 CMES, vol.118, no.2, pp.291-315, 2019

Context-Based Intelligent Scheduling and Knowledge Push Algorithms for AR-Assist Communication Network Maintenance

Lanlan Rui[1], Yabin Qin[1,*], Biyao Li[1] and Zhipeng Gao[1]

Abstract: Maintenance is an important aspect in the lifecycle of communication network devices. Prevalent problems in the maintenance of communication networks include inconvenient data carrying and sub-optimal scheduling of work orders, which significantly restrict the efficiency of maintenance work. Moreover, most maintenance systems are still based on cloud architectures that slow down data transfer. With a focus on the completion time, quality, and load balancing of maintenance work, we propose in this paper a learning-based virus evolutionary genetic algorithm with multiple quality-of-service (QoS) constraints to implement intelligent scheduling in an edge network. The algorithm maintains the diversity of the population and improves the speed of convergence using a fitness function and a learning-based population generation mechanism. The test results demonstrate that the algorithm delivers good performance in terms of load balancing and QoS guarantee. We also propose a knowledge push algorithm based on a context model for intelligently pushing relevant knowledge according to the given conditions. The simulation results demonstrate that our scheme can improve the efficiency of on-site maintenance.

Keywords: Internet of things (IoT), edge computing, Augmented Reality (AR), maintenance, communication network.

1 Introduction

Efficient and stable operation all equipment in a communication network is important. The safety, quality, and efficiency of on-site maintenance are direct consequences of the effectiveness of maintenance work, because of which stringent requirements govern these factors [Ahmed, Hasan, Pervez et al. (2017)]. However, environments for the maintenance of communication networks are complex for several reasons. A maintenance site is usually remote. The mobility of on-site maintenance personnel may imply an intermittent Internet connection. Poor data exchange results in the failure of maintenance work orders to arrive in time, resulting in unreasonable task scheduling, and low utilization and efficiency [Zhao, Chen, Li et al. (2014)]. Moreover, work order

[1] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China.

[*] Corresponding Author: Lanlan Rui. Email: llrui@bupt.edu.cn.

scheduling uses manual allocation and relies on experience. The diversity of the skills and capabilities of workers is rarely considered. Network data analysis is still based on traditional empirical observations. The on-site maintenance of non-intelligent attributes (dumb resources, such as equipment and lines) that constitute the network requires substantial human involvement. The relevant work relies on personal experience and local information.

To solve the above problems, researchers have proposed using augmented reality (AR) to aid maintenance. AR technology superimposes computer-generated virtual objects or information on the environment in real time to present to the user a scenario with sensory effects and rich scene-related information [Yang, Gong, Song et al. (2017); Gupta (2018)]. This coexistence of real and virtual scenes provides users with flexible and intuitive guidance in the complex equipment maintenance process, such that unskilled workers can maintain equipment to shorten its maintenance cycle and reduce costs [Hou, Lu, Dey et al. (2017)].

Although many studies related to maintenance work for communication networks have been undertaken, many problems in the area remain unsolved. For example, it is important to push auxiliary information to the devices intelligently according to the on-site environment. The contributions of this paper are as follows:

- We propose an AR-based edge maintenance architecture, where on-site maintenance work is carried out in the edge network of a communication network using AR technology. Compared with the widely used cloud architecture, this improves the speed of information circulation, meets the needs of real-time interaction with AR devices, and helps on-site maintenance personnel obtain auxiliary information through the integration of real and virtual information on AR devices.
- We develop a model that describes the complex environment and resource relationships of maintenance site in a communication network. There are three types of nodes in the context model: user, task, and equipment. They correspond to on-site maintenance personnel, maintenance tasks, and equipment to be maintained at the maintenance site.
- Based on the above context model, we propose a learning-based virus evolutionary genetic algorithm (L-VEGA) with multiple quality-of-service (QoS) constraints to implement intelligent scheduling in an edge network. We consider time, quality, and load balancing as multiple QoS goals by considering the impact of multiple factors on the quality of the work. We use the virus evolutionary genetic algorithm (VEGA) for horizontal and vertical search, and introduce a learning mechanism to avoid the loss of the optimal solution and improve the efficiency of evolution. Following the completion of the assignment of personnel, tasks, and equipment, we propose a knowledge push algorithm based on the context model (CMKP) to push related knowledge according to the given conditions. By comparing the similarity between the on-site maintenance context model and the knowledge context model, and filtering based on popularity, auxiliary information is pushed to reduce the cost of knowledge acquisition for on-site maintenance personnel. These two measures improve the efficiency of on-site maintenance personnel.

The remainder of this paper is organized as follows: Section 2 explains related research,

and Section 3 presents our AR-based edge maintenance architecture. Context-based intelligent scheduling and knowledge push algorithms are explained in Section 4, followed by a discussion of the evaluation of our research in Section 5. Section 6 concludes the paper and proposes future work in the area.

2 Related work

Interactive AR guidance has been widely used in maintenance work to improve the efficiency of maintaining complex equipment. The results of many studies have been reported. Fiorentino et al. [Fiorentino, Uva, Gattullo et al. (2014)] proposed an empirical study in which 14 participants performed four tasks related to the maintenance of motorcycle engines with the help of a paper manual and AR-based instructions. The results showed that enhanced instructions significantly reduce overall execution time and the error rate of the participants. In Palmarini et al. [Palmarini, Erkoyuncu, Roy et al. (2017)], the authors analyzed the state of the art in AR maintenance and the most relevant technical limitations. Amo et al. [Amo, Erkoyuncu, Roy et al. (2018)] proposed an information framework to integrate AR into maintenance systems.

Some results on experiments related to task scheduling have been reported, but many defects persist. Qi et al. [Qi and Zha (2013)] applied the VEGA algorithm to the work order scheduling of locomotive maintenance, with the skill of personnel and time as the targets of measurement. However, although the algorithm used vertical and horizontal search, which is an improvement over the genetic algorithm (GA), it had poor population diversity and may lose the optimal solution. To meet the needs of users and ensure efficient use of cloud resources, Yang et al. [Liu, Bi, Yang et al. (2016)] proposed a scheduling mechanism based on double fitness and load balancing. They introduced variance to calculate the loads on workers and weighed a multi-fitness function. The scheduling ensures load balancing and reduces the cost of task completion. In Dai et al. [Dai, Lou, Lu et al. (2015)], the authors proposed a task scheduling algorithm based on the GA and the ant colony optimization algorithm with multiple QoS constraints (MQoS-GAAC) in cloud computing environments. It improves user QoS and balances loads on resources. These two algorithms are an improvement on the GA [Rodriguez, García-Martínez, Blum et al. (2012)], which is intended to solve the high rate of utilization of cloud resources in a cloud computing environment. When the GA solves a scheduling problem, it transmits evolution-related information vertically to ensure the global convergence of the algorithm. However, its high-adaptation strategy retains both efficient and inefficient genes, leading to a decline in the diversity of subsequent generations and premature convergence.

Knowledge push is a way to actively provide knowledge to users during the maintenance process. It can help improve the efficiency and quality of maintenance work. Methods of pushing knowledge used in current maintenance work are generally based on users and content. Wang et al. [Wang, Tian, Wu et al. (2016)] proposed such a method based on the intent of the design and user interest to improve the usability of knowledge. Liu et al. [Liu, Wang, He et al. (2016)] proposed an intelligent method for measuring the cohesion between the contexts of workflow in process-aware information systems and their use for recommending task knowledge at runtime. Xu et al. [Xu, Yin, Nie et al. (2013)] proposed

an active knowledge service framework based on capturing collaborative intent to assist in the conceptual design of products. However, in the on-site maintenance scenario, knowledge is context dependent. The context here includes the business process and task characteristics as well as the personality characteristics of users and the equipment. Knowledge matching is an important part of knowledge push, and aims to find knowledge that best meets the requirements of related tasks or users. In Zhang et al. [Zhang and Li (2016)], the authors proposed a general method to push business process knowledge using a multi-dimensional hierarchical context model. By creating a representative context model, they considered the two dimensions of user and task, and proposed different combinations of types of schemes to calculate text similarity and semantic similarity. Wang et al. [Wang, Tian, Wu et al. (2016)] proposed three types of semantic similarity calculation and a functional behavior structure. The work in Gu et al. [Gu, Hu, Wu et al. (2015)] developed a mapping relationship between cases and knowledge to obtain appropriate knowledge by comparing the similarities between the given tasks and past cases. However, these methods are mainly based on the similarity of text or symbols, and do not consider the similarity of equipment. Moreover, in the knowledge push process, knowledge matching updates need to be considered.

Based on the results of prevalent research, we propose an AR-based edge architecture maintenance scheme with context-based intelligent scheduling and knowledge push algorithms.

3 Architecture of AR-based edge maintenance

Most maintenance systems in communication networks use a cloud service architecture [Lojka, Bundzel, Zolotova et al. (2016)]. Maintenance personnel collect resource information, such as operation status and fault information, and forward it to the cloud server for processing and analysis. The cloud server remotely guides the maintenance work on-site. However, given that most cloud servers are stationed far from terminal devices and users [Bonchi, Brogi, Canciani et al. (2017)], the mobility of on-site maintenance personnel may lead to an intermittent Internet connection. Moreover, AR glasses require fast real-time performance. The resource and bandwidth limitations of AR glasses may mean that they do not adapt well to cloud architecture.

As a result, we use the concept of edge computing in the maintenance architecture of a communication network [Qiao, Ren, Dustdar et al. (2018); Schneider, Rambach, Stricker et al. (2018)]. Edge computing migrates applications, data resources, and computing services originally provided by the central node to edge nodes, providing nearest-end services at nearby locations [Ai, Peng, Zhang et al. (2017); Cicirelli, Guerrieri, Spezzano et al. (2017); Aazam and Huh (2016); Dolui and Datta (2017)]. This method of providing nearby computing services meets technical and application-related requirements in terms of rapid connection and response, and real-time analysis [Al-Shuwaili and Simeone (2017)]. As shown in Fig. 1, we propose an AR-based edge maintenance architecture for a communication network.

On-site maintenance personnel are provided with AR glasses. Smart glasses have a voice-controlled display screen that can visualize real-time data, pictures, and video information. On-site maintenance personnel can be connected in real time with remote

experts to gain valuable knowledge and skills during the work process. Regardless of the time of day or location of the workers, interconnected technology can be used for their assistance. This implies that on-site maintenance personnel will have decades of accumulated expertise that can be accessed through commands in simple language. Such a solution can keep workers safe and make them more efficient.

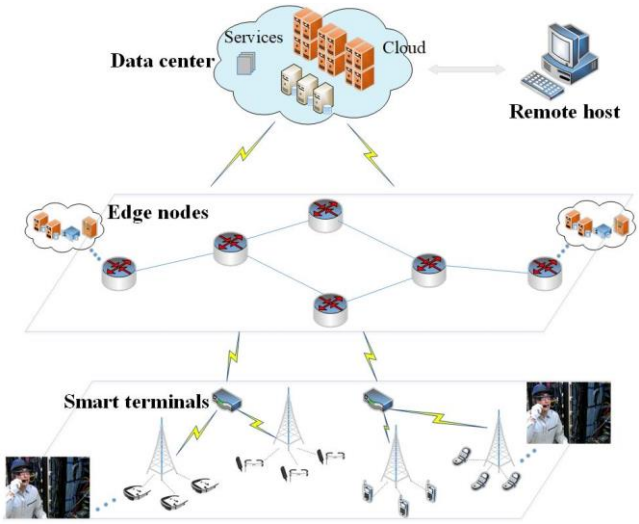


Figure 1: AR-based edge maintenance architecture

The intelligent scheduling and push scenario of on-site maintenance in communication networks are shown in Fig. 2. An edge server assigns work orders to the on-site maintenance personnel in the covered area. Workers, equipment, and tasks must be considered when assigning work orders. During the work, the edge server pushes relevant knowledge to the user to assist them based on the context. Once the tasks have been completed, the on-site maintenance personnel upload a work record to the edge nodes, and the central server implements data synchronization using them.

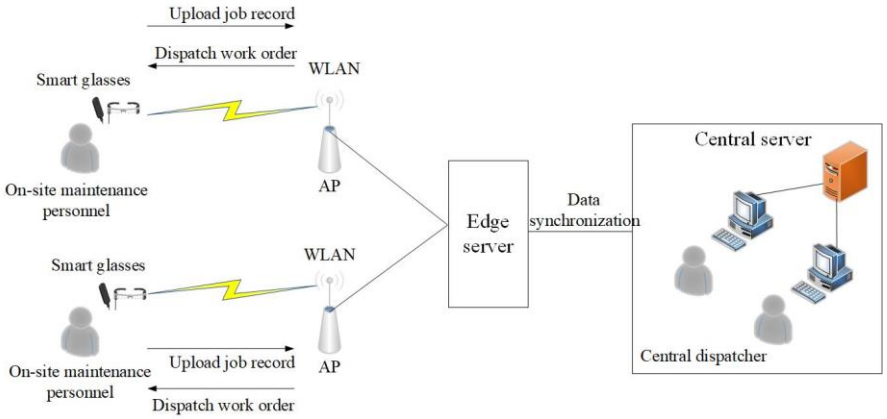


Figure 2: Maintenance scenario in communication networks

4 Context-based intelligent scheduling and knowledge push algorithms

4.1 Context model

The context model is an abstraction of the maintenance work in a complex communication network, and includes user context, equipment context, and task context. The user model is mainly composed of basic information and usage records. The equipment model is an abstraction of the maintained resources, and includes equipment type, location, and components. Different equipment maintenance tasks require assistance with diverse knowledge such that the equipment must be modeled. In maintenance work, the tasks include on-site inspection, on-site overhaul, and on-site troubleshooting. The information required for separate working periods is different. Therefore, task models should be considered when modeling context models.

The attributes of user context, equipment context, and task context can be dynamically adjusted based on the maintenance conditions. Fig. 3 shows a generic context model.

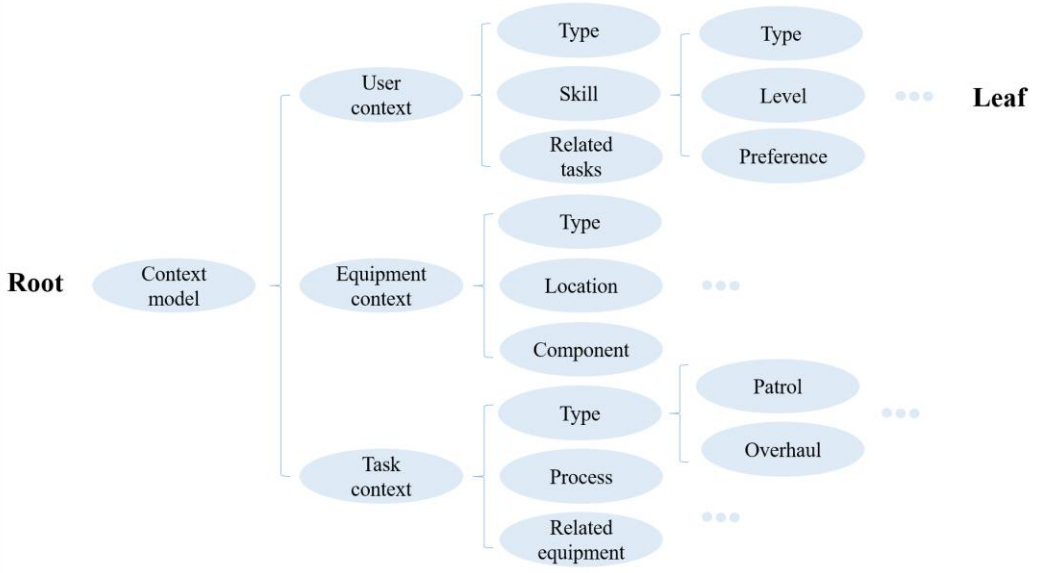


Figure 3: Multi-dimensional context model

4.2 Intelligent scheduling of the communication network

4.2.1 Problem description

For the task set $P=\{P_1, P_2, \ldots, P_n\}$, there are n different tasks, each containing k processes. J is a collection of k processes. $J=\{J_1, J_2, \ldots, J_k\}$,, where J_1 and J_k are the start and end virtual tasks. Let bt_i and et_i be the start and execution times of the process; then, $bt_1=0, et_1=et_k=0$. Considering the importance and difficulty of the process, the weight of the i-th process is set to ω_i, $\omega_i \in \{1, 2, \ldots, 10\}$, $\omega_1=\omega_k=0$.

We use the directed acyclic graph $G=(V, E)$ to describe timing constraints between processes. The vertex set is $V \subseteq J$. Set $E=\{(J_a, J_b), J_a, J_b \in J, a \neq b\}$ indicates that process J_b

must start after the completion of process J_a. The set of processes completed before process J_i is PJ_i, and the set of processes completed after process J_i is NJ_i. PJ_i and NJ_i can be obtained from the adjacency matrix of G, as shown in Fig. 4. The second row is the set of processes completed after J_2, that is, NJ_2. The fourth column is the set of processes completed before J_4, that is, PJ_4.

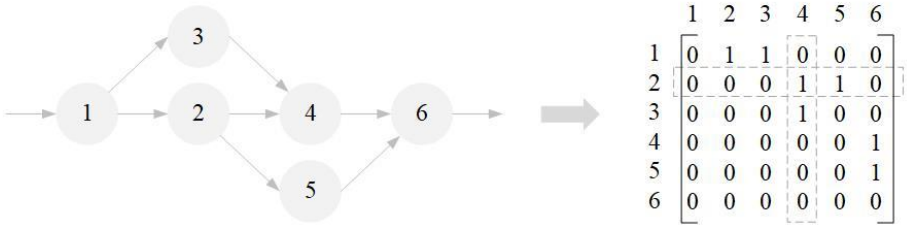


Figure 4: Timing constraints on tasks

The user set is $M=\{M_1, M_2, \ldots, M_m\}$. The n tasks are completed by m maintenance personnel, each with skills of different types at different levels. Maintenance personnel are assigned to different departments or groups according to their majors, or skills, to accept different maintenance tasks. S is a collection of f skills, $S=\{S_1, S_2, \ldots S_f\}$. The skill level ranges from one to ten, and the higher the level, the better the mastery of the relevant skill.

We use matrices to represent the demand relationship between tasks and user skills. MS is a user-skill relationship matrix. The value of MS_{ij} is the level at which the i-th maintenance person has the j-th skill. $MS_{ij}=0$ means that the i-th maintenance person does not have the j-th skill. JS is a task–skill relationship matrix. $JS_{ij}=1$ means that process i must be completed by the maintenance personnel who has skill j. Otherwise, $JS_{ij}=0$.

Equipment set $E=\{E_1, E_2, \ldots, E_l\}$. There is a many-to-many mapping between tasks and devices. A task can involve multiple devices, and a device can carry multiple tasks. s_{ij} is the distance between the maintenance personnel and the device displayed by the AR device according to positioning information.

The constraints are as follows: There is no constraint on the process between tasks, but there are constraints within a task. Assignments must meet skill requirements of the tasks at hand. If an available candidate set is empty, the relevant task must be suspended for a period of time and allocated once the resources are released. A process is completed by a worker, and a worker performs only one task at a time. There is no interruption in process execution.

4.2.2 Objective function

The goal of intelligent scheduling is to find a reasonable scheduling scheme that achieves the shortest completion time, the highest quality, and the most balanced load [Hu and Jiang (2017); Lopes (2017)]. When solving scheduling problems, most algorithms use the minimum completion time as a goal, and ignore the impact of workers on the quality of

the work. We propose an intelligent scheduling method for on-site work orders with multiple QoS constraints. Multi-QoS targets are evaluated by fitness functions.

The completion time of a maintenance task depends mainly on the execution time of the task, and the distance between the maintenance personnel and the equipment to be maintained. For a given scheduling scheme, the shortest execution time for tasks can be represented as the minimum start time of the virtual end process J_k. This is calculated as follows:

$$T_a = max\left\{\left(bt_i + et_i \mid J_i \in PJ_k\right)\right\} \tag{1}$$

The time it takes for maintenance personnel to arrive at the maintenance site is as follows:

$$T_b = \frac{s_{ij}}{v} \tag{2}$$

Thus, the shortest completion time as goal can be expressed as:

$$f_1 = min\left(T_a + T_b\right) \tag{3}$$

The assignment of maintenance personnel directly influences the quality of the maintenance tasks. Let a and b be the maximum and minimum values of skill levels for all maintenance personnel that can execute process J_i. The quality evaluation of process J_i performed by maintenance personnel at skill levels a and b are q_{ia} and q_{ib}, respectively. Then, the quality evaluation of process J_i when executed by maintenance personnel with skill level j is q_{ij}.

$$q_{ij} = q_{ib} + \frac{\left(j-b\right)\left(q_{ia} - q_{ib}\right)}{a-b} \tag{4}$$

According to the weight and quality of each process, the overall quality of a task can be obtained as follows:

$$Q = \frac{\sum_{i=2}^{k-1} \omega_i \theta_{ij} q_{ij}}{\sum_{i=2}^{k-1} \omega_i} \tag{5}$$

In Eq. (3), ω_i is the weight of process J_i. θ_{ij} is used to indicate whether J_i is assigned to M_j. If θ_{ij}=1, J_i is assigned to M_j. Otherwise, θ_{ij}=0. q_{ij} is the quality evaluation value when M_j executes process J_i.

Thus, the goal with the highest completion quality can be expressed as:

$$f_2 = min\left(\frac{1}{Q}\right) \tag{6}$$

Maintenance personnel with the same skills at same skill levels should have the same workload. Maintenance personnel with the same skills at different skill levels need to be assigned according to need. Load balancing can be measured by the mean-squared deviation of work done by maintenance personnel per unit time.

Let a and b be the maximum and minimum values of the skill levels, respectively, of all maintenance personnel who can execute process J_i. The execution time of process J_i

performed by the maintenance personnel at skill levels a and b are t_{ia} and t_{ib}, respectively. Then, the execution time of process J_i when executed by maintenance personnel at skill level j is t_{ij}.

$$t_{ij} = t_{ib} + \frac{(j-b)(t_{ia}-t_{ib})}{a-b} \tag{7}$$

The amount of work done by M_j per unit time is:

$$num_{lsj} = \frac{\sum_{i=2}^{k-1} \theta_{ij}\omega_i}{\sum_{i=2}^{k-1} \theta_{ij} t_{ij}} \tag{8}$$

In Eq. (8), θ_{ij} is used to indicate whether J_i is assigned to M_j. If θ_{ij}=1, J_i is assigned to M_j. Otherwise, θ_{ij}=0. ω_i is the weight of process J_i.

Thus, the goal of load balancing can be expressed as:

$$f_3 = \sum_{l=1}^{f} \sqrt{\sum_{s=1}^{10}\sum_{j=1}^{m}\sum_{i=2}^{k-1} \frac{\theta_{ij}\left(num_{lsj} - \overline{num_{ls}}\right)}{N_{ls}}} \tag{9}$$

In Eq. (9), N_{ls} is the number of personnel with skill S_l at skill level s, and $\overline{num_{ls}}$ is the average amount of work done.

Then, the three-dimensional (3D) QoS objective function is:

$$f = w_1 f_1 + w_2 f_2 + w_3 f_3 \tag{10}$$

In Eq. (10), w_1, w_2, and w_3 are the weights of f_1, f_2, and f_3, respectively. We use expert scoring to determine the weight of each objective function [Chao, Meng, Tingting et al. (2016)]. We solicited the opinions of 30 industry experts to score the 3D QoS evaluation indicators. We considered evaluation indicators {f_1, f_2, f_3} as a fuzzy concept, treating each indicator as an element and scoring it from one to 10.

Each expert independently provided a score for element f_i, and we obtained a 3×30 adjacency matrix. Once the scores had been average weighted and normalized, we obtained the weights below:

Table 1: Weights of three-dimensional QoS evaluation indicators

Indicator	f_1	f_2	f_3
Weight	0.55	0.3	0.15
Rank	1	2	3

4.2.3 Learning-based VEGA algorithm with multiple QoS constraints

The chromosome of a host individual contains $n \times k$ genes, where n is the number of tasks and k is the number of processes per task. Each gene represents a worker assigned to the process of a task. The coding of a virus is obtained by adding a wildcard * related to the coding of the host population.

Fitness is used to assess the quality of individuals in a population. In this paper, the reciprocal of the 3D QoS objective function is chosen as the fitness function of the host generation, *I*, and is expressed as follows:

$$Fitness(I) = \frac{1}{f} \tag{11}$$

In Eq. (11), *f* is the 3D QoS objective function. The smaller the value of the objective function, the larger that of the fitness function, and the higher the corresponding chromosome fitness.

A virus can infect several host individuals. The fitness of a virus is expressed by the change in fitness before and after infection of the hosts it infects. The fitness function of a virus is:

$$fitvirus_i = \sum_{j \in S} \left(fithost_j^{'} - fithost_j \right) \tag{12}$$

In Eq. (12), $fitvirus_i$ is the fitness of virus *i*, *S* is the set of hosts infected by *i*, and $fithost_j$ and $fithost_j^{'}$ are the fitness values of the host individuals before and after infection.

The traditional VEGA algorithm randomly selects the initial population. The selection of the initial population influences the efficiency of the algorithm. We use a learning-based population initialization method. When initializing the population, we first query the knowledge base to avoid losing the optimal solution and improve evolution capability.

When generating the initial population, we search the database to find the results of calculations of the same or similar conditions. If these are available, a portion of the chromosome is randomly selected, and another portion of it is randomly generated. If not, the initial population is randomly generated, as shown in Fig. 5. This ensures that the initial population has a higher average fitness value, that individual diversity is maintained in the initial population, and that the loss of the optimal solution is prevented as much as possible.

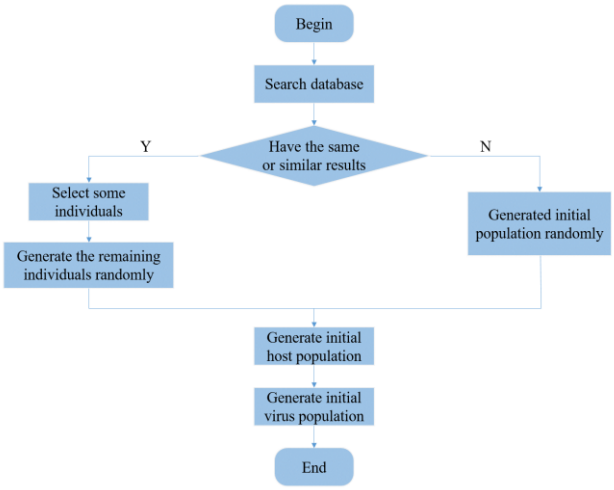


Figure 5: Generation of initial population

The genetic operation involves the following three steps:

(1) Selection: The selection step consists of choosing suitable individuals from the host generation and eliminating relatively inferior ones. It is based on a fitness assessment. Individuals with greater fitness are more likely to be selected. We use roulette selection to determine their selection probability.

(2) Crossover: In the host generation, two individuals are randomly selected and crossed using a single-point crossover method with crossover probability P_{cross} to obtain sub-chromosomes and enter the next-generation set. The value of the crossover probability is usually between 0.6 and 0.9.

(3) Mutation: For each gene of the population obtained following the crossover, we adopt the shift variation method to determine whether to mutate according to the preset mutation probability P_{mut}.

The viral population infects the host population with P_{infect}. If the fitness of the infected population increases, the infected individuals replace the host individuals; otherwise, a new virus is generated to replace the corresponding individual in the original virus generation.

In the traditional VEGA algorithm, a new virus is generated only when the vitality of a previous virus decays to zero. When a virus infects a generation of individuals, its vitality may not have weakened to zero. However, with multiple infections of the same virus, the diversity of the population is difficult to maintain, and the probability of mutation is challenging to select. We use a learning-based virus update mechanism to maintain the diversity of the population and improve the speed of convergence of the algorithm.

Let Gen_{ik} and Gen_{jk} be the k-th genes on chromosomes i and j, respectively. The Euclidean distance between the chromosomes of the host population is as follows:

$$Dis_{ij} = \sqrt{\sum_{k=1}^{n} \left(Gen_{ik} - Gen_{jk} \right)^2} \tag{13}$$

The similarity between the chromosomes is as follows:

$$Simil_{ij} = \frac{1}{1 + Dis_{ij}} \tag{14}$$

If the host population achieves the similarity standard, new viruses are generated in the next-generation genetic operation to maintain the diversity of the population. If not, no new virus is generated.

The above analysis relates to the genetic and evolutionary processes of the virus. On this basis, we propose a learning-based VEGA algorithm with multiple QoS constraints.

The algorithm is as follows.

Step 1: initialize the parameters of the algorithm.

P_{mut}, P_{copy}, P_{cross}, P_{cut}, P_{infect}, γ, POP_{host}=100, POP_{virus}=10, t=0.

Step 2: initialize the generation. Search the database to find the results of calculation for the same or similar conditions. If there is any, a portion of the chromosome is randomly selected, and the other portion of the chromosome is randomly generated. If not, the

initial population is randomly generated.

Step 3: perform a geneticoperation on the host population.

Step 4: perform an evolutionary operation on the virus.

Step 5: generate new viruses. Determine whether a new virus is to be generated based on the similarity between the chromosomes.

Step 6: evaluate the host population based on fitness and retain the best individuals.

Step 7: if the termination conditions are met, the algorithm ends. Otherwise $t=t+1$, and move to Step 3.

The brief algorithm is as follows:

Algorithm 1: Learning-based VEGA algorithm with multiple QoS constraints

```
Input: List of tasks, on-site maintenance personnel and the equipment to be maintained
Output: The optimal solution for task allocation
Steps:
1:  Initialize algorithm parameters. P_mut, P_copy, P_copy, P_cut, P_infect, γ, POP_host=100, POP_virus=10, t=0.
2:  Initialize generation
3:  Search database
4:  If(The same or similar conditions exist)
5:      Select some individual
6:      Generate the remaining individuals randomly
7:  Else if(There is no identical or similar condition)
8:      Randomly generate initial population
9:  End if
10: While(t<=T)
11:     Perform genetics operation on the host population
12:     Perform evolutionary operation on the virus
13:     Generate new viruses
14:     Calculate chromosome similarity of the host population
15:     If(The host population did not reach the specified similarity)
16:         Do not generate new viruses
17:     Else
18:         Generate new viruses
19:     End if
20:     Select regenerative individuals based on fitness
21:     t=t+1
22: End while
23: Return the optimal solution
```

4.2.4 Analysis of algorithms

The proposed algorithm consists of two stages. First, we traverse individuals in the knowledge base to find the same or similar conditions. This consists of only traversing the candidates once. The complexity of this process is $O(n)$. Following genetic manipulation, a similarity measure is applied to determine if a new virus must be produced. The complexity of this process is $O(n^2)$. Therefore, we obtain the complexity of the entire algorithm as follows:

$$\Omega = O\left(n^2\right) + O\left(n\right) \tag{15}$$

4.3 Maintenance knowledge push based on context model

The information required for each period of maintenance is different, because of which it is necessary to study the push mechanism for maintenance knowledge to achieve accurate push matching with maintenance personnel, equipment, and tasks.

4.3.1 Problem description

It is difficult for maintenance personnel to fully understand information concerning each device due to the complexity of resource devices. In the absence of information about the corresponding equipment, maintenance work may not be carried out, which significantly affects maintenance efficiency. Knowledge refers to information that can guide practice. By intelligently pushing relevant knowledge to on-site maintenance personnel, two-way information flow between the maintenance site and the edge server can be implemented, the cost of knowledge acquisition can be reduced, and the efficiency of on-site maintenance personnel can be improved.

4.3.2 Maintenance knowledge push based on context model

Similarity is the degree of sameness between a and b, expressed as *Simil(a, b)*, *Simil(a, b)* $\in$[0, 1]. The larger the value, the greater the similarity between *a* and *b*. By calculating the similarity between the maintenance context model and the knowledge context model, knowledge suitable for a given task is pushed to on-site maintenance personnel, thereby improving the quality and efficiency of on-site maintenance work.

The context model has three dimensions: user, equipment, and task. Each dimension consists of several related attributes, which may have different weights. When calculating similarity between context models, the similarity of their leaf nodes is first calculated, following which similarity between intermediate nodes is calculated according to weights, and finally, the similarity of the root node is obtained. The similarity of the root node is the similarity between context models.

The AR-based maintenance mode generally assists on-site maintenance personnel by using AR glasses. Thus, the context model generally contains two types of attributes: text and images. Due to the complexity of the components of the device being maintained, if only text is used for description, the state of the device being maintained cannot be accurately described at any given time. AR-based maintenance collects images of the maintained device through the AR device, and combines it with user and task information to obtain more accurate push knowledge.

Once on-site maintenance personnel arrive at the maintenance site, the AR device automatically identifies the device to be maintained, obtains related component information for the device, and takes photos and uploads them to the edge server. The edge server acquires device information according to the images and establishes a maintenance context model in combination with the given user, device, and task information. It then pushes auxiliary information for the AR device based on the context model. At this time, the similarity between the user and the task is calculated by the text, and similarity calculation for the device is based on images supplemented by text. The

similarity of attributes is calculated by type, and more accurate results can thus be obtained.

The different types of similarity calculations are as follows:

(1) Leaf attribute of text type

$$Simil(u,v)=\left|\frac{\sum_{i=1}^{n}\left(u_i-\bar{u}\right)\left(v_i-\bar{v}\right)}{\sqrt{\sum_{i=1}^{n}\left(u_i-\bar{u}\right)^2}\sqrt{\sum_{i=1}^{n}\left(v_i-\bar{v}\right)^2}}\right| \tag{16}$$

As shown in Eq. (16), the attribute values of u and v are $(u_1, u_2, \ldots, u_n)$ and $(v_1, v_2, \ldots, v_n)$, respectively. The larger the value, the greater the similarity.

(2) Leaf attribute of image type

$$Simil(u,v)=\sum_{i=1}^{n}\sqrt{p(i)q(i)} \tag{17}$$

In Eq. (17), p and q respectively represent the image histogram data of context models u and v, and the calculated result is the similarity value of the images. The larger the value, the greater the similarity.

(3) Non-leaf attribute

$$Simil(u,v)=\frac{\sum_{i=1}^{n}w_i Simil\left(u_i,v_i\right)}{\sum_{k=1}^{n}w_k} \tag{18}$$

Suppose u and v are non-leaf attributes, and the number of their sub-attributes is n. In Eq. (18), u_i and v_i are the i-th sub-attributes of u and v, respectively, w_i are the weights of u_i and v_i, respectively, and $Simil(u_i, v_i)$ is the similarity of their corresponding i-th sub-attributes.

The similarity between context models is calculated recursively by the above method of similarity calculation. The knowledge context model is sorted by similarity. Several knowledge context models with large similarities are selected as candidate models.

Suppose n knowledge context models are in the knowledge base. u is the root node of the maintenance context model and v_i is the root node of a knowledge context model. The greatest similarity of the context model is:

$$MaxSimil=\max\left(\sum_{i=1}^{n}Simil\left(u,v_i\right)\right) \tag{19}$$

The candidate knowledge context models are then sorted by popularity, and the model with the greatest popularity is selected, and its knowledge is pushed to the given device. The popularity of knowledge is calculated as follows:

$$KPop_i = \frac{N(i)}{N_{\max}} \tag{20}$$

In Eq. (20), *N(i)* is the number of uses of knowledge item *i*, and N_{max} is the maximum number of uses of all knowledge.

Once maintenance work has been completed, the weight of the knowledge context model attribute can be updated according to feedback to ensure the dynamic update of the knowledge base. For example, the user model can be modified based on the historical behavior of on-site maintenance personnel to better match the user's skills and knowledge preferences.

The updated weight of each attribute in the knowledge context model is related to the weight before the update as well as the weight of the corresponding attribute in the matching maintenance context. The method of updating the weights is as follows:

$$w_c = (1-\alpha) w_c^{'} + \alpha \cdot w_m \tag{21}$$

In Eq. (21), $w_c^{'}$ and w_c are the weights before and after the update of the attributes in the knowledge context, and w_m is the weight of the corresponding attribute in the matching maintenance context model. α is the coefficient of the weight update, which is usually set to 0.05.

The framework for knowledge push is as shown in Fig. 6.

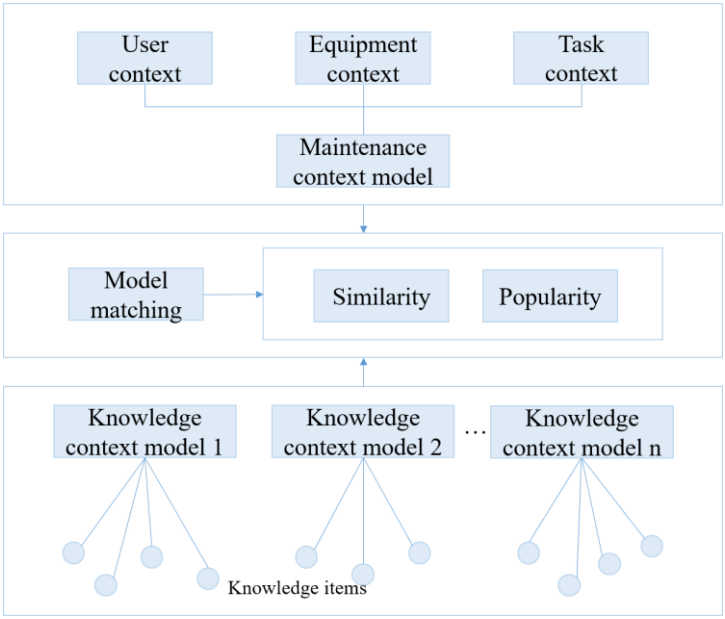


Figure 6: Framework for knowledge push

4.3.3 Steps of the algorithm

The brief algorithm is as follows:

Algorithm 2: Maintenance knowledge push based on context model

Input: A user, a task and the equipment to be maintained
Output: Knowledge pushed to AR devices
Steps:
1: Knowledge in the knowledge base is initialized for modeling
2: Build a maintenance context model
3: For(leaf nodes u and v in the context models)
4: If(u and v are texts)
5: Calculate their similarity according to Eq. (16)
6: Else if(u and v are images)
7: Calculate their similarity according to Eq. (17)
8: End if
9: End for
10: For(non-leaf nodes u and v in the context models)
11: Calculate their similarity according to Eq. (18)
12: If(u and v are root nodes)
13: Break
14: End if
15: End for
16: Select the knowledge with large similarity as candidate knowledge
17: Select the most popular knowledge from the candidate knowledge
18: Return the knowledge pushed to AR devices
19: Update the weight of attributes in the knowledge context model

4.3.4 Analysis of algorithms

The proposed algorithm has two stages. First, we calculate similarity between root attributes recursively by calculating similarity between the leaf attributes of the given context models. The complexity of this process is *O(nlogn)*. We then compare the popularity of knowledge based on similarity, and select the knowledge with the greatest popularity. The complexity of this process is *O(n)*. Therefore, we obtain the complexity of the entire algorithm as follows:

$$\Omega = O(nlogn) + O(n) \tag{22}$$

5 Result and analysis

5.1 Simulation environment

We conducted simulations of our proposed methods using Visual C++ and MATLAB. Some of the parameter settings for the L-VEGA are shown in Tab. 2.

Table 2: Parameter explanation

Symbol	Meaning	Value
POP_{host}	Size of the host generation	100
P_{mut}	Probability of mutation	0.05
P_{cross}	Probability of crossover	0.89
POP_{virus}	Size of the virus generation	10
P_{copy}	Probability of copy	0.2
P_{cut}	Probability of cut	0.15
P_{infect}	Probability of infection	0.02
γ	Decline in vitality rate	0.9

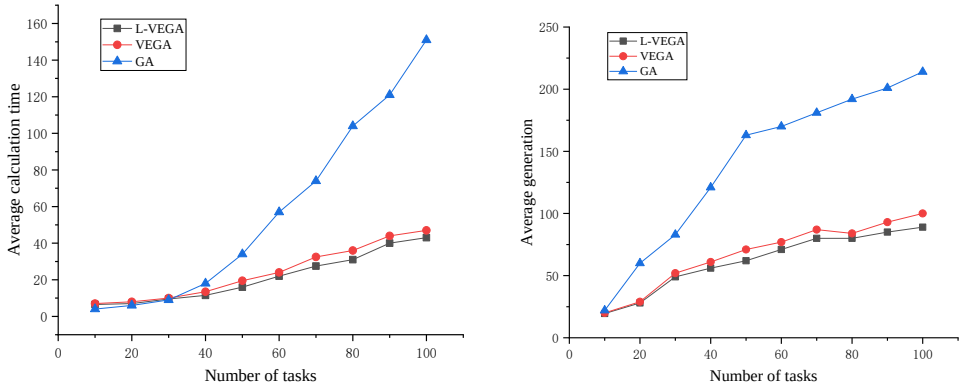
5.2 Results of simulation of L-VEGA

5.2.1 Average performance for different numbers of tasks

We randomly generated 100 instances, and calculated the average calculation times and generations using different algorithms. Here, we compare L-VEGA with GA and VEGA. The results are shown in Tab. 3, Tab. 4 and Fig. 7.

As shown in Fig. 7(a), when the number of tasks is small, the calculation time of the GA is slightly shorter than that of VEGA and L-VEGA. However, as the number of tasks increases, they average time grows exponentially, whereas the average performance of VEGA and L-VEGA does not change much. The average time spent by L-VEGA is slightly less than that spent by VEGA. Because the local search capability of GA is poor, the search efficiency is low in the late evolution. In maintenance work, the number of tasks is usually large. Therefore, compared with GA and VEGA, L-VEGA can obtain an optimal solution with a shorter calculation time.

Like in the analysis of the average calculation time, the average generation of the GA is larger than that for the other two algorithms. The average generation for L-VEGA is slightly less than that of VEGA as shown in Fig. 7(b). As the number of tasks increases, the difference between these three algorithms grows larger. As a result, L-VEGA obtains the optimal solution with the shortest calculation generation.



(a) (b)

Figure 7: Average performance for different numbers of tasks. (a) Average calculation times for different numbers of tasks. (b) Average calculation generations for different numbers of tasks

Table 3: Average calculation times for different numbers of tasks

Number of tasks / Algorithms	10	20	30	40	50	60	70	80	90	100
GA	4	6	9	18	34	57	74	104	121	151
VEGA	7	8	10	13	19	24	32	36	44	47
L-VEGA	6	7	9	11	16	22	27	31	40	43

Table 4: Average generations for different numbers of tasks

Algorithms \ Number of tasks	10	20	30	40	50	60	70	80	90	100
GA	22	60	83	121	163	170	181	192	201	215
VEGA	20	29	52	61	71	77	87	84	93	100
L-VEGA	19	28	49	56	62	71	80	80	85	89

5.2.2 Fitness of different generations

We calculated the fitness of 25 to 200 generations. Tab. 5 and Fig. 8 show changes in fitness of generations of populations. Of them, the GA obtained the highest fitness. We see that it had the worst results, and its fitness was in the range of 0.80 to 0.84. VEGA was slightly better, in the range of 0.75 to 0.83. We think that because it used a learning mechanism, L-VEGA had a strong global optimization resistance to precocity, and its fitness was lowest.

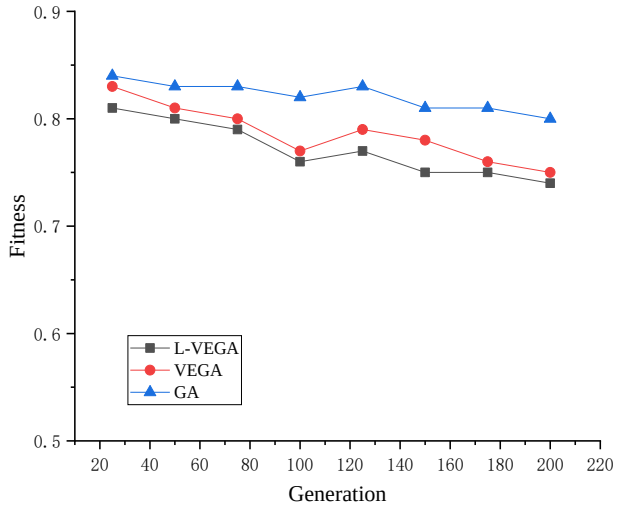


Figure 8: Fitness of different generations

Table 5: Fitness of different generations

Fitness \ Generation	25	50	75	100	125	150	175	200
GA	0.84	0.83	0.83	0.82	0.83	0.81	0.81	0.80
VEGA	0.83	0.81	0.80	0.77	0.79	0.78	0.76	0.75
L-VEGA	0.81	0.80	0.79	0.76	0.77	0.75	0.75	0.74

5.2.3 Average optimum

The average optimum is the average ratio of the optimal value to the maximum of the given instance. The initialization parameters were the same as above. The size of the host generation was kept at 100, and the average optimum (avg. opt.) of 100 instances was used as evaluation index.

As shown in Fig. 9, when the number of viruses is small, the optimal solution can be obtained by properly increasing the infection probability of the virus. However, it is easy to locate a local optimum if the infection probability is large. Increasing the size of the virus generation helps obtain the optimal solution, but the computation cost is high. In this experiment, when the size of the virus population was 100 and the maximum probability of infection was 0.1, L-VEGA generated the optimal order of scheduling and resource mode at a faster rate.

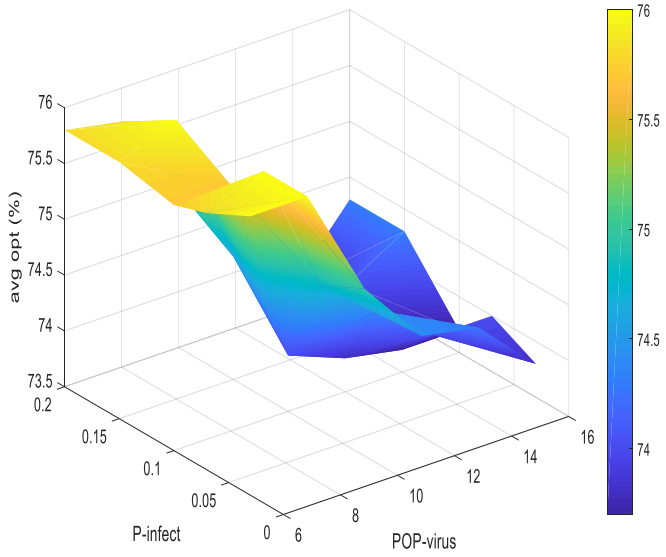


Figure 9: Impact of the size of the virus population and the maximum probability of infection

5.2.4 Average variance of task quantity

We randomly selected five on-site maintenance personnel to calculate the variance in the number of tasks assigned over a period of time. As shown in Fig. 10, variance for the GA is large and fluctuating, while L-VEGA and VEGA have smaller variance values. This is because the GA uses only the vertical search method, which easily reaches a local optimum. L-VEGA and VEGA use a combination of horizontal and vertical search to achieve a global optimum. L-VEGA considers load balancing in the objective function and uses a learning-based mechanism. This improves the speed of convergence and attains load balancing for tasks.

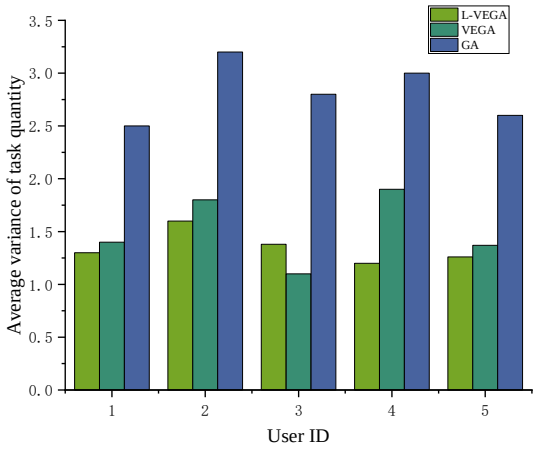


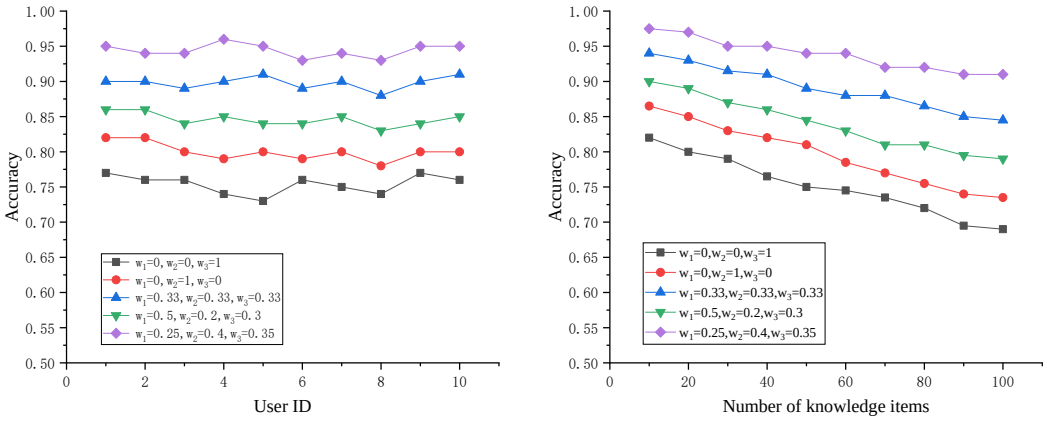
Figure 10: Average variance in task quantity

5.3 Results of simulation of CMKP

5.3.1 Accuracy under different conditions

To explore the impact of the user, equipment, and task on the accuracy of CMKP, we set different weights for them, as w_1, w_2, and w_3, respectively. The number of knowledge items was fixed at 50. When w_2=1, only the similarity of equipment was considered, and when w_3=1, only the similarity of tasks was.

In maintenance work for a communication network, knowledge push must consider the impact of the users, equipment, and tasks. Tab. 6 shows a comparison in terms of accuracy of knowledge for 10 random users under different weight settings. We see that regardless of weight setting, the general trend of CMKP is similar. As shown in Fig. 11(a), when w_1=0.25, w_2=0.4, and w_3=0.35, the accuracy of knowledge push was higher than in other cases. If only a single factor is considered, lower accuracy obtains.



(a) (b)

Figure 11: Accuracy under different conditions. (a) Different users. (b) Different items of knowledge

Tab. 7 and Fig. 11(b) show a comparison of accuracy of knowledge under different weight settings for different items. We see that as the number of knowledge items increases, accuracy under different weight settings declines. When w_1=0.25, w_2=0.4, and w_3=0.35, the accuracy of knowledge push is higher than for other settings.

Combining the above two conditions, CMKP has the highest accuracy when w_1=0.25, w_2=0.4, and w_3=0.35.

Table 6: Accuracy for different users

Weights \ User ID	1	2	3	4	5	6	7	8	9	10
w_1=0,w_2=0,w_3=1	0.77	0.76	0.76	0.74	0.73	0.76	0.75	0.74	0.77	0.76
w_1=0,w_2=1,w_3=0	0.82	0.82	0.80	0.79	0.80	0.79	0.80	0.79	0.80	0.80
w_1=0.33,w_2=0.33,w_3=0.33	0.90	0.90	0.89	0.90	0.91	0.89	0.90	0.88	0.90	0.91
w_1=0.5,w_2=0.2,w_3=0.3	0.86	0.86	0.84	0.85	0.84	0.84	0.85	0.83	0.84	0.85
w_1=0.25,w_2=0.4,w_3=0.35	0.95	0.94	0.94	0.96	0.95	0.93	0.94	0.93	0.95	0.95

Table 7: Accuracy for different numbers of knowledge items

Weights \ Number of knowledge items	10	20	30	40	50	60	70	80	90	100
w_1=0,w_2=0,w_3=1	0.82	0.80	0.79	0.76	0.75	0.74	0.73	0.72	0.69	0.69
w_1=0,w_2=1,w_3=0	0.86	0.85	0.83	0.82	0.81	0.78	0.77	0.75	0.74	0.73
w_1=0.33,w_2=0.33,w_3=0.33	0.94	0.93	0.91	0.91	0.89	0.88	0.88	0.86	0.85	0.84
w_1=0.5,w_2=0.2,w_3=0.3	0.90	0.89	0.87	0.86	0.84	0.83	0.81	0.81	0.79	0.79
w_1=0.25,w_2=0.4,w_3=0.35	0.97	0.97	0.95	0.95	0.94	0.94	0.92	0.92	0.91	0.91

5.3.2 Comparisons of popularity of knowledge

We use the rate of knowledge adoption to measure the popularity of knowledge. The higher the adoption rate of knowledge, the higher its intrinsic quality, and the greater the consequent assistance to maintenance work.

Tab. 8 and Fig. 12 show a comparison of the knowledge adoption rate achieved by eight random users using different knowledge push methods. We see that the feature points-based knowledge push algorithm has the worst results, and its knowledge adoption rate is in the range of 0.7 to 0.8. Content-based knowledge push algorithm is slightly better, and is in range of 0.8 to 0.9. We think these results obtained because the factors of similarity and popularity were considered when pushing knowledge, with the result that the adoption rate of the knowledge pushed by CMKP was higher than the other two algorithms.

Table 8: Adoption rates of pushed knowledge for different users

Algorithms \ User ID	1	2	3	4	5	6	7	8
CMKP	0.92	0.89	0.94	0.91	0.93	0.88	0.90	0.92
Content-based knowledge push	0.83	0.82	0.85	0.84	0.86	0.81	0.82	0.80
Feature points-based knowledge push	0.76	0.73	0.78	0.73	0.79	0.75	0.72	0.74

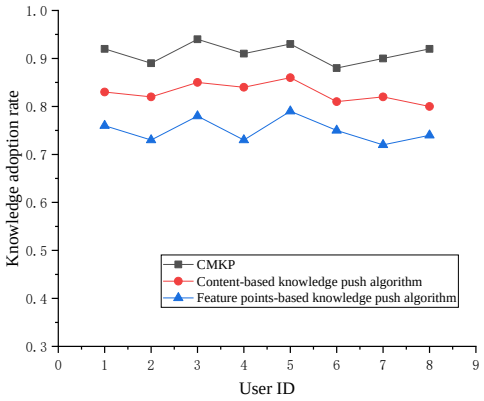
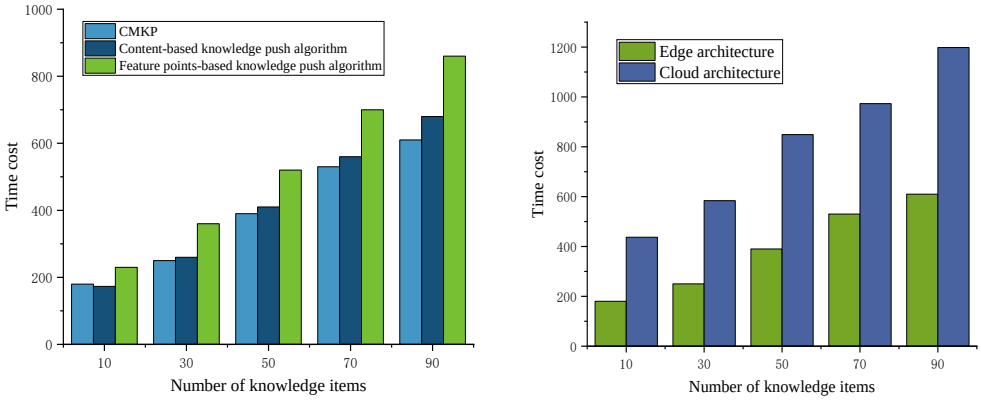


Figure 12: Adoption rates of pushed knowledge for different users

5.3.3 Time cost under different conditions

Fig. 13(a) shows the time cost of the three knowledge push algorithms with different numbers of knowledge items. It can be observed that the time cost of CMKP and the content-based knowledge push algorithm is similar, and less than the time cost of the feature points-based knowledge push algorithm. As the number of knowledge items increases, the difference becomes more apparent. This is explained by the fact that the feature points-based knowledge push algorithm must extract the feature points of the devices, which increases its complexity. Thus, the time cost is greater.

In Fig. 13(b), a time cost comparison of the CMKP algorithm based on two architectures is shown. Time cost based on the edge architecture is approximately half of that of the cloud architecture because with edge architecture, the relevant calculations of the CMKP algorithm can be processed at the edge server, which is closer to the users, and thus no data need to be passed to the data center for calculation.



(a) (b)

Figure 13: Time costs under different conditions. (a) Time costs under different numbers of knowledge items. (b) Time costs of CMKP under different architectures

6 Conclusion

In this study, we proposed an AR-based maintenance architecture using the concepts of edge computing, and context-based intelligent scheduling and knowledge push algorithms to quickly assist on-site maintenance personnel. The results show that L-VEGA and CMKP perform well for the maintenance work in a communication network.

Challenges persist in this domain, and require research, such as those relating to the optimization of database search, the dynamic update mechanism in context models, the methods of determining weights, and knowledge relationship in context models.

Acknowledgement: The work presented in this paper was supported by National Key Research and Development Program of China (2016YFE0204500), and Industrial Internet Innovation and Development Project 2018 of China.

References

Ali, A. S.; Osvaldo, S. (2017): Energy-efficient resource allocation for mobile edge computing-based augmented reality applications. *IEEE Wireless Communications Letters*, vol. 6, no. 3, pp. 398-401.

Amo, I. F. D.; Erkoyuncu, J. A.; Roy, R. (2018): Augmented reality in maintenance: an information-centred design framework. *Procedia Manufacturing*, vol. 19, pp. 148-155.

Aazam, M.; Huh, E. N. (2016): Fog computing: the cloud-IoT/IoE middleware paradigm. *IEEE Potentials*, vol. 35, no. 3, pp. 40-44.

Ai, Y.; Peng, M.; Zhang, K. (2017): Edge cloud computing technologies for internet of things: a primer. *Digital Communications and Networks*, vol. 4, no. 2, pp. 77-86.

Ahmed, W.; Hasan, O.; Pervez U. (2017): Reliability modeling and analysis of communication networks. *Journal of Network & Computer Applications*, vol. 78, pp. 191-215.

Bonchi, F.; Brogi, A.; Canciani, A. (2017): Simulation-based matching of cloud applications. *Science of Computer Programming*, vol. 162, pp. 110-131.

Cicirelli, F.; Guerrieri, A.; Spezzano, G. (2017): An edge-based platform for dynamic smart city applications. *Future Generation Computer Systems*, vol. 76, pp. 106-118.

Chao, H. E.; Meng, L. I.; Tingting, L. I. (2016): Comparison and analysis of the four methods of determining weights in multi-objective comprehensive evaluation. *Journal of Hubei University*, vol. 38, no. 2, pp. 172-178. (In Chinese)

Dai, Y. Y.; Lou, Y. S.; Lu, X. (2015): A task scheduling algorithm based on genetic algorithm and ant colony optimization algorithm with multi-QoS constraints in cloud computing. *IEEE International Conference on Intelligent Human-machine Systems & Cybernetics*, vol. 2, pp. 428-431.

Dolui, K.; Datta, S. K. (2017): Comparison of edge computing implementations: fog computing, cloudlet and mobile edge computing. *Global Internet of Things Summit*, pp. 1-6.

Fabian, B.; Christian, B. (2017): Knowledge-driven architecture composition: case-

based formalization of integration knowledge to enable automated component coupling. *IEEE International Conference on Software Architecture Workshops*, pp. 108-111.

Fiorentino, M.; Uva, A. E.; Gattullo, M. (2014): Augmented reality on large screen for interactive maintenance instructions. *Computers in Industry*, vol. 65, no. 2, pp. 270-278.

Gu, C. C.; Hu, J.; Wu, K. J. (2015): Quantitative behavioral knowledge modeling for functional case adaptation. *Research in Engineering Design*, vol. 26, no. 4, pp. 309-326.

Gupta, R. (2018): Appropriate AR modeling for surface electromyogram signals and its application in hand activity classification. *International Conference on Computing and Communication Technologies for Smart Nation*, pp. 276-280.

Hu, J.; Jiang, Z. (2017): Job scheduling integrated with imperfect preventive maintenance considering time-varying operating condition. *IEEE International Conference on Industrial Engineering and Engineering Management*, pp. 578-582.

Hou, X.; Lu, Y.; Dey, S. (2017): Wireless VR/AR with edge/cloud computing. *International Conference on Computer Communication and Networks*, pp. 1-8.

Liu, F.; Bi, L.; Yang, J. (2016): An improved genetic algorithm for cloud computing resource scheduling. *Computer Measurement & Control*, vol. 24, no. 5, pp. 202-206. (In Chinese)

Lopes, I. S.; Senra, P.; Neto, B. (2017): Multi-criteria classification for prioritization of preventive maintenance tasks to support maintenance scheduling. *IEEE International Conference on Industrial Engineering and Engineering Management*, pp. 2102-2106.

Lojka, T.; Bundzel, M.; Zolotova, I. (2016): Service-oriented architecture and cloud manufacturing. *Acta Polytechnica Hungarica*, vol. 13, no. 6, pp. 25-44.

Liu, T. Y.; Wang, H. F.; He, Y. (2016): Intelligent knowledge recommending approach for new product development based on workflow context matching. *Concurrent Engineering*, vol. 24, no. 4, pp. 318-329.

Palmarini, R.; Erkoyuncu, J. A.; Roy, R. (2017): A systematic review of augmented reality applications in maintenance. *Robotics and Computer-Integrated Manufacturing*, vol. 49, pp. 215-228.

Qi, J.; Zha, X. (2013): Optimization model and algorithm for China railway high-speed maintenance job work order scheduling. *International Conference on Computer Sciences and Applications*, pp. 233-236.

Qiao, X.; Ren, P.; Dustdar, S. (2018): A new era for web AR with mobile edge computing. *IEEE Internet Computing*, vol. 22, no. 4, pp. 46-55.

Rodriguez, F. J.; García-Martínez, C.; Blum, C. (2012): An artificial bee colony algorithm for the unrelated parallel machines scheduling problem. *International Conference on Parallel Problem Solving from Nature*, pp. 143-152.

Schneider, M.; Rambach, J.; Stricker, D. (2018): Augmented reality based on edge computing using the example of remote live support. *IEEE International Conference on Industrial Technology*, pp. 1277-1282.

Wang, Z. S.; Tian, L.; Wu, Y. H. (2016): Personalized knowledge push system based on design intent and user interest. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 230, no. 11, pp. 1757-1772.

Xu, Y. H.; Yin, G. F.; Nie, Y. (2013): Research on an active knowledge push service based on collaborative intent capture. *Journal of Network and Computer Applications*, vol. 36, no. 6, pp. 1418-1430.

Yang, K.; Gong, X.; Song, S. (2017): A fast AR system with high accuracy deployed on mobile devices. *International Conference on Parallel and Distributed Systems*, pp. 113-120.

Zhang, F.; Li, L. (2016): Research on knowledge push method for business process based on multidimensional hierarchical context model. *IEEE International Conference on Industrial Engineering and Engineering Management*, vol. 29, no. 4, pp. 751-758.

Zhao, Y.; Chen, L.; Li, Y. (2014): Efficient task scheduling for many task computing with resource attribute selection. *China Communications*, vol. 11, no. 12, pp. 125-140.