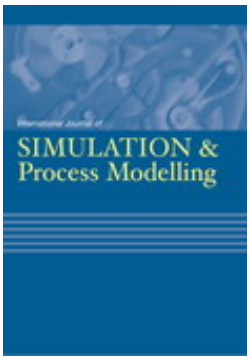
INDERSCIENCE PUBLISHERS
Linking academia, business and industry through research



International Journal of Simulation and Process Modelling

ISSN online: 1740-2131 - ISSN print: 1740-2123
https://www.inderscience.com/ijspm

A computational offloading algorithm for cloud-edge collaboration in smart agriculture

Feng Li, Yiyuan Li, Yuqing Pan

DOI: 10.1504/IJSPM.2024.10066644

Article History:

Received:	22 November 2022
Last revised:	11 May 2023
Accepted:	05 September 2023
Published online:	04 October 2024

A computational offloading algorithm for cloud-edge collaboration in smart agriculture

Feng Li*, Yiyuan Li and Yuqing Pan

School of Computer Science and Communication Engineering,
Jiangsu University,
Zhenjiang, Jiangsu, China
Email: fengli@ujs.edu.cn
Email: lyynothing@126.com
Email: panyq@ujs.edu.cn
*Corresponding author

Abstract: To solve the problem that the massive amount of information and real-time processing in the IoT system puts pressure on the computing resources of the whole system, the industry often adopts the computation offloading algorithm for cloud-edge collaboration. However, at this stage, conventional computation offloading algorithms frequently fail to account for dynamic conflicts between terminal tasks and offloaded tasks, and the results of offloading are inefficient. To that purpose, this paper introduces the task volume prediction model, the response time model and the power consumption model to represent the whole computing process of agricultural activities from various angles. On that basis, the DQN algorithm is used to solve the model's mixed-integer nonlinear optimisation issue. Experiments show that the algorithm proposed in this paper can increase the offloading accuracy and the speed and accuracy of cloud-edge collaborative computing.

Keywords: smart agriculture; cloud-edge collaboration; computation offloading; deep reinforcement learning.

Reference to this paper should be made as follows: Li, F., Li, Y. and Pan, Y. (2024) 'A computational offloading algorithm for cloud-edge collaboration in smart agriculture', *Int. J. Simulation and Process Modelling*, Vol. 21, No. 2, pp.121–129.

Biographical notes: Feng Li obtained his PhD in Engineering in 2011 from the Institute of Computing Technology, Chinese Academy of Sciences and a post-doctoral researcher of computer application technology.

Yiyuan Li is a Master candidate at the School of Computer Science and Communication Engineering, Jiangsu University.

Yuqing Pan is an Associate Professor at the Jiangsu University.

1 Introduction

Smart agriculture has several challenges and new opportunities compared to traditional agriculture, which is pointed out by Ayaz et al. (2019).

However, with the further advancement of agricultural wisdom, many difficult practical problems have arisen in the process of agricultural operations, the most prominent of which is how to process a large amount of perceived information in a timely manner, providing real-time data support for various agricultural operations processes. Since the computing power of terminal farming machines often does not match the computing power required for actual computing tasks, large amounts of perceived information are usually transmitted to the cloud for analysis and computation, which in turn leads to the need to concentrate computing resources in the cloud. Considering the computational time delay and data transmission time delay in the cloud, this will result in the farm being unable to obtain the corresponding data processing results quickly. If we simply increase the computing power of the farm machine terminal, the cost is high on the one hand, and it is difficult to realise the collaboration between the farm machine terminals on the other hand. Therefore, how to deal with a large number of perceived tasks and information while controlling costs is still an urgent problem in the field of agricultural wisdom.

With the development and popularisation of communication technologies such as 5G, as well as the improvement of various cloud, edge and end computing capabilities, a computing model based on cloud-edge-end collaboration provides the possibility of timely analysis and processing of massive heterogeneous sensory information generated by various agricultural operation processes

(Gao et al., 2022; Paraforos and Griepentrog, 2021). Instead of uploading all the computing tasks to the cloud, the computing capabilities of edge-end nodes can be used for partial task computation through cloud-edge-end collaboration. This is the cloud-edge collaboration model, which reduces the overall time delay and obtains computation results faster. Based on the cloud-edge collaboration model, sensory information and agricultural data computation tasks can be automatically assigned to the cloud or the edge, which improves the overall computation efficiency by utilising the computing power of the edge while greatly reducing transmission costs (Ruan et al., 2019). Finally, the operation efficiency of the overall intelligent agricultural system is improved. Thus, this paper proposes a computation offloading algorithm for cloud-edge collaboration in smart agriculture.

At present, many scholars have proposed many effective ideas in computation offloading algorithms for cloud-edge collaboration. Meyer et al. (2021) investigated a method for classifying applications based on interference levels and proposed a static interference model and policy for scheduling co-hosted cloud applications, with preliminary results showing an average 27% improvement in resource utilisation efficiency when applying their classification method in a cloud-side collaboration scenario; Zhang et al. (2021) designed a fine-grained serverless pipeline, thus facilitating fine-grained adaptive partitioning of cloud-edge workloads with multiple concurrent query pipelines. This particular collaborative cloud-edge design reduced the overhead of the cloud-only solution by 86.9% and the data transfer overhead by 74.4%, and increased the analytic throughput of the edge-only solution by 20.6%; Zhou et al. (2021) improved the model based on time delay using the shortest response time priority algorithm and optimised the update strategy of expected response time based on their model, and their results showed that it outperformed the traditional serial processing framework in terms of processing time delay; Xuewen and Jingxian (2022) present a computational model that integrates latency, energy consumption, user priority, and computational resource cost, and introduces convex optimisation conditional computation to compute the offloading optimal solution so that the model can sense user priority, and emergency users have higher utility and lower latency; Ren et al. (2021) designed a collaborative task offloading and resource scheduling framework, including base station collaboration space (BSCS) and micro BSCS, in order to enhance the collaboration of nodes in the edge environment to balance the computational and caching resources in heterogeneous wireless networks, and this model was successfully applied in the field of wireless surveillance cameras, which improved the timeliness and accuracy of the original scheme to a large extent; Zhang et al. (2017) developed an access controller management model to further reduce computational duplication and data transmission redundancy in mobile edge computing (MEC) in 5G networks, while proposing a new algorithm to trade-off between energy consumption and offloaded data volume under the constraint of total computation time and calculate the optimal solution of partial critical path (PCP) by executing the proposed dynamic programming algorithm, and their experimental results proved the correctness of the new approach.

However, the above papers of the computational offloading algorithm for cloud-edge collaboration use static modelling mechanisms, which only consider the current task volume and related states, and do not consider the future task computation characteristics, so for some tasks with periodic nature, evaluating offloading tasks based only on the current states may bring problems such as inefficiency and task conflicts. Agricultural activities are often characterised by obvious seasonality and periodicity, so the common computational offloading algorithms for cloud-edge collaboration are difficult to meet the actual computing needs of smart agriculture activities. Because of this, it is an urgent problem to design a computational offloading algorithm for cloud-edge collaboration that is more real-time, more accurate, and better adapted to the needs of smart agriculture activities.

To address the above issues, we design a computational offloading algorithm for cloud-edge collaboration in smart agriculture, in which we first fully consider the real-time, accuracy, dynamics, power consumption, and task volume problems of the whole system. We transform the actual problems of the system into a basic mathematical model and design an adaptive prediction task volume algorithm model. The abstract problem is concretised into the problem of finding the optimal solution to a mixed-integer nonlinear optimisation problem while overcoming the conflicts arising from traditional static modelling. Finally, the defined adaptive function and deep Q-network (DQN) algorithm are used to find optimal solutions, which reduce conflicts dynamically. At the same time, it enhances the real-time and accurate calculation of agricultural terminal tasks. The following tasks are completed in this paper:

1 a task volume prediction model is proposed to address the conflict problem arising from static modelling and to enhance the accuracy of offloading

2 response time and power consumption models are designed to capture the overall computational process of agricultural activities from multiple perspectives

3 the DQN algorithm is employed to solve the mixed-integer nonlinear optimisation problem in the model, achieving improved optimisation results.

Experimental results demonstrate that the proposed algorithm in this paper can significantly improve the accuracy of offloading, as well as the speed and accuracy of cloud-edge collaborative computing.

This paper makes several contributions. Firstly, it proposes a cloud-based collaborative computing offload algorithm for smart agriculture, which addresses the challenges of computational pressure and real-time computational tasks in farm systems. Secondly, it models the computational process of agricultural operations from multiple perspectives to reduce the conflict rate of

offloading computations. Finally, it introduces the DQN algorithm to improve the speed of finding the optimal offloading solution within the proposed model.

The remaining part of the paper is organised as follows: Section 2 briefly introduces some mixed-integer nonlinear optimisation problems for solving algorithms; Section 3 explains in detail the overall process of constructing task volume prediction models, response time models, and power consumption models and using DQN algorithm to find the optimal solution; Section 4 discusses the experimental results; finally, the whole paper is concluded in Section 5 by summarising the whole paper while discussing ideas for the future.

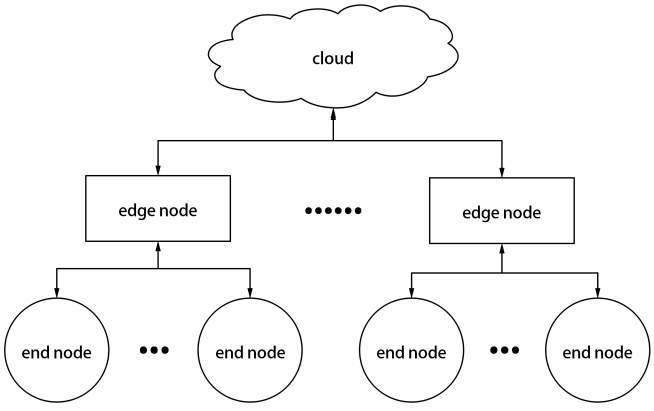
2 Fundamental knowledge

To facilitate a better understanding of the algorithmic model and process presented in this paper, this section introduces and discusses the relevant theories. Firstly, we discuss the cloud-edge collaboration, followed by the computation offloading.

2.1 Cloud-edge collaboration

The term 'cloud-edge collaboration' refers to the collaboration between the cloud and the edge nodes in which some computing tasks are offloaded to the nodes closer to the production environment. By placing these tasks at the edge or end nodes, which are in close proximity to the production environment, the transmission distance is reduced, resulting in lower latency (Zeyu et al., 2020). This facilitates faster computation and enables the system to obtain the required results more quickly, providing critical data support for real-time decision making.

Figure 1 Process of end node task upload



The cloud-edge collaboration framework consists of three parts: the cloud, edge nodes, and end nodes (Zhao et al., 2019), as shown in Figure 1. These nodes are distributed in a tree structure, where multiple end nodes are connected to an edge node, and multiple edge nodes are connected to a cloud. All kinds of nodes have computing capabilities, which means that they can all serve as computing nodes. As this paper's algorithm is focused on the field of smart agriculture, the end nodes in this framework are agricultural terminals (or farm machines), and the tasks generated by these agricultural terminals are referred to as agricultural terminal tasks (or end node tasks).

2.2 Computation offloading

Computation offloading involves transferring computational tasks from one subject to another for processing and then returning the result to the original subject. During the early stages of cloud computing, terminals and edge devices were the primary entities used to offload computations. This involved transmitting computing tasks from terminals or edge devices through networks to the cloud, where the computations were performed. The results of the computations were then sent back to the terminals or edge devices, or directly stored in the cloud.

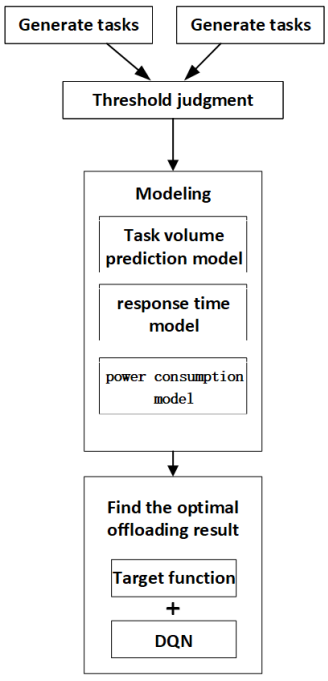
In this paper, computation offloading refers to the partial or complete transfer of computation tasks from end nodes to the edge or cloud for processing.

3 Algorithm and models

3.1 The overall process of algorithm

The overall process of this algorithm is shown in Figure 2. The whole is divided into three major parts, the first part is to upload the task information to the cloud from each agricultural terminal node; the second part is to build the model in the cloud; the third part is to find the optimal solution for the model and to distribute the calculated offloading results.

Figure 2 The overall process of algorithm

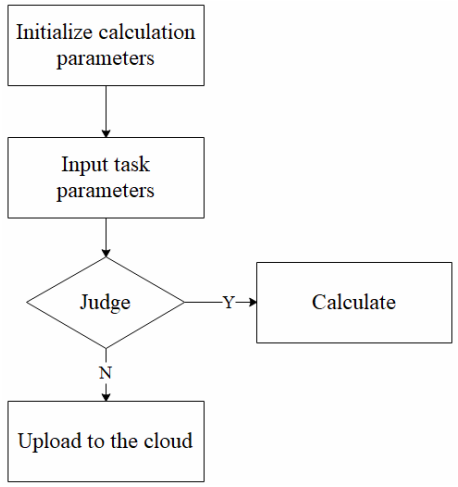


Firstly, each agricultural terminal node generates tasks and uploads them to the cloud for unified distribution if they cannot be judged by local thresholds. The cloud builds three abstract models, namely the task volume prediction model, the response time model, and the power consumption model. After the model is built, the DQN algorithm is used to find the optimal solution for the model through a custom target function. Finally, the optimal computation offloading result is obtained.

3.2 End node task upload

For each agricultural terminal node, the first step of the system is to upload the computing data of each node and the task information generated by the agricultural terminal node to the cloud. To abstract this process, we give the following definition: define a task as i, $i \in \{1, 2, \ldots, n\}$. A task is abstracted into three task parameters $Ta_i\{Da_i, C_i, S_i\}$, where Da denotes the task data size, C denotes the number of CPU cycles required for this task, and S denotes the sensitivity of the task, i.e., the maximum tolerance time of the task; define a computing node (including terminal node, edge node, and cloud) as j, $j \in \{1, 2, \ldots, m\}$, a computational node is abstracted into three task parameters $Cal_j\{M_j, F_j, Me_j\}$, where M denotes the total memory of this computing device, F denotes the main frequency of this computing device and Me denotes the used memory of this computing device. Before the task is uploaded, each agricultural terminal node needs to perform a threshold judgement for the task, as described in Section 3.5 boundary constraints below.

Figure 3 Process of end node task upload



As shown in Figure 3, regarding the part of end node task upload, the details are:

1 terminal nodes initialise their own computational parameters

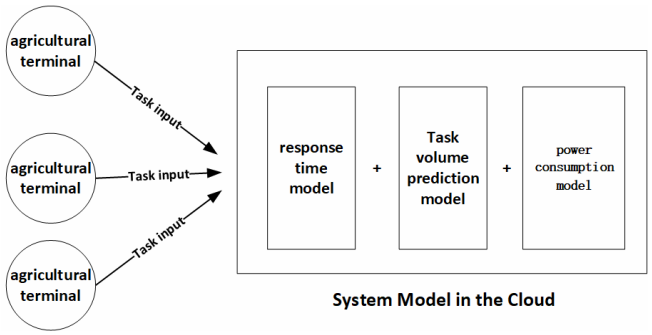
2 each terminal node inputs the relevant task parameters

3 terminal nodes judge the local threshold value of the input task, and if the condition is satisfied, they directly perform local calculation, and if not, they upload the task parameters to the cloud.

3.3 System model building in the cloud

To process the data uploaded by the agricultural terminal node to obtain optimal computation offloading results, we need to build the system model in the cloud according to the environment of agricultural activity. The overall model in the cloud consists of three sub-models, which are the response time model, task volume prediction model, and power consumption model, as shown in Figure 4.

Figure 4 Task input system model in the cloud



After the statistics of each edge node parameter Cal and each task parameters Ta in the cloud, the cloud needs to build the system model. This section specifies the three sub-models that make up the system model.

3.3.1 The response time model construction

To more accurately calculate the time cost of tasks offloading, the response time is modelled as follows:

Define D as the cost of time to transfer data between cloud, edge and end; define D_k^e as the cost of time to transfer data from agricultural terminal nodes to the edge node; define D_k^c as the cost of time to transfer data from agricultural terminal node to the cloud.

Define the channel bandwidth as $B = \{B_k^e, B_k^c\}$, the transmission rate of data can be expressed by Shannon's formula, which is pointed out by Shannon (1948):

$$C = B * \log_2^{1+S/N}(\text{bit/s}) \tag{1}$$

then the general formula for D can be calculated as:

$$D = Da_{\text{i}} \Big/ \left(B * \log_2^{((1+p*h)/(g+w))}\right) \tag{2}$$

where w is the noise power, h is the channel gain between the user equipment and the wideband channel. p is the user uplink transmission power, and g is the interference caused by other tasks in the channel to the task i.

Defining the response time equation Q_i

$$Q_{\text{i}} = u_1 * (D_{\text{k}}^{\text{e}} + C_{\text{i}}/F_{\text{j}}) + u_2 * (D_{\text{k}}^{\text{c}} + C_{\text{i}}/F_{\text{j}}) \tag{3}$$

where u_1 is 1 when there is data transfer to the edge node, and 0 otherwise; u_2 is 1 when there is data transfer to the cloud, and 0 otherwise.

3.3.2 The task volume prediction model construction

To alleviate errors and conflicts in the offloading process and to allow the model to calculate offloading decisions more accurately, a task volume prediction model is added to the model to take advantage of the periodicity and concentrated characteristics of agricultural activities with the basic idea being that historical records predict the future. It makes the model dynamically adjusted for feedback on the outcome of each offloading decision. This task forecasting model is based on the Q-learning algorithm and combined with a custom task forecasting function called Static(t) [this function is given later in equation (6)].

To make the task prediction model adaptive to real-time feedback, we design our T-value by borrowing the Q-value from the Q-learning algorithm (Clifton and Laber, 2020). Similar to the Q-value in the Q-learning algorithm, the T-value is also divided into $the\ realistic\ value\ of\ T$ and $the\ actual\ record\ of\ T$.

Define $the\ realistic\ value\ of\ T$:

$$\begin{cases} T^{\text{real}}\ (S_{n+1}) = R_{n+1} + \gamma * \text{Static}(t) \\ \gamma \in [0,1] \end{cases} \tag{4}$$

where γ is the decay value; R is the feedback value, i.e., the actual effect statistics for each agricultural terminal node; Static(t) is the prediction function presented in equation (6) later. In $the\ realistic\ value\ of\ T$, a prediction is also included. This means that the value of this step not only contains the estimate of the change for the next step but also incorporates the return value of this step.

Define $the\ actual\ record\ of\ T$:

$$T\left(S_{\text{n}+1}\right) = \text{Static(t)} + \alpha * \left[T^{\text{real}}\ \left(S_{\text{n}+1}\right) - \text{Static(t)}\right] \tag{5}$$

$the\ actual\ record\ of\ T$ is defined using cumulative variables for future developments, not just realistic values for the next step, thus enabling forecasting of future task volumes, bringing a base value to the mode, and avoiding the conflicts associated with static modelling.

For the task volume prediction function Static(t), two tables are generated for it: the first table is a queue table recording $the\ actual\ record\ of\ T$ at time t for each day; the other table is a two-dimensional array table with vertical coordinates in years and horizontal coordinates in offsets in days, of length m, used to store $the\ actual\ record\ of\ T$ for the year. For the first table, we use the feedback values and the forecast values based on the statistical values of the second table to generate the T-values to calculate the horizontal forecast; for the second table, we use $the\ actual\ record\ of\ T$ recorded in the first table to average them to calculate the values for our vertical forecast.

Define the basic task volume prediction function Static(t):

$$\alpha_1 * \left(\sum_{\text{i}=1}^{\text{m}} T_{\text{w}}^{\text{kt}}\right) \Big/ m + \alpha_2 * \left(\sum_{\text{j}=1}^{\text{n}-1} T_{\text{v}}^{\text{t}}\right) \Big/ (n-1) \tag{6}$$

where the statistical prediction function at time t is Static(t), T_w^{kt} is the value of the two-dimensional array table with offset k at year w, T_v^t is the value at time t of the queue list. where α_1 and α_2 are the proportionality factors for the value of the longitudinal forecast and the share of the cross-sectional forecast, respectively. Sum of α_1 and α_2 is 1.

For computational convenience, the task volume prediction model is incorporated into the response time model described above, and the response time model incorporated into the predicted task volume model is defined as Q, then Q is defined as follows:

$$\begin{cases} Q = \sum_{i=1}^{n} \left(h_1 * Q_{\text{i}} + h_2 * \text{Static}(t)\right) \\ h_1 + h_2 = 1 \end{cases} \tag{7}$$

where h_1 and h_2 are the proportionality factors for response time and predicted task volume, respectively.

3.3.3 The power consumption model construction

In order to achieve a more result calculation of the resource offload cost, a model was developed for the power consumption as follows:

The power consumption model proposed by Basmadjian et al. (2011) is used here:

$$E_i = P_{\text{idle}}\ + (P_{\text{max}} - P_{\text{idle}}\ ) * \frac{Me_{\text{j}}/M_j}{100} \tag{8}$$

where Me_j is the used memory of the computing device as defined previously, M_j is the total memory of the computing device as defined previously, and P_{max} and P_{idle} indicate the maximum power consumption (100% utilisation) and the idle power consumption (no activity) of the processor respectively.

According to E_i, the overall power consumption of the system can be defined as:

$$E = \sum_{j=1}^{n} \left(P_{\text{idle}}\ + (P_{\text{max}} - P_{\text{idle}}\ ) * \frac{Me_{\text{j}}/M_{\text{j}}}{100}\right) \tag{9}$$

Lo_j is defined as follows:

$$Lo_{\text{j}} = Me_{\text{j}}/M_{\text{j}} * 100\% \tag{10}$$

where Lo_j indicates load factor.

3.4 Calculate the optimal offloading solution

In the course of each iteration, each time the information about agricultural terminal tasks is entered, the optimal offloading decision for this iteration should be obtained. As the system model in the cloud has already been constructed in the previous section, we can use the constructed system model to define a suitable target function and then use the algorithm to find the optimal solution of this target function to obtain the optimal offloading solution for this iteration.

Among these algorithms used to solve the target function, the particle swarm algorithm and the Q-learning algorithm are commonly used, but both algorithms have their drawbacks when applied here, and we finally used the DQN algorithm to solve the target function.

3.4.1 Define the target function

We describe the offloading and resource allocation of the overall system of agricultural activities as an optimisation problem, and the objective of this part is to minimise the total cost of the combination of task volume prediction, response time and power consumption in the system, so the following equation is defined to represent the target function of the solution. Also to ensure that the gap between Q and E is not too large, Define m as critical value.

$$\begin{cases} \text{Target } = Q + E \\ Q/m < E < m * Q \\ Q > 0, E > 0 \end{cases} \tag{11}$$

For the target function, we only need to choose a suitable algorithm to find the optimal solution to the target function. As this target function is a mixed integer nonlinear optimisation problem, it is not possible to obtain the optimal value by derivation, so the DQN algorithm is used to obtain the optimal solution of this function.

3.4.2 DQN algorithm solving

In order to avoid the slow speed problem of the particle swarm algorithm and the tendency to fall into local optima, we used the Q-learning algorithm for the solution. However, the Q-learning algorithm suffers from the dimensional catastrophe problem, and we eventually used the DQN algorithm instead of a particle swarm or Q-learning method to solve the target function.

For the DQN algorithm, we need to consider three key elements, namely $state$, $action$, $reward$ (Long and He, 2020).

- $state$: the system $state$ consists of two components, $state$ = (tc, enc). We define tc(total costs) as the total cost of the whole system, i.e., the optimal solution for our target, $tc = Target$; we define enc(each node capability) as an array representing the remaining computational capacity of each computational node.
- $action$: for the $action$ of the algorithm, which consists of one part: the part is the offloading decision for each, denoted by A, which is defined as $A = [a_1, a_2 \ldots\ldots, a_n]$.
- $reward$: after each iteration, a reward value $R(s, a)$ is generated. For the overall system, our goal is to obtain as small a tc as possible and as large a reward value as possible subject to the boundary constraints. Therefore, we set the reward to be tc negatively correlated. We define the immediate reward as $(-|tc_{local} - tc|/tc_{local})$, the tc_{local} here is the sum of the task volume running costs.

3.5 Boundary constraints

In the process of performing the boundary condition determination of the agricultural terminal and in the process of performing the boundary condition determination of the target function, we need to impose boundary constraints on some of its values. That is, the computation offloading under the system needs to take into account task volume prediction, response time, power consumption, load and boundary conditions. The specific boundary constraints are as follows:

Performing boundary condition judgements at the agricultural terminal, i.e., making S_{ac} (accuracy), S_{lo} (load volume) and S_{lc} (local response time) threshold judgements:

1 For each computational node, the sum of the memory used by the calculation task P_i and the used memory capacity should be less than the total amount available:

$$\mathrm{P_i} + Me_{\mathrm{j}} < M_{\mathrm{j}} \tag{12}$$

2 The pre-load value should be less than the set load value threshold:

$$(\mathrm{P_i} + Me_{\mathrm{j}})/M_{\mathrm{j}} < S_{\mathrm{lo}} \tag{13}$$

3 The expected local response time is less than the local response time threshold set in advance:

$$C_{\mathrm{i}}/F_{\mathrm{j}} < S_{\mathrm{lc}} \tag{14}$$

Performing boundary condition judgements on the target function, i.e., performing S_{last} (time cost calculated from task offload), M_{last} (memory cost calculated from task offload) and S_{lc} (local response time) threshold judgements:

4 For the calculated time cost, the time cost per task needs to be less than the pre-determined sensitivity of each task:

$$S_{\mathrm{last}} < S_{\mathrm{i}} \tag{15}$$

5 For the calculated memory cost, the memory cost per task allocated to a compute node needs to be less than the maximum memory capacity that each compute node can handle:

$$\mathrm{M_j} - Me_{\mathrm{j}} < M_{\mathrm{last}} \tag{16}$$

3.6 Algorithm pseudo-code

The algorithm pseudo-code is as shown in Table 1.

Table 1 The procedure of algorithm

A computation offloading algorithm for cloud-edge collaboration in smart agriculture
Initialise T(s) arbitrarily
Repeat (for each episode):
Initialise calculation parameters S
Repeat (for each step of the episode):
Generation task parameter ta
Generating functions and models
Calculate the optimal solution by DQN
Get solution and feedback value
Calculate T
Update T of two tables
End Repeat
End Repeat

4 Experimental results and discussion

In this section, we present simulation experiments to evaluate the performance of the proposed algorithm.

4.1 Experimental environment

The experiments in this paper were completed in a Python 3.7 environment. The computer used was hosted with an Intel i7-8750H processor, 32 GB of RAM, an RTX2060 graphics card with 12 GB of video memory and Pycharm 2020.3 simulation software.

4.2 Experimental parameter settings

A simulated farm situation is as follows. We use a bandwidth size of B = 10 MHz and deploy an end server (which can be connected directly to the cloud) located at the centre of the farm. The various farm terminals are randomly distributed within 200 metres of the server. The memory M = 64 G, the CPU frequency of each agricultural terminal node is F_n^i= 1 GHz/s, memory M_n^i = 2 G. For the task the calculated data size is Da and its value lies between (300, 500).

4.3 Experimental results and analysis

First of all, we need to determine the values of α_1 and α_2 in equation (6) for the weights of the following prediction functions. Since $\alpha_1 + \alpha_2 = 1$, the independent variable is α_1. In order to facilitate comparison of the advantages and disadvantages between different definitions of α_1 the conflict is defined as when the sum of the input and unload tasks of the farm machine is greater than the computational capacity of the farm machine, and the conflict rate is defined as the number of conflicts divided by the number of agricultural terminals. The impact on the conflict rate was examined by α_1 taking [0.1, 0.9] respectively, and the experimental results are shown in Figure 5.

From Figure 5, it can be seen that the conflict rate is lowest and the model works best when $\alpha_1 = 0.4$, that is, when $\alpha_2 = 0.6$. At this point $\alpha_1 > \alpha_2$, i.e., when the weight of recent behaviour is greater, the predicted conflict rate is lower, which is logical. Therefore, all subsequent experiments took the value of 0.4 for α_1 and 0.6 for α_2.

The experiments in this paragraph compare the advantages and disadvantages of three models for solving the problem of conflicting assignment of agricultural tasks in smart agriculture: FullLocal model, no task volume prediction model and task volume prediction model. Defines a model where tasks are not offloaded and are only computed locally as a FullLocal model (FullLocal); define a system model with only the response time model and the power consumption model and no the task volume prediction model as noPreModel. Define a system model with the task volume prediction, the response time model and the power consumption model as PreModel. Four experiments were carried out, with 200 iterations each, and the results are illustrated in Figure 6.

Figure 5 Influence of α_1 values on conflict rate (see online version for colours)

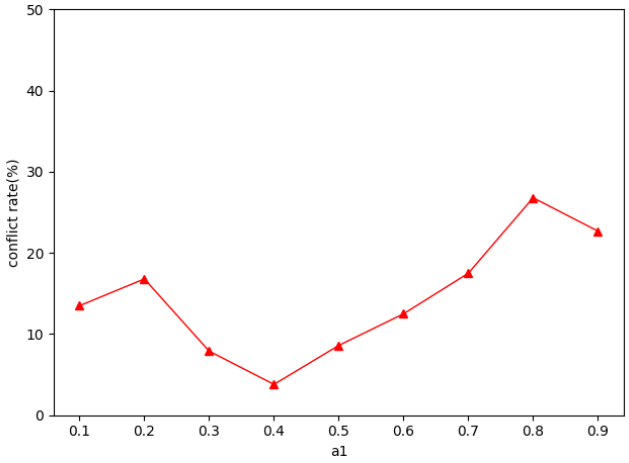


Figure 6 Comparison of the three models corresponding to the conflict rate (see online version for colours)

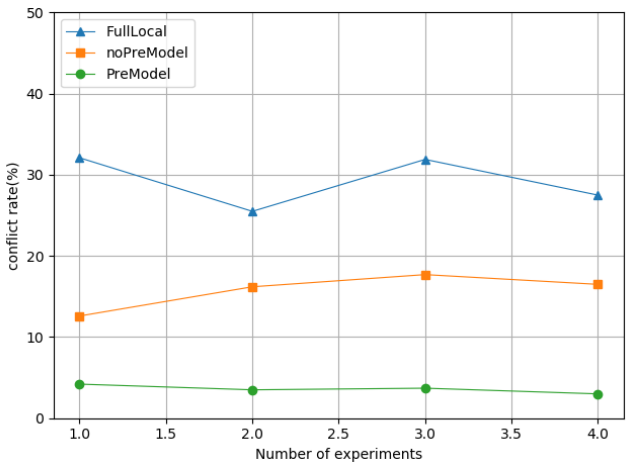


Table 2 Comparison of the three models corresponding to the conflict rate

Number	*FullLocal*		*noPreModel*		*PreModel*	
	Conflict rate	*Variance*	*Conflict rate*	*Variance*	*Conflict rate*	*Variance*
1	32.10%	0.78	12.60%	0.65	4.20%	0.61
2	25.50%	0.72	16.20%	0.69	3.50%	0.6
3	31.90%	0.77	17.70%	0.69	3.70%	0.62
4	27.50%	0.76	16.50%	0.68	3.00%	0.61

Figure 7 Different algorithms corresponding to sum cost (see online version for colours)

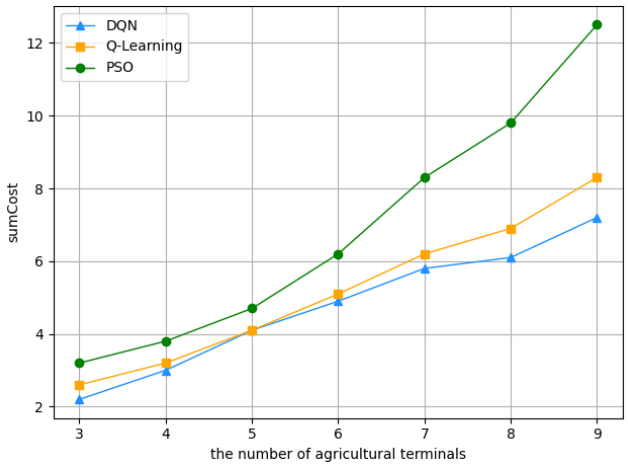
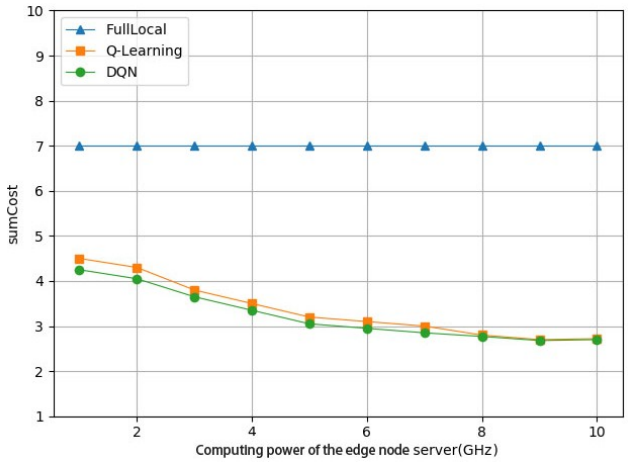


Figure 8 Computing power of the edge node server corresponds to sum cost (see online version for colours)



It can be seen from Table 2 and Figure 6 that for the original FullLocal model, there are frequent cases where its conflict rate remaining between 25% and 30%. This is because the agricultural terminal node may be stopped for a short period of time, when its computational resources are plentiful, and static modelling assigns computational tasks to it. However, when the owner runs the agricultural terminal node, agricultural terminal node will generate tasks. At this time, machine's own computational resources are occupied, resulting in a conflict where the required computational resources are much greater than the computational resources it has; the improved model, whether without or with the predictive model, works much better than the FullLocal model, where the conflict rate with the PreModel is maintained between 3% and 5%, largely avoiding the conflict problem.

In order to compare the advantages and disadvantages of PSO algorithm, Q-learning algorithm, and DQN algorithm for finding optimal solutions, we compare them in terms of the total cost of solving the problem (taking the target value above as the total cost). The model we have jointly selected here is PreModel, the task volume prediction model.

Figure 7 shows a graph of the change in total cost as different algorithms, where the computing power of the edge nodes is set to 5 GHz/s. Overall, the total cost of these algorithms is increasing as the number of agricultural terminals continues to increase. As the number of agricultural terminals in agricultural activities increases, so does the number of tasks, which is why the total cost is increasing. In Figure 7, the DQN algorithm gives the best results for reducing sum cost, followed by Q-learning with a smaller gap. Without considering the dimensional explosion problem of the Q-learning algorithm, the results are similar for both algorithms. The total cost of the PSO algorithm is slightly higher than that of Q-learning algorithm and DQN algorithm when the total number of agricultural terminals is in [3, 6], but the total cost of PSO tends to increase more and more as the number of agricultural terminals increases. This is because the multi-machine computing task is certainly much better than single-machine computing when the number of agricultural terminals increases, so in this case, it is particularly important to choose a strategy for offloading.

Figure 8 shows the relationship between the computing power of the edge node server and sum cost when the number of agricultural terminals is 5. As can be seen from Figure 8, for the FullLocal, it does not change as the computing power of the edge node servers increases, as this approach does not use the cloud-edge computing offload algorithm to distribute tasks to multiple compute nodes. For the Q-learning and DQN algorithms the sum cost decreases as the computing power of the edge node server increases. As the computing power of the edge node servers increases to a certain level, the two algorithms are generally consistent.

5 Conclusions

In this paper, we present a computation offloading algorithm for cloud-edge collaboration in smart agriculture

and provide a detailed explanation of the specific process involved. To address the conflict problem resulting from static modelling, we developed our own T-value inspired by the Q-value designed by the Q-learning algorithm and replaced the original maximum prediction value of Q with a prediction function of our own design. Additionally, we introduced the DQN algorithm to find the optimal solution for the model and expedite the search for the optimal offloading strategy. The experiments show that the algorithm we designed overcomes the conflicts generated by the previous static modelling, improves the accuracy of the computed offload results, and makes the real-time performance of the terminal task computation better; in the experiment of testing the conflict rate, it was shown that the prediction model we designed is effective and greatly reduces the conflict rate of the task offload. In the comparative experiment between FullLocal and computation offloading, the sum cost of computation offloading was much lower than FullLocal, highlighting the superiority of computation offloading. In comparative experiments with other algorithms for finding the optimal solution of the model, the use of the DQN algorithm has a better performance compared to some other algorithms. In the subsequent research, the optimisation of the DQN model can be considered to find optimal unloading solution faster.

References

Ayaz, M., Ammad-Uddin, M., Sharif, Z., Mansour, A. and Aggoune, E.H.M. (2019) 'Internet-of-things (IoT)-based smart agriculture: toward making the fields talk', *IEEE Access*, No. 7, pp.129551–129583.

Basmadjian, R., Ali, N., Niedermeier, F., De Meer, H. and Giuliani, G. (2011) 'A methodology to predict the power consumption of servers in data centres', *Proceedings of the 2nd International Conference on Energy-Efficient Computing and Networking*, May, pp.1–10.

Clifton, J. and Laber, E. (2020) 'Q-learning: theory and applications', *Annual Review of Statistics and Its Application*, No. 71, pp.279–301.

Gao, J., Chang, R., Yang, Z., Huang, Q., Zhao, Y. and Wu, Y. (2022) 'A task offloading algorithm for cloud-edge collaborative system based on Lyapunov optimization', *Cluster Computing*, No. 26, pp.1–12.

Long, Y. and He, H. (2020) 'Robot path planning based on deep reinforcement learning', *2020 IEEE Conference on Telecommunications, Optics and Computer Science (TOCS)*, December, pp.151–154.

Meyer, V., Kirchoff, D.F., Da Silva, M.L. and De Rose, C.A. (2021) 'ML-driven classification scheme for dynamic interference-aware resource scheduling in cloud infrastructures', *Journal of Systems Architecture*.

Paraforos, D.S. and Griepentrog, H.W. (2021) 'Digital farming and field robotics: internet of things, cloud computing, and big data', *Fundamentals of Agricultural and Field Robotics*, No. 1, pp.365–385.

Ren, J., Hou, T., Wang, H., Tian, H., Wei, H., Zheng, H. and Zhang, X. (2021) 'Collaborative task offloading and resource scheduling framework for heterogeneous edge computing', *Wireless Networks*, pp.1–13.

Ruan, J., Jiang, H., Zhu, C., Hu, X., Shi, Y., Liu, T., ... and Chan, F.T.S. (2019) 'Agriculture IoT: emerging trends, cooperation networks, and outlook', *IEEE Wireless Communications*, Vol. 26, No. 6, pp.56–63.

Shannon, C.E. (1948) 'A mathematical theory of communication', *The Bell System Technical Journal*, Vol. 27, No. 3, pp.379–423.

Xuewen, W. and Jingxian, L. (2022) 'Game-based resource allocation and task offloading scheme in collaborative cloud-edge computing system', *Journal of System Simulation*, Vol. 34, No. 7, pp.1468–1481.

Zeyu, H., Geming, X., Zhaohang, W. and Sen, Y. (2020) 'Survey on edge computing security', *2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, June, pp.96–105.

Zhang, M., Wang, F., Zhu, Y., Liu, J. and Wang, Z. (2021) 'Towards cloud-edge collaborative online video analytics with fine-grained serverless pipelines', *Proceedings of the 12th ACM Multimedia Systems Conference*, July, pp.80–93.

Zhang, Z., Wu, J., Jiang, G., Chen, L. and Lam, S.K. (2017) 'QoE-aware task offloading for time constraint mobile applications', *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*, October, pp.510–513.

Zhao, Y., Wang, W., Li, Y., Meixner, C.C., Tornatore, M. and Zhang, J. (2019) 'Edge computing and networking: a survey on infrastructures and applications', *IEEE Access*, pp.101213–101230.

Zhou, B., Huang, H., Xu, Y., Li, X., Gao, H., Chen, T., ... and Xu, J. (2021) 'Parallel task scheduling algorithm based on collaborative device and edge in UAV delivery system', *Jisuanji Jicheng Zhizao Xitong*, Vol. 27, No. 9, pp.2575–2582.