ingegneria sismica
International Journal of Earthquake Engineering
Anno XLIII - Num. 1 - 2026

Design and Reliability Analysis of Power Plant Intelligent Inspection System Based on 5G MEC Technology

Jianbo Liu[1,*]

[1] Guoneng Guohua (Beijing) Gas-Fired Cogeneration Power Co., Ltd., Beijing, 100025, China

SUMMARY: *In this paper, we take the background of intelligent inspection in power plants and face the demand of minimizing the task completion rate and average energy consumption of inspection machines within their inspection scenarios. 5G MEC technology is utilized to construct an inter-node migration model for MEC to coordinate the coupling of policy formulation of multiple inspection machines. As a solution to the problem of unbalanced loading and underutilization of the edge server for deployment in inspection machine implementation, this research proposes the enhancement of Top-K algorithm. The enhanced algorithm considers base stations with the highest number of tasks for edge server deployment and ensures coverage of all the base stations by the edge server. In the offloading problem, the GT offloading decision is proposed by integrating game theory and reinforcement learning, and the GT offloading decision scheme is implemented using a decentralized algorithm to establish a distributed offloading algorithm. In addition, the SLA-based reinforcement learning method is used to promote the effective connection between inspection machines and edge servers to form the task offloading strategy of multiple inspection machines-multiple edge servers under the non-cooperative game.Under the task offloading algorithm proposed by this research, the experimental inspection system operates at the transmission delay and energy cost range of 151.19 to 1400.00 depending on the increasing number of intelligent terminals ranging between 150 and 550., and the ratio of idle CPU is only 13.40 to 67.36 during this period. The inspection system supported by the method in this paper achieves high resource utilization at a lower cost and improves the training effect while reducing the inspection cost.*

KEYWORDS: *intelligent inspection of power plants; GT offloading decision; improved Top-K algorithm; inter-node migration model; 5G MEC*

1 Introduction

Safe and stable operations of equipment in the power plant are a crucial basis in guaranteeing the safety of power plant production. In the past few years, the magnitude of power production has kept expanding, the complexity and number of power plant equipment has increased exponentially, which requires more detailed work of the power plant testing department [1-3]. For the power industry, the availability of monitoring information is far from adequate. The ability to communicate between individuals and organizations is still weak, and the information silo effect is a prominent problem of the management system, which lacks inter-departmental communication and information sharing, and many data

*liujianbo88882025@163.com
https://doi.org/10.65102/is2026481
Published: 30 April, 2026

cannot accurately restore the real situation [4]. Conventional power plant inspection mostly adopts manual inspection method, which often consumes too much manpower in the implementation process and is not efficient. Relative to electronic inspection tools, paper inspection records are often not easy to record and query, the process is inefficient, due to the error-prone information transfer process, inspection information can not be reported in time, and can not meet the requirements of work efficiency; drone inspection method is based on the latest pattern recognition analysis technology to analyze the collected on-site image data, but from the point of view of the current technical conditions, the pattern recognition algorithms have to be improved [5-9]. In order to solve the issue above, 5G technology, MEC technology, as well as the combination of 5G and MEC have been applied in power plant inspection applications increasingly.

With the help of 5G technology, data can be transmitted quickly and processed promptly, which helps accelerate the response speed of the whole system. According to the work of Zhou et al. [10], 5G technology and AI technology were merged to form a fully automated 3D inspection technique for wind power production using UAVs. Hua et al [11] developed a 5G+ robotic autonomous inspection system in smart power plants and introduced intelligent image processing technology to process and analyze multiple state information of power plant equipment to comprehensively reflect the real situation of power plants. Chuang et al. [12] presented a data analysis architecture for real-time surveillance and intelligent optimization of the production processes of power plants, using the 5G infrastructure, a hybrid dominant behavior and reward neural network model, and a two-level deep Q-network. This architecture enables effective surveillance of power plants along with optimal allocation of resources. Huang et al. [13] noted that smart power plants using 5G technology can create an IoT network covering all areas and fitting the conditions of power plants. The IoT network can incorporate the demands of power plant communication, power production, and grid systems, thus improving the safety monitoring and intelligent diagnostic performance. While Zanzi et al [14] added MEC to the IoT ecosystem to support enhanced IoT deployments, reduce operational latency on IoT systems, and increase computational power for IoT system performance optimization.

MEC technology is based on the network topology information between devices and edge nodes, combined with real-time network conditions, to dynamically direct device requests to the most appropriate edge servers for low-latency data processing. Li et al. [15] introduced a real-time surface defects detection model based on the MEC technology and transfer learning, and a spatial-channel attention mechanism, which is capable of providing equipment surface subtle defects in industrial environments with high-accuracy and low-latency detection services. Krishnan et al [16] utilized software-defined networking to improve the MEC technology and created an infrastructure distributed denial-of-service cyberattack threat monitoring and security framework, with up to 96% detection accuracy for different cyberattack intensities. Gradually, researchers are combining 5G technology with MEC technology for the power plant sector. Rekoputra et al [17] showed that the inclusion of MEC technology in 5G networks can achieve superior network feasibility and efficiency and help in the implementation of smart factories as compared to MEC technology. The researchers Wang et al. [18] employed 5G-MEC technology to enable robot detection and developed a smart computing offloading mechanism. The model enhances the speed and efficiency of robot detection when operating in power system applications, without compromising its energy costs and delays.

In this paper, we first propose a multi-training robot computation migration network structure, design its local computation model, migration computation model, and MEC inter-node model in turn, mathematically model the average energy minimization problem,

and form the MEC inter-node migration model. Afterwards, an improved Top-K algorithm is chosen as the basis for designing the deployment plan for edge servers. After that, based on the principles of game theory, the decentralized algorithm is used as the execution carrier to establish the GT offloading decision in the multi-inspection machine scenario. And based on the SLA reinforcement learning method, it builds the environment interaction system between SLA and multi-inspection machine-multi-edge server to realize the optimal selection of edge server by inspection machine. Lastly, the effectiveness of the edge server deployment plan using the improved Top-K algorithm is tested by comparing the model with other related methods in terms of minimum access delay and network reliability. Additionally, the efficiency of the task offloading method is proven by evaluating its average delay and residual energy performance under various environmental conditions. Moreover, intelligent inspection simulations are carried out to determine the total cost of the task offloading technique for different weighted cost rates, the delay rate during the data transfer process, and the energy cost associated with the use of idle computing resources.

2 MEC inter-node migration model and problem description

2.1 Network model

This section presents the network model. Let $\mathcal{D}=\{D_1, D_2, D_3 \cdots D_n\}$ denote the set comprising all n inspection robots, and $\mathcal{S}=\{S_1, S_2, S_3 \cdots S_m\}$ denote the set of all MEC service nodes. The architecture of the multi-inspection robot computational migration network is illustrated in Fig. 1. Within this architecture, MEC service nodes supply both computational resources and wireless resources to support the inspection robots. Beyond these shared service nodes, a dedicated MEC control node governs information sharing and algorithm scheduling across the entire system.

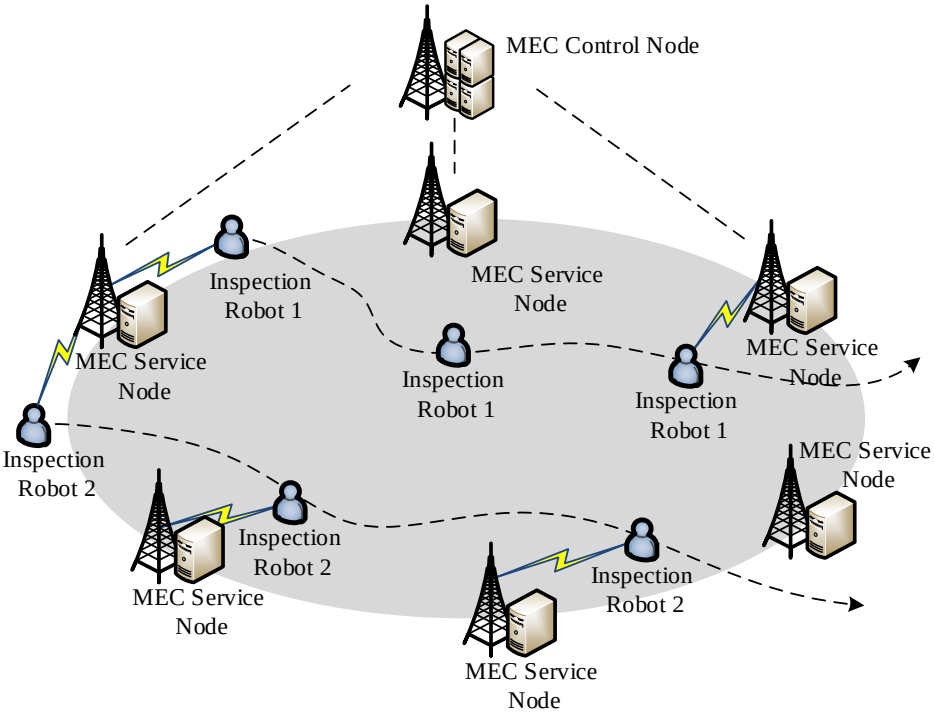

Figure 1: The computational migration network structure of multiple inspection robots

All inspection robots perform inspection tasks periodically, assuming that the task data generated by inspection robot D_i in a certain cycle is expressed as a quaternion $t_i = \{I_i, d_i, O_i, C_i\}$, $I_i\left(bit\right)$ denotes the amount of data for this task, $d_i\left(cycles/bit\right)$ denotes the computational density of this subtask, O_i (bit) denotes the amount of output data of this task, and $C_i\left(s\right)$ denotes the latency threshold of this task. Before starting the migration, each inspection robot will formulate its own migration strategy based on the service node information sent by the control node. Let the inspection path of D_i be $Path_i = \{LR_{i,k} \mid LR_{i,k} = (x_{i,k}, y_{i,k}), k \in \varphi\}$, and the speed be V_i.

2.2 Local computational models

Let the computational power of the inspection robot D_i be $f_i^l\left(cycle/s\right)$. The computational delay of the locally executed task is equation (1):

$$T_i^l = \frac{I_i \times d_i}{f_i^l} \tag{1}$$

Let the power for local computation be P_i^l, then the energy consumption for local computation is equation (2):

$$E_i^l = T_i^l \times P_i^l \tag{2}$$

2.3 Migration calculation model

When a MEC service node processes assigned tasks, the energy consumption of an inspection robot can be broken down into two components: upload energy consumption and idle time energy consumption. The constituent elements of delay and energy consumption follow the same structure. Idle time refers specifically to the period during which a task is being computed on the MEC service node or transferred between MEC service nodes. Given that the transmission power p_i^{tra} of inspection robot i remains constant, and that the wireless communication between the robot and the base station employs orthogonal frequency division multiple access technology, simultaneous transmissions from multiple inspection robots do not cause mutual interference. Based on Shannon's formula, the uplink data transmission rate between inspection robot D_i and MEC service node S_j can be expressed as equation (3):

$$R_{ij} = \frac{W}{N} \times \log_2\left(1 + \frac{p_i^{tra} H_{ij}}{\sigma^2}\right) \tag{3}$$

where W is the total uplink channel bandwidth, and N is the number of inspection robots transmitting tasks to the MEC service node S_j. $H_{ij} = d_{ij}^{-\alpha}$ is the channel gain and σ^2 is the channel noise power. Then the transmission energy consumption and transmission delay when the inspection robot D_i uploads a task are equal to equations (4)-(5):

$$T_{i,j}^{up} = \frac{I_i}{R_{ij}} \tag{4}$$

$$E_{ij}^{up} = T_{i,j}^{up} \times p_i^{tra} \tag{5}$$

Let F_j^e be the computational capacity of MEC service node j. The computational delay of the task processed on this node is then given by equation (6):

$$T_{ij}^{exe} = \frac{I_i \times d_i}{f_j^e} \tag{6}$$

where f_j^e is the computational power that inspection robot i can share, $f_j^e = F_j^e / K_j$, and K_j is the computational load of MEC service node S_j, i.e., the number of simultaneous computational tasks in a certain cycle. Let p_i^f be the idle power of inspection robot D_i. The energy consumption of the inspection robot for task migration calculation is equation (7):

$$E_{ij}^{exe} = T_{ij}^{exe} \times p_i^f \tag{7}$$

2.4 Migration model between MEC nodes

When the robot receives the results, there might be situations where the robot will be outside the area covered by its original MEC service node, or the area where it can enter the area covered by another new MEC service node, making data migration essential. There are two ways of data migration, which can migrate the calculation results between MEC service nodes or migrate the tasks, i.e., virtual machine migration. It is worth noting that in the whole industrial inspection area, the coverage of MEC service nodes is dead-end, in order for this to happen, the robot must always be within the service area of at least one MEC service node at any time. This leads to the classification of the entire data migration process in three different cases depending on the location of the robot when uploading tasks and receiving results:

(1) During the period of transmitting tasks until the robot receives the results, the robot does not leave the area covered by the original MEC service node and enters the area of the new MEC service node; therefore, no data migration takes place and its migration cost is zero.

(2) During the period of transmitting tasks until the robot receives the results, the robot stays within the area covered by the original MEC service node, but when it receives the results, it enters the area of another new MEC service node.

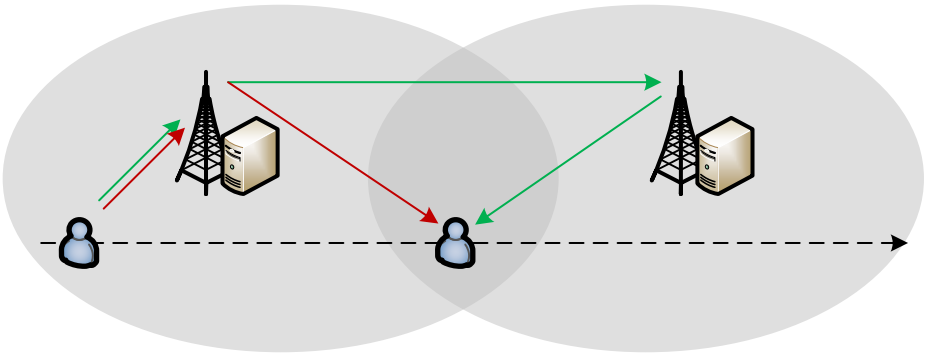

Figure 2: Migration Model between MEC Nodes

At this point, the inspection robot has three options:

1) There is no VM migration and no result migration required here, similar to case 1).

2) The task itself is entirely migrated to the target service node and represents VM

migration. The migration delay and energy consumed during such migration are determined by formulas (8) and (9):

$$T_i^{vm} = \frac{I_i}{v_b} \tag{8}$$

$$E_i^{vm} = T_i^{vm} \times P_i^f \tag{9}$$

v_b is the inter-node transmission rate.

(3) The task computation occurs on the original service node, followed by migration of the result to the target service node, after which the latter sends the result to the inspection robot. Delay and energy required for migration can be estimated using formulas (10) and (11):

$$T_i^{res} = \frac{O_i}{v_b} \tag{10}$$

$$E_i^{res} = T_i^{res} \times P_i^f \tag{11}$$

(4) At the moment the results are received, the inspection robot is already outside the coverage area of the original MEC service node and within the coverage area of another node. Two solutions exist to guarantee that the results are received properly: (1) VM migration, the migration delay is the same as the energy consumption and the 2) method in case (2). (2) Result migration with the same migration latency as the 3) method in energy consumption and situation (2).

2.5 Description of the problem

In this chapter, only one task migration per cycle is considered, and each inspection robot requires at most two MEC service nodes throughout the computation and migration process. One is the target MEC node designated at the time of upload, and the other is the MEC node responsible for carrying out the actual task computation. Let $a_i \in [0, m]$ and $b_i \in [0, m]$ denote the two decision variables of inspection robot D_i within a given migration cycle, where a_i represents the decision of inspection robot i to select a target service node for uploading, and b_i represents the MEC service node on which the task of inspection robot i is ultimately computed. Here, $a_i = 0$ indicates that the inspection robot opts to perform the task computation locally. When no VM migration takes place, the uploaded target MEC node is the same node on which the task is computed, giving $a_i = b_i$. If $a_i \neq b_i$, VM migration occurs. Incorporating the MEC inter-node migration model, the delay and energy consumption associated with inter-node migration can be derived from equations (12) and (13):

$$T_i^{migrate} = I_{\{a_i = b_i\}} \times T_i^{res} + I_{\{a_i \neq b_i\}} \times T_i^{VM} \tag{12}$$

$$E_i^{migrate} = I_{\{a_i = b_i\}} \times E_i^{res} + I_{\{a_i \neq b_i\}} \times E_i^{VM} \tag{13}$$

The objective of this chapter is to minimize the average energy consumption of all inspection robots within a single cycle. To this end, the energy consumption expression for

each individual inspection robot must first be derived. Prior to obtaining this expression, the locations of the inspection robots at the moments of task upload and result reception need to be determined. Since distance serves as a limiting condition governing the selection of MEC service nodes by the inspection robots, it also constitutes a significant factor influencing the transmission rate. Assume that the coordinates of inspection robot D_i at the beginning of the cycle are $LR_{i,h} = (x_{i,h}, y_{i,h}), i \in n, h \in \varphi$, at this time, D_i uploads the task, and due to the short uploading latency, the change of the inspection robot's position can be ignored, and the time elapsed by inspection robot from the upload to the receiving of the result is equation (14):

$$\sum_{j=1}^{m} I_{\{b_i=f\}} \times T_{i,j}^{exe} + T_i^{migrate} \tag{14}$$

By incorporating the moving speed V_i of the inspection robot, the coordinate system grid edge length a, and the number of grids num_i traversed by inspection robot D_i from the end of uploading to the reception of the result, the corresponding displacement can be calculated as in equation (15):

$$num_i = \frac{\left(\sum_{j=1}^{m} I_{\{b_i=f\}} \times T_{i,j}^{exe} + T_i^{migrate}\right) \times V_i}{a} \tag{15}$$

Get the position $LR_{i,k'} = (x_{i,k} + num_i, y_{i,k} + num_i)$ at the time of accepting the result.

By the same approach, the position of the inspection robot at the start of the next cycle can be determined, given that each inspection cycle is of equal duration. The energy consumption and delay incurred by inspection robot i in accomplishing its task through computational migration within a single cycle are then obtained as expressed in equations (16) and (17):

$$T_i^m = \sum_{j=1}^{m} \left(I_{\{a_i=j\}} \times T_{i,j}^{up} + I_{\{b_i=j\}} \times T_{i,j}^{exe} \right) + T_i^{migrate} \tag{16}$$

$$E_i^m = \sum_{j=1}^{m} \left(I_{\{a_i=j\}} \times E_{i,j}^{up} + I_{\{b_i=j\}} \times E_{i,j}^{exe} \right) + E_i^{migrate} \tag{17}$$

In accordance with the objective established in this chapter, namely minimizing the average energy consumption of all inspection robots when processing tasks across each cycle, the problem is modeled as the following Eqs. (18)-(23), using $a_i, b_i \in [0, m]$ to denote the set of decisions for all inspection robots:

$$\min_{a_i, b_i} \left(I_{\{a_i=0\}} \times E_i^l + I_{\{a_i \neq 0\}} \times E_i^m \right) \tag{18}$$

$$\sum_{j=1}^{m} I_{\{a_i=j\}} \times d_{i,j} \leq R_j \quad i \in n, j \in m \tag{19}$$

$$\sum_{j=1}^{m} I_{\{b_i=j\}} \times d_{i,j} \leq R_j \quad i \in n, j \in m \tag{20}$$

$$\sum_{j=1}^{m} I_{\{a_i=j\}} \leq 1 \quad i \in n, j \in m \tag{21}$$

$$\sum_{j=1}^{m} I_{\{b_i=j\}} \leq 1 \quad i \in n, j \in m \tag{22}$$

$$T_i^m < C_i \quad i \in n \tag{23}$$

These equations (19) and (20) ensure that the selection of the MEC service nodes will not be made outside of their corresponding service coverage regions. These equations (21) and (22) indicate that task migration can be carried out towards one particular service node at any point in time, without being migrated toward multiple service nodes simultaneously. This equation (23) indicates that the overall time taken for the completion of the task must be less than a given delay limit.

3 Edge server deployment scheme based on improved Top-K

3.1 Algorithmic principles

The Top-K algorithm is essentially a search and sort algorithm which helps in finding out the largest K elements from a given set. Conventionally speaking, the Top-K algorithm has found widespread applicability in domains like information retrieval, data mining, and database management. In the case of server deployment, the basic idea of the Top-K algorithm is to select K base stations that have the largest task volume as server locations, after clustering them based on the server coverage range and computational threshold. However, conventional Top-K algorithm when implemented for server deployment purposes often suffers from server load imbalance due to irregularly distributed tasks.

Therefore, some changes to the Top-K algorithm include:

Step 1: Characterize the workload of each base station quantitatively.

Step 2: Allocate one server to the sole base station with the greatest workload, discovered via the top-1 approach.

Step 3: Classify the base stations according to the computational capability of the allocated server and its coverage region.

Step 4: Upon classifying the base stations into clusters, discard all base stations within the server's coverage region. Repeat steps 1, 2, and 3 iteratively until all base stations are covered by an edge server. The number of iterations required to complete this process is represented by K.

Step 5: Determine the weight ratio of each base station relative to its cluster of base stations.

Step 6: Treat each base station from step 5 as the potential location of server deployment and compute the distances between the candidate base station and every other base station within the cluster. Multiply the calculated distances by the weight ratios of each base station from step 5, and compute their sums.

Step 7: Select the base station with the smallest sum from step 6 as the final server deployment location.

3.2 Algorithm design

This chapter considers that the inspection robots will upload the task volumes to the server through the base station cluster, and the amount of tasks uploaded can have random fluctuations. The process of deployment with the help of the improved Top-K algorithm can be divided into three phases: Phase I: Initialize the base station cluster according to the

improved Top-K algorithm; Phase II: Optimize the base station cluster; Phase III: Plan the location of the edge server for deployment.

(1) Initialize the base station cluster according to the improved Top-K algorithm.

1)The base station clusters are randomly generated within the simulation experiment domain. Then, the task volumes uploaded by the inspection robots to the server through the base station can be obtained from the historical data.

2) Deploy edge server s_j at base station b_i with the largest task volume. Whether the condition is satisfied is then determined by locating the nearest base station b_j to base station b_i, calculating the distance between the two, and summing their respective task volumes. If the condition is met, base station b_j is assigned to edge server s_j, and base station b_j is subsequently blocked.

3) Repeat step 2) until none of the conditions can be satisfied.

4) Classify all base stations that meet the conditions in step 3) as a base station cluster, and denote the cluster as $c_j = \{c_1, c_2, ..., c_k\}, k \le n$, where the total number of base station clusters is n_i.

5) Block all base stations within base station cluster c_j and repeat steps 2) through 4) until every base station is covered by an edge server.

(2) Cluster optimization of base stations

The algorithm for optimizing the cluster of base stations modifies the size of the cluster by assigning the farthest base station to another cluster. There are two reasons behind this action; first, the algorithm attempts to balance the distribution of tasks among the servers. Second, the algorithm tries to minimize the latency caused by transmitting data from inspection robots to the server via the farthest base station in the cluster.

1) Identify base station b_{cj} within base station cluster $c_j, j \in [1, k]$ that is located furthest from edge server $c_j, j \in [1, k]$, and denote the distance between b_{cj} and s_j as $dis(b_{cj}, s_j)$.

2) Calculate the distance from base station b_{cj} to edge server $s_m, (m \ne j)$, denoted as $dis(b_{cj}, s_m)$. If $dis(b_{cj}, s_m) < dis(b_{cj}, s_j)$ and the computational capacity of edge server s_m satisfies $dis(b_{cj}, s_m) < dis(b_{cj}, s_j)$, then base station b_{cj} is reassigned to the cluster of base stations in which edge server b_{cj} resides.

3) Repeat steps 1) and 2) until Pareto optimality is reached.

(3) Algorithm for final positioning of edge server

The algorithm for final positioning of the edge server is developed to minimize the uploading latency associated with each base station in the cluster. Assuming that the deployment of the server has been done using the optimized Top-K algorithm, the position of the server in each cluster is refined based on the calculation of the summation obtained by multiplying the weight ratio of the base station in the cluster by its distance from the server, following a row-minimum method.

1) Calculate the distance between base station b_i, where the server is deployed, and every other base station $b_i, i \ne l$. Then compute the weight ratio occupied by base station b_i within base station cluster c_j, and accumulate the products of each weight ratio and its corresponding distance to obtain the cumulative sum dis_{gj}.

2) Deploy the server at each base station within cluster c_j other than b_i, and repeat step 1) to calculate dis_{gj} for each alternative base station.

3) · Repeat steps 1) and 2) across all base stations contained in the base station cluster, computing p values of dis_{sj}, where p denotes the total number of base stations in the cluster. This process is carried out to identify Deploy the edge server at the base station where $dis_{gj}^{\min}$ is located..

4) Deploy the edge server at the base station where $dis_{gj}^{\min}$ is located.

3.3 Algorithm analysis

(1) The time complexity of the improved Top-K algorithm is $O\left(km\log n+mn^2+n^2\right)$.

During the initialization of the base station cluster using the improved Top-K algorithm, one base station is selected to deploy an edge server, and the first m base stations nearest to that base station are drawn from all base stations. These $m+1$ base stations collectively form a base station cluster, a procedure that carries a time complexity of $O(n\log n)$. Since k edge servers must be placed in total, the time complexity upon completion of this stage reaches $O(km\log n)$. When the base station cluster optimization algorithm is executed, the base station located furthest from the server within each cluster is reassigned to another cluster following the clustering procedure, giving a time complexity of $O(n^2)$. The edge server location planning algorithm then performs fine-tuning of server deployment positions. Within each base station cluster, it computes the cumulative sum of the products of the weight ratios of individual base stations and their respective distances, subsequently identifying the minimum cumulative sum. This step carries a time complexity of $O(mn^2)$. The total time complexity of the algorithm is therefore $O\left(km\log n+(m+1)n^2\right)$.

(2) Edge server deployment is an NP-hard problem.

Proof: The mapping between the server deployment approach and the K-median problem is done. As the K-median problem is NP-hard, it is proved that edge server deployment is also NP-hard through this mapping. In K-median, there is the set of clients $F=\{F_1,F_2,...,F_n\}$ and the set of devices $C=\{C_1,C_2,...,C_k\}$, devices provide services to customers, a customer only needs one device and a device can provide services to multiple customers. Given a network topology graph $G'=(V',E')$, where $V'=F\cup C=\{V_1,V_2,...,V_{k+n}\}$, E' denotes the association between a device and a customer, there is a customer at $V_i\in V'$, and there is a device at $V_j\in C$, and if $V_i=F_i\in F,V_j=C_j\in C$, the device V_i serves the customer V_j for a cost of d_{ij}, the K-median problem is to place k devices at kk k locations in V' to minimize the total service cost of the device providing cost to the customer.

By drawing an analogy between the MEC edge server deployment problem and the K-median problem, construct $G=(B\cup S,E)$ from G', where $B=\{b_1,b_2,...,b_n\}=V',S=\{s_1,s_2,...,s_k\}\in V'$, and $E=E'$. The edge server occupies the same location as the base station, and since $n>k$, we have $V'=\{V_1,V_2,...,V_n\}$. Base station $b_i\in V'$ is deployed at V_i, and server $s_j\in V'$. Each base station belongs to one and only one server, and each server serves multiple base stations. This chapter considers deploying k servers across n base stations to maximize server utilization and achieve load balancing, based on server coverage and computational capacity. The optimal solution G for edge server deployment is equally the optimal solution G' for the K-median problem, which is an NP-hard problem.

4 Multi-Patrol Machine-Multi-Edge Server Task Offload Policy

4.1 GT offloading decisions

This chapter will analyze the development of a highly effective distributed algorithm used for offloading computing tasks onto edge servers or base stations that are near. Every local decision taken by one single device directly affects the performance of other patrolling machines running task applications within the same network. When several machines decide to use the exact offloading technique within the same network, a decrease in the network throughput leads to increased transmission delays. In addition, as long as the rate of transferring data is small, more energy is needed for the process of offloading data. To solve this challenge, a distributed algorithm developed based on the game theory model will be presented in this chapter.

4.1.1 GT-based offloading strategy

The main reason for choosing game theory as a research method is that each network node may have different requirements and pursue different interests, so this chapter proposes a distributed strategy where each inspection machine chooses the best strategy to achieve its goal. In game theory, a solution is required where all participants are mutually satisfied and no participant has an incentive to change strategies individually, this combination of strategies is known as a Nash equilibrium.

A non-cooperative game in the form of a strategy called $\mu(D,A,G)$ is used; where D is the number of inspection machines, A is the inspection machine execution strategy, G is the global cost function, and α is the strategy that is decided to be executed by the inspection machine μ_i, which consists of local execution, offloading to the edge servers and base stations. The utility function G_j generated by the inspection machine in executing the above policies is expressed as in equation (24):

$$G_j(a_i,a_{-i})=\begin{cases} Z_{Local}=\alpha E_{Local}+\beta T_{Local}, & if A_i=0 \\ Z_{Server}=\alpha E_{Server}+\beta T_{Server}, & if A_i=1 \end{cases} \tag{24}$$

where A_i is defined as $A_i=\{a_i,\forall i\}$ (0 for local execution, 1 for offloading to the MEC server); and a_{-i} denotes the execution policy of the inspection machines other than the inspection machine d_i.

4.1.2 Nash equilibrium

In order to reach the Nash equilibrium in a non-cooperative game, players should have a mutual agreement. This state of the game can be termed as the Nash equilibrium. This means that none of the participants can unilaterally change their strategy to improve their utility function. The goal of the game in this chapter, then, is to determine the state in which the participants (μ_j,μ_{-j}) can converge to a Nash equilibrium. This stable state is an optimal strategy, which is chosen by the participants, and can significantly reduce costs while achieving the computational task.

This chapter utilizes the concept of a potential game to prove the existence and ultimately the convergence of the game. Since, the potential game has at least one solution for NE, with

the help of a potential game, it is possible to prove whether the offloading strategy of this chapter achieves NE. The search of the Nash equilibrium state can be viewed as finding the maximum of a certain function, which is called the potential function.

Argument: $\mu(D,A,G)$ is a potential game

Proof: if the gain function of a game is represented as a potential function as in equation (25):

$$\begin{aligned}&\varphi(a_i,a_{-i})-\varphi(a_i',a_{-i}')=G_j(a_i,a_{-i})-G_j(a_i',a_{-i}')\\&\varphi(a_i,a_{-i})=\arg\min_{a\in A_j}G_j(a_i,a_{-i})=\psi_i(a_{-i})\\&\varphi(a_i',a_{-i}')=\psi_i(a_{-i}')\end{aligned}\tag{25}$$

where $\psi_i(a_{-i})$ is the optimal payoff of the participant μ_j in executing the strategy a_{-i}, which can also be expressed as equation (26):

$$\psi_i(a_{-i})=\begin{cases}\arg\min_{a\in A_j}Z_{Local}, & ifA_i=0\\ \arg\min_{a\in A_j}Z_{Server}, & ifA_i=1\end{cases}\tag{26}$$

Since equation (26) satisfies the definition of a potential function, which is the unique solution that ensures an optimal balance between low overhead and accomplishing the requested task, $\mu(D,A,G)$ constitutes a potential game. The Nash equilibrium solution is therefore unique and equals $\psi_i(a_{-i})$. In the actual experiment, each participant selects from two candidate values, $\arg\min_{a\in A_j}Z_{Local}$ and $\arg\min_{a\in A_j}Z_{Server}$, choosing between these two possible values to determine the final decision.

4.1.3 Offloading algorithms

It can be seen clearly from the above analysis that once the equilibrium is attained, all the decisions made by the players will become constant. However, there should be a decentralized algorithm that will be used in the implementation of the distributed computing offloading model discussed in this chapter so that eventually, patrolling machines will reach a point where they are satisfied with each other.

The algorithm for the distribution offloading suggested in this paper has been illustrated in Fig. 3. The basic principle behind the offloading algorithm can be understood through the concept of convergence that can be achieved from the Nash equilibrium theorem adopted in this model. In such a way, only a limited number of iterations will be sufficient to achieve the aforementioned steady state of the platform. The decision making process will take place simultaneously in all the nodes within the network before the task is performed. Similar algorithms have been adopted in the areas of cloud computing and mobile cloud computing before. Since all the players will perform their actions in a simultaneous and selfish manner, it is imperative that an efficient messaging protocol be adopted in this chapter. It is important to emphasize that only one update request can be approved at a time after the acknowledgement message in each iteration.

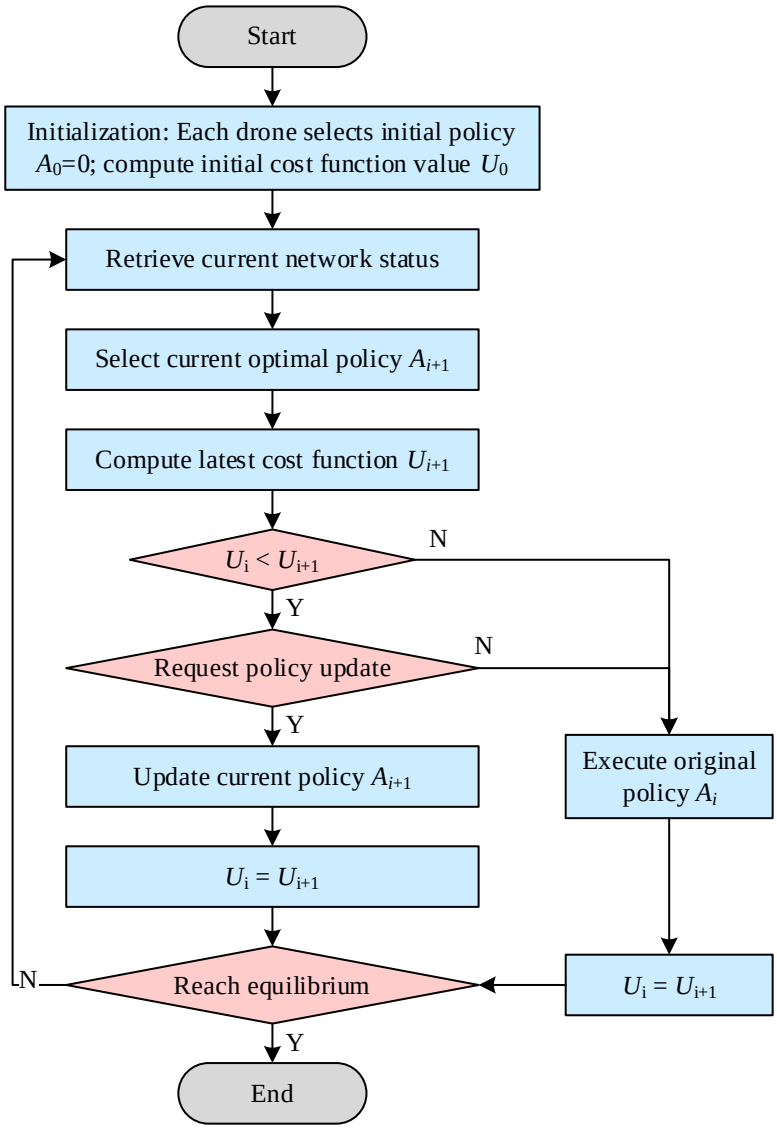

Figure 3: Distributed Offload Algorithm

4.2 Reinforcement learning based MEC server selection

The inspection machine's offloading strategy consists of two main parts: (1) which offloading method the inspection machine chooses, which is determined by a distributed offloading algorithm specified by the concept of a non-cooperative game, and (2) the inspection machine's choice of edge servers, which is solved in this chapter using reinforcement learning. The inspection machine's choice of edge service likewise has a large impact on the latency and energy consumption of the offloading task.

SLA belongs to the category of reinforcement learning, which mainly solves stochastic optimization problems.SLA is an automaton that changes its behavior by interacting with the stochastic environment. At a certain moment, SLA selects a behavior from the data set according to a certain probability based on the current state, which enters the stochastic environment, and the stochastic environment gives feedback on this behavior, and SLA receives the feedback and modifies the state until it selects a suitable behavior. Each inspection machine acts as a stochastic learning machine, and by interacting with the environment, it intelligently measures the computational power and relative distance of each

edge server and the QoS provided by the server for the inspection machine, and selects an optimal MEC server to offload the computational tasks. Figure 4 illustrates the interaction between the SLA and the environment. Assume that each inspection machine has a set of actions in each operation gap $\alpha(t)=\{\alpha_1,\alpha_2...\alpha_j\}$, which represents the decision made by the inspection machine and outputs the determination set $\beta(t)=(r^{(t)},d^{(t)})$ through the interaction with the environment, where $r^{(t)}$ denotes the reputation scores of the edge servers, and $d^{(t)}$ denotes the distance of the inspection machine from the edge servers.

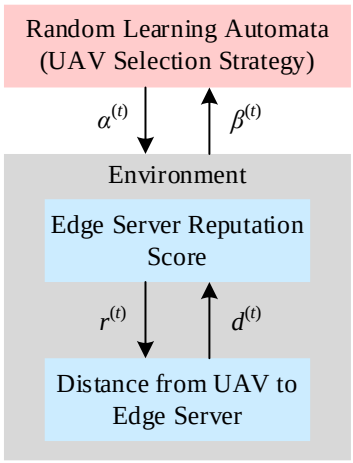

Figure 4: An Interactive System for SLA and Environments

Each MEC server will have a reputation score. Reputation scores increase due to an increase in the relative computational capacity of the MEC server, the rise in the utility of the served user, and decrease in the relative distance between the MEC server and the user. Reputation scores of the MEC server can be mathematically represented using the following formula (27):

$$r_s=\left(\frac{F_s}{\sum_{s\in S}F_s}\frac{\sum_{\forall d\in D}U_{d,sd}}{\sum_{s\in S}\sum_{\forall d\in D}U_{d,sd}}\right)\Bigg/\left(\frac{\sum_{\forall d\in D}d_{d,sd}}{\sum_{s\in S}\sum_{\forall d\in D}d_{d,sd}}\right) \tag{27}$$

Here, Sd refers to the volume of data that inspection machine dd d transmits to the base station; $F_s=F_{Server}^{cpu}$ represents the spatial distance between the inspection machine and the base station through which service is being provided; and $F_s=F_{Server}^{cpu}$ captures the CPU operating frequency of server $s\in S$.

In order to deal with the dynamic nature of the system that consists of many inspection machines and many edge servers, an SLA based reinforcement learning approach will be adopted. Thus, each inspection machine will be able to determine the best MEC server for data offloading on their own through the probabilistic approach of equations (28) and (29).

$$\Pr_{d,s}(t+1)=\Pr_{d,s}(t)+br_s(t)(1-\Pr_{d,s}(t)),\quad s^{(t+1)}=s^{(t)} \tag{28}$$

$$\Pr_{d,s}(t+1)=\Pr_{d,s}(t)-br_s(t)\Pr_{d,s}(t),\quad s^{(t+1)}\neq s^{(t)} \tag{29}$$

where: b denotes the parameter of convergence to SLA, $0<b<1$; Eq. (28) denotes the

probability that the inspection machine selects the same MEC server as time period t in time period $t+1$; and Eq. (29) denotes the probability that the inspection machine selects a different MEC server than the previous time period t in time period $t+1$. In this way, each inspection machine identifies the Nash equilibrium and chooses the best MEC server to offload their data at each stage.

5 Feasibility Assessment of Intelligent Inspection System

5.1 Effectiveness of Edge Server Deployment Programs

5.1.1 Minimum access delay

(B2) RNOESPA, (B3) KMCA, and (B4) RESPA are chosen as the reference approaches for comparing their performance. At first, there will be only one edge server deployed; this quantity of edge servers will then be increased by four times. After that, there will be five edge servers. The average access latency performance of these three reference methods and the proposed method for deploying edge servers is shown in Fig. 5. For all four algorithms, it can be seen that the average access latency gradually reduces as the number of deployed edge servers increases. Specifically, for the proposed algorithm and its two similar methods (B1) RNOESPA, and (B2) KMCA, the numerical value and declining rate show good similarity; their average access latency stays between (1, 2) ms.

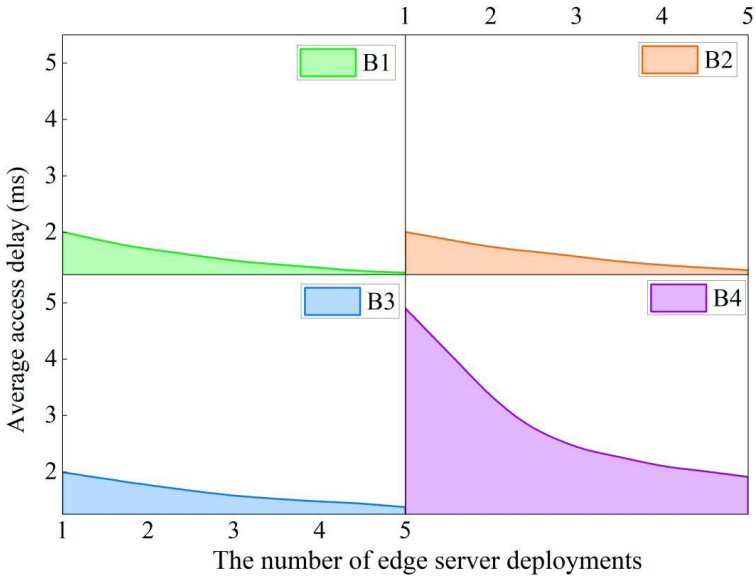

Figure 5: The average access delay performance of the four algorithms

The runtime performances of all the four algorithms in the average access delay comparison experiment are further depicted in Fig. 6. With regard to runtime growth trend, algorithm (B3) KMCA and algorithm (B4) RESPA show a steep runtime growth trend due to an increasing number of servers, while algorithm (B1) edge server deployment scheme proposed by this paper and algorithm (B2) RNOESPA maintain relative stability in terms of runtime during the whole experiment. With respect to the total value of runtime, algorithm (B4) RESPA and algorithm (B2) RNOESPA require much more runtime, each ranging from 20,000 ms or higher, with algorithm (B4) RESPA even taking as long as 143,433 ms. The runtime of algorithm (B3) KMCA ranges between 20 to 36 ms, while that of algorithm (B1)

edge server deployment scheme proposed by this paper is about 4.8 ms. With an increase in the number of edge servers, the ranking of these four algorithms' computational complexity, from low to high, will be: (B1) edge server deployment scheme in this paper < (B3) KMCA < (B2) RNOESPA < (B4) RESPA.

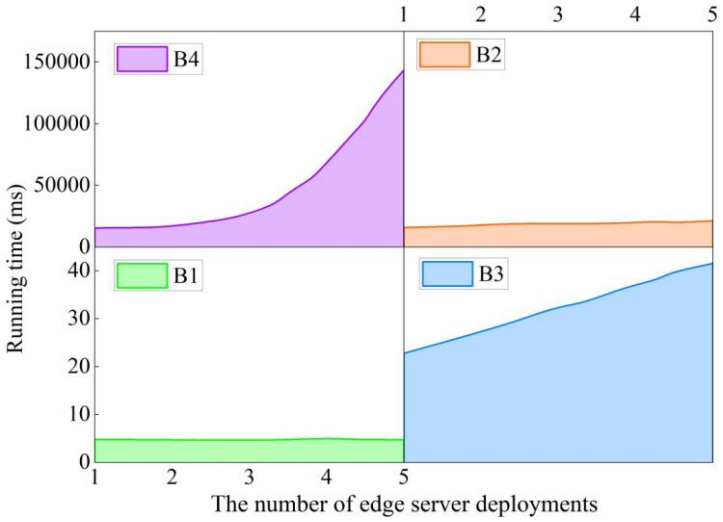

Figure 6: The computational complexity performance of the four algorithms

5.1.2 Network reliability

The network reliability comparison experiments of the four algorithms carried out under edge server network sizes of 10, 20, 30, 40, and 50 are shown in Fig. 7. Overall, the network reliability of the algorithm shows a fluctuating downward trend with the increase of edge server network size, in which the network reliability of (B1) the edge server deployment scheme in this paper, (B2) RNOESPA, and (B3) KMCA are stable at 0.9500 and above, and (B4) RESPA has a network reliability of 0.7996 when the size of the edge server network is 10, and its reliability shows a significant decrease, fluctuating around 0.35, when the network size starts to increase. The reliability of (B4) RESPA is 0.7996 at a network size of 10 edge servers, and when the network size starts to increase, the reliability decreases dramatically and fluctuates around 0.35.

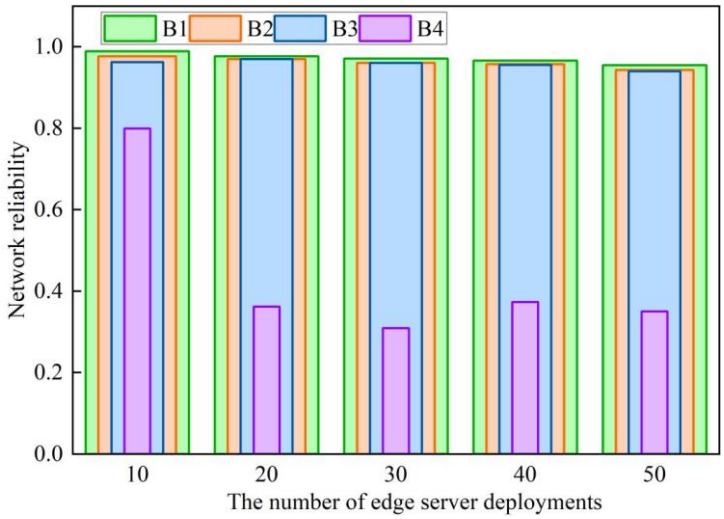

Figure 7: The network reliability performance of four algorithms

Further, the network reliability performance of the four algorithms when the edge server access delay is plotted to be limited to 1.5~2.5ms is shown in Fig. 8. As the same as the network reliability comparison experiment based on the edge server network size, the overall values of the three methods (B1) edge server deployment scheme in this paper, (B2) RNOESPA, and (B3) KMCA still show more excellent network reliability. The difference is that the network reliability of (B1) this paper's edge server deployment scheme, (B2) RNOESPA, and (B3) KMCA is around 0.26 when the average access delay is limited to 1.5ms, and with the relaxation of the average access delay, the network reliability rises gradually to around 0.97 and stabilizes. The network reliability of (B4) RESPA method is in the range of (0.2,0.7).

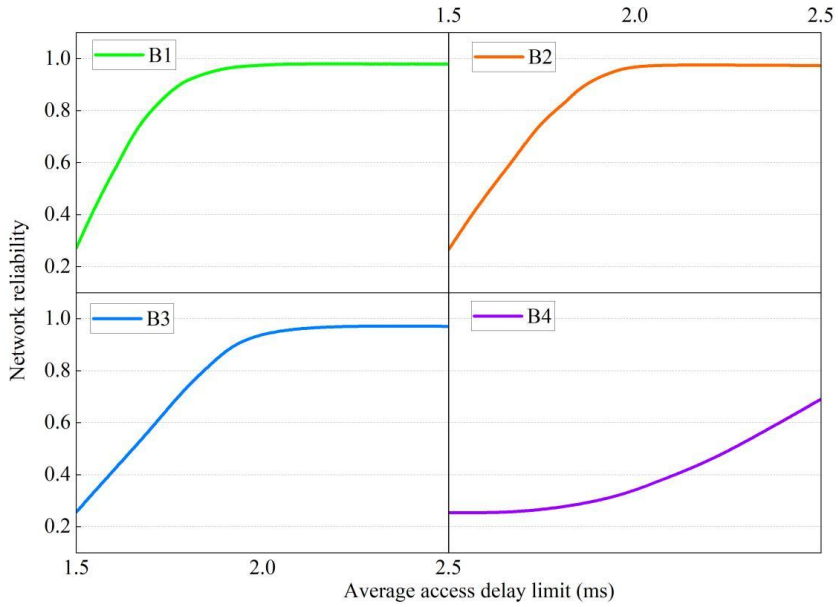

Figure 8: Network reliability under access delay constraints

5.2 Operational performance of the task offloading strategy

5.2.1 Average delay

Taking five active node numbers: 5, 10, 15, 20 and 25, (C1) Random, (C2) K-means and (C3) MDIKW offloading strategies are selected as comparisons to unfold the average latency experiments with the nodes determined by (C4) this paper's offloading strategy are shown in Fig. 9. The average latency of the (C1) Random strategy for different numbers of active nodes are all in the range of 0.94ms floating around, which is higher latency despite being more stable. (C2)K-means strategy is more jumpy in comparison, the average delay of the algorithm is characterized by a growing trend, regardless of the number of nodes, but still staying in the interval of [0.30, 0.65) ms. The MDIKW offloading scheme (C3) and the offloading algorithm presented in this work (C4) give similar outcomes. In spite of the growth of the average delay due to an increase in the number of nodes, and (C4) this paper offloading strategy average delay is the lowest, in the range of 0.1887~0.3043, which verifies the effectiveness of this paper's offloading strategy in the network deployment offloading service.

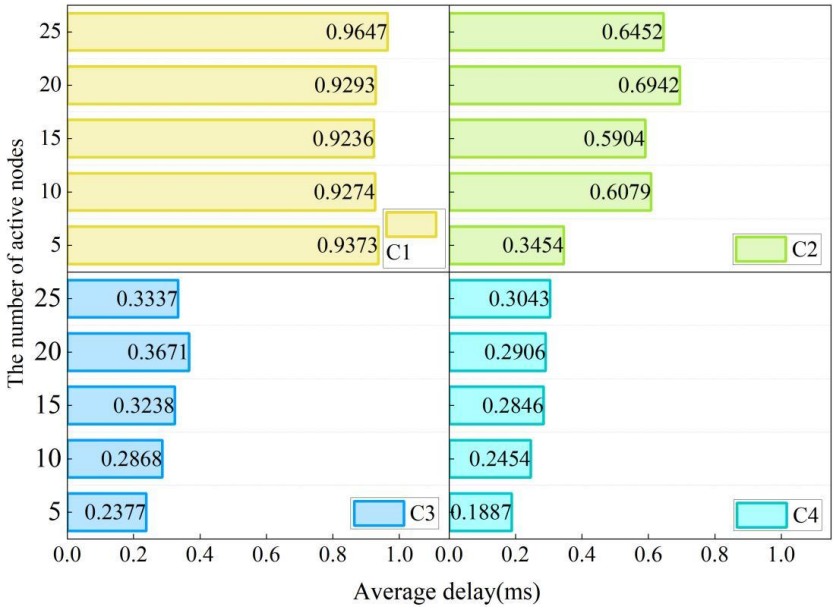

Figure 9: Average delay comparison

5.2.2 Residual energy

Let the time change to 24h, each aggregation node offloads the computational tasks to the nodes every minute, calculate the energy of the network nodes in each hour for the three offloading strategies is shown in Fig. 10. (C1) Random strategy is eliminated in this section due to its high average latency and low performance. It can be seen that the nodes take energy only from 8 to 17h, so the network residual energy reaches the lowest point around 7h and the network residual energy reaches the highest point around 17h. And in 24h, the network residual energy of (C4) offloading strategy of this paper is higher than the remaining two strategies, between 2.93~7.57J, which has the least energy consumption and thus the longest network survival time.

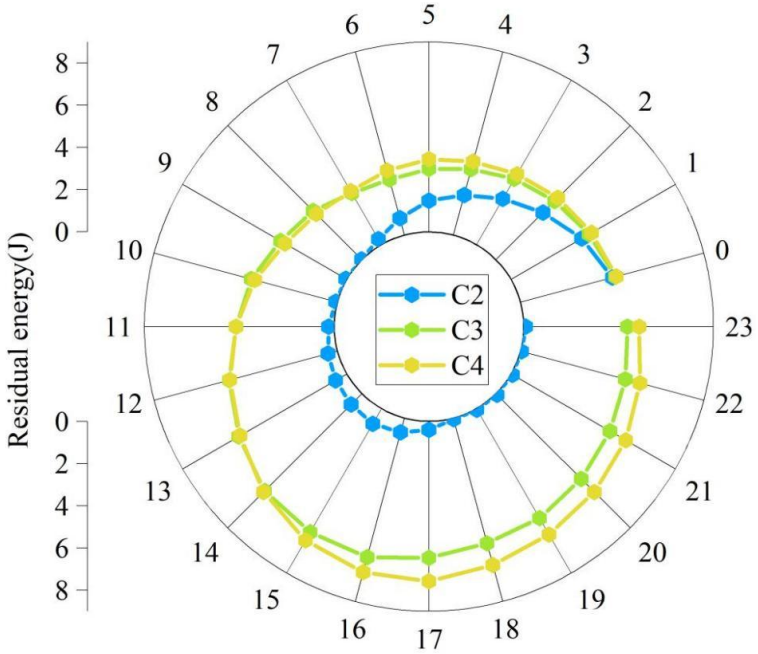

Figure 10: Residual energy comparison

5.3 Simulation Experiments of Task Offloading Algorithm

In this section, the inspection area of a power plant is used as a simulation environment, in which the base station is located in the center of the inspection area and equipped with edge servers, so that the inspection machine equipped with the comprehensive task offloading algorithm of this paper and the inspection system with two task offloading algorithms, D2D and AC, will automatically cruise in the inspection area, and calculate the performance of a number of costs during the work of the inspection system under different task offloading algorithms.

5.3.1 Total cost with different weighted cost ratios

If the weighted sum of the time and energy consumption cost α is set to 0.55, time cost β as 0.00, and energy consumption cost θ as 1.10, the results of the total cost of (D1) task offloading algorithm in this paper and (D2) D2D, (D3) AC altogether two similar strategies with different weighted cost ratios are shown in Fig. 11(a)-(c). Then, as the number of smart terminals increases, all three algorithms' overall costs show an upward trend. Although the total cost performance of the three algorithms is closer but still different, with (D3) AC strategy having the highest total cost, (D2) D2D strategy coming next, and (D1) task offloading algorithm in this paper consuming less total cost. And all three algorithms have the lowest total cost within 1700 when the time cost β is 0.00.

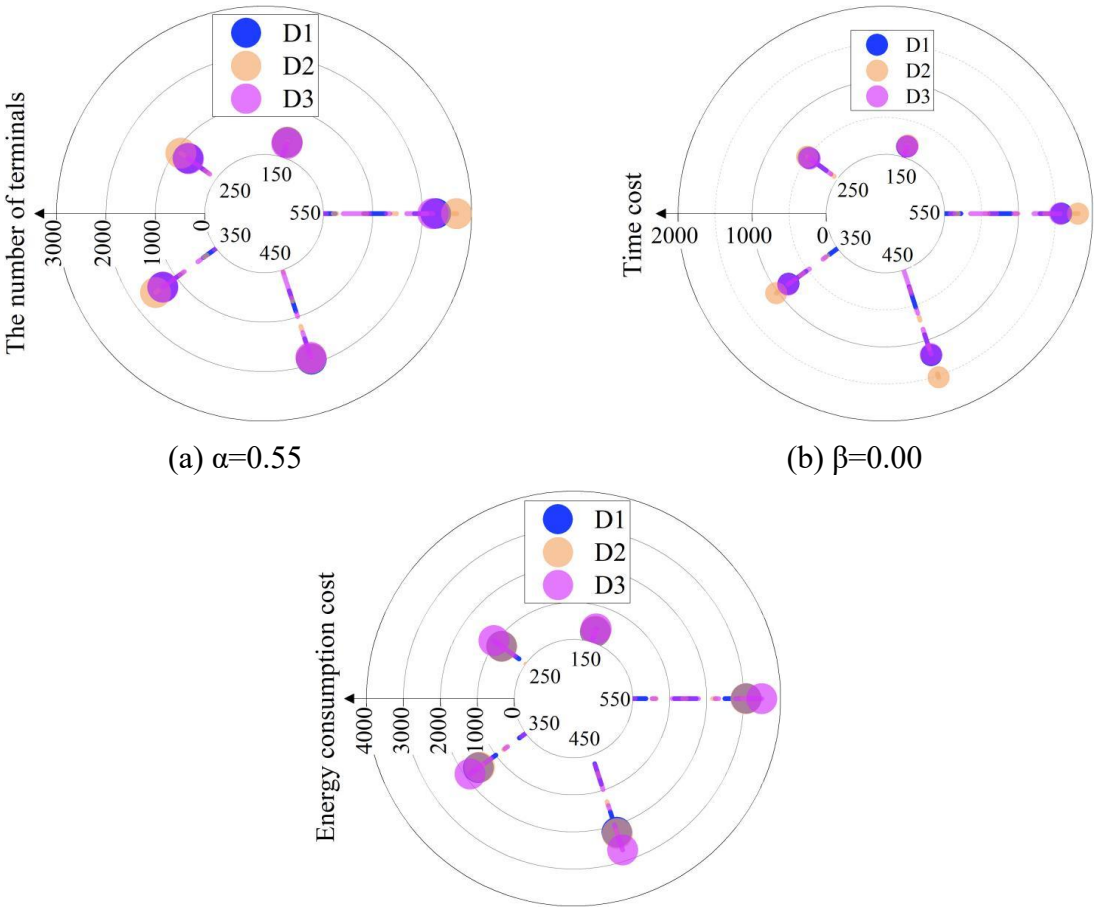

Figure 11: Comparison of algorithm performance under different cost weight ratios

5.3.2 Transmission Delay and Its Energy Consumption Cost

The transmission delay and its energy consumption cost performances of the three algorithms, when the weighted sum of time and energy consumption cost α is 0.55, are shown in Fig. 12. The transmission delay and energy consumption cost performances of all three algorithms are in accordance with the experimental results discussed in the previous section, and the energy consumption is (D1)this paper policy<(D2)D2D<(D3)AC in the order of low to high.The consumption cost sums of the three algorithms are all increased with the smart terminals, the sum of consumption costs of (D1) this paper task offloading algorithm ranges from 151.19 to 1400.00, the sum of consumption costs of (D2) D2D strategy ranges from 170.06 to 1472.17, and the sum of consumption costs of (D3) AC strategy ranges from 176.64 to 1655.43.

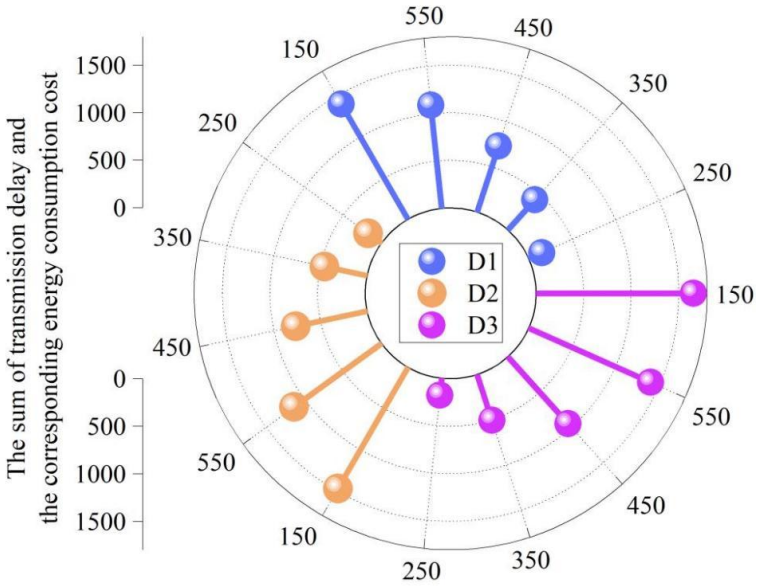

Figure 12: Transmission delay and corresponding energy consumption cost

5.3.3 Utilization of free computing resources

The extent of use of the idle computing resources available by each of the algorithms is evaluated through the calculation of the free CPU percentage for each of the algorithms. The free CPU percentages for all the three algorithms are shown in Fig. 13. As the number of intelligent terminals increases, the three algorithms exhibit an increase in the use of spatial computing resources in their corresponding base station coverage areas. In terms of the growth rate, (D1) this paper task offloading algorithm < (D2) D2D < (D3) AC, and in terms of the proportion of idle CPU, (D3) AC (85.86~533.31) < (D2) D2D (31.90~276.24) < (D1) this paper task offloading algorithm (13.40~67.36), and (D1) this paper task offloading algorithm has the highest resource utilization.

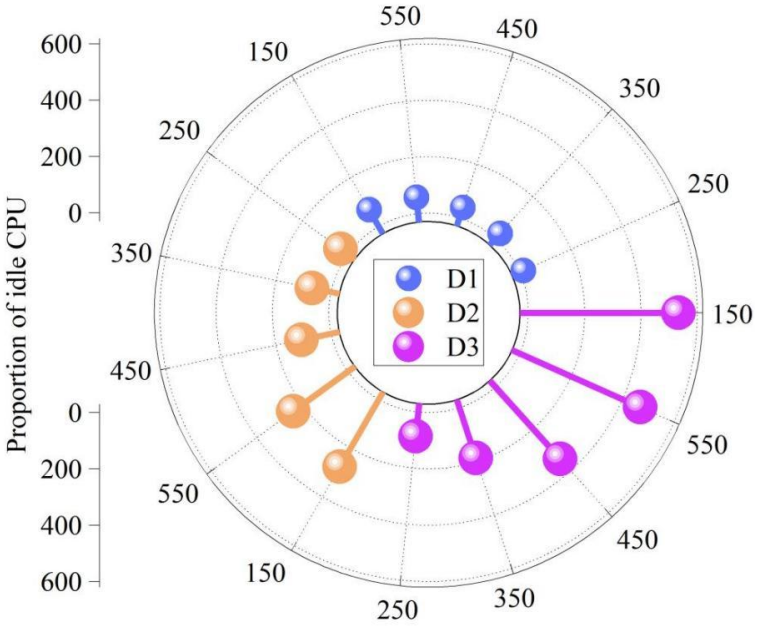

Figure 13: The utilization of idle computing resources by the algorithm

6 Conclusion

For intelligent inspection tasks in power stations, a server node migration model is presented in this paper. In addition, based on this, an approach for deploying an edge server using the improved Top-K method and a task offloading policy in the context of multiple inspections and multiple edge servers are designed respectively.

When conducting edge server deployment tests under different deployment scenarios, the average access delay can be obtained in the interval of (1, 2) ms, and the runtime stabilizes around 4.8 ms. When conducting test cases with different numbers of active nodes, the minimum access delay for the task offloading policy fluctuates in the range of 0.1887 to 0.3043, and the network residual energy ranges from 2.93 to 7.57 J. When performing simulation test cases for the task offloading algorithm, the overall cost of the algorithm is the lowest compared with other algorithms of the same type. The cost consisting of the sum of the transmission delay and energy consumption ranges from 151.19 to 1400.00, and the CPU space utilization ratio varies from 13.40% to 67.36%. This paper designs the various intelligent inspection technology, have a more outstanding performance in the field, and can be flexibly adapted to the different scales and demands inspection scenarios, is a more successful power plant intelligent inspection program design.

About the Author

Jianbo Liu, born on 1970, male, native place of Longhui, Hunan. He holds a bachelor's degree and graduated from Changsha Electric Power Institute with a major in Power System Automation from 1991 to 1995. He is a senior engineer, and his main research directions include safety production management of power plants and clean energy power generation.

References

[1] Jagtap, H. P., & Bewoor, A. K. (2017). Use of analytic hierarchy process methodology for criticality analysis of thermal power plant equipments. Materials today: proceedings, 4(2), 1927-1936.

[2] Majumdar, D., & Pasqualetti, M. J. (2019). Analysis of land availability for utility-scale power plants and assessment of solar photovoltaic development in the state of Arizona, USA. Renewable energy, 134, 1213-1231.

[3] Balijepalli, R., Chandramohan, V. P., & Kirankumar, K. J. R. E. (2020). Development of a small scale plant for a solar chimney power plant (SCPP): A detailed fabrication procedure, experiments and performance parameters evaluation. Renewable Energy, 148, 247-260.

[4] Boente, D. R., & Lustosa, P. R. B. (2020). Efficiency of electricity distribution companies. RAUSP Management Journal, 55(2), 177-193.

[5] Romanov, A. M., Gyrichidi, N., Volkova, M. A., Eroshenko, S. A., Matrenin, P. V., & Khalyasmaa, A. I. (2024). Automated mission planning for aerial large-scale power plant thermal inspection. Journal of Field Robotics, 41(5), 1313-1348.

[6] Chen, J., Xue, C., Wang, Y., Qian, C., & Zhu, Q. (2022, December). Analysis of Inspection and Maintenance Methods for Thermal Power Plant. In International Conference on Communication, Devices and Networking (pp. 585-593). Singapore: Springer Nature Singapore.

[7] Singh, S. N., & Pretorius, J. H. C. (2017). Development of a sem-quantitative approach for risk based inspection and maintenance of thermal power plant components. SAIEE Africa Research Journal, 108(3), 128-138.

[8] Mesas-Carrascosa, F. J., Verdú Santano, D., Pérez Porras, F., Meroño-Larriva, J. E., & García-Ferrer, A. (2017). The development of an open hardware and software system onboard unmanned aerial vehicles to monitor concentrated solar power plants. Sensors, 17(6), 1329.

[9] Wang, H., & Meng, F. (2019). Research on power equipment recognition method based on image processing. EURASIP Journal on Image and Video Processing, 2019(1), 57.

[10] Zhou, Y., Chen, J., Xu, R., Cao, Z., Jin, Z., Guo, Z., ... & Li, K. (2022, December). Three dimensional fully autonomous inspection method for wind power employing unmanned aerial vehicle based on 5G wireless communication and artificial intelligence. In 2022 IEEE 5th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC) (Vol. 5, pp. 800-805). IEEE.

[11] Hua, T. I. A. N., Wei, Z. H. A. N. G., Bin, F. E. N. G., Wei, W. A. N. G., Xiaowei, M. E. N. G., & Tao, Q. I. U. (2023). Design and Development of 5G+ Robot Autonomous Patrol Inspection System in Intelligent Power Plant. SOUTHERN ENERGY CONSTRUCTION, 10(6), 34-42.

[12] Chuang, X., Li, L., Zhu, L., Wei, M., Qiu, Y., & Xin, Y. (2025). The design of a

real-time monitoring and intelligent optimization data analysis framework for power plant production systems by 5G networks. Energy Informatics, 8(1), 1-27.

[13] Huang, G., Ye, M., Zhou, H., Li, J., Wang, G., & Zhang, X. (2021, June). 5G network architecture and networking analysis of smart power plant. In Proceedings of the 2021 3rd International Conference on Information Technology and Computer Communications (pp. 120-126).

[14] Zanzi, L., Cirillo, F., Sciancalepore, V., Giust, F., Costa-Perez, X., Mangiante, S., & Klas, G. (2019). Evolving multi-access edge computing to support enhanced IoT deployments. IEEE Communications Standards Magazine, 3(2), 26-34.

[15] Li, H., Li, X., Fan, Q., Xiong, Q., Wang, X., & Leung, V. C. (2023). Transfer learning for real-time surface defect detection with multi-access edge-cloud computing networks. IEEE Transactions on Network and Service Management, 21(1), 310-323.

[16] Krishnan, P., Duttagupta, S., & Achuthan, K. (2019). SDNFV based threat monitoring and security framework for multi-access edge computing infrastructure. Mobile Networks and Applications, 24(6), 1896-1923.

[17] Rekoputra, N. M., Tseng, C. W., Wang, J. T., Liang, S. H., Cheng, R. G., Li, Y. F., & Yang, W. H. (2023). Implementation and evaluation of 5G MEC-enabled smart factory. Electronics, 12(6), 1310.

[18] Wang, W., Qu, R., Liao, H., Wang, Z., Zhou, Z., Wang, Z., ... & Guizani, M. (2023). 5G MEC-based intelligent computation offloading in power robotic inspection. IEEE Wireless Communications, 30(2), 66-74.