{off}_m = \sum_{i=1}^{|D_m - I_m|} t^{off}_{m,i} \tag{3}$$

In addition, if AGV_m cannot offload normally according to conflicts, it may wait at ESes for its turn. T_m is the upper bound of time an AGV receives tasks, offload tasks, and travels in each round. It varies due to the offloading scheduling (t_{wait}). If it is too long and longer than the period of cyclical tasks, it will decrease the number of offloaded tasks. Therefore, the total time T_m required in one cycle of AGV_m is:

$$T_m = 2t^{on}_m + L_{total}/v + t_{wait} \approx L_{total}/v + t_{wait} \tag{4}$$

Because the offloading time is very small compared with the moving time, waiting time, and processing time, and equal across AGVs, it can be regarded as a constant (we set it to 0 in the experiment). When AGV_m does not need to wait in a cycle, t_{wait} is also 0.

Processing time: It is used to evaluate the time required for ESes to process offloaded tasks from AGVs. This paper assumes that the computing power and

capacity of each ES are given and different. These parameters together with the distance variation will lead AGVs to assign different number of tasks to ESes. For the i-th ES, the task processing time $t_{m,i}^{cal}$ for tasks of AGV_m is:

$$t_{m,i}^{cal} = Task_i^m / E_i \tag{5}$$

Then, in a particular cycle, the final task finishing time is the largest one among all tasks:

$$t_m^{cal} = \max\left(t_{m,i}^{cal}\right), \forall i, i \in |D_m - I_m| \tag{6}$$

This paper aims to maximize the multi-AGV cyclical task offloading scheduling problem to maximize the total number of processed tasks offloaded by multiple AGVs in a given time. Therefore, the objective function is:

$$Task^{total} = maximize \sum_{i=1}^{M} Task^i * T^{total} / T_i^{single} \tag{7}$$

To maximize the total number of tasks that can be offloaded in a given amount of time, we propose the following algorithm design.

Algorithm design

The AGV assisted offloading scheduling problem is divided into two parts using an efficient multi-AGV cyclical offloading optimization algorithm (MCOO). In particular, the best offloading plan that only takes into account one AGV is determined in the first place. Considering the task processing efficiency and capacity limitation of ESes and the time spent by each AGV in a cycle without conflict, the number of tasks that each AGV can offload successfully during this period is calculated. Any AGV can select the greedy offloading strategy if the offloading demands of all AGVs in a cycle can be satisfied. The weighted polling method is used to distribute tasks to achieve a load balance state if the offloading demand of AGVs in a cycle cannot be satisfied. Then, in the second part, based on the optimal offloading scheme obtained for a single AGV, the game theory is used to optimize the offloading decisions of AGVs considering the impact of multiple AGVs offloading to the same ES at different times. Finally, to solve the problem distributively, reinforcement learning is used to train each AGV asynchronously, so that each AGV can learn the optimal offloading scheme, which helps achieve the goal of this paper. The implementation of the proposed method is introduced in detail below.

Cyclical offloading optimization of single AGV: For the offloading scheduling of periodic tasks, the offloading scheme of an AGV in the previous cycle may affect the offloading scheme of subsequent cycles. Thus, a single AGV cyclical offloading optimization algorithm that combines load balancing and the greedy algorithm is proposed in this section. Firstly, it calculates the time taken for each AGV to complete a cycle of operation and offloading without conflict, and then calculates the number of tasks that each ES can handle within this time. If the task cannot be offloaded in this case, the weighted polling algorithm is used to calculate the load-balancing offloading scheme of AGVs in a cycle, which can also reduce the impact on the offloading scheme in subsequent cycles. Through the joint optimization of the greedy algorithm and load balancing algorithm, the optimal offloading scheme of each AGV without conflict can be obtained. This detail is explained below.

Each AGV carries some tasks and starts from the starting point at a speed v and drives circularly according to the planned path. When it passes through an ES node during the driving process, it offloads some of the tasks to the ES for processing. When the AGV reaches the last ES, it needs to offload the rest of the tasks it carries, and then the AGV returns to the starting point to receive tasks again and starts the next round of operation and offloading. Firstly, we calculate the time required for one round of AGV operation and offloading. From the above, it can be obtained that the time required for one round of the AGV cycle is $T_m = L_{total}/v + t_{wait}$. therefore, for the i-th ES passed by an AGV, the task quantity $Task_i^{greedy}$ that ES can handle during this period is:

$$Task_i^{greedy} = \begin{cases} E_i * T, & E_i * T \le C_i \\ C_i, & E_i * T > C_i \end{cases} \tag{8}$$

Then the weighted polling algorithm is used to calculate the load-balancing distribution scheme of AGVs at each ES. Because the AGV passes through different ESes in sequence, the time when each ES receives the task and starts processing is different. Therefore, when the AGV travels to the i-th ES, the ES_x before the i-th ES is already processing previous tasks. Then, for any ES before the i-th ES, the time t_x^{cal} that starts before the i-th ES is:

$$t_x^{cal} = \sum_{L_{j,k} \in \{ES_x\} \to \{ES_i\}} L_{j,k}/v, \{ES_x\} \to \{ES_i\} \subseteq P_m \tag{9}$$

The number of tasks $Task_x^{cal}$ that ES_x can handle in this part of time is:

$$Task_x^{cal} = min\left(RC_x, E_x * t_x^{cal}, RT\right) \tag{10}$$

where RT is the remaining tasks of the AGV. Therefore, when the AGV reaches D at the last ES, the number of tasks assigned to the first $|D - I| - 1$ ESes are:

$$Task^{cal} = \sum_{i=1}^{|D-I|-1} Task_i^{cal} \tag{11}$$

The remaining tasks RT to be assigned of the AGV are:

$$RT = Task - Task^{cal} \tag{12}$$

These remaining tasks can also be allocated according to the computing capacity and remaining capacity of each ES. If the ES is fully loaded, it will refuse to accept the task. Therefore, the sum of the efficiency of ES under full load is calculated as:

$$E_{total} = \sum_{i=1}^{|D-I|} E_i, \text{if } ES_i \text{ is fully loaded, } E_i = 0 \tag{13}$$

Thus, it can be calculated that the task amount RT_i can be reassigned to the i-th ES is:

$$RT_i = \begin{cases} E_i/E_{total} * AR, & E_i/E_{total} * RT \leq RC_i \\ RC_i, & other \end{cases} \tag{14}$$

where RC_i represents the remaining capacity after the i-th ES receives the task amount of $Task_i^{cal}$:

$$RC_i = C_i - Task_i^{cal} \tag{15}$$

If the remaining capacity of ES fails to meet the assigned task volume during the allocation process, the excess task volume RT' needs to be allocated again according to the above steps. Therefore, the number of tasks allocated to the i-th ES using the weighted polling algorithm is:

$$Task_i^{balance} = Task_i^{cal} + RT_i \tag{16}$$

As mentioned above, because an AGV will operate multiple cycles, combined with greedy allocation and weighted polling allocation under restrictive conditions, the amount allocated by AGV to the i-th ES is:

$$Task_i = \begin{cases} Task_i^{greedy}, & Task_i^{greedy} \geq Task_i^{balance} \\ Task_i^{balance}, & Task_i^{greedy} < Task_i^{balance} \end{cases} \tag{17}$$

The algorithm is shown in Algorithm 1. After the task of each AGV is reasonably allocated by using the single AGV cyclical offloading scheduling algorithm, the scheme with no or little impact on the subsequent cycle can be obtained. Similarly, the offloading scheduling that has no impact on the subsequent cycle allocation can also be obtained. On this basis, the offloading schemes of multiple AGVs at the same ES can be coordinated by collecting information of multiple cycles, so that the offloading allocation schemes of multiple AGVs can reach an excellent balance state. Next, we will use the idea of game theory combined with multi-agent reinforcement learning to optimize the problem of multi-AGV cyclical offloading.

Input:
Path set L_set and carried task set $Task\_set$ of each AGV, capacity set C_set and efficiency set E_set of each ES;
Output:
The set of offloading schemes for each AGV
$RT^m = Task^m$
Calculate $Task_i^{greedy}$ from Eq. (8)
Calculate $Task^{cal}$ from Eq. (11)
Update RC_i, RT
while $RT^m \neq 0$ **do**
  Calculate RT_i^m from Eq. (14)
  Offloading the task of $Task_i^m$ at ES_i
  Update $Task_i^m$
  $RT^m = RT^m - Task_i^m$
end while

Algorithm 1 Single AGV Cyclical Scheduling Algorithm

Cyclical offloading optimization of multi-AGV: As mentioned above, when multiple AGVs perform periodic mobile offloading, due to the first mover advantage between AGVs, later AGVs may not offload according to their planned offloading scheme, thus affecting the scheduling efficiency. This kind of situation with first mover advantage is also a common phenomenon in game theory. To achieve the final goal of this paper, we introduce a dynamic cooperative game.

The game model generally includes some participants, a strategy set and a utility function. We regard each AGV as a participant in the game, with a total number of M. When each AGV arrives at an ES, it can know the decision-making information of other previous arriving participants and uses it as a basis to determine its offloading amount. The decision-making of each AGV when it encounters insufficient resources at an ES is taken as the strategy set. Therefore, the "strategy" of the game is whether the previous arriving AGV gives up resources for the second comer. Therefore, the policy set of the m-th participant at N shared ESes is $S_m = \{S_m^1, S_m^2, ..., S_m^N\}$, where S_m^i is either S^{refuse} or S^{off}, and the policy set of all participants is $S = \{S_1, S_2, ..., S_M\}$. The utility function in the game now can be transformed into the objective function of this paper.

In the cooperative game, multiple AGV groups are called alliances, which are composed of AGVs for collaborative interests, that is, a larger total offloading volume. The impact of an AGV on the group is realized through the offloading decision of the first arrivals (whether to choose to give up resources for the later arrivals). Here we use this model to solve the offloading decision problem of multiple AGVs.

Specifically, for two AGVs that will pass through the same ES, AGV_1 will arrive at the ES first and offload $Task_1$. The capacity of the ES is assumed to be C. When AGV_2 arrives at the ES, it needs to offload $Task_2$. If $Task_1$ and $Task_2$ can be offloaded normally, AGV_1 does not need to make the decision to give up resources when the next round arrives, which

is S^{refuse}. If offloading $Task_2$ will exceed the capacity of the ES, but AGV_2 will offload this part of the task to other ESes and will not affect the offloading scheme of the next round, AGV_1 will decide S^{refuse}. If offloading $Task_2$ will exceed the capacity of the ES and assigning this part of the task to other ESes will affect the offloading scheme of the next round, AGV_2 will leave a message at that ES. When AGV_1 arrives in the next round, it will calculate whether the resources will affect its subsequent offloading scheme. If not, AGV_1's decision is S^{off}. If so, AGV_1 will consider whether giving up resources will increase the benefits to the alliance of AGV_1 and AGV_2. If it can increase the overall benefits of the alliance, AGV_1 will decide S^{off}, otherwise, it will decide S^{refuse}. As can be seen from the description above, the first arrivals benefit from delegating decision-making, and various processing techniques can be used depending on the state information and overall income of the second arrivals.

According to the maximization objective function of the above formula, the AGV needs to consider maximizing the current income when making decisions, that is, the offloading volume of the AGV alliance. This objective is equal to that each round of AGV transportation and offloading take the least amount of time. That is, try to avoid waiting for offloading in the process of one round and reduce the total waiting time of the AGV alliance. In addition, when an AGV chooses to refuse or give up resources when making offloading decisions, it also needs to consider how much to give up to maximize benefits.

In terms of multi-agents, the difficulty of learning is much higher than that of a single agent. Because a single agent only depends on the action of one agent during state transition, while in a multi-agent system, the state transition of an agent will be affected by the joint actions of all agents. Therefore, in the multi-agent system, the MDP attribute will become invalid, and it is difficult to evaluate the impact of each agent on the overall results. The multi-agent training optimization problem is solved in this paper using a multi-agent actor-critic algorithm. Among them, multiple "actors" are trained asynchronously, and then a centralized "critic" evaluates the results to optimize the behavior of the agent.

The model in this paper can be formalized as a quad $< AM, S, A_m, R >$, where AM is the number set of all AGVs, S is the global state space, A_m is the action space of the m-th AGV, and R is the reward. In each step, each AGV takes action according to the obtained information and strategy π. After execution, it will get a reward R, and then state s will move to the next state s'. The goal of each AGV is to maximize its expected return:

$$J_m(\pi_m) = \sum r(s, a_1, a_2, ..., a_M) \tag{18}$$

Action a is the offloading amount of an AGV at the shared ES. To achieve the goal of maximizing the total offloading amount of AGVs within a given time T^{total}, the global reward can be set as:

$$R = \sum_{i=1}^{M} T^{total} / T_i^{single} * Task_i^{single} \tag{19}$$

As was already mentioned, the Actor-Critic algorithm (AC algorithm) combines two reinforcement learning algorithms based on strategy and value, enabling it to more effectively update in one step while also choosing the best course of action in continuous or high-dimensional action space. For the M AGVs in the model, the hypothetical strategy $\pi = \pi_1, \pi_2..., \pi_M$, parameter $\phi = \phi_1, \phi_2, ..., \phi_M$, the return function for the m-th AGV is:

$$J(\phi_m) = V^{\pi_m}(s) = E_{\pi_m}[V] \tag{20}$$

Then the strategy function is derived to obtain:

$$\nabla_{\phi_m} \pi_m(s, a) = \pi_m(s, a) \nabla_{\phi_m} \log \pi_m(s, a) \tag{21}$$

Thus, the expected income gradient of the m-th AGV is:

$$\nabla_{\phi_m} J(\phi_m) = E_{\pi_m}\left[\nabla_{\phi_m} \log \pi_m(s, a) Q_\theta(s, a)\right] \tag{22}$$

where $Q_\theta(s, a)$ is the critic's Q function. $Q_\theta(s, a)$ takes the global state s and joint action a as the input and outputs of the global Q value. At this time, the critic's function Q_θ can be updated as:

$$L(\theta) = \sum_{i=1}^{M} E_{(s,a,s',a')}\left[\left(y - Q_\theta(s, a)\right)^2\right] \tag{23}$$

where y is expressed as:

$$y = r + \gamma E_{a'}\left(Q_{\theta'}\left(s', a'\right)\right) \tag{24}$$

The A3C (Asynchronous Advantage Actor Critic) algorithm is presented in this paper to speed up the algorithm's convergence and reduce data correlation. Based on the framework of the AC algorithm, the A3C algorithm introduces the idea of asynchronous training, which can make multiple agents train and learn asynchronously in multiple environments. Therefore, there is no correlation in the data itself, and there is no need for the empirical playback of the DQN algorithm to stabilize the learning process. In addition to saving storage space, this greatly accelerates training time and improves the uniformity of sample distribution.

In addition, to clearly identify the impact of agents on the global reward, we propose to use the differential reward. The differential reward can be described as:

$$D_m(s,a) = r(s,a) - r\left(s,\left(a_m, a_m^c\right)\right) \tag{25}$$

where $r(s,a)$ is the global reward and $r\left(s,\left(a_m,a_m^c\right)\right)$ is the global reward when the m-th agent takes the default behavior. Therefore, agent m can improve D_m's behavior and increase the overall reward. The distinct offloading scheme of each AGV obtained in the previous section is trained here as the agent's default behavior.

In this algorithm, multiple agents share the global network parameters ϕ and ϕ_v, which represent the actor network and critical network respectively, and each agent has its two network parameters ϕ' and $\phi_v^{'}$, as well as the global counter T and each agent's counter t. At the beginning of the algorithm, the agent counter is initialized first, and then the global network parameters are used to initialize the network parameters in each agent thread, and the initial state s_t is obtained. In the iterative process, the agent uses the policy function $\pi\left(a_t|s_t;\phi'\right)$ to get and execute action a_t to obtain the next state s_{t+1} and the corresponding reward R. At this time, the value function of each state can be calculated through the critical network. After several iterations, the parameters of each agent thread are used to update the global network parameters. The basic description of multi-AGV cyclical offloading optimization algorithm (MCOO) is described in Algorithm 2.

1: Randomly initialize environments with M agents
2: Initialize agent thread network parameters $\phi' = \phi$ and $\phi_v^{'} = \phi_v$
3: Initialize thread period $t = 1$
4: **for** each episode **do**
5: Initialize the thread state s_t
6: **for** each time period t **do**
7: Perform a_t according to policy $\pi\left(a_t|s_t;\phi'\right)$
8: Receive reward r_t and new state s_{t+1}
9: $t = t+1, T = T+1$
10: **end for**
11: Get the value function for each state:

$$R = \begin{cases} 0, \ reached\ the\ termination\ state \\ V\left(s_t,\phi_v^{'}\right), \ otherwise \end{cases}$$

12: Calculate the value function each episode: $R_t = r_t + R_{t+1}$
13: Update thread's network parameters
14: Update the difference rewards
15: **end for**
16: Update global parameters
17: Update offloading decision making and calculate global benefits

Algorithm 2 Multi-AGV Cyclical Offloading Optimization Algorithm (MCOO)

Experiments

Results from simulations are presented in this section to attest to the effectiveness of the proposed algorithms. We first introduce the setting of various parameters, then conduct experiments to observe the total revenue and service delay of multi-AGV cyclical offloading, then compare it with related algorithms to verify the effectiveness, and finally observe the robustness on performance by changing parameters.

Data set

In this experiment, we comprehensively consider the realistic factors, such as the transportation and offloading routes of AGVs, the computing and storage capacity of edge servers, and periodic tasks in accord with the industrial scenario. Firstly, as shown in Fig. 2, there are 5 AGVs initially in the experiment. The plant floor is divided into grids. The distance of each grid is 50m. Each AGV starts from the task receiving point at the speed of 5m/s and travels along the given route. An AGV will pass through some edge servers during driving. The number of tasks carried by an AGV is randomly selected between {200, 300, 500}kB, the distance between edge servers is randomly selected between [100, 300]m, and the upper bound of computing power of edge servers is taken as {50, 60, 70, 80, 90}kB/min, and the storage capacity is randomly taken as {100, 150, 180}kB. Please be noted, Fig. 2 is only a demonstration of one slice of multiple rounds. The number of AGVs, number of ESes, computing power of ESes, and other parameters will change from round to round. The simulation parameters are summarized in Table 2.

We must train AGVs for numerous cycles to obtain a stable offloading scheme. According to the area of the scenario and the speed of AGVs, each round could be finished around 10 minutes. Thus, this paper sets the total evaluation time to one hour. This time can be changed without affecting the performance characteristics. The cyclical offloading of a single AGV is trained first, followed by the training of the entire multi-AGV and comparisons with the greedy algorithm, the DQN algorithm, and the basic AC algorithm to accurately assess the performance in complex scenes. Evaluation metrics include the convergence speed and results of the algorithm, which is very important for delay-sensitive tasks. In addition, we also observe the impact on the performance of the algorithm by changing the parameters of the algorithm and servers.

Simulation results

Optimal offloading scheme of single AGV for periodic tasks: We firstly optimize the offloading allocation of single AGVs. Table 3 shows the average scheduling time of each cycle and the cumulative offloading volume of one hour.

Table 3 shows the number of tasks received by 5 AGVs at the beginning of each cycle, and the average time

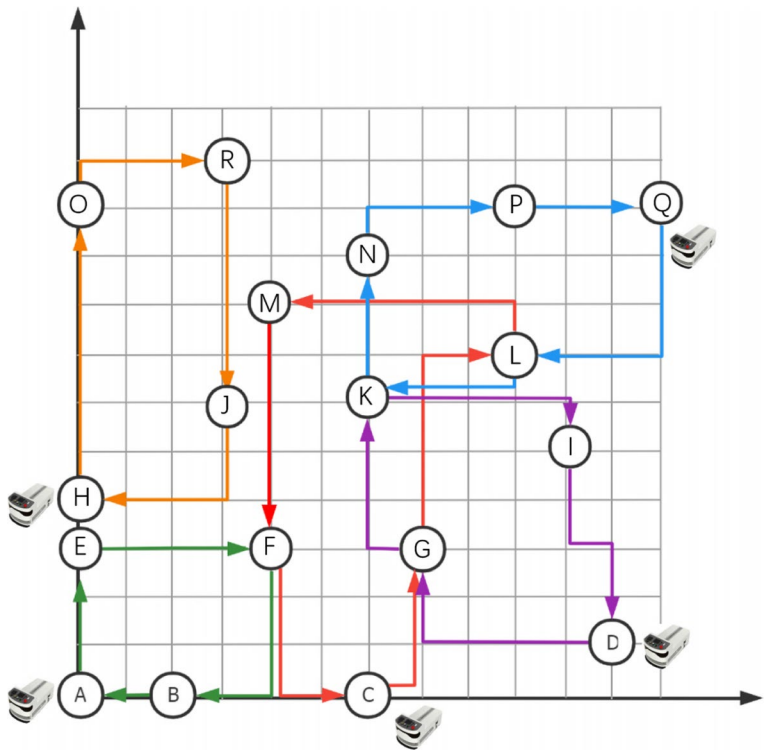

Fig. 2 Path of AGV mobile offloading

Table 2 Parameter setting

Parameter	Value Range
M	[2, 10]
N	[15, 20]
$Task$	{200, 300, 500}kB
v	5m/s
L	[100, 300]m
C	{100, 150, 180}kB
E	{50, 60, 70, 80, 90}kB/min

Table 3 Optimal offloading scheme of single AGV

	Tasks received by AGV per cycle (kB)	Average time consumed per cycle (min)	Total offloading volume of AGV in one hour (KB)
AGV_1	500	5	6000
AGV_2	300	5	4500
AGV_3	200	4	3000
AGV_4	300	4.5	4000
AGV_5	500	6	5000

consumed by each AGV for traveling and offloading, and the total offloaded tasks within one hour. Figure 3 shows the offloading amount in one hour under different algorithms. The non-optimized algorithm is actually a greedy algorithm, in which the AGV always tries to offload tasks as many as possible at the first convenience. As can be seen from Fig. 3, the single AGV offloading optimization algorithm proposed in this paper can keep the time consumed in each cycle relatively stable, while the baseline algorithm cannot achieve the optimal offloading. For the baseline algorithm, with the increasing of elapsed time, the time consumed by AGVs in one cycle will increase, thus gradually reducing the offloading amount of tasks.

Comparison of training efficiency and results of different algorithms: In this section, we apply four different algorithms to comprehensively evaluate the performance of multiple AGVs. The four algorithms are the greedy algorithm, DQN algorithm, AC algorithm, and multi-AGV cyclical offloading optimization algorithm (MCOO). Here, the total number of tasks offloaded by multiple AGVs in one hour is selected as the metric to measure the performance of the algorithms. Since the results of the greedy algorithm will not change with the process of training, it is not used in comparison for training performance. For simplicity, the same learning rate $\alpha = 0.01$ and the same loss function $\gamma = 0.95$ are selected

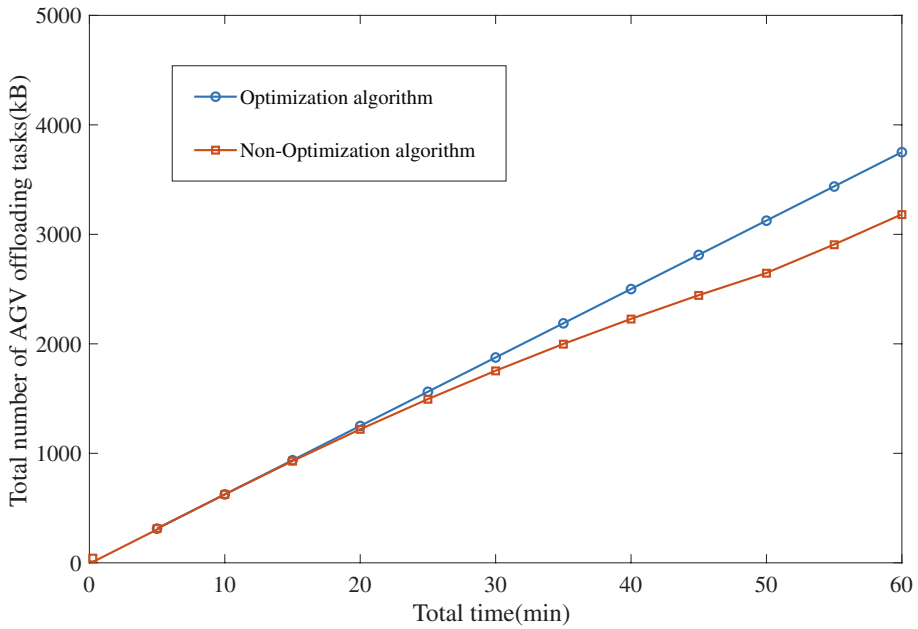

Fig. 3 Task offloading volume of single AGV in one hour

for the other three algorithms in training, and the mean value of rewards is taken as the result, as shown in Fig. 4.

As can be seen from the above figure, the proposed MCOO algorithm shows the best training speed which is much faster than the other two algorithms. Except that, the acquired total reward is also larger than the competitors. This is because that the algorithm can make better decisions in a shorter time, which is very important for delay-sensitive tasks.

Now, we evaluate and compare the performance by changing the parameters. The first parameter is the number of AGVs. We set the number of AGVs to 2, 4, 6, 8, and 10 respectively, and conducted 200 independent experiments for each group of AGVs to take the mean value. The results are shown in Fig. 5. As mentioned above, when multiple AGVs offload tasks at different times to the same ES, the remaining capacity of the ES is often not able to meet the offloading requirements of the later AGVs, which affects the offloading scheme of the later AGVs. This leads to idle waiting for offloading, which will affect the overall offloading volume. It can be seen from the figure that the result of the greedy scheduling strategy is the worst because all AGVs choose to offload as many as possible in the offloading process. In this way, in the shared ES, the first AGV always preempts resources, resulting in the later AGV being prone to waiting for offloading. Therefore, the final result is not good. And with the increase in the number of AGVs, the effectiveness of greedy scheduling is getting worse and worse. The DQN algorithm and AC algorithm have poor convergence speed when training multiple agents. Also, with the increase of the number of agents, the training effectiveness also gets worse. MCOO algorithm combines the game theory with the A3C algorithm to optimize the offloading decision of AGVs at shared ESes. Because multiple threads are used to train each AGV, not only the training speed but also the final result are improved.

The influence of other factors on performance: In this part, we study the influence of the number of ESes and the computing power of ESes on the performance of the algorithm. Here, the number of AGVs are fixed to 5, the number of ESes and the computing power of ESes are adjusted respectively. 200 groups of experiments are carried out, and the mean value are taken as the final results. The routes of AGVs are different in each round. Figure 6 compares the results when the number of ESes passed by each AGV increases from 4 to 7. I.e., the routes of AGVs are randomly generated, but the maximum number of ESes they pass is fixed between 4 and 7. It can be seen from Fig. 6 that with the increase of the number of ESes, the impact of the offloading scheme of AGV at the shared ESes on the subsequent rounds and other AGVs becomes smaller and smaller, while the offloading time of each AGV is more and more stable. When the maximal number of ESes passed by an AGV increases to 6, all algorithms can achieve the maximum offloading volume.

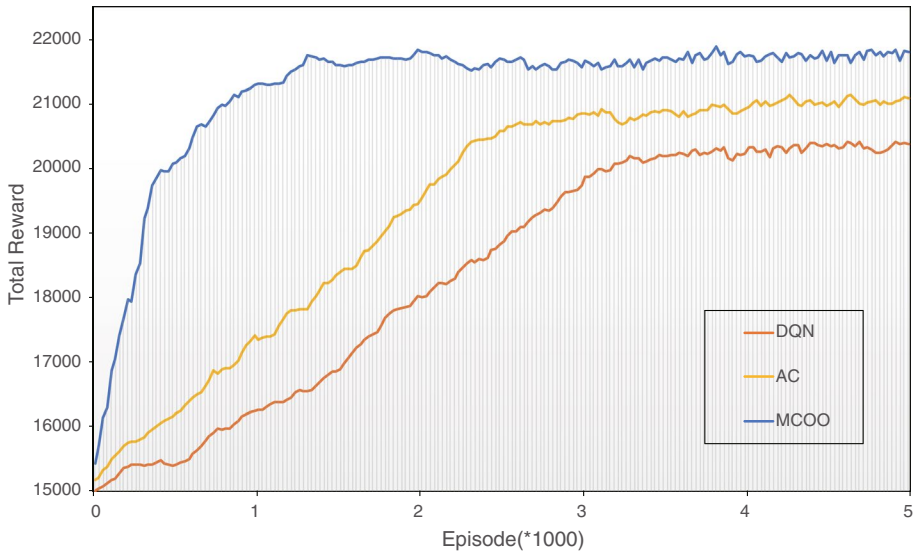

Fig. 4 Comparison of total rewards of different algorithms

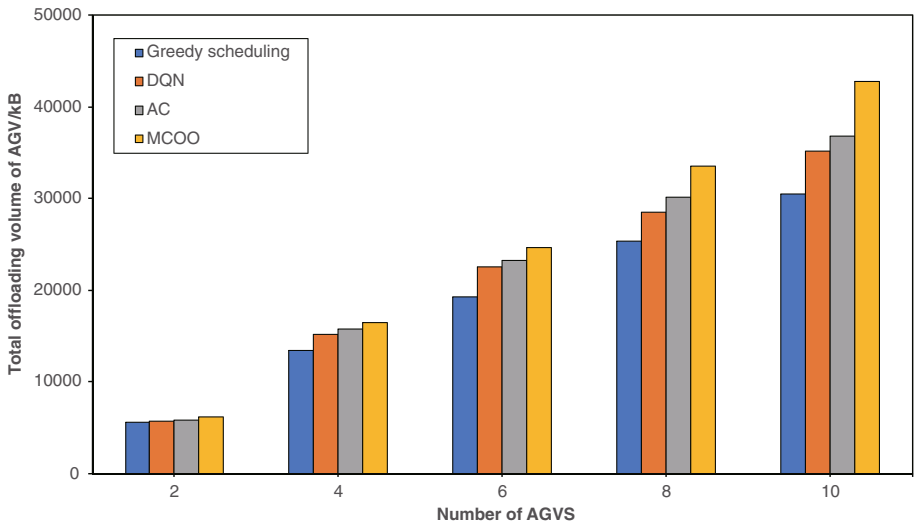

Fig. 5 Comparison of total offloading volume of different algorithms

Figure 7 compares the results of various algorithms when the upper bound of the processing efficiency of ESes increases from 50kB/min to 90kB/min. Similar to the result in Fig. 6, when the computing power of ESes reaches 80kB/min, which exceeds the offloading requirements of AGVs, all algorithms can achieve the maximum offloading efficiency. During the process, the proposed MCOO algorithm always needs less ESes to achieve the same offloading amount or the largest offloading amount with the same computing power, compared with other algorithms.

Conclusion

This paper studies the task offloading allocation problem of multiple AGVs aiming at more total offloaded tasks in a given time period. In the proposed scenario, there are AGVs, periodic tasks, and edge servers/cloudlets/multi-cloud. When offloading tasks to the edge server, AGVs should take into account not only the effects of other AGVs' offloading decisions but also their own on subsequent cycles. When solving this problem, this paper decomposes the problem into two parts: in the first part, the optimal offloading allocation of a single AGV is carried

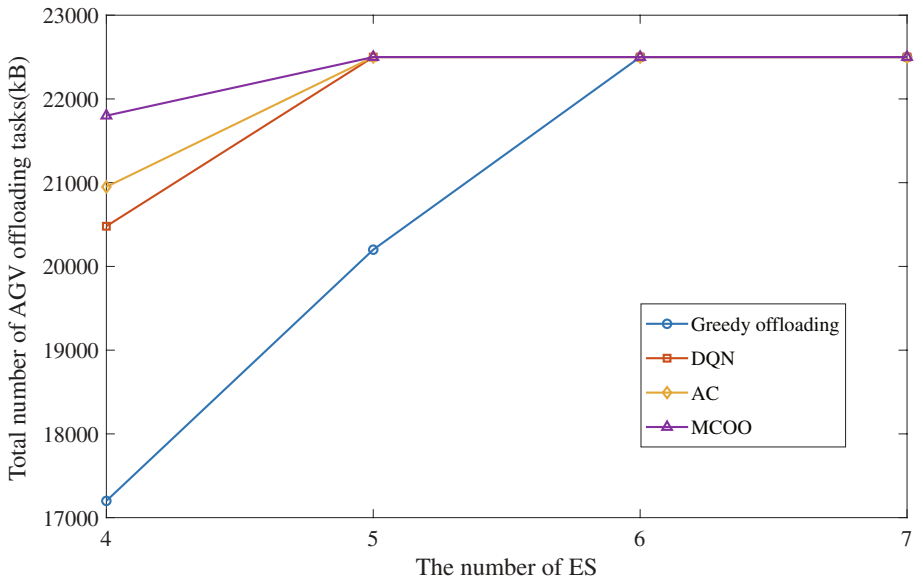

Fig. 6 The effect of the number of ESes on the results

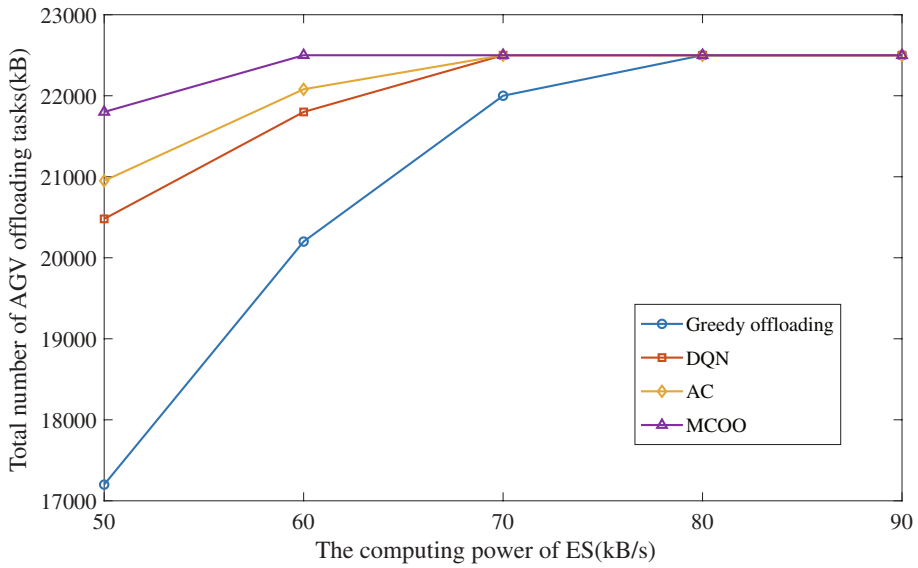

Fig. 7 The effect of the computing power of ESes on the results

out separately, and the offloading scheme with no or minimal impact on other cycles is obtained. Then, based on the first part, considering the overall benefits of multiple AGVs, the game theory is introduced to solve the offloading decision of AGVs at the shared ESes. Then, based on the distributed idea, the Multi-Agent reinforcement learning algorithm is used to train each AGV asynchronously, and then the central system optimizes the whole to obtain the optimal offloading scheme of each AGV. Simulation results show that the multi-AGV offloading optimization algorithm proposed in this paper can effectively improve the overall performance of multi-AGV mobile offloading in the scenario of periodic task scheduling.

As privacy has become a major concern in edge-cloud IIoT, federated learning has been proven very effective in the existing work [33]. To achieve privacy preservation, future work will consider combining federated learning with reinforcement learning. Another trend is to use

AGVs for edge caching [34], which is very important when the amount of generated data is very large.

Abbreviations

AGV: Auto guided vehicle; ES: Edge server; MCOO: Multi-AGV cyclical offloading optimization; A3C: Asynchronous advantage actor critic.

Acknowledgements

The authors thank the editor and anonymous reviewers for their helpful comments and valuable suggestions.

Authors' contributions

Peng Liu proposed the solution to the target problem and led the write-up of the manuscript. Zhe Liu and Zifu Wu completed most of the writing of this manuscript and conducted the experiment. Ji Wang found the target problem from his working experience and built the system model. Peng Li took part in the discussion of the solution and gave many useful suggestions. Huijuan Lu helped in revising the paper. All authors have read and approved the manuscript.

Authors' information

Peng Liu received his B.S. and M.S. in Computer Science and Technology from Hangzhou Dianzi University respectively in 2001 and 2004, and Ph.D. in Computer Science and Technology from Zhejiang University in 2007, China. Currently, he is an associate professor at Hangzhou Dianzi University. His research interests include the Internet of things, edge computing, and vehicular ad-hoc networks. Zhe Liu is a graduate student at Hangzhou Dianzi University. Her research interest is the Internet of vehicles. Ji Wang received his M.S. in Computer Science and Technology from Hangzhou Dianzi University, China in 2005. He is now the CEO of Yuxiang Technology (Hangzhou) Co Ltd, China and the chief engineer in R&D team as well. His research interests include industrial automation and big data. Zifu Wu was a graduate student at Hangzhou Dianzi University. His research interest is edge computing and the industrial Internet of things. Peng Li received his BS degree from Huazhong University of Science and Technology, China, in 2007, the MS and PhD degrees from the University of Aizu, Japan, in 2009 and 2012, respectively. Dr. Li is currently an associate professor at the University of Aizu, Japan. His research interests mainly focus on cloud computing, Internet-of-Things, big data systems, as well as related wired and wireless networking problems. Huijuan Lu received her Ph.D. degree from the China University of Mining and Technology, China, in 2012. Since 1999, she has been with the College of Information Engineering, China Jiliang University, Hangzhou, China, where she is currently a professor in computer science and technology. Her current research interests include big data, distributed computing, bioinformatics, data mining, pattern recognition, and artificial intelligence.

Funding

This work is supported by the Natural Science Foundation of China under Grant 62172134.

Availability of data and materials

Please contact the corresponding author for available data and materials.

Declarations

Competing interests

The authors declare that they have no competing interests.

Author details

[1]School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, China. [2]Yuxiang Technology (Hangzhou) Co Ltd, Hangzhou, China. [3]HDU-ITMO Joint Institute, Hangzhou Dianzi University, 310018 Hangzhou, China. [4]School of Computer Science and Engineering, University of Aizu, Aizu, Japan. [5]Key Laboratory of Electromagnetic Wave Information Technology and Metrology of Zhejiang Province, College of Information Engineering, China Jiliang University, Hangzhou, China.

Received: 18 August 2022 Accepted: 21 October 2022
Published online: 10 November 2022

References

1. Chen Y, Gu W, Xu J et al (2022) Dynamic task offloading for digital twin-empowered mobile edge computing via deep reinforcement learning. China Commun
2. Li K, Zhao J, Hu J et al (2022) Dynamic energy efficient task offloading and resource allocation for noma-enabled iot in smart buildings and environment. Build Environ. https://doi.org/10.1016/j.buildenv.2022.109513
3. Tomarchio O, Calcaterra D, Modica GD (2020) Cloud resource orchestration in the multi-cloud landscape: a systematic review of existing frameworks. J Cloud Comput 9:49
4. Ren Q, Liu K, Zhang L (2021) Multi-objective optimization for task offloading based on network calculus in fog environments. Digit Commun Netw. https://doi.org/10.1016/j.dcan.2021.09.012
5. You Q, Tang B (2021) Efficient task offloading using particle swarm optimization algorithm in edge computing for industrial internet of things. J Cloud Comput 10(1):41
6. Wu Y, Dai HN, Wang H, Xiong Z, Guo S (2022) A survey of intelligent network slicing management for industrial iot: Integrated approaches for smart transportation, smart energy, and smart factory. IEEE Commun Surv Tutorials 24(2):1175–1211
7. Cheng N, Lyu F, Quan W, Zhou C, He H, Shi W, Shen X (2019) Space/aerial-assisted computing offloading for iot applications: A learning-based approach. IEEE J Sel Areas Comput 37(5):1117–1129
8. Chen Y, Zhao F, Chen X, Wu Y (2022) Efficient multi-vehicle task offloading for mobile edge computing in 6g networks. IEEE Trans Veh Technol 71(5):4584–4595
9. Chen J, Chen S, Luo S, Wang Q, Cao B, Li X (2020) An intelligent task offloading algorithm (itoa) for uav edge computing network. Digit Commun Netw 6(4):433–443
10. Yang T, Kong L, Zhao N, Sun R (2021) Efficient energy and delay tradeoff for vessel communications in sdn based maritime wireless networks. IEEE Trans Intell Transp Syst 22(6):3800–3812
11. Liu Y, Peng M, Shou G, Chen Y, Chen S (2020) Toward edge intelligence: Multiaccess edge computing for 5g and internet of things. IEEE Internet Things J 7(8):6722–6747
12. Xu J, Li D, Gu W et al (2022) Uav-assisted task offloading for iot in smart buildings and environment via deep reinforcement learning. Build Environ. https://doi.org/10.1016/j.buildenv.2022.109218
13. Shu C, Zhao Z, Han Y, Min G, Duan H (2020) Multi-user offloading for edge computing networks: A dependency-aware and latency-optimal approach. IEEE Internet Things J 7(3):1678–1689
14. Chen Y, Zhao F, Lu Y, Chen X (2021) Dynamic task offloading for mobile edge computing with hybrid energy supply. Tsinghua Sci Technol. https://doi.org/10.26599/TST.2021.9010050
15. Guo H, Liu J (2020) Uav-enhanced intelligent offloading for internet of things at the edge. IEEE Trans Ind Inform 16(4):2737–2746
16. Shang B, Liu L (2020) Mobile-edge computing in the sky: Energy optimization for air—ground integrated networks. IEEE Internet Things J 7(8):7443–7456
17. Wu Y, Wu J, Chen L, Yan J, Luo Y (2020) Efficient task scheduling for servers with dynamic states in vehicular edge computing. Comput Commun 150:245–253
18. Zhang J, Zhou L, Zhou F, Seet BC, Zhang H, Cai Z, Wei J (2020) Computation-efficient offloading and trajectory scheduling for multi-uav assisted mobile edge computing. IEEE Trans Veh Technol 69(2):2114–2125
19. Sadatdiynov K, Cui L, Zhang L, Huang JZ, Salloum S, Mahmud MS (2022) A review of optimization methods for computation offloading in edge computing networks. Digit Commun Netw. https://doi.org/10.1016/j.dcan.2022.03.003
20. Ali L, Muyeen SM, Bizhani H, Simoes MG (2021) Game approach for sizing and cost minimization of a multi-microgrids using a multi-objective optimization. 2021 IEEE Green Technologies Conference (GreenTech). IEEE, Denver, pp 507–512
21. Cao B, Xia S, Han J, Li Y (2020) A distributed game methodology for crowdsensing in uncertain wireless scenario. IEEE Trans Mob Comput 19(1):15–28
22. Wang W, Lu B, Li Y, Wei W, Li J, Mumtaz S, Guizani M (2021) Task scheduling game optimization for mobile edge computing. ICC 2021 - IEEE International Conference on Communications. IEEE, Montreal, pp 1–6

23. Wan S, Ding S, Chen C (2022) Edge computing enabled video segmentation for real-time traffic monitoring in internet of vehicles. Pattern Recog 121:108146
24. Raza S, Liu W, Ahmed M, Anwar MR, Mirza MA, Sun Q, Wang S (2020) An efficient task offloading scheme in vehicular edge computing. J Cloud Comput 9:28
25. Zhou Y, GE H, Ma B, Zhang S, Huang J (2022) Collaborative task offloading and resource allocation with hybrid energy supply for uav-assisted multi-clouds. J Cloud Comput 42:11
26. Chen R, Cui L, Wang M, Zhang Y, Yao K, Yang Y, Yao C (2021) Joint computation offloading, channel access and scheduling optimization in uav swarms: A game-theoretic learning approach. IEEE Open J Comput Soc 2:308–320
27. Yan Z, Ge J, Wu Y, Li L, Li T (2020) Automatic virtual network embedding: A deep reinforcement learning approach with graph convolutional networks. IEEE J Sel Areas Commun 38(6):1040–1057
28. Chen Y, Gu W, Li K (2022) Dynamic task offloading for internet of things in mobile edge computing via deep reinforcement learning. Int J Commun Syst. https://doi.org/10.1002/dac.5154
29. Huang J, Gao H, Wan S et al (2023) Aoi-aware energy control and computation offloading for industrial iot. Futur Gener Comput Syst 139:29–37
30. Jiang Q, Xu X, He Q, Zhang X, Dai F, Qi L, Dou W (2021) Game theory-based task offloading and resource allocation for vehicular networks in edge-cloud computing. 2021 IEEE International Conference on Web Services (ICWS). IEEE, Chicago, pp 341–346. https://doi.org/10.1109/ICWS53863.2021.00052
31. Tran-Dang H, Bhardwaj S, Rahim T, Musaddiq A, Kim DS (2022) Reinforcement learning based resource management for fog computing environment: Literature review, challenges, and open issues. J Commun Netw 24(1):83–98
32. Yuan Y, Lei L, Vu TX, Chatzinotas S et al (2021) Energy minimization in uav-aided networks: Actor-critic learning for constrained scheduling optimization. IEEE Trans Veh Technol 70(5):5028–5042
33. Huang J, Tong Z, Feng Z (2022) Geographical poi recommendation for internet of things: A federated learning approach using matrix factorization. Int J Commun Syst. https://doi.org/10.1002/dac.5161
34. Chen Y, Xing H, Ma Z, et al. (2022) Cost-efficient edge caching for noma-enabled iot services. China Commun

Publisher's Note