Wireless Communications and Mobile Computing

Smart Edge in Intelligent Industrial Systems

Lead Guest Editor: Xiaoxian Yang
Guest Editors: Muddesar Iqbal and Yuyu Yin

Smart Edge in Intelligent Industrial Systems

Wireless Communications and Mobile Computing

Smart Edge in Intelligent Industrial Systems

Lead Guest Editor: Xiaoxian Yang
Guest Editors: Muddesar Iqbal and Yuyu Yin

Contents

Malicious Code Classification Method Based on Deep Residual Network and Hybrid Attention Mechanism for Edge Security
Yanli Shao, Yang Lu, Dan Wei, Jinglong Fang, Feiwei Qin, and Bin Chen
Research Article (19 pages), Article ID 3301718, Volume 2022 (2022)

Hierarchical Social Recommendation Model Based on a Graph Neural Network
Zhongqin Bi, Lina Jing, Meijing Shan, Shuming Dou, and Shiyang Wang
Research Article (10 pages), Article ID 9107718, Volume 2021 (2021)

Popularity-Aware In-Network Caching for Edge Named Data Network
Jiliang Yin, Congfeng Jiang, Hidetoshi Mino, and Christophe Cérin
Research Article (13 pages), Article ID 3791859, Volume 2021 (2021)

Parallel Swarm Intelligent Motion Planning with Energy-Balanced for Multirobot in Obstacle Environment
Shoubao Su, Wei Zhao, and Chishe Wang
Research Article (16 pages), Article ID 8902328, Volume 2021 (2021)

BSVMS: Novel Autonomous Trustworthy Scheme for Video Monitoring
Xuan Wang, Jingjing Xu, Jiaxin Wang, Wei Ou, Jung Yoon Kim, and Wenbao Han
Research Article (15 pages), Article ID 6518743, Volume 2021 (2021)

An Approach to Modeling and Analyzing Reliability for Microservice-Oriented Cloud Applications
Zheng Liu, Guisheng Fan, Huiqun Yu, and Liqiong Chen
Research Article (17 pages), Article ID 5750646, Volume 2021 (2021)

A Combining Method for Wireless Protocol Conformance Testing: A Empirical Case
Lin Wei-Wei, Zeng Hong-Wei, and Jung Yoon Kim
Research Article (14 pages), Article ID 6378139, Volume 2021 (2021)

A Graph Convolutional Method for Traffic Flow Prediction in Highway Network
Tianpu Zhang, Weilong Ding, Tao Chen, Zhe Wang, and Jun Chen
Research Article (8 pages), Article ID 1997212, Volume 2021 (2021)

Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 3301718, 19 pages
https://doi.org/10.1155/2022/3301718

Research Article

Malicious Code Classification Method Based on Deep Residual Network and Hybrid Attention Mechanism for Edge Security

Yanli Shao, Yang Lu, Dan Wei, Jinglong Fang, Feiwei Qin, and Bin Chen

Key Laboratory of Complex Systems Modeling and Simulation, School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China

Correspondence should be addressed to Dan Wei; weiwd@hdu.edu.cn

Received 17 July 2021; Revised 8 June 2022; Accepted 17 June 2022; Published 7 July 2022

Academic Editor: Chakchai So-In

Edge computing is a feasible solution for effectively collecting and processing data in industrial Internet of Things (IIoT) systems, and edge security is an important guarantee for edge computing. Fast and accurate classification of malicious code in the whole lift cycle of edge computing is of great significance, which can effectively prevent malicious code from attacking wireless sensor networks and ensure the stable and secure transmission of data in smart devices. Considering that there is a large amount of code reuse in the same malicious code family, making their visual feature similar, many studies use visualization technology to assist malicious code classification. However, traditional malicious code visual classification schemes have the problems such as single image source, weak ability of deep-level feature extraction, and lack of attention to key image details. Therefore, an innovative malicious code visual classification method based on a deep residual network and hybrid attention mechanism for edge security is proposed in this study. Firstly, the malicious code visualization scheme integrates the bytecode file and assembly file of the malware and converts them into a four-channel RGBA image to fully represent malicious code feature information without increasing the computational complexity. Secondly, a hybrid attention mechanism is introduced into the deep residual network to construct an effective classification model, which extracts image texture features of malicious code from two dimensions of the channel and spatial to improve the classification performance. Finally, the experimental results on the BIG2015 and Malimg datasets show that the proposed scheme is feasible and effective and can be widely applied used in various malicious code classification issues, and the classification accuracy rate is relatively higher than the existing better-performing malicious code classification methods.

1. Introduction

In recent years, the fast expansion of the Internet of Things (IoT) has led to the industrial IoT (IIoT). Edge computing as a feasible solution for efficient collection and processing of data in IIoT has received great attention from academia, industry, and government departments and has been widely used in industries such as power, transportation, manufacturing, and smart cities. With the continuous deepening of the digital transformation process of the industry, the evolution of the edge computing network architecture will inevitably lead to an increasing number of security attacks on edge computing nodes, and edge security issues have become one of the obstacles restricting the development of the edge computing industry [1]. Nowadays, a large number of smart devices and sensors constitute a large wireless sensing network that can monitor, sense, and collect information from various monitored objects, while computing and storing these massive data. However, malware running on these ubiquitous sensors and smart devices can affect data security and cause other potential threats to data and IIoT devices [2]. Currently, the rapid increase in the types and quantity of malicious code not only brings property and economic losses but also gradually threatens national security [3]. For example, in May 2017, a computer ransomware called WannaCry spreads in more than 100 countries around the world. Many universities were infected and severely spreads to large public service areas such as airports, customs, and public safety networks [4]. In view of the weak security protection mechanism and limited computing resources of edge computing nodes, the detection and prevention of malicious code in the entire life cycle of edge computing is of great significance [5]. Malicious code classification is the key to preventing malicious code from running and improving information security and provides an important basis for malicious code detection, control, and removal.

Despite the continuous advancement of malicious code detection and classification technologies, malicious code has continuously evolved to generate new variants to avoid detection and quickly copied and spread, resulting in frequent security incidents in recent years. In most cases, the malicious code is generated or improved in an automated or semiautomated manner, and its core modules are reused during the generation process. Since the vast majority of new malicious codes are derived from the known malicious code mutations, there are generally less than 2% code differences between malicious codes of the same family [6]. This provides information security researchers with the basis for malicious code classification, that is, the detection and classification of different malicious code families can be achieved through visual feature similarity detection of malicious code core modules. The current mainstream malicious code detection and classification methods mainly include static analysis methods [7] and dynamic analysis methods [8]. The former refers to the analysis of malicious code without executing binary programs, which often fails to effectively solve the impact of packing and obfuscation technologies, while the latter refers to the use of program debugging tools to track and observe malicious code when it is executed and to verify the static analysis results according to the working process of the malware. This method is often inefficient and has a single execution path when dealing with large amounts of malicious code. Limited by computing power and resource consumption, traditional solutions perform poorly invariant similarity analysis of large-scale malicious code family samples. With the rapid development of deep learning technology and the increase in types and quantities of malicious code, researchers gradually began to convert the malicious code classification problem into an image classification problem. Malicious code visualization scheme based on deep learning has become a current research hotspot [9–11].

Malicious codes of the same family have similarities in visual features, but different families are different, which can be used as the basis for malicious code detection and classification. Thus, a malicious code visualization scheme transforms the problem of malicious code classification into an image classification problem and applies deep learning technology to solve it. Since different malicious code images reflect the differences in code data structure and information volume, the generation method of malicious code images is very important for malicious code classification. Currently, most of the existing malicious code visualization schemes only use bytecode files or assembly files and convert them into grayscale or RGB images for classification [10]. Some visualization schemes choose to calculate information entropy to enhance image information to further improve classification accuracy [11]. However, these methods have problems such as the single source of malicious code images and the large computational complexity of enhanced information, which increase the classification difficulty and reduce the classification accuracy to a certain extent. In addition, malicious code often exists in the local location of the program, manifesting as local image features. In malicious code visualization scheme, the commonly used convolutional neural network (CNN) pays more attention to the global image features and does not consider the detailed image features of the key regions. Therefore, it is necessary to introduce an attention mechanism to assign different weights to different regions in the image, so that the neural network can fully exploit and utilize the local detailed feature information of the malicious code image. In this way, the key image feature information is extracted through the attention mechanism, thereby improving the accuracy of subsequent malicious code detection and classification.

Based on the above analysis, this study proposes an innovative malicious code visualization classification method to further improve the classification accuracy and efficiency and then supports the detection and prevention of malicious code in edge computing. On the one hand, this scheme uses both the bytecode file and assembly file of the malware to visualize the malicious code as an RGBA image without additional calculation of code information entropy, which makes up for the defects of a single source of malicious code image information, insignificant image features, and excessive calculation. On the other hand, the hybrid attention mechanism is combined with the deep residual network to build a more accurate classification model. The deep residual network improves the classification accuracy while using shortcut connections to alleviate the gradient disappearance problem, accelerate model convergence, and improve the model's discriminative ability. Especially, each residual unit adopts a hybrid attention mechanism to extract more critical deep features from the two dimensions of channel and spatial to further improve the classification accuracy.

This study is organized as follows: Section 2 reviews the related work on malicious code classification. Section 3 introduces the core method, showing the detailed implementation of the proposed malicious code classification method from the malicious code visualization module and classification module, respectively. Section 4 presents the related experimental verification and performance analysis. The last section is the conclusion and future work.

2. Related Work

As mentioned above, edge security is an important guarantee for edge computing, wherein malicious code detection and prevention in the entire life cycle of edge computing is of great significance [1]. At present, malicious code visualization schemes have been developed on the basis of static analysis and dynamic analysis. Researchers have conducted extensive exploration and research on classification methods based on malicious code visualization. The key to improving the classification accuracy lies in how to extract reasonable and effective feature images to represent the program features of original malicious code as much as possible.

Conti et al. [12] pointed out that the malicious sample visualization method can help security analysts to quickly identify malicious code files. On this basis, Nataraj et al. [13] proposed a complete visual classification scheme for malicious samples, which mapped the malicious code to a grayscale image and extracted GIST features from it and finally implemented the malicious code classification through the K nearest neighbor (KNN) algorithm. They [14] also pointed out that the malicious code images of the same family have similar texture features, while the texture features of malicious code images of different families are quite different. Kornish et al. [15] found that appropriate improvements to images can improve the malware classification accuracy. Since then, malicious code visualization schemes have been enriched, the source of image information was no longer limited to bytecode files, and RGB images [16, 17] and RGBA images [18] were widely used. Wang et al. [16] divided the binary sequence of the malicious code file into RGB three-color channel values and converted the malicious sample into RGB images. Meanwhile, Sun et al. [17] used ASCII character information and PE structure information to convert malicious samples into RGB images and used VGG16 model to train and predict malicious code images. Chen et al. [18] used the bytecode file and local information entropy to convert the malware into RGBA images with larger information capacity, but this scheme increases the amount of calculation, and the image information source is single. These malicious code visualization schemes based on image features make up for the shortcomings of static analysis methods that are difficult to solve the problem of sample packing and confusion, as well as the long feature extraction time of dynamic analysis methods. Most of the aforementioned visualization schemes still follow Nataraj's grayscale scheme, using only bytecode files or assembly files, converting them to grayscale or RGB images for classification, or choosing to calculate information entropy to enhance image information to improve the classification accuracy. However, there are still the problems of single source of code images and high computational complexity. The feature information of malicious code images is not fully utilized, and the classification accuracy and efficiency still need to be improved.

As mentioned before, deep learning technology has power feature learning and expression ability, which makes it has outstanding advantages in extracting global features and contextual information of images. Currently, deep learning technology is widely used in various classification and prediction problems in different fields, such as hyperspectral image classification, IIoT security, and malicious code classification and detection [19–22]. Cheng et al. [9] explored an ensemble interpretable framework for automatic and efficient malicious code detection based on the knowledge graph of malware. Peng and Lu [23] proposed a discriminative extreme learning machine with supervised sparsity preserving (SPELM) model and verified the effectiveness of this model on four widely used image benchmark datasets. Pitolli et al. [24] proposed a novel approach for malware family identification based on an online clustering algorithm, which efficiently updates clusters as new samples are fed without rescanning the entire dataset. Cakir and Dogdu [25] used a shallow neural network based on the Word2Vec vector space model to represent the malicious code and finally applied the gradient search algorithm to classify the malicious code. Turnip et al. [26] proposed the eXtreme Gradient Boosting (XGBoost) to identify Android malware types. Liu et al. [27] combined graph neural networks with expert knowledge to realize smart contract vulnerability detection. Choi [28] proposed a malicious PowerShell detection method using GCN, which increased the detection rate of malicious PowerShell by approximately 8.2%. Wu et al. [29] proposed an attack-agnostic method based on cascaded self-supervised learning models [30] and achieved effective defense performance. With the development of the IIoT technology [31, 32], more and more users are beginning to use smart mobile terminal devices. Jaigirdar et al. [33] proposed the Prov-IoT model to maintain the data security of IoT devices. Zhou et al. [34] proposed a security defense system to protect the security of intelligent systems. However, the Android system is often attacked by malware due to its open source. Multimodal deep learning (MDL) performs well in complex scenes chosen to detect Android Malware by Kim et al. [35] and Vasu and Pari [36]. Ghouti and Imam [37] used principal component analysis (PCA) to extract the category and structural features of the malicious code and then used an optimized SVM to achieve malicious code classification. However, due to the structural characteristics of deep neural networks, such as focusing on global features and ignoring local details, some emerging research needs to be introduced to compensate for structural defects to comprehensively extract malicious code features and further improve the classification accuracy.

Since using the attention module in the CNN can focus on key information and improve the representation ability of convolution [38], more and more researchers [39] introduce attention mechanism into the field of malicious code classification and detection. Yakura et al. [40] built an ACNN malicious code detection model by combining CNN and attention mechanism to reduce the workload of analysts. Wang et al. [41] proposed a Depthwise Efficient Attention Module (DEAM) and combined it with a DenseNet to propose a new malware detection and family classification model. However, these schemes did not conduct in-depth research on the classification of malicious code families; there is still a huge potential research space for the application research of attention mechanism in malicious code visualization-based classification schemes.

3. Malicious Code Visual Classification Method

3.1. Method Overview. In order to solve the abovementioned problems in existing malicious code classification methods, this study proposes a malicious code visualization classification method based on a deep residual network and hybrid attention mechanism to achieve the accurate and efficient classification of malicious code. The overall flowchart is shown in Figure 1, and the details are as follows:

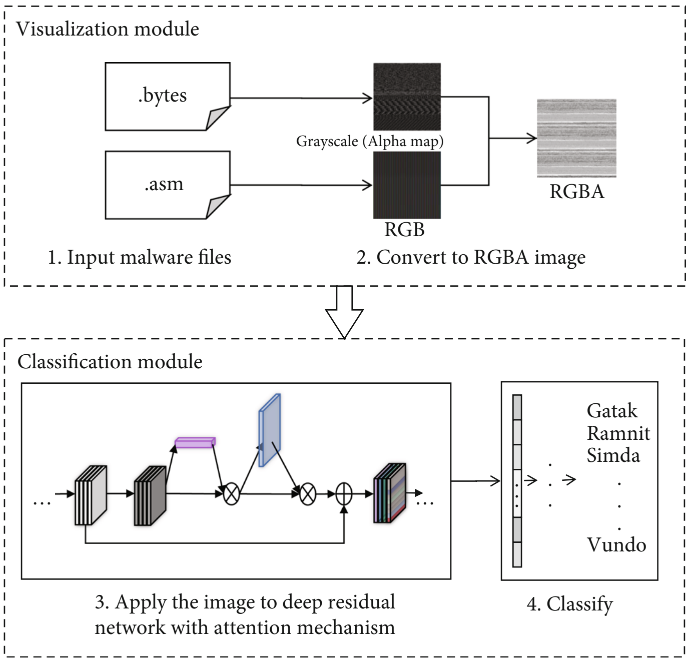


Figure 1: Method overview

(1) Malicious code visualization module: to compensate for the single source of malicious code image information, insignificant image features, and excessive computational complexity, a malicious code visualization scheme is proposed, which combines malware bytecode files and assembly files to form the RGBA images to enhance image information. This method converts the bytecode file into a grayscale image with a specified pixel size and also converts the assembly file into an RGB image of the same size to facilitate subsequent image fusion. Then, the value of the grayscale image as the transparency channel value is merged with the RGB image to form an RGBA image, so as to realize the visualization of malicious code while enhancing the effective information of the image without increasing the complexity of information calculation

(2) Malicious code classification module: in order to fully consider the key features of malicious code images and further improve the classification accuracy, a malicious code classification method combining a hybrid attention mechanism and a deep residual network is proposed. This method uses ResNet50 as the backbone network since the residual network can increase the accuracy by increasing the considerable network depth. The internal residual module uses shortcut connection to alleviate the problem of gradient disappearance caused by increasing depth of the network. Then, the channel attention module and the spatial attention module constitute a hybrid attention module, which is added to the residual unit of each convolution part of ResNet50 to improve the representation ability of the convolutional network. Combine the two to build a classification model, train the malicious code image dataset, and finally, realize the effective classification of malicious code

3.2. Malicious Code Visualization Module

3.2.1. Visual Problem Analysis. In an image, as the carrier of the malicious code file, each pixel contains a lot of code file information, and different malicious code images have different malicious code data structures and information amounts. For example, the images of malicious code of the Kelihos_ver1 family, the Vundo family, and the C2LOP.-gen!g family are shown in Figure 2. It can be seen that there are visual similarities between the malicious code images corresponding to the malicious code variants of the same family, while there are obvious visual differences between the malicious code images corresponding to different family variants. This difference in visual features shows that malicious code classification based on image similarity is feasible and effective. Thus, the generation method of malicious code images is very important for malicious code classification.

Generally, bytecode is a complied intermediate binary code that is independent of specific machine code and implementation platform. The assembly code is a low-level hardware-related assembly instruction compiled from the source code, which has poor cross-platform performance but relatively high execution performance. Bytecode and assembly code reflect different information about the code, but there is a close correlation between them. As a low-level language, assembly code has high scalability and lengthy code, so the assembly file of the same malicious code is longer and more informative than the bytecode file. Therefore, in order to comprehensively use image information to support the effective and accurate classification, in the malicious code visualization

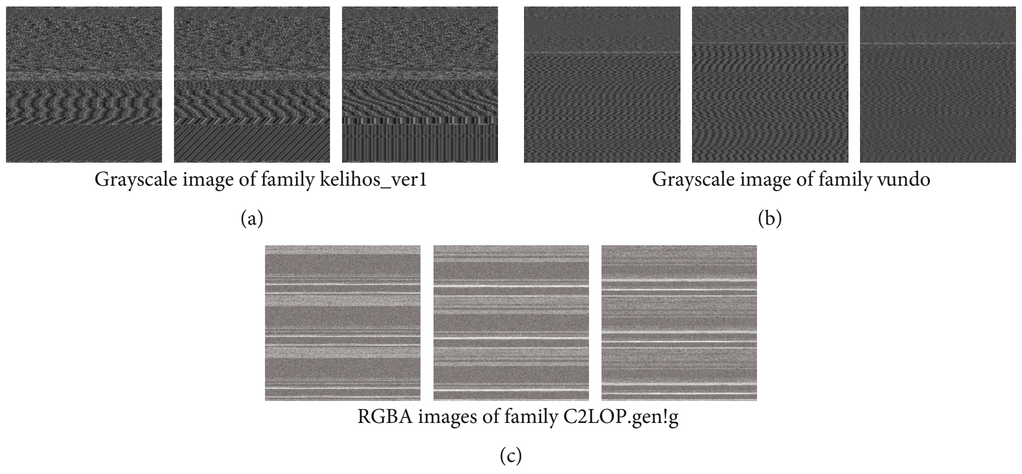

Grayscale image of family kelihos_ver1
(a)
Grayscale image of family vundo
(b)
RGBA images of family C2LOP.gen!g
(c)

Figure 2: Images of different families of malicious code.

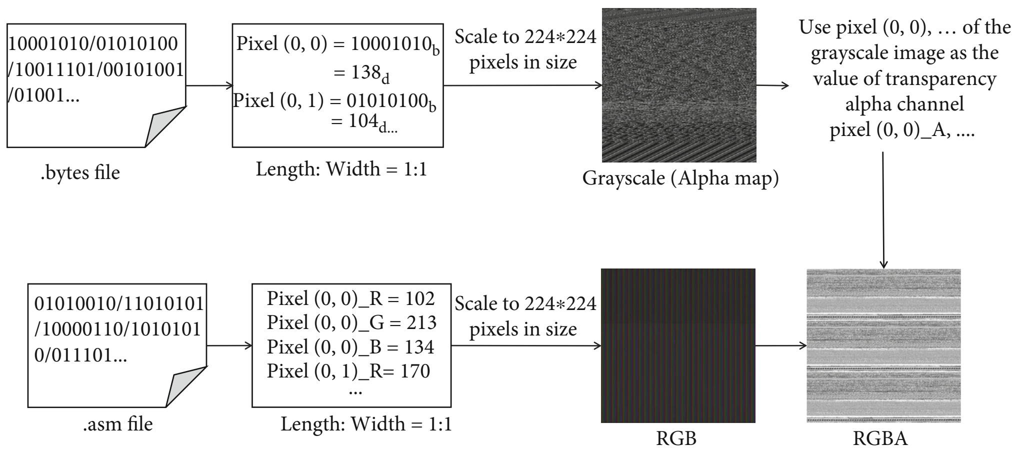


Figure 3: RGBA image generation flowchart.

module, we innovatively choose to use both the bytecode file and assembly file information to convert the malicious code into an RGBA image for subsequent classification. Compared with the grayscale image with only one channel, an RGBA image is an image with four channels by adding a transparency channel to an RGB image with three channels. Consequently, the effective information amount of RGBA images is 4 times that of grayscale images, which can provide more comprehensive and accurate image features for subsequent detection. Furthermore, RGBA images can not only carry more channel features but also can effectively fuse bytecode and assembly files.

3.2.2. RGBA Image Generation. Considering the differences in the amount of information between the assembly file and the bytecode file and the composition of the RGBA image, firstly, the assembly file and the bytecode file are converted to the same size ($224 * 224$ pixels) RGB image and grayscale image, respectively. Then, the grayscale image value is used as the transparency channel and merged with the RGB image generated by the assembly file to form an RGBA image. The RGBA image contains 4 channels, which are red, green, and blue color channels and transparency channel. Each channel has 8 bits and a total of 256 color levels. The malicious code file is read according to the binary data stream, and each 8-bit length ranges from 0 to 256, which exactly matches the length of each channel. In this case, the bytecode and assembly files are not added or deleted, and the bytecode and assembly features are similar in each local detail of the RGBA image after they are converted to images and fused. Thus, RGBA images can not only carry more channel features but also can effectively fuse bytecode and assembly files.

Based on the above analysis, suppose that the malware's .byte file is .byte = (..., 01101110, 10011100, 11010011, ...) = (..., 110, 156, 211, ...), and the .asm file is asm = (..., 01101100, 10011101, 11010010 ...) = (..., R:108, G:157, B:210, ...). The RGBA image generation flowchart and algorithm are shown in Figure 3 and Algorithm 1, and the specific steps are as follows:

(1) Read the malware's .bytes file by reading a binary data stream, and every 8 bits is converted to an unsigned integer vector. The value range of 8-bit unsigned integer is 0~255, which exactly corresponds to the pixel gray value 0~255. According to

length : width equal to 1 : 1, to generate a grayscale image

$$m_{\text{gray}} \in R^{1\times n\times n} = \begin{bmatrix} \cdots & 110 & 156 & 211 \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix}, \quad (1)$$

$n = \text{sqrt(file.length)}$, and then, scale the original grayscale image to gray image $m_{\text{gray}} \in R^{1\times 224\times 224} = \text{Image.resize}((224, 224), \text{Image.ANTIALIAS})$

(2) Read the malware's .asm file by reading the binary data stream as well, each 8 bits corresponds to the R, G, and B values of a pixel ($R_k = \sum_{i=0}^{7} b_{i+16} \times 2^i$; $G_k = \sum_{i=0}^{7} b_{i+8} \times 2^i$; $B_k = \sum_{i=0}^{7} b_i \times 2^i$), according to length : width equal to 1 : 1 to generate RGB image

$$m_{rgb} \in R^{3\times n\times n} = \left\{ \begin{bmatrix} \cdots & R:108 & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix} \cdot \begin{bmatrix} \cdots & G:157 & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix} \cdot \begin{bmatrix} \cdots & B:210 & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix} \right\}, \quad (2)$$

$n = \text{sqrt(file.length/3)}$ and then scale the original RGB image to RGB image $m_{rgb} \in R^{3\times 224\times 224} = \text{Image.resize}((224, 224), \text{Image.ANTIALIAS})$

(3) The gray value of the gray image is used as the transparency channel of the RGBA image, and it is merged with the RGB image generated by the .asm file to form an RGBA image $m_{rgba} \in R^{4\times 224\times 224}$

When the malicious code file is converted into an image, the zero-padded operation is performed instead of intercepting part of the file content, which ensures the source integrity of the malicious code image information to a certain extent. The parameter of the resize() function is set to Image.ANTIALIAS, which will perform high-quality compression on the image to ensure that the image quality will not be reduced when the image size changes. In this way, the RGBA image contains 4 channels, and the effective information contained is 4 times that of the grayscale image, which can provide more potential malicious code features and effectively support the subsequent malicious code classification.

3.3. Malicious Code Classification Module. After obtaining the malicious code image dataset through the above malicious code visualization module, the next step is to build an accurate malicious code classification model. In the malicious code classification module, we propose an innovative malicious code classification model based on the deep residual neural network - ResNet50 [42] and the attention module structure of Woo et al. [38], which combines a hybrid attention mechanism with the deep residual network to further improve classification accuracy. On the one hand, the deep neural network is used to improve classification accuracy by increasing the structural depth. Meanwhile, the residual structure can effectively avoid the problem of gradient disappearance through the shortcut connection. On the other hand, a hybrid attention mechanism is applied and injected into the residual network to effectively capture the key features of malicious code images and assign different learning weights, so that the model can learn the image features that need to be focused to further improve the classification accuracy. Moreover, the application of the attention mechanism adds less parameters and calculation amount, which can ensure the classification effect of the model without affecting the classification efficiency.

The overall network architecture of the classification model Mcs - ResNet is shown in Figure 4, containing 5 convolution parts (conv1~conv5). Among them, conv2_x, conv3_x, conv4_x, and conv5_x are formed by adding a hybrid attention module to the residual unit of the convolution part, to ensure the full integration of the hybrid attention module and the deep residual network to further enhance the mining of deep features. Moreover, the detailed parameter information of the model is shown in Table 1. Here, a 50-layer ResNet model with 3 layers of bottleneck blocks is chosen as the base network for malicious code classification. Therefore, the model complexity is about 3.8 billion FLOPs (floating-point operations) and so is the parameter size.

The convolutional layer implements the feature extraction and feature mapping, weight sharing, and local connection of the input image through the convolution filter in CNN. Generally, in the convolution process, the convolution filter often has multiple channels, and the filters of multiple channels usually perform feature extraction at the same time. For example, when the input image is $m_{i,j,k} (0 \le i \le W, 0 \le j \le H, 0 \le k \le K)$, that is, the image size is $W \times H$ and the channel is K, the convolution processing is shown in

$$m'_{i,j,k} = \sum_{l=0}^{K-1} \sum_{p=0}^{M-1} \sum_{q=0}^{M-1} m_{i+p,j+q,k+l} w_{p,q,l} + b_{i,j,k}, \quad (3)$$

where $b_{i,j,k}$ represents the bias of the neural network and $w_{p,q,l}$ represents the weight of K convolution filters with a size of $M \times M$.

```
Input: The bytecode file $\text{file}_{\text{bytes}}$ and assembly file $\text{file}_{\text{asm}}$ of the malicious code;
Output: The final training dataset RGBA images $m_{rgba} \in R^{4\times224\times224}$.
For each sample $\text{file}_{\text{bytes}}$:
        Calculate the width of the image $\text{width}_{\text{bytes}} = \text{sqrt}(\text{file}_{\text{bytes}}.\text{length})$;
        Calculate the gray value corresponding to each pixel, $\text{gray}_n = \sum_{i=0}^{7} b \times 2^i$, form a grayscale image;
        Scale the original grayscale image to 224 * 224 pixel size gray image $m_{\text{gray}} \in R^{1\times224\times224} = \text{Image.resize}((224, 224), \text{Image}$
.ANTIALIAS)..
End
For each sample $\text{file}_{\text{asm}}$:
    Calculate the width of the image $\text{width}_{\text{asm}} = \text{sqrt}(\text{file}_{\text{asm}}.\text{length}/3)$;
    Calculate the R, G, B value of each pixel, $R_k = \sum_{i=0}^{7} b_{i+16} \times 2^i$; $G_k = \sum_{i=0}^{7} b_{i+8} \times 2^i$; $B_k = \sum_{i=0}^{7} b_i \times 2^i$, form an RGB image;
    Scale the original RGB image to 224 * 224 pixel size RGB image $m_{rgb} \in R^{3\times224\times224} = \text{Image.resize}((224, 224), \text{Image}.$
ANTIALIAS).
End
For each image $m_{\text{gray}}$ and $m_{\text{rgb}} \in R^{3\times224\times224}$:
    The gray value of $m_{\text{gray}}$ is used as the $A$ value of the RGBA image;
    Merged with $m_{\text{rgb}}$ to form an RGBA image $m_{rgba} \in R^{4\times224\times224}$.
End
```

Algorithm 1:RGBA image generation algorithm.

As shown in the lower part of Figure 4, the hybrid attention module is composed of a channel attention module and a spatial attention module to simultaneously obtain the channel feature weights and spatial feature weights of the malicious code image, thereby enhancing the obtained important features. After that, the enhanced features and the original input image features are connected through the shortcut connection structure to obtain the final output features. The channel attention module and the spatial attention module emphasize the special regions of the malicious code image to enhance the accuracy of malicious code image classification. The following describes the residual module and hybrid attention module and their combination in detail.

3.3.1. Residual Module. The deep learning model is usually composed of multiple layers, and its deep structure has powerful learning capabilities and efficient feature expression capabilities to automatically learn features from a large amount of data. It is widely used in image recognition, speech recognition, and other fields, and has become an important part of computer vision technology. The network depth of a deep learning model determines whether it can extract deeper features, but as the network depth continues to deepen, it will cause network degradation and gradient disappearance problems. The residual network proposes a shortcut connection technique to solve the above problems. The input is transferred across layers and added to the result of the convolution, and the identity mapping is added, as shown in Figure 5(a). When the network input is x, the learned feature is $F(x) + x$, that is, the unit input and output are directly added, and then activated by the ReLU activation function. This network structure does not add additional parameters, which facilitates the subsequent network optimization and greatly improves the training efficiency. Based on these characteristics of the residual network, the attention module is injected into the residual network to construct a residual attention network to simultaneously utilize the advantages of both, as shown in Figure 5(b).

Here, the proposed malicious code classification model Mcs-ResNet uses the ResNet50 residual network as the backbone network. ResNet50 is a deep residual network formed by adding a shortcut connection mechanism on the basis of the VGG19 network. The network structure of the traditional CNN model is directly stacked, which is equivalent to multiplication calculation. In this ResNet model, the network structure is connected through a shortcut connection, and the calculation is changed from multiplication to addition. The feature calculation under this structure will be more stable, so the original feature information in the malicious code image and the key feature information processed by the attention module will flow to the next layer more stably, and the malicious code image classification will be more efficient.

Based on the above analysis, the expression of the RGBA image $m \in R^{4\times224\times224}$ processed by the residual module is as follows:

$$m' = m + f(F_S(F_C(m))), \tag{4}$$

where f represents operations such as feature mapping, activation, and attention weighting; F_S is the spatial attention weight; and F_C is the channel attention weight. The specific calculation of F_S and F_C will be described in the next section. At this time, the features of the RGBA image are not compressed, so that the channel and spatial features can be learned more fully after adding the attention module. This ensures that more critical deep features in the two dimensions flow more stably to the next layer.

3.3.2. Hybrid Attention Module. As mentioned earlier, the use of attention mechanism in CNNs can focus on key

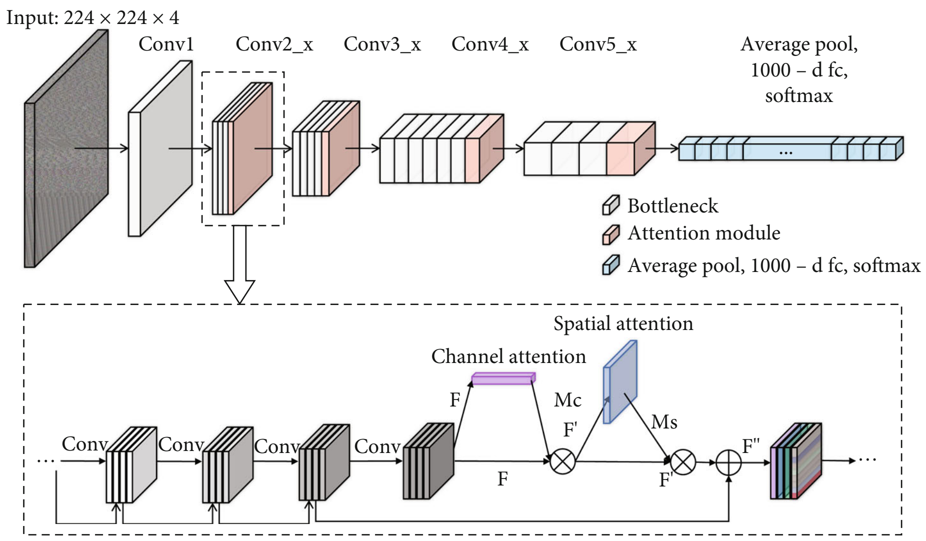


Figure 4: The network architecture of Mcs-ResNet.

Table 1: Network architecture parameter information.

Layer name	Output size	Model
conv1	112×112	7×7, 64, stride 2
		3×3 max pool, stride 2
conv2_x	56×56	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3 \oplus$ **Attention**
conv3_x	28×28	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4 \oplus$ **Attention**
conv4_x	14×14	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6 \oplus$ **Attention**
conv5_x	7×7	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3 \oplus$ **Attention**
	1×1	Average pool, 1000-d fc, softmax
FLOPs		3.8×10^9

information and improve the convolutional representation ability. Therefore, an attention module is added after the residual module to focus on key information and weaken the useless information. The one-dimensional channel attention feature matrix and the two-dimensional spatial attention feature matrix are derived in turn, and then, the generated attention feature matrix is multiplied with the original input feature matrix to form the output feature matrix, which enables the classification model to focus on key areas with higher correlation with malicious behaviors for more accurate classification.

(1) Channel Attention Module. Compared with grayscale images or RGB images, RGBA images contain richer information and more channels. Using the CNN with channel attention for classification can assign different weights to each channel, thereby effectively improving the classification accuracy of malicious code. In the CNN, the two-dimensional malicious image will generate an image feature matrix (H, W, C) after the convolution kernel operation, where H, W represent the image height and width, and C represents the image feature channel. Introducing the channel attention mechanism into the malicious code classification model can effectively strengthen the model's extraction of global texture features of malicious code images. The channel attention module can pay attention to the importance of different feature channels of the input image. By modeling the importance of each feature channel, assign different weights to the channel features, and strengthen or suppress different channels according to the degree of correlation with malicious behavior.

The operation process of the channel attention module is shown in Figure 6, and the specific steps are as follows: firstly, the output feature matrix of the previous layer of convolution is used as the intermediate input feature. Then, the intermediate feature matrix obtains two-channel descriptions in the form of $1 \times 1 \times C$ through average pooling and maximum pooling based on spatial dimensions to compress the spatial dimensions of the input feature matrix and gather spatial information. The feature information is extracted from different angles, the importance of each feature channel is modeled, and the channel features are assigned weights, thereby effectively utilizing the special interaction relationship between the channels of the intermediate feature matrix obtained after convolution. Afterwards, through the adjustment of the shared network multilayer perceptron, the output vector dimension should match the number of channels of intermediate feature matrix, and the adjusted vector elements are added together and activated by the

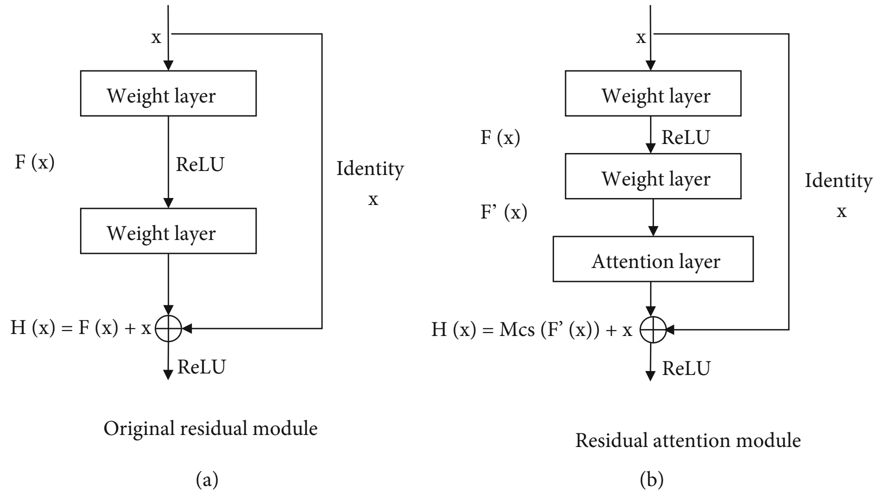


Figure 5: Residual module.

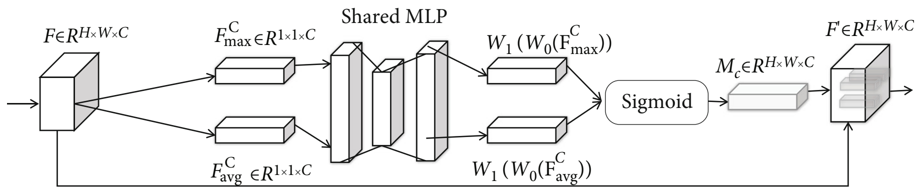


Figure 6: The operation process of the channel attention module.

Sigmoid function, realizing the enhancement or suppression of different channels as needed. The expression of the channel attention module is shown in formula (5).

$$M_C(F) = \sigma(\mathrm{MLP}(\mathrm{AvgPool}(F)) + \mathrm{MLP}(\mathrm{MaxPool}(F))),$$
$$= \sigma\left(W_1\left(W_0\left(F^C_{\mathrm{avg}}\right)\right) + W_1\left(W_0\left(F^C_{\mathrm{max}}\right)\right)\right), \tag{5}$$

where σ denotes the Sigmoid function, $W_0 \in R^{(C/r)\times C}$ and $W_1 \in R^{C\times(C/r)}$, r is the compression ratio. Note that W_0 and W_1 are weights of the multilayer perceptron (MLP), shared by the input features and the ReLU activation function of W_0. F^C_{avg} and F^C_{max} represent the spatial matrix generated by the average pooling and the maximum pooling.

Finally, the channel attention module output matrix and the input intermediate feature matrix are weighted and summed channel by channel to complete the channel attention calculation of the output feature matrix. On the basis of the residual module, combined with the channel attention module, it can retain more global texture information in the input malicious code image, greatly improving the malicious feature representation ability.

(2) Spatial Attention Module. Since most new malicious codes are derived from existing malicious code mutations, their core modules are repeatedly rewritten to generate new malicious code. Hence, the key to malicious code variant detection is how to extract the core module feature information and how to assign different weights to different regions in the image to focus on key feature information to improve the detection and classification accuracy of malicious code variants. The spatial attention module focuses on the importance of different feature spatial locations, generates spatial attention weights for the output feature map, and enhances the spatial location features with higher correlation with malicious behavior according to the feature weights.

The operation process of the spatial attention module is shown in Figure 7, and the specific steps are as follows: firstly, take the feature matrix processed by the channel attention as the intermediate input feature, and perform average pooling and maximum pooling, respectively, based on channel dimensions to obtain two spatial description matrices in the form of $H \times W \times 1$. This will not only consider the contribution of local malicious code image space but also can capture the contribution of global space. Next, the two spatial description matrices are merged into a feature matrix, and a two-dimensional spatial attention map is generated through the convolutional layer to better fit the spatial complexity correlation. Thus, adding a spatial attention module to the classification model can improve the learning ability in key regions with higher correlation with malicious behavior and complements the channel attention, thereby further improving the classification accuracy. Finally, the spatial attention map can be generated after the activation of the Sigmoid function. The spatial attention module is shown in formula (6), where σ represents the

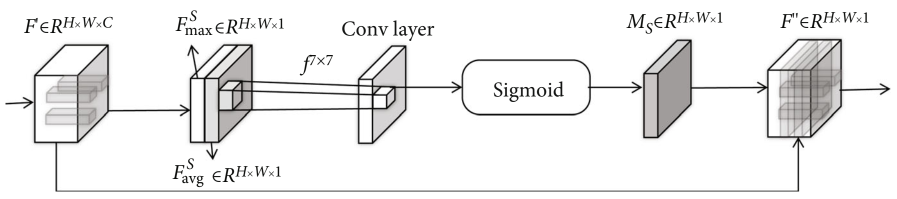

$F \in R^{H \times W \times C}$
$F^S_{max} \in R^{H \times W \times 1}$
Conv layer
$f^{7 \times 7}$
Sigmoid
$M_S \in R^{H \times W \times 1}$
$F'' \in R^{H \times W \times 1}$
$F^S_{avg} \in R^{H \times W \times 1}$

Figure 7: The operation process of the spatial attention module.

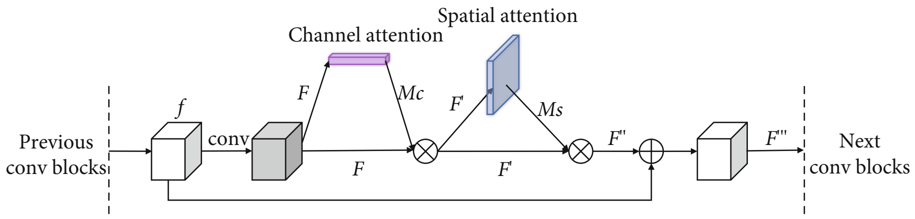


Figure 8: Local structure of the classification model.

sigmoid function, $f^{7\times7}$ represents the convolution operation, and the size of the convolution kernel is 7×7. F^S_{avg} and F^S_{max} also represent the matrices generated by the average pooling and maximum pooling.

$$\begin{aligned} M_S(F) &= \sigma\left(f^{7\times7}(\mathrm{AvgPool}(F)\,;\mathrm{MaxPool}(F))\right), \\ &= \sigma\left(f^{7\times7}\left(F^S_{avg}\,;F^S_{max}\right)\right). \end{aligned} \tag{6}$$

(3) Hybrid Attention Mechanism. The classification model proposed in this study extracts malicious code features by fusing channel attention and spatial attention. The channel attention module focuses on the global feature information between each channel, and the spatial attention module focuses on the local feature information within the channel. The combination of the two forms a hybrid attention mechanism, which supports the learning of key features and further improves classification accuracy. Woo et al. [38] proved that the channel attention module and the spatial attention module can be arranged in parallel or sequentially, but the sequential arrangement has better performance, and the model performance with the channel attention module priority is slightly better than the spatial attention module priority. The reason is that the channel attention focuses on "what" is critical and meaningful in an input image, and the spatial attention focuses on "where" is an informative part, which is complementary to the channel attention. Therefore, the priority order of the channel attention module is used in the proposed classification model.

3.3.3. Classification Model Structure. Based on the above analysis, the classification model structure that combines the residual module and hybrid attention mechanism is shown in Figure 8. Firstly, perform a convolution operation on the features generated in the previous layer to generate the input feature F. F passes through the channel attention module to obtain the importance of each feature channel, so that the model pays more attention to the channel related to malicious behavior with high weight and suppresses the channel with low correlation, so as to obtain the channel attention feature Mc. The corresponding matrix elements are multiplied by F and Mc to extract the features from the spatial dimension, improve the classification model's ability to extract local texture features, and obtain the new feature F'. Then, the F' is used as the input feature of the spatial attention module to obtain the spatial attention feature Ms. Ms and F' are multiplied by the corresponding matrix elements to obtain the mixed feature F''. Finally, F'' is added to the features generated in the previous layer to generate feature F''' as the input of the next module.

The whole attention calculation process is shown in formulas (7)–(9). This process strengthens the feature information between channels in the global features of the malicious code and the local location information within the channels, thereby improving the classification performance.

$$F' = M_C(F) \otimes F, \tag{7}$$

$$F'' = M_S\left(F'\right) \otimes F', \tag{8}$$

$$F''' = F'' \oplus f. \tag{9}$$

In order to fully learn the image characteristics of malicious code and improve the performance of the attention module, a hybrid attention module is added after each residual unit instead of just adding it once. Therefore, when the next module performs the deep convolution operation, the features learned by the attention module in the previous module will be retained to continue learning. Moreover, although channel attention and spatial attention are arranged sequentially, they are also connected by identity mapping, which can prevent information of different dimensions from interfering with each other.

Table 2: Malware dataset.

Dataset	Family	Number of samples	Family	Number of samples
BIG2015	Gatak	1013	Ramnit	1541
	Kelihos_ver1	398	Simda	42
	Kelihos_ver3	2942	Tracur	751
	Lollipop	2478	Vundo	475
	Obfuscator.ACY	1288		
Malimg	Adialer.C	125	Lolyda.AA2	184
	Agent.FYI	116	Lolyda.AA3	123
	Allaple.A	2949	Lolyda.AT	159
	Allaple.L	1591	Malex.gen!J	136
	Alueron.gen!J	198	Obfuscator.AD	142
	Autorun.K	106	Rbot!gen	158
	C2LOP.gen!g	200	Skintrim.N	80
	C2LOP.P	146	Swizzor.gen!E	128
	Dialplatform.B	177	Swizzor.gen!I	132
	Dontovo.A	162	VB.AT	408
	Fakerean	381	Wintrim.BX	97
	Instantaccess	431	Yuner.A	800
	Lolyda.AA1	213		

Table 3: Experimental equipment environment.

Hardware	Description	Software	Description
GPU	GTX1060 6GB	CUDA	10.0
CPU	Intel i7-7700	cuDNN	7.6.5
RAM	16 GB	Language	Python

4. Experiment and Performance Analysis

4.1. Experimental Preparation

4.1.1. Experimental Dataset. The experimental dataset used in this study are the BIG2015 dataset (https://www.kaggle.com/c/malware-classification/data) and the Malimg dataset (https://www.kaggle.com/keerthicheepurupalli/malimg-dataset9010). The BIG2015 dataset is a 500 G malware file dataset released by Microsoft on Kaggle during its malware classification challenge in 2015, which includes assembly files and bytecode files of more than 20,000 malware samples. In addition to providing services in Kaggle competitions, the BIG2015 dataset has become a standard benchmark for studying malware behavior modeling. So far, it has been cited by more than 50 research papers. Therefore, this dataset is used here to verify the performance of the proposed malicious code classification model. In order to facilitate the performance verification, the labeled training dataset which consists of 10868 malware samples from 9 families is selected as the experimental dataset, as shown in the upper part of Table 2, and divided into a training dataset and test dataset according to the ratio of 8 : 2.

The Malimg dataset is released by the Advanced Visualization Research Project of the Visual Research Laboratory under the University of California-Santa Barbara. They first proposed a malicious code visualization method for malicious code detection and classification. In 2011, this team constructed the Malimg dataset and published the code visualization method to promote software security research. This dataset contains a total of 9342 samples from 25 family categories, as shown in the lower part of Table 2. Furthermore, the Malimg dataset is composed of grayscale images converted from malware bytecode files.

4.1.2. Experimental Settings. The experimental environment is shown in Table 3.

The stochastic gradient descent (SGD) algorithm with momentum can effectively suppress the oscillation of SGD and accelerate the convergence speed. The data distribution of the model in this paper is relatively uniform and can be well adapted to the SGD algorithm for model optimization. Therefore, the experiment uses the SGD algorithm with momentum optimization to update the model parameters to improve the computational efficiency, and the momentum is set to 0.9. A total of 2000 epochs are trained, and the training batch samples are 16. The dynamic attenuation learning rate is used, and the initial learning rate is set to 0.01, and the classification function is softmax.

Since the classification of malicious code families is a multiclassification problem, in order to facilitate comparison with other models and better measure the classification performance, the arithmetic average of the accuracy of various malicious code families is taken as the standard for performance evaluation. Here, TP is defined as the number of malicious samples classified as malware, TN is defined as the number of benign samples classified as benign, FP is the number of benign samples classified as malware, and FN is the number of malicious samples classified as benign. Thus, accuracy (Acc) is defined as follows:

$$\mathrm{Acc} = \frac{1}{n}\sum_{i=1}^{n}\frac{\mathrm{TP}_i + \mathrm{TN}_i}{\mathrm{TP}_i + \mathrm{FN}_i + \mathrm{TN}_i + \mathrm{FP}_i}. \tag{10}$$

4.2. Ablation Experiment and Analysis. Two sets of ablation experiments are conducted to verify the feasibility and effectiveness of the proposed malicious code visualization and classification module, which includes visualization scheme validity verification and hybrid attention module performance analysis. The comparative experiments all use the classification accuracy (Acc) as the evaluation index to facilitate comparison.

4.2.1. Visualization Scheme Validity Verification. The first experiment applied the malicious code images obtained from different visualization schemes to the classic classification models and the proposed classification model - Mcs-ResNet for comparative analysis. The classic classification models include VGG16, VGG19, and ResNet50 pretraining models for feature extraction. The KNN model is used as the classifier, where k is 5. This group of experiments uses the bytecode files and assembly files provided by the

Table 4: Classification effect of different visualization schemes.

No.	Image	VGG16+KNN5	VGG19+KNN5	ResNet50+KNN5	Mcs-ResNet
1	Byte-gray	88.05	87.33	89.17	89.03
2	Byte-RGB	88.59	88.41	89.98	91.83
3	ASM-gray	89.98	90.77	92.07	89.12
4	ASM-RGB	91.53	90.19	94.34	90.04
5	Byte-gray (224 ∗ 224)	93.64	93.84	93.01	96.72
6	Byte-RGB (224 ∗ 224)	92.54	92.52	91.79	92.98
7	ASM-gray (224 ∗ 224)	93.95	93.49	93.86	96.98
8	ASM-RGB (224 ∗ 224)	94.26	94.41	94.70	96.70
9	RGBA (224 ∗ 224)	**94.69**	**95.23**	**95.12**	**97.21**

BIG2015 dataset for verification. In addition to using the proposed visualization scheme to convert it into RGBA images, only bytecode files or assembly files are converted into grayscale and RGB images, and the image size is changed to show the classification effect based on different visualization schemes.

The experimental results are shown in Table 4, wherein the grayscale image and RGB image generated from the bytecode file are marked as Byte-gray and Byte-RGB, respectively, and the grayscale image and RGB image generated from the assembly file are marked as ASM-gray and ASM-RGB, respectively, and the RGBA images are RGB images that contain transparency information, and 224 ∗ 224 represents the image size. And according to the experimental results, the following conclusions can be drawn:

(1) According to the experimental results of nos. 1-4, the classification effect of converting bytecode files into grayscale or RGB images is almost the same, while the classification effect of converting assembly files into RGB images is better than that of grayscale images. For example, in the classification model ResNet50+KNN5, the accuracy of the bytecode file converted into the two types of images is 89.17% and 89.98%, respectively, with a difference of only 0.81%, while the accuracy of RGB image converted from the assembly file is 2.27% higher than that of grayscale image. The reason is that the assembly file size of the same malware code is much larger than that of the bytecode file. The grayscale image can effectively represent the bytecode file but not the assembly file. Therefore, the classification effect of the bytecode file converted into two kinds of images is similar, and the classification effect of the assembly file converted into RGB image is better.

(2) From the comparison of nos. 5-8 and nos. 1-4, it can be seen that the classification effect of prescaling the image to a uniform size (224 ∗ 224 pixels) with high quality is significantly better than that of directly inputting the original pixel size image. In the classification model composed of VGG19 and KNN5, the accuracy on the original images are 87.33%, 88.41%, 90.77%, and 90.19%, while the accuracy on the corresponding 224 ∗ 224 pixel images improved by 6.51%, 4.11%, 2.72%, and 4.22%, respectively. Meanwhile, in the Mcs-ResNet model, it also improves 7.69%, 1.15%, 7.86%, and 6.66%, respectively, all of which are significantly improved. The reason is that when the image is scaled in the preprocessing, the parameter of the resize() function is set to *Image.ANTIALIAS*. This is an operation for high-quality image compression, and the original image features can be preserved to the greatest extent. But the original image is directly compressed to 224 ∗ 224 pixel size when the data is loaded. This is low-quality processing, resulting in the loss of a large amount of effective information in the original image and poor classification effect. In addition, the performance of the Mcs-ResNet model is better than other models, i.e., in the no. 7 and no. 8 experiments; its accuracy is 3.49% and 2.29% higher than that of the VGG19+KNN5 model, respectively.

(3) It can be seen from no.8 and no. 9 that the classification effect based on RGBA images is better than those based on other images. All four models achieved the best classification accuracy on RGBA images. Especially, the classification accuracy of the proposed model is 97.21%, which is 2.95%, 2.8%, and 2.51% higher than the previous three models based on RGB images (no.8). It is also 2.52%, 1.98% and 2.09% higher than the previous three models based RGBA images (no.9). The information source of RGBA image is composed of bytecode files and assembly files. Therefore, the information source is richer, and the amount of information contained is larger than the grayscale image and RGB image, which can better describe the features of malicious code images.

Based on the above analysis, the proposed visualization scheme is feasible and effective and shows good classification performance in different classifiers. Thus, when fusing bytecode files and assembly files, it is a reasonable choice to use bytecode files as the data source of the transparency channel and the assembly files as the R, G, and B channel data sources. This operation can deeply exploit and utilize the

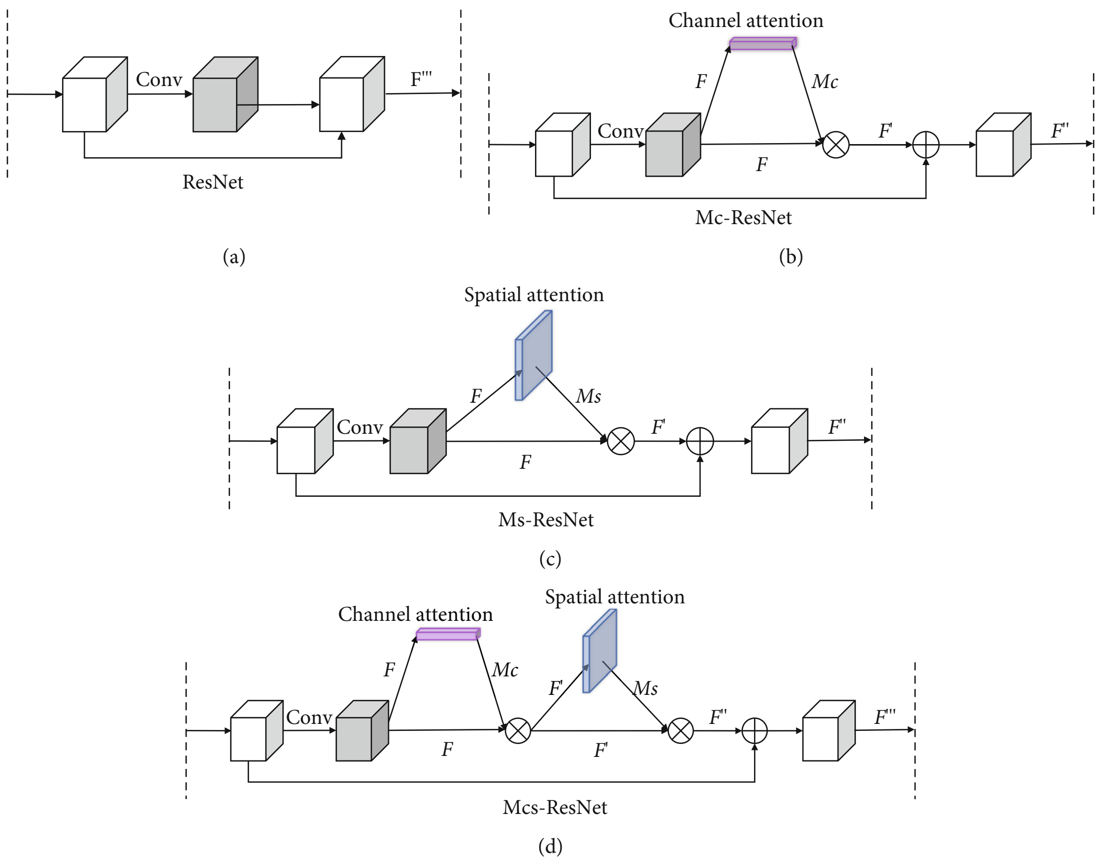


FIGURE 9: Four models with different attention module structures.

feature information of malicious code images and effectively support the accurate classification of malicious codes.

4.2.2. Hybrid Attention Module Performance Analysis. In order to verify the classification performance of different attention mechanisms on malicious code image classification, four sets of comparative experiments are conducted here, including the ResNet model without an embedded attention module; the ResNet model with only the channel attention module embedded (Mc-ResNet); the ResNet model with only the spatial attention module embedded (Ms-ResNet); and the ResNet model with the hybrid attention module (Mcs-ResNet), as shown in Figure 9.

Except for the embedded attention module, the other model parameters are consistent with those shown in Section 4.1.2. The experimental results are shown in Table 5. It can be seen that:

(1) The Acc value of the ResNet model without the attention module is significantly lower than the other three residual neural network models embedded with the attention module. The ResNet model has the lowest accuracy rate on the Byte-Gray dataset of 78.32%, and the average accuracy rate is 87.39%, which is the lowest among the four models, and 3.7%, 4.87%, and 6.01% lower than the other three models, respectively. It also works best on RGBA images in each dataset, with an accuracy rate of 95.14%, and it is still 0.64%, 1.95%, and 2.07% lower than other models.

(2) The average Acc value of the Mc-ResNet model and the Ms-ResNet model is 3.7% and 4.87% higher than that of the ResNet model. The experiments show

TABLE 5: Classification results under different attention modules.

No.	Image	ResNet	Mc-ResNet	Ms-ResNet	Mcs-ResNet
1	Byte-gray	78.32	85.96	88.51	**89.03**
2	Byte-RGB	80.44	87.15	88.98	**91.83**
3	ASM-gray	79.38	85.08	87.42	**89.12**
4	ASM-RGB	85.49	88.11	87.97	**90.04**
5	Byte-gray (224 ∗ 224)	94.27	95.78	96.58	**96.72**
6	Byte-RGB (224 ∗ 224)	86.88	89.64	92.21	**92.98**
7	ASM-gray (224∗224)	95.05	96.43	96.38	**96.98**
8	ASM-RGB (224 ∗ 224)	91.54	95.88	95.19	**96.70**
9	RGBA (224 ∗ 224)	95.14	95.78	97.09	**97.21**
Average		87.39	91.09	92.26	93.40

Table 6: Running time (s) under different attention modules.

Dataset	Model	Training time	Prediction time
BIG2015	ResNet	15.1398	10.2272
	Mc-ResNet	19.3584	12.0378
	Ms-ResNet	18.5872	11.3385
	Mcs-ResNet	21.7059	13.5375
Malimg	ResNet	16.4576	3.9875
	Mc-ResNet	20.5867	4.7621
	Ms-ResNet	19.8541	4.5753
	Mcs-ResNet	22.3302	5.7758

that the introduction of the attention mechanism is helpful to the extraction of key image features and can effectively improve classification accuracy. Similarly, the above two models also work best on RGBA images.

(3) The average Acc value of the proposed Mcs-ResNet embedded with the hybrid attention module is 6.01% higher than the ResNet model. Moreover, it is 2.31% and 1.14% higher than the Mc-ResNet model and Ms-ResNet model which are embedded with a single attention module. In general, Mcs-ResNet, which embeds both the channel attention module and spatial attention module, achieves the best classification performance even on different visualization schemes. Especially on RGBA images, its classification accuracy is still the best with an accuracy rate of 97.21%. As shown in the 9th experiment, when using RGBA images, the model after embedding hybrid attention improves the classification accuracy by 2.07% compared to the model with only residual network. Moreover, the classification accuracy is also improved compared with the model embeds channel or spatial attention alone. This further verifies the effectiveness of the proposed malicious code visualization scheme.

The running times of the above four models are shown in Table 6. It can be seen that with the introduction of channel attention and spatial attention mechanisms, the training time and prediction time of the model are longer. But given the improvement in accuracy, the increase in prediction time is small and acceptable within the expected range. In addition, the training time on the two datasets are relatively close, 21.7059 seconds and 22.3302 seconds, while the prediction time are 13.5375 seconds and 5.7758 seconds, respectively, and the residual network model parameter scale is close to 3.8 billion FLOPs. Therefore, the experimental results show that the proposed model can achieve better prediction results within a reasonable running time.

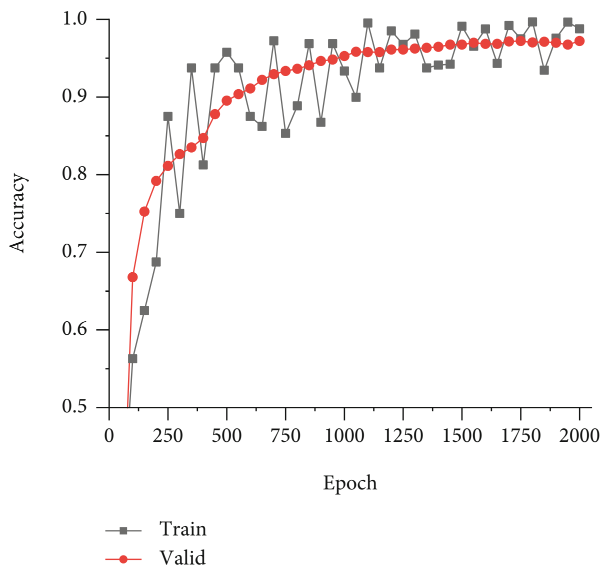
Based on the above analysis, the attention mechanism can improve the classification effect, and different attention mechanisms have different effects on the model. Especially, the introduction of the hybrid attention mechanism in the deep residual network can effectively improve the classification accuracy. The reason is that a single attention mechanism is not enough to fully characterize key features. If channel attention is ignored, the ability to extract global texture features will suffer. If the spatial attention is ignored, it will have an impact on local texture feature learning, thus ignoring the local texture information. The hybrid attention mechanism can learn different weights from the two dimensions of channel and spatial, extract the deep texture feature of malicious code images from the whole and local perspective, and strengthen the model's ability to extract key features. Overall, the fusion of channel and spatial attention mechanism enables the deep features of malicious code images to be fully represented, enabling the classification model to have better classification accuracy for different malicious families.

4.3. Overall Performance Experiment and Analysis. In this section, two groups of experiments are conducted to analyze the overall performance of the model: (1) performance analysis on different datasets: verify the general applicability of the proposed classification model on different experimental datasets; (2) performance comparison analysis: compare the proposed scheme with other malicious code classification schemes to verify the superiority of the proposed scheme.

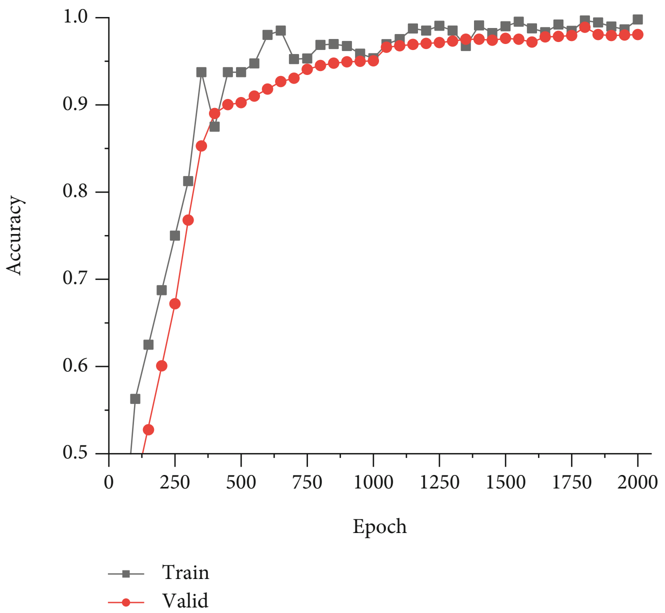
4.3.1. Performance Analysis on Different Datasets. In order to verify the general applicability of the proposed classification model, the BIG2015 dataset and the Malimg dataset are selected for comparative experiments, and the experimental results are shown in Figure 10 and Table 7. The BIG2015 dataset provides both bytecode files and assembly files that can be directly used in the proposed classification model, while the Malimg dataset provides the image format of the malicious code after gray-scale processing. Since the malicious code file is truncated and other operations that cause information loss, reverse processing can be performed to completely restore the image file to a bytecode file. And then use the IDA PRO disassembly tool (https://www.hex-rays.com/products/ida/) to analyze the bytecode files to obtain the corresponding assembly file and finally use the Mcs-ResNet model to classify malicious code.

It can be concluded from the experimental results that when the epoch is 250, the validation accuracy of the BIG2015 dataset is 81.13%, and the validation accuracy of the Malimg dataset is 67.19%, so in the initial stage of training, the optimization of the training effect on the BIG2015 dataset is slightly faster. When the epoch is 500, the validation accuracy of these two datasets reaches 89.53% and 90.25%, respectively; now, the prediction effect on the Malimg dataset is slightly better. And both datasets can reach stability at 1250 epochs, and the accuracy is basically close to the maximum. Finally, it achieved an average classification accuracy of 97.21% on the BIG2015 dataset and 98.06% on the Malimg dataset.

In addition to the accuracy rate, precision, recall, $F1$ -score, and confusion matrix are also used to evaluate the



(a) BIG2015 dataset



(b) Malimg dataset

Figure 10: The performance of Mcs-ResNet on different datasets.

model performance; the formula and experimental results are shown below (Table 8).

$$\text{Precision} = \frac{1}{n}\sum_{i=1}^{n}\frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i}, \quad (11)$$

$$\text{Recall} = \frac{1}{n}\sum_{i=1}^{n}\frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i}, \quad (12)$$

$$F1\_\text{score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (13)$$

It can be seen that the proposed model performs well under different evaluation metrics. The number of malicious code families Skintrim.N and Swizzor.gen!E in Malimg dataset is only 80 and 128, and the classification effect on these two families is unstable and is largely affected by the data imbalance. Data imbalance is not the focus of this paper and will be studied in the follow-up work. In summary, the proposed classification model shows good generalization performance and is not limited to a specific dataset, which can achieve better classification results on different datasets while ensuring the classification efficiency (Figure 11).

4.3.2. Performance Comparison of Malicious Code Classification Schemes. The last set of experiments compares the proposed Mcs-ResNet model with several models that currently perform well in malicious code classification to verify its classification performance. The experimental results on two datasets are shown in Table 9. Among them, Nataraj et al. [13] convert the bytecode file into a grayscale image, extract the GIST features, and use the KNN algorithm for classification; Wang et al. [16] convert the bytecode file into an RGB image and use VGGNet model for classification; Cui et al. [43] convert the malicious code into grayscale images and use CNN for classification; Cakir and Dogdu [25] use the assembly file of malicious code, extract features based on Word2Vec, and then, use Gradient Boosting Machine (GBM) for classification; Ma et al. [39] use both bytecode files and assembly files of malicious codes and classify them based on SVM.

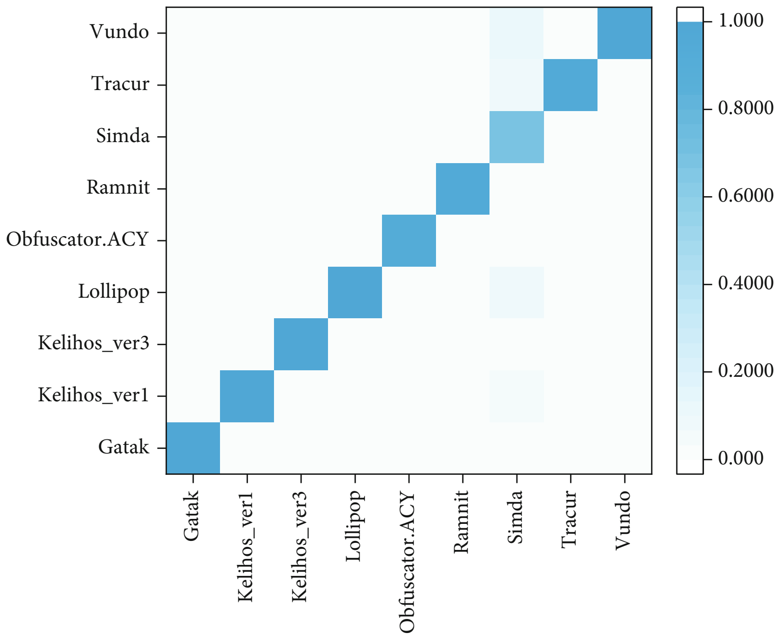
It can be seen from Table 9 that the proposed Mcs-ResNet model reaches 97.21% and 98.06% classification accuracy on the two datasets, respectively. Compared with other methods that only use .bytes files, such as Nataraj et al. [13] and Cui et al. [43], the classification accuracy rate

Table 7: Accuracy in different training epochs.

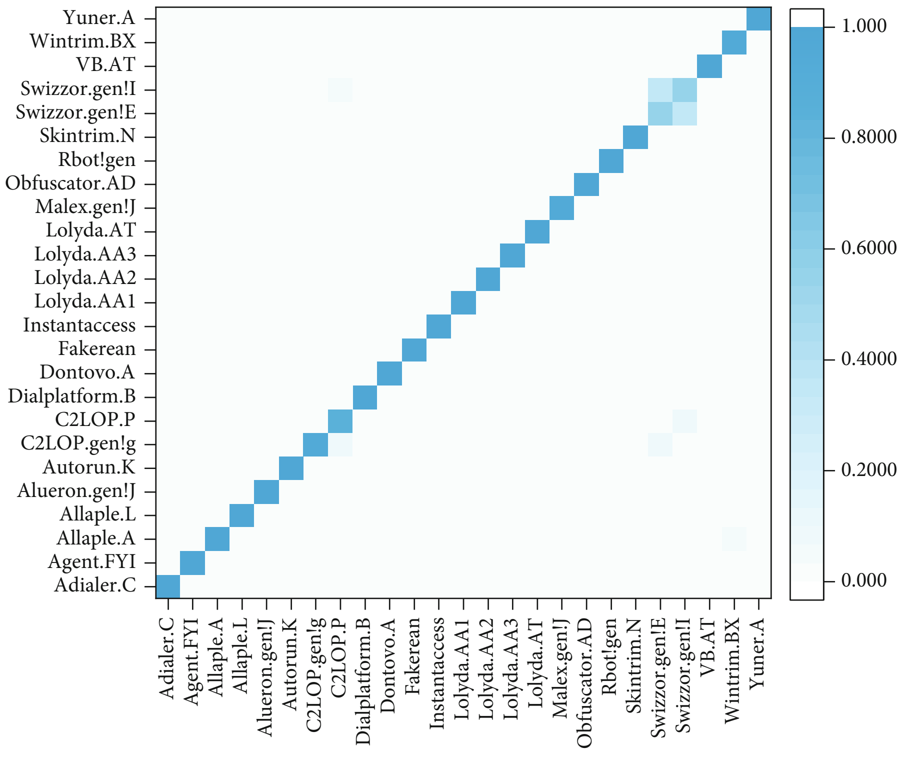
Epoch	BIG2015		Malimg	
	Train	Valid	Train	Valid
250	87.5	81.13	75	67.19
500	95.75	89.53	93.75	90.25
750	85.32	93.35	95.32	94.07
1000	93.35	95.28	95.35	95.04
1250	96.77	96.11	99.07	97.14
1500	99.11	96.75	99.01	97.63
1750	97.49	97.21	98.49	97.95
2000	98.79	97.21	99.79	98.06

Table 8: The result of precision, recall, and $F1$-score.

	Precision	Recall	$F1$-score
BIG2015	96.55	96.24	96.39
Malimg	97.11	96.93	97.02



(a) BIG2015



(b) Malimg

Figure 11: Confusion matrix of different datasets.

is increased by 1~3%. Compared with methods that only use .asm files, such as Cakir nad Dogdu [25], the classification accuracy rate is improved by 1.07%. Therefore, the experimental results of using both .byte files and .asm files are better than using only one of them, indicating that more file types can provide more information and further improve the subsequent classification accuracy. And compared with the Ma et al. [39] method that uses both .bytes files and .asm files, the classification accuracy rate is also improved by 1.12%. Ma et al. only use the global attention mechanism to extract weights of each assembly statement, without considering the key channels and regions of intermediate feature maps of the classification model. Therefore, the hybrid attention module composed of channel attention and spatial attention outperforms the global attention mechanism used by Ma et al. Overall, the proposed method has

Table 9: Comparative experimental results.

Method	Files used	Dataset	Accuracy
Nataraj et al. [13]	.bytes	Malimg	97.18
Wang et al. [16]	.bytes	Malimg	96.16
Cui et al. [43]	.bytes	Malimg	94.50
Cakir and Dogdu [25]	.asm	BIG2015	96.14
Ma et al. [39]	.bytes+.asm	BIG2015	96.09
Mcs-ResNet (ours)	.bytes+.asm	BIG2015	**97.21**
Mcs-ResNet (ours)	.bytes+.asm	Malimg	**98.06**

certain advantages in classification accuracy on different datasets over other malicious code classification methods. The reason is that this method uses both bytecode files and assembly files to form RGBA images to obtain more malicious code information, and the proposed hybrid attention mechanism pays more attention to the extraction of key regions and local features, which further improves the classification accuracy.

5. Conclusion and Future Work

Edge security is an important guarantee for edge computing, and it is of great significance to classify malicious code quickly and accurately in the entire life cycle of edge computing. Therefore, a malicious code visualization classification method based on a deep residual network and hybrid attention mechanism for edge security is proposed to effectively support the detection and accurate classification of malicious code. The main contributions are as follows:

(1) A visualization scheme that converts malicious code into RGBA images is proposed to improve the deep feature representation ability of malicious code images. This scheme effectively integrates the bytecode file and assembly file of the malware, deeply exploits and utilizes the image feature information, and solves the problem of a single source of code images in other visualization solutions without adding additional computational complexity

(2) A classification model - Mcs-ResNet that combines a hybrid attention mechanism and deep residual network is proposed to accurately classify malicious code. Due to its powerful feature extraction capability and shortcut connection architecture, the deep residual network improves classification accuracy while alleviating the problem of model degradation and gradient disappearance. The hybrid attention mechanism including channel and spatial attention can effectively extract the key feature information of malicious code images. The combination of the two can further improve the classification accuracy and effectiveness

The experimental results on the BIG2015 and Malimg datasets demonstrate the feasibility and effectiveness of the proposed visualization scheme and classification model. Compared with the existing malicious code classification methods, the proposed model performs better in classification accuracy and generalization performance. Future work will start with the serialization of malicious code. Consider associating the bytecode file of the malware with the assembly file and extracting the features of the sequence information in the vertical direction and the associated information in the horizontal direction to fully utilize the malicious code information. How to better combine the attention mechanism with malicious code classification is also the focus of the future work.

Data Availability

The datasets used in the experimental part include the BIG2015 dataset and the Malimg dataset, from the following websites: https://www.kaggle.com/c/malware-classification/data and https://www.kaggle.com/keerthicheepurupalli/malimg-dataset9010.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors appreciate the support from the Zhejiang Provincial Natural Science Foundation of China (LY20F020015 and LY21F020015), Zhejiang Province Key R&D Project (2021C02012), the National Science Foundation of China (61902345, 61972121, 61902099, 61702517, and 61802101), the Defense Industrial Technology Development Program (no. JCKY2019415C001), and the Open Project Program of the State Key Lab of CAD&CG (grant no. 2109), Zhejiang University.

References

[1] X. Luo, Q. Qin, X. Gong, and M. Xue, "A survey of adversarial attacks on wireless communications," *Champions*, vol. 437, pp. 83–91, 2022.

[2] Z. Guo, Y. Lu, H. Tian, J. Zuo, and H. Lu, "A security evaluation model for multi-source heterogeneous systems based on IOT and edge computing," *Cluster Computing*, vol. 24, 2021.

[3] Y. Wang, G. Yang, T. Li, L. Zhang, and X. Yu, "Optimal mixed block withholding attacks based on reinforcement learning," *International Journal of Intelligent Systems*, vol. 35, no. 12, pp. 2032–2048, 2020.

[4] Wikipedia org, "Wikipedia's official website," 2022, https://en.wikipedia.org/wiki/WannaCry_ransomware_attack.

[5] S. Shen, K. Zhang, Y. Zhou, and S. Ci, "Security in edge-assisted Internet of Things: challenges and solutions," *Science China Information Sciences*, vol. 63, no. 12, article 220302, 2020.

[6] S. Greengard, "Cybersecurity gets smart," *Communications of the ACM*, vol. 59, no. 5, pp. 29–31, 2016.

[7] P. Seshagiri, A. Vazhayil, and P. Sriram, "AMA: static code analysis of web page for the detection of malicious scripts," *Procedia Computer Science*, vol. 93, pp. 768–773, 2016.

[8] T. Shibahara, T. Yagi, M. Akiyama, D. Chiba, and T. Yada, "Efficient dynamic malware analysis based on network behavior using deep learning," in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, Washington, DC, USA, 2016.

[9] J. Cheng, J. Zheng, and X. Yu, "An ensemble framework for interpretable malicious code detection," *International Journal of Intelligent Systems*, vol. 36, 2020.

[10] D. Vasan, M. Alazab, S. Wassan, B. Safaei, and Q. Zheng, "Image-based malware classification using ensemble of cnn architectures (imcec)," *Computers & Security*, vol. 92, article 101748, 2020.

[11] S. Ni, Q. Qian, and R. Zhang, "Malware identification using visualization images and deep learning," *Computers & Security*, vol. 77, pp. 871–885, 2018.

[12] G. J. Conti, E. Dean, M. Sinda, and B. Sangster, "Visual reverse engineering of binary and data files," in *International Workshop on Visualization for Computer Security*, J. R. Goodall, G. Conti, and K. L. Ma, Eds., vol. 5210 of Lecture Notes in Computer Science, pp. 1–17, Springer, Berlin, Heidelberg, 2008.

[13] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: visualization and automatic classification," in *Proceedings of the 8th international symposium on visualization for cyber security*, pp. 1–7, Pittsburgh, Pennsylvania, USA, 2011.

[14] L. Nataraj and B. S. Manjunath, "SPAM: signal processing to analyze malware [applications corner]," *IEEE Signal Processing Magazine*, vol. 33, no. 2, pp. 105–117, 2016.

[15] D. Kornish, J. Geary, V. Sansing, S. Ezekiel, L. Pearlstein, and L. Njilla, "Malware classification using deep convolutional neural networks," in *2018 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pp. 1–6, Washington, DC, USA, 2018.

[16] B. Wang, H. H. Cai, and Y. Su, "Classification of malicious code variants based on VGGNet," *Journal of Computer Applications*, vol. 40, no. 1, pp. 162–167, 2020.

[17] B. Sun, P. Zhang, M. Y. Cheng, X. T. Li, and Q. Li, "Malware detection method based on enhanced code images," *Journal of Tsinghua University (Science and Technology)*, vol. 60, no. 5, pp. 386–392, 2020.

[18] T. Chen, B. Xiang, L. V. Mingqi, B. Chen, and X. Jiang, "Android malware detection method based on byte-code image and deep learning," *Telecommunications Science*, vol. 35, pp. 9–17, 2019.

[19] D. Hong, N. Yokoya, J. Chanussot, and X. X. Zhu, "An augmented linear mixing model to address spectral variability for hyperspectral unmixing," *IEEE Transactions on Image Processing*, vol. 28, no. 4, pp. 1923–1938, 2018.

[20] H. Gao, W. Huang, and Y. Duan, "The cloud-edge-based dynamic reconfiguration to service workflow for mobile ecommerce environments," *ACM Transactions on Internet Technology (TOIT)*, vol. 21, pp. 1–23, 2021.

[21] Y. Yin, Z. Cao, Y. Xu, H. Gao, and Z. Mai, "QoS prediction for service recommendation with features learning in mobile edge computing environment," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 4, pp. 1136–1145, 2020.

[22] F. Qin, N. Gao, Y. Peng, Z. Wu, S. Shen, and A. Grudtsin, "Fine-grained leukocyte classification with deep residual learning for microscopic images," *Computer Methods and Programs in Biomedicine*, vol. 162, pp. 243–252, 2018.

[23] Y. Peng and B. L. Lu, "Discriminative extreme learning machine with supervised sparsity preserving for image classification," *Neurocomputing*, vol. 261, pp. 242–252, 2017.

[24] G. Pitolli, G. Laurenza, L. Aniello, L. Querzoni, and R. Baldoni, "MalFamAware: automatic family identification and malware classification through online clustering," *International Journal of Information Security*, vol. 20, no. 3, pp. 371–386, 2021.

[25] B. Cakir and E. Dogdu, "Malware classification using deep learning methods," in *Proceedings of the ACMSE 2018 Conference*, pp. 1–5, Richmond, Kentucky, 2018.

[26] T. N. Turnip, A. Situmorang, A. Lumbantobing, J. Marpaung, and S. I. Situmeang, "Android malware classification based on permission categories using extreme gradient boosting," in *Proceedings of the 5th International Conference on Sustainable Information Engineering and Technology*, pp. 190–194, Malang, Indonesia, 2020.

[27] Z. Liu, P. Qian, X. Wang, Y. Zhuang, L. Qiu, and X. Wang, "Combining graph neural networks with expert knowledge for smart contract vulnerability detection," in *IEEE Transactions on Knowledge and Data Engineering*, IEEE Xplore, 2021.

[28] S. Choi, "Malicious powershell detection using graph convolution network," *Applied Sciences*, vol. 11, no. 14, p. 6429, 2021.

[29] H. Wu, X. Li, A. T. Liu, Z. Wu, and H. Y. Lee, "Adversarial defense for automatic speaker verification by cascaded self-supervised learning models," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6718–6722, Toronto, ON, Canada, 2021.

[30] D. Hong, L. Gao, J. Yao, N. Yokoya, and B. Zhang, "Endmember-Guided Unmixing Network (EGU-Net): A General Deep Learning Framework for Self-Supervised Hyperspectral Unmixing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, pp. 1–14, 2021.

[31] G. Cheng, Y. Chen, S. Deng, H. Gao, and J. Yin, "A blockchain-based mutual authentication scheme for collaborative edge computing," *IEEE Transactions on Computational Social Systems*, vol. 9, no. 1, pp. 146–158, 2021.

[32] H. Gao, X. Qin, R. J. D. Barroso, W. Hussain, Y. Xu, and Y. Yin, "Collaborative learning-based industrial IoT API recommendation for software-defined devices: the implicit knowledge discovery perspective," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 1, pp. 66–76, 2022.

[33] F. T. Jaigirdar, C. Rudolph, and C. Bain, "Prov-IoT: a security-aware IoT provenance model," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 1360–1367, Guangzhou, China, 2020.

[34] C. Zhou, Y. Yu, S. Yang, and H. Xu, "Intelligent immunity based security defense system for multi-access edge computing network," *China Communications*, vol. 18, no. 1, pp. 100–107, 2021.

[35] T. Kim, B. Kang, M. Rho, S. Sezer, and E. G. Im, "A multimodal deep learning method for android malware detection using various features," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 773–788, 2019.

[36] B. Vasu and N. Pari, "Combining multimodal DNN and SigPid technique for detecting malicious android apps," in *2019 11th International Conference on Advanced Computing (ICoAC)*, pp. 289–294, Chennai, India, 2019.

[37] L. Ghouti and M. Imam, "Malware classification using compact image features and multiclass support vector machines," *IET Information Security*, vol. 14, no. 4, pp. 419–429, 2020.

[38] S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, "Cbam: convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, Munich, Germany, 2018.

[39] X. Ma, S. Guo, H. Li et al., "How to make attention mechanisms more practical in malware classification," *IEEE Access*, vol. 7, pp. 155270–155280, 2019.

[40] H. Yakura, S. Shinozaki, R. Nishimura, Y. Oyama, and J. Sakuma, "Neural malware analysis with attention mechanism," *Computers & Security*, vol. 87, article 101592, 2019.

[41] C. Wang, Z. Zhao, F. Wang, and Q. Li, "A novel malware detection and family classification scheme for IoT based on DEAM and DenseNet," *Security and Communication Networks*, vol. 2021, Article ID 6658842, 16 pages, 2021.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, Las Vegas, NV, USA, 2016.

[43] Z. Cui, F. Xue, X. Cai, Y. Cao, G. G. Wang, and J. Chen, "Detection of malicious code variants based on deep learning," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3187–3196, 2018.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 9107718, 10 pages
https://doi.org/10.1155/2021/9107718

Research Article

Hierarchical Social Recommendation Model Based on a Graph Neural Network

Zhongqin Bi,[1] Lina Jing,[1] Meijing Shan,[2] Shuming Dou,[1,3] and Shiyang Wang[1]

[1]*College of Computer Science and Technology, Shanghai University of Electric Power, Shanghai 200090, China*
[2]*Institute of Information Science and Technology, East China University of Political Science and Law, Shanghai 201620, China*
[3]*China Electronic Systems Engineering Corp, Beijing 100141, China*

Correspondence should be addressed to Shuming Dou; d2313776747@163.com

Received 9 July 2021; Revised 30 July 2021; Accepted 17 August 2021; Published 31 August 2021

Academic Editor: Xiaoxian Yang

With the continuous accumulation of social network data, social recommendation has become a widely used recommendation method. Based on the theory of social relationship propagation, mining user relationships in social networks can alleviate the problems of data sparsity and the cold start of recommendation systems. Therefore, integrating social information into recommendation systems is of profound importance. We present an efficient network model for social recommendation. The model is based on the graph neural network. It unifies the attention mechanism and bidirectional LSTM into the same framework and uses a multilayer perceptron. In addition, an embedded propagation method is added to learn the neighbor influences of different depths and extract useful neighbor information for social relationship modeling. We use this method to solve the problem that the current research methods of social recommendation only extract the superficial level of social networks but ignore the importance of the relationship strength of the users at different levels in the recommendation. This model integrates social relationships into user and project interactions, not only capturing the weight of the relationship between different users but also considering the influence of neighbors at different levels on user preferences. Experiments on two public datasets demonstrate that the proposed model is superior to other benchmark methods with respect to mean absolute error and root mean square error and can effectively improve the quality of recommendations.

1. Introduction

With the rapid development of the Internet and the explosive growth of information, recommendation systems help information consumers find the content that they are interested in from a large amount of information, effectively relieving the pressure of information overload. Collaborative filtering [1] is one of the most commonly used recommendation methods. It uses historical data of user-item interactions to learn users' preferences for projects without explicit user and project information [2]. In addition to user-project interactions, social relationships between users also provide potential information for modeling user preferences. According to the social theory, people in a social network are affected by their social relationships, resulting in the homogeneity of preferences of social neighbors. Therefore, the influence of social relations on recommendation systems has attracted increasing attention [3]. The general approach of recommendation algorithms based on social information is to integrate social relations based on the rating matrix information. By contrast, social relations are the mapping of natural social relations and reflect the users' natural social relations [4–6].

In recent years, researchers have used the methods of convolutional networks, recurrent networks, and deep autoencoders to define and design neural network structures for processing graph data. Such deep neural networks are called graph neural networks (GNNs) [10–14]. Some recommendation systems use GNNs as feature learning tools to extract useful features from social information [7–9]. The main idea is to use neural networks to iteratively aggregate feature information from the local graph neighborhood. At the same time, node information can be transmitted and aggregated through graphs after transformation. In addition, the data in the social recommendation system can be expressed as user-user social graphs and user-item graphs that provide potential advantages for the learning of GNNs. Some studies use GNNs to incorporate social network information into users and item potential factors to learn. For example, GraphRec [6] proposed a social recommendation framework of GNN that aggregates user-item interaction information and social relationship information simultaneously when performing predictions. DiffNet [44] is a hierarchical influence propagation structure for the simulation of the recursive dynamic diffusion process in social recommendations. Through this structure, both users and projects can be expressed as embedded information containing collaboration and functional content.

Although these recommendations based on GNNs have been very successful, they do not take full advantage of social network information. First, most of the methods simply link the user representation extracted from the social relationship to the user or item information and lack in-depth learning of the social relationship. Second, most of them only consider the role of direct (local) neighbors and ignore the influence of indirect neighbors that come from a few hops away. When direct neighbors are sparse, indirect neighbors are also helpful for user representation, and the recommendation system needs to take this indirect social relationship into account. Finally, not all neighbors are useful when recommending users. For example, a user who likes Macs may not be indicative that all of his neighbors like Macs. If the influence of these neighbors is treated equally, then the recommendation effect will be reduced. Therefore, it is necessary for the recommendation algorithm to distinguish strong and weak social relations and reasonably allocate the influence weight of neighbors on users.

In this paper, we use a GNN to design a hierarchical social collaborative filtering framework called GHSCF. The model includes three modules: the embedded propagation layer, the embedded layer, and the sequential learning layer. The framework can make full use of social network information for recommendations. Compared with other social recommendation methods, this model has the following advantages. First, an embedded propagation method is designed to learn the influence of neighbors at different levels. When direct neighbors have no valuable information to share, users may use indirect neighbor information. In addition, the attention mechanism is introduced to distinguish strong and weak social connections, screen helpful neighbors, and effectively capture the influence weights of different neighbors. Second, the user preference sequence generated by different levels of communication learning represents user modeling of different levels of neighbors. The long-term and short-term memory (LSTM) network is used for serial modeling and makes final recommendations based on the influence of neighbors at different levels.

In summary, the contributions of this article are as follows:

(1) We designed a hierarchical social recommendation model based on the graph neural network to capture user relationships along social networks and simulate changes in influence between users at different levels to improve the accuracy of user representation

(2) We propose an embedded propagation based on GNN that introduces a user relationship metric for the most relevant items and alleviates the problem of inconsistent user preferences for target items

(3) We combine the attention mechanism with the bidirectional LSTM network and propose a bidirectional LSTM with an attention mechanism that can help sequence modeling

(4) We conduct comprehensive experiments on two standard datasets and compare existing social recommendation frameworks in order to evaluate and demonstrate the effectiveness of the proposed approach, proving the superior performance of our method. Compared with the existing methods, our method not only considers the information from the neighbors but also considers the information of the original neighbors. Our method reasonably distinguishes strong and weak social relations

The rest of this article is organized as follows: in Section 2, we briefly introduce the related work. In Section 3, we provide some preliminary information. In Section 4, we introduce our GHSCF architecture model in detail. Section 5 describes the experimental results. Section 6 concludes the paper.

2. Related Work

The collaborative filtering recommendation algorithm [1] is the most widely used method in recommendation systems to predict the interaction between users and products. Using the historical interaction information between users and products, the users and products of interest can be modeled. In addition to user-item interaction, the use of social relationships to make suggestions has attracted great attention [20, 21]. Many social recommendation methods [22–27] have demonstrated the effectiveness of incorporating social relationships into recommendations. SoDimRec [26] uses the heterogeneity of social relations and the weak dependence on social networks to make recommendations. SOReg [17] proposes two kinds of social recommendation algorithms and uses a social regularization conditional constraint matrix to decompose the objective function. TRUSTMF [18] simulates the interaction between users and maps users to two low-dimensional spaces by decomposing the social trust network into the trust space and trusted space. Although the social recommendation method based on matrix decomposition has achieved a good recommendation effect, its linearity is insufficient to reveal the complex nonlinear relationship between users and products.

Recently, deep neural networks have been used to enhance recommendation systems [28, 29, 43]. Most of these systems use deep neural networks as feature learning tools to extract features from auxiliary information, such as the text description of items [30, 32, 33, 46]. Neural matrix factorization (NeuMF) [19] proposes a neural cooperative filtering (NCF) framework to learn the nonlinear interactions between users and projects. Later, researchers attempted to incorporate social relationship information into the NCF model to improve recommendation performance. Neural social collaborative ranking (NSCR) [31] uses social network information as graph regularization to extend the NeuMF model so that nearby neighbors have similar potential vectors. The deep neural network model on social relations (DeepSoR) [12] learns the nonlinear characteristics of each user from social relations and integrates them into probability matrix factorization for rating prediction. The deep social collaborative filtering (DSCF) [45] framework uses a random walk layer to generate the item perceived social sequence and extract relevant information from distant neighbors for user-item nonlinear fusion. All of these works address the cross-domain task of social recommendation and thus differ from traditional social recommendation systems.

GNNs have demonstrated the ability to learn graph structure data [11, 34, 35, 42, 45, 48]. In the social recommendation task, the relationship between users employs typical graph data. Thus, GNNs have a natural advantage when used for this task. In [6, 36, 37, 47], SocialGCN [36] was proposed to capture user preferences by utilizing the advantages of GCNS and modeling the diffusion process of preference information in the figure. However, SocialGCN does not consider the relative importance of different neighbors and the degree of trust of the users for different neighbors when buying different items. GraphRec [6] implements a new graphical neural network framework for social recommendation. In particular, a principled approach is designed to capture interaction and rating information jointly and to simulate both types of graphs and heterogeneous intensities coherently. However, GraphRec does not take into account the influence of remote neighbors. This paper proposes a hierarchical social recommendation model that can make full use of social network information to make recommendations. This model can simulate the influence process of user preference information spread in social networks and model user preference considering the importance of neighbors at different levels.

3. Preliminaries

3.1. Attentional Mechanism. The attention mechanism [39] refers to the human visual attention and thinking mode to help the model quickly screen valuable information from a large amount of content. It uses the scoring method to extract the weight of key information and is widely used in various deep learning tasks, such as natural language processing, image classification, and speech recognition [40, 41].

Each layer of the model uses an input information vector $X = [x_1, x_2, \cdots, x_n]$ to search and select important information in X in a given query vector q. A "soft" selection mechanism is usually adopted to capture the weight distribution of each piece of information in the input information. The weight of x_1 is calculated as follows:

$$e_i = \mathbf{v}^T \tanh\left(W\mathbf{x}_i + b_i\right), \quad \alpha_i = \frac{\exp(e_i)}{\sum_{j=1}^{N}\exp(e_j)}. \tag{1}$$

The probability vector constituted by α_i is called the attention distribution. W, U, and V are learnable network parameters. The input information is summarized by the weighted average method and is calculated as follows:

$$s = \sum_{i=1}^{N} \alpha_i x_i. \tag{2}$$

3.2. LSTM. An LSTM network [38] is a sequential convolutional network derived from a cyclic neural network that alleviates the vanishing and exploding gradient problems of cyclic neural networks by using cell states and gate mechanisms. LSTM can learn long-term dependence, making it suitable for solving many long sequence learning problems. The excellent performance of LSTM arises from its unique internal structure. Specifically, LSTM uses three gate structures to control the flow of the information state. In each time step t, the current cell state c_{t-1} and the hidden layer state h_{t-1} are updated through an input sequence element value x_t, the cell state c_t, and the hidden layer state h_t of the time step $t-1$. The formula is as follows:

$$\begin{aligned} \begin{bmatrix} i_t \\ f_t \\ o_t \end{bmatrix} &= \begin{bmatrix} \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(W[h_{t-1}; x_t] + b\right), \\ \widehat{c}_t &= \tanh\left(W_c[h_{t-1} : x_t] + b_c\right), \\ c_t &= f_t \odot \widehat{c}_{t-1} + i_t \odot \widehat{c}_t, \\ h_t &= o_t \odot \tanh\left(c_t\right), \end{aligned} \tag{3}$$

where i_t, f_t, and o_t represent gate activation, σ is the sigmoid activation function, $\tan h$ is the hyperbolic tangent activation function, and $\odot$ represents vector element multiplication. Intuitively, ForgetGate f_t controls the extent to which the previous memory unit is forgotten, and input gate i_t controls how many input gates are updated by each unit. The output gate o_t controls the output of the internal memory state. The hidden layer state h_t represents the output information of the internal memory unit of the LSTM unit.

The prediction may need to be determined based on both previous and subsequent inputs, improving the performance of the network. A more common approach is to use bidirectional LSTM to simulate textual semantics from the forward and backward states of the hidden layer. For a sequence vector $[x_1, x_2, \cdots, x_T]$, the forward LSTM reads the sequence x_1 to x_T, and the backward LSTM reads the sequence x_T to x_1 and then connects the forward and backward hidden layer states $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$, h_t integrating all of the information of the sequence around x_t.

4. Model

In this section, we describe in detail the proposed model, namely, GHSCF. The framework of the model is shown in Figure 1. The model includes three modules: the embedded propagation layer captures the intensity of different levels

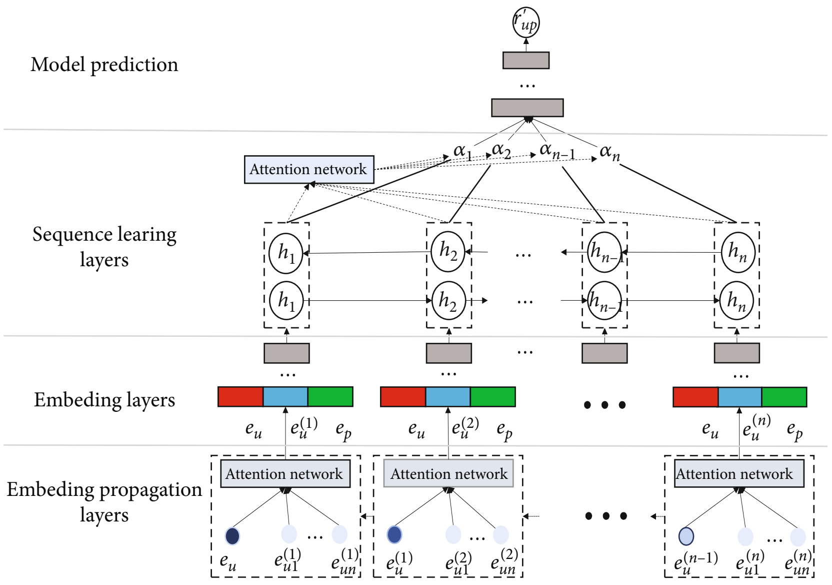


FIGURE 1: Hierarchical recommendation model based on a graph neural network.

of neighbor social relations and carries out user preference modeling, the embedding layer models the user-item interaction, and the sequence learning layer uses LSTM to fuse user-item interaction sequences composed of different levels. The remainder of this section describes each module in detail.

4.1. Embedded Propagation Layer. In this layer, a GNN-based messaging structure is constructed so that the model can capture user relationships along with the social network and optimize user preferences with different levels of neighbors, as shown in Figure 2. To better illustrate the internal structure of the embedded propagation layer, we first introduce the propagation structure of one layer and then extend it to multiple layers.

In social recommendation, when trying to recommend an item for a given user u, the user's preferences are influenced by direct neighbors, while indirect neighbors can also provide valuable help. At the same time, the connection between users can be transmitted through social networks to influence the behavior of other users further. Therefore, the recommendation algorithm should include all of these neighbors in the potential factors of user preferences and introduce the attention mechanism to select representative social neighbors to describe the social information of users. Therefore, this paper proposes a user potential representation method based on social space that aggregates the potential factors of neighboring users to learn the preferences of target users. Specifically, it iteratively learns the potential representation of user u according to its neighbor set N_u, as follows:

$$e_u^{(1)} = \text{LeakyReLU}\left(W_1 e_u + W_2 \sum_{k\in N_u} a_{uk} e_{u_k}\right). \quad (4)$$

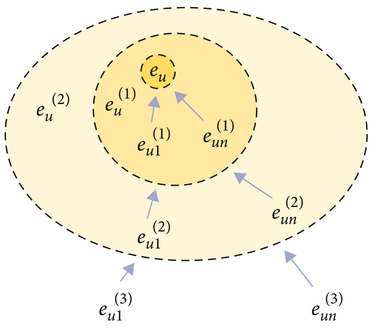


FIGURE 2: Embedded propagation model.

In the formula, $e_u^{(1)}$ represents the representation of user u obtained after the first embedded propagation layer, a_{uk} represents the influence weight of neighbors, that is, $1/|N_u|$, e_{uk} represents the potential representation of neighbor k, and LeakyReLU [15] is the activation function that allows the use of positive signals and small negative signals to encode messages. Notably, in addition to integrating the neighborhood information, the model also preserves the user's original characteristics.

Although users can be modeled based on neighborhood information, no explicit recommendations are specified; that is, users share social relationships with their neighbors' individual interacting items. However, the preferences of target users and their neighbors may not be the same. Sometimes, the neighbors' information does not help recommend target items, and only the information related to the item contributes. Therefore, the model needs to select an item related to the target item for each user to realize similarity modeling between the user and a neighbor. It is important to note that only the most relevant items contribute to the recommendation of a particular item. This is because the most relevant items are the most important influencing factors for the target item, while other items may not be helpful or contribute to noise. Specifically, for the target item p, we select the item most related to p from the item interaction set of user u:

$$p_u = \arg\ \max_{p_i \in V_u} \cos\ (p_i, p). \quad (5)$$

In the formula, v_u is the user's item interaction set, and $\cos\ (p_i, p)$ is a function to measure the similarity between item p and item p_i. Cosine similarity is used according to experience:

$$\cos\ (p_i, p) = \frac{f(p_i)^T f(p)}{|f(p_i)||f(p)|}. \quad (6)$$

In the formula, the f function generates the appropriate feature representation for the item. In this paper, the NeuMF model is used to extract the representation of objects and to calculate the similarity.

As mentioned earlier, there are strong ties and weak ties in a social network, and users may share more similar items with strong ties. Therefore, through the most similar items between users, we can learn the relationship strength.

$$\beta_{uk} = \text{LeakyReLU}\left(v^T \cdot \left[\text{We}_{p_k} \oplus \text{We}_{p_u}\right]\right),$$
$$a_{uk} = \frac{\exp\ (\beta_{uk})}{\sum_{i \in N_u} \exp(\beta_{ui})}, \quad (7)$$

where e_{p_k} represents the embedding representation of the most relevant item of user k and e_{p_u} represents the embedding representation of the most relevant item p of user u. a_{uk} is the degree of correlation between users. With the first layer of direct user modeling, the simulation network propagation mode overlays more indirect neighbors, leading to deeper user preference modeling, which is crucial for the subsequent serialization modeling.

Through n embedded propagation layers, user u can receive information transmitted by neighbors on n layers. In the NTH layer, user u is recursively expressed as

$$e_u^{(n)} = \text{LeakyReLU}\left(W_1^n e_u^{(n-1)} + W_2^n \sum_{k \in N_u} a_{uk}^n e_k^{(n)}\right), \quad (8)$$

where $W_1^n W_2^n$ is a trainable transformation matrix, e_u^{n-1} is the user preference of the upper layer generated by the transmission signal of the previous layer and aggregates the neighbor information of the upper layer, e_k^n is the neighbor K embedding vector of the N^{th} layer, and a_{uk}^n is the influence factor. Through the propagation of social networks, on the basis of direct neighbor modeling, the modeling information of indirect neighbors' user preferences is accumulated layer by layer, which is represented as a user preference sequence $[e_u^{(1)}, e_u^{(2)}, \cdots, e_u^{(n)}]$ that reflects that users at different friend levels will be affected by different neighbors.

4.2. Embedded Layer. The user preference sequence is composed of user modeling by neighbors and users in a social network under the influence of the strength of item interaction. Therefore, it is necessary to integrate the user preference sequence into the user-item interaction first, so that the user's various preference information is fully considered. Clearly, the bad and good comments from the user's social neighbors reflect the user's choice of items in a sequential manner. The interaction between the user and the item is highly nonlinear, and a multilayer perceptron (MLP) can fuse the interaction information with the user preference information. The input of the MLP includes user embedding e_u, single-layer user preference embedding $e_u^{(i)}$, $i \in [1, n]$, item embedding as input e_p, and output user-item interactive embedding as m_i, $i \in [1, n]$, that is given by

$$m_i = f_{u,p}\left(e_u \oplus e_p \oplus e_u^{(i)}\right), \quad i \in [1, n]. \quad (9)$$

Using an MLP, a sequence of all user-item interactions can be obtained from the neighbors $[m_u^1, m_u^2, \cdots, m_u^n]$.

4.3. Sequential Learning Layer. The purpose of the sequence learning layer is to integrate the sequence of rating vectors further or to integrate the information of rating vectors under different levels to obtain a unified representation. Since all of the neighbors in the sequence will affect the prediction, for the distant neighbors, the model needs to capture the remote social information between these neighbors and the user u. In addition, the influence between users in a social networks is bidirectional, and bidirectional LSTM has inherent advantages in sequence modeling:

$$\overrightarrow{h}_i = \overrightarrow{\text{LSTM}}(m_i), \quad i \in [1, n],$$
$$\overleftarrow{h}_i = \overleftarrow{\text{LSTM}}(m_i), \quad i \in [1, n]. \quad (10)$$

The output feature $h_i = [\overrightarrow{h}_i ; \overleftarrow{h}_i]$ is obtained by connecting the forward hidden layer state $\overrightarrow{h}$ and the backward hidden layer state $\overleftarrow{h}$ of the sequence. Then, the attention mechanism is used to learn the important features in the sequence:

$$s = \sum_{i=1}^{n} \alpha_i h_i. \quad (11)$$

In the formula, α_i is the weight of attention. α_i is used to assign the importance of information at different levels of the sequence:

$$a_i = v_a^T \tanh\ (W_a h_i + b_a),$$
$$\alpha_i = \frac{\exp\ (a_i)}{\sum_{i \in N_u} \exp(a_i)}. \quad (12)$$

4.4. Model Learning. In this work, we apply the proposed model to the recommendation task of rating prediction

Table 1: Statistics for the two datasets.

Dataset	Users	Items	Ratings	Links	Rating density	Social connection density
Ciao	7,317	10,4975	283,319	111,781	0.0368%	0.2087%
Epinions	18,088	261,649	764,352	355,813	0.0161%	0.1087%

and finally use the MLP to predict user u's rating score for item p.

$$r'_{up} = f(s). \tag{13}$$

Since the task focused on in this work is rating prediction, the model selects a commonly used objective function:

$$\text{Loss} = \frac{1}{2|O|} \sum_{u,p \in O} \left(r'_{up} - r_{up} \right)^2, \tag{14}$$

where $|O|$ is the number of ratings observed and r_{up} is the true rating of user u on item p.

5. Experiment

5.1. Experimental Data. In this experiment, we selected two commonly used public datasets to evaluate the effectiveness of this method, namely, Ciao and Epinions. They are item rating datasets taken from two different social platforms that contain user rating records and social relationship information. The user rating in the dataset is recorded in intervals of 1 for each user to view the items, with [1, 5] as the rating interval. The social relationship information gives the set of all social neighbors for each user. The statistics for these two datasets are shown in Table 1.

For the convenience of subsequent use, 80% of the data in each dataset were randomly selected for training, 10% were selected for validation, and 10% were selected for testing. All hyperparameters were selected based on the validation set. In this model, Minh and Salakhutdinov's method [16] was selected as the optimization algorithm, and the initial learning rate was set to 0.005. At the same time, to alleviate the overfitting problem, dropout technology was used in the training process, and the ratio was set to 0.6. Unless otherwise mentioned, three hidden layers were used for all neural networks, and the number of neighboring layers was set to 4. For the embedded size, the model tests the values of {8,16,32,64,128,256} and searches for the batch size and learning rate in {16,32,64,128,512} and {0.0005, 0.001, 0.005, 0.01, 0.05, 0.1}. After repeated tests, the embedded size of the model was set to 64, the batch size to 128, and the learning rate to 0.005. In addition, following a rule of thumb, the size of the hidden layer was set to the embedded size (the dimension of the underlying factor), and the activation function is set to ReLU. Finally, the parameters of the baseline algorithm were initialized and repeatedly adjusted through testing to achieve the best performance.

To evaluate the quality of the recommendation algorithm, two commonly used prediction accuracy indexes, namely, the mean absolute error (MAE) and root mean square error (RMSE), were used in this paper. Smaller MAE and RMSE values indicate better prediction accuracy.

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{I=1}^{N} \left(y\wedge^{(i)} = y^{(i)} \right)^2}, \qquad \text{MAE} = \frac{1}{N} \sum_{i=1}^{N} \left| y\wedge^{(i)} - y^{(i)} \right|, \tag{15}$$

where N represents the number of test examples.

5.2. Contrast Experiment

(i) PMF [16]: probabilistic matrix factorization is a standard rating prediction model that uses only ratings for collaborative filtering

(ii) SoRec [17]: social regularization models social network information as regularization terms and constrains the matrix factorization framework

(iii) SocialMF [18]: this model effectively integrates social information into the conventional matrix decomposition framework. However, it assumes that the trust relationship between all users is the same and simulates social influence based on that assumption

(iv) NeuMF [19]: this method is a state-of-the-art neural CF model that uses multiple hidden layers above the elements and connects embedded users and objects to capture their nonlinear feature interactions

(v) DeepSoR [12]: this model uses a deep neural network to learn the representation of each user from social relations and integrates it into probability matrix factorization for rating prediction

(vi) DSCF [8]: a deep social collaborative filtering framework that extracts valuable information from social relationships for recommendation and generates an item-aware social sequence for integration into user-item interaction

(vii) DiffNet [44]: a social recommendation framework based on a GNN. It proposes a hierarchical diffusion method to simulate the transmission process of the users' potential preferences in social trust networks

Table 2: RMSE and MAE experimental results of the models.

Models	Ciao		Epinions	
	MAE	RMSE	MAE	RMSE
PMF	0.9021	1.1238	0.9952	1.2128
SoRec	0.8611	1.0848	0.9119	1.1703
SocialMF	0.8270	1.0501	0.8837	1.1328
NeuMF	0.8062	1.0617	0.9072	1.1476
DeepSoR	0.7739	1.0316	0.8383	1.0972
DSCF	0.7270	0.9867	0.8275	1.0667
DiffNet	0.7194	0.9782	0.8158	1.0741
GHSCF	0.7206	0.9194	0.7968	0.9731

5.3. Results and Analysis. As shown in Table 2, probabilistic matrix factorization (PMF) performed poorly on both datasets. This indicates that the internal product is insufficient for capturing the complex relationship between the user and item, further limiting performance. NeuMF consistently outperformed PMF in all cases, demonstrating the importance of nonlinear characteristic interactions between the user and item embedding. Both SoRec and SocialMF utilize ratings and social network information, while PMF only uses rating information. These results indicate that social network information is complementary to rating information. DeepSoR, DSCF, and DiffNet outperform SoRec and SocialMF, and all of these methods use ratings and social network information. However, DeepSoR, DiffNet, and DSCF are based on neural network structures, further demonstrating the superiority of neural networks in making recommendations. In contrast to the other methods, DiffNet shows quite a strong performance, implying that GNNs can deeply mine the underlying information of graphic data. Notably, GHSCF consistently outperformed all baseline approaches. Compared to DeepSoR and DSCF, GHSCF provides advanced modeling components to combine different levels of neighbors in social network information. Further analysis was performed to better understand the contributions of the model components.

5.4. Model Analysis

5.4.1. Effectiveness of the Number of Embedded Propagation Layers. To investigate whether the model can benefit from multiple embedded propagation layers, the model performance is tested by changing the number of layers of the model. We explore model performance at levels 1 through 4 according to the experimental results shown in Table 3.

With increasing model layers, the RMSE and MAE decrease gradually and tend to become stable, indicating that the model can not only effectively utilize the direct neighbor information but also capture the influence of indirect neighbors on the modeling of user preferences. It is important to note that in dataset Ciao, overfitting occurs, possibly due to the noise that an architecture that applies too deeply can generate for the learning representation. When the number of propagation layers is changed, the model is always superior to other methods on the two datasets, indicating that the propagation mechanism in this model can effectively capture the neighbor information at different levels, which is helpful for improving user preference modeling.

Table 3: RMSE and MAE of different communication levels.

Layers	Ciao		Epinions	
	MAE	RMSE	MAE	RMSE
GHSCF-1	0.7271	0.9605	0.8081	1.0558
GHSCF-2	0.7238	0.9598	0.8075	1.0247
GHSCF-3	0.7206	0.9194	0.8071	0.9731
GHSCF-4	0.7229	0.9601	0.8068	1.0331

5.4.2. Neighbor Relationship Modeling and LSTM Validity Analysis. To better understand the proposed model, this experiment compared three variants of the GHSCF model: GHSCF-β, GHSCF-α, and GHSCF-Item. These three variants are defined as follows:

(i) GHSCF-β: the purpose of this variant is to study the effect of the attention factor β on the strength of the learning user's relationship, assuming that each neighbor has the same influence on the target user

(ii) GHSCF-α: in this variant, the bidirectional LSTM structure is replaced by the serial average; that is, the user preferences of different levels are integrated by averaging, and the differences between different levels are ignored

(iii) GHSCF-item: this variant only uses the user vector to evaluate the strong or weak relationship between the user and his neighbors; that is, the influence of the most relevant items is no longer considered

The results on the Ciao dataset are shown in Figure 3. The results using the Epinions dataset are not shown here because similar observations can be made. First, the performance of the GHSCF and GHSCF items is better than that of the GHSCF β. This indicates that not all neighbors have the same influence on target users, and correctly distinguishing the strong and weak relations of neighbors is beneficial for user preference modeling. In addition, the performance of the GHSCF items is worse than that of the GHSCFs. This is because adjacent users may not have an item intersection, and it is not sufficient to capture the strength of user relationships only by considering the user's own social network information. Therefore, not all information from neighbors is useful when recommending a particular project, and it is more valuable to select neighbor-related projects to measure user relationships. Second, the performance of GHSCF-α is significantly lower than that of GHSCF. This indicates that the bidirectional LSTM component better learns the representation of the sequence of user-item interactions at different levels. Therefore, it is further proven that target users in social networks will have different degrees of preferences for target products due to the influence of neighbors separated by different distances.

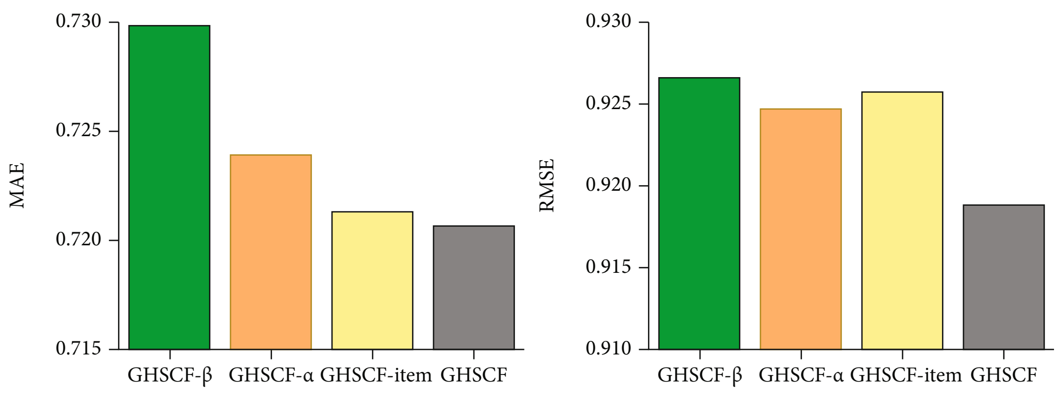


Figure 3: Results of MAE and RMSE experiments for different variants.

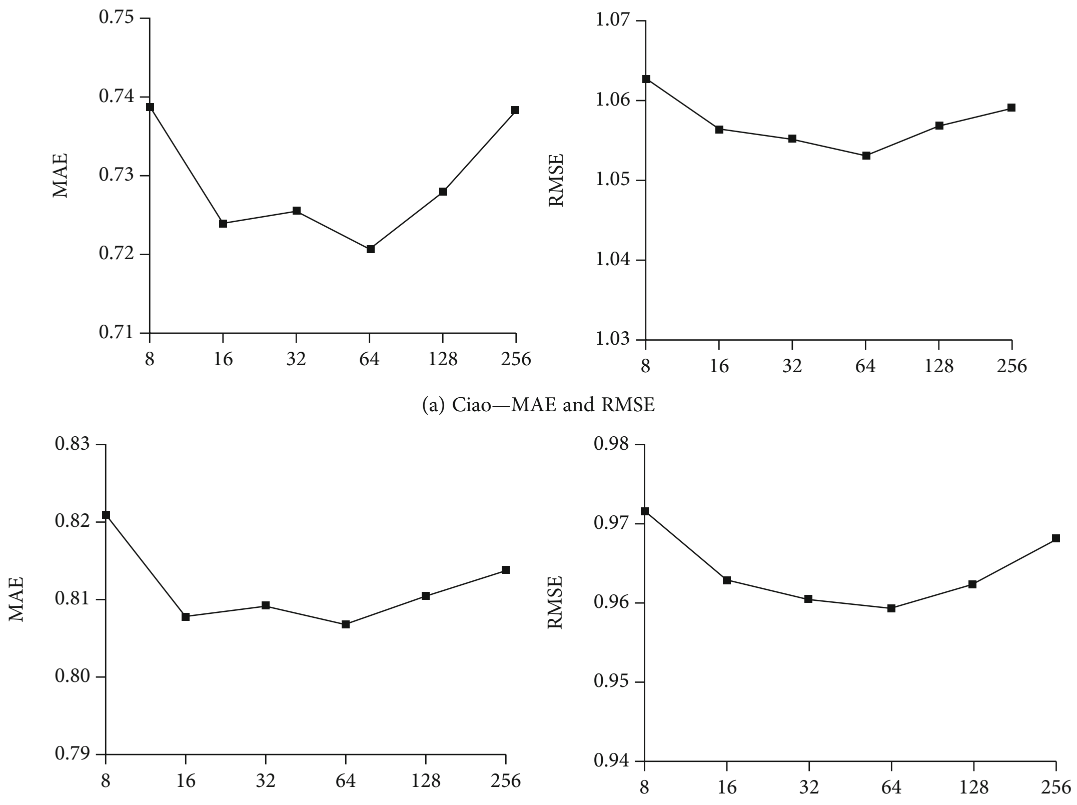


Figure 4: Effect of different embedded sizes.

5.4.3. Effect of the Embedded Size. Figure 4 shows the impact of the embedded sizes of users and items on model performance. As the embedded size increases, the recommended performance first increases and then decreases. When the embedded size is increased from 8 to 64, the performance is significantly improved. However, when the embedding size is 256, the performance is degraded. This shows that using a large embedded size has a powerful representation. If the length of embedding is too large, then the complexity of the model will increase significantly, and overfitting may occur. Therefore, the correct embedding length must be found to balance performance and complexity.

6. Conclusion and Future Work

We propose a hierarchical social recommendation model that designs an embedded propagation layer. This model takes into account information not only from immediate neighbors but also from distant neighbors, helping to model user preferences at different neighborhood levels. In addition, this paper introduces an object perception method to capture the strength of neighborhood relationships, taking into account the importance of close objects between users. Finally, we propose a bidirectional LSTM with an attention mechanism to extract user-item interaction sequences with different preferences. In addition, to solve the overfitting problem, this paper uses the dropout strategy. Experiments show that neighbors at different levels play an important role in social information modeling. Finally, the feasibility of these changes is tested by a comparative experiment that shows that the GNN can learn more accurate social influence and further improve the recommendation performance.

In this work, we only use user-item interaction information to measure the similarity between projects. By contrast, tangential information associated with the project, such as text description and visual content of images, may be more valuable. In addition, we now consider ratings with social information to be static. However, ratings and social information are dynamic. Therefore, in future work, we will consider the establishment of a dynamic GNN for social recommendation.

Data Availability

The experimental data, named Ciao and Epinions, are from previously reported studies and datasets, which have been cited. Ciao is available at https://www.librec.net/datasets.html, and Epinions is available at http://www.trustlet.org/epinions.html.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National Nature Science Foundation of China (No. 61972357).

References

[1] B. Sarwar, G. Karypis, and J. Konstan, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, pp. 285–295, 2001.

[2] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in artificial intelligence. Advances in artificial intelligence*, vol. 2009, pp. 1–19, 2009.

[3] J. Tang, X. Hu, and H. Liu, "Social recommendation: a review," *Social Network Analysis and Mining*, vol. 3, no. 4, pp. 1113–1133, 2013.

[4] E. Bakshy, I. Rosenn, and C. Marlow, "The role of social networks in information diffusion," in *Proceedings of the 21st international conference on World Wide Web*, pp. 519–528, 2012.

[5] W. Fan, T. Derr, and Y. Ma, "Deep adversarial social recommendation," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019.

[6] W. Fan, Y. Ma, and Q. Li, "Graph neural networks for social recommendation," in *The World Wide Web Conference*, pp. 417–426, 2019.

[7] R. Berg, T. N. Kipf, and M. Welling, *Graph Convolutional Matrix Completion*, 2017, arXiv preprint arXiv:1706.02263.

[8] W. Fan, Y. Ma, and D. Yin, "Deep social collaborative filtering," in *Proceedings of the 13th ACM Conference on Recommender Systems*, pp. 305–313, 2019.

[9] T. N. Kipf and M. Welling, *Semi-supervised Classification with Graph convolutional networks*, 2016, arXiv preprint arXiv:1609.02907.

[10] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1025–1035, 2017.

[11] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in Neural Information Processing Systems*, vol. 29, pp. 3844–3852, 2016.

[12] W. Fan, Q. Li, and M. Cheng, "Deep modeling of social relations for recommendation," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[13] X. Wang, X. He, and M. Wang, "Neural graph collaborative filtering," in *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, pp. 165–174, 2019.

[14] H. Gao, L. Kuang, Y. Yin, B. Guo, and K. Dou, "Mining consuming behaviors with temporal evolution for personalized recommendation in mobile marketing apps," *Mobile Networks and Applications*, vol. 25, no. 4, pp. 1233–1248, 2020.

[15] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," *Proc. ICML*, vol. 30, no. 1, 2013.

[16] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," *Advances in Neural Information Processing Systems*, pp. 1257–1264, 2008.

[17] H. Ma, D. Zhou, and C. Liu, "Recommender systems with social regularization," in *Proceedings of the fourth ACM international conference on Web search and data mining*, pp. 287–296, 2011.

[18] M. Jamali and M. Ester, "A matrix factorization technique with trust propagation for recommendation in social networks," in *Proceedings of the fourth ACM conference on Recommender systems*, pp. 135–142, 2010.

[19] X. He, L. Liao, and H. Zhang, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, pp. 173–182, 2017.

[20] J. Tang, C. Aggarwal, and H. Liu, "Recommendations in signed social networks," in *Proceedings of the 25th International Conference on World Wide Web*, pp. 31–40, 2016.

[21] B. Yang, Y. Lei, J. Liu, and W. Li, "Social collaborative filtering by trust," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 8, pp. 1633–1647, 2017.

[22] G. Guo, J. Zhang, and N. Yorke-Smith, "Trustsvd: collaborative filtering with both the explicit and implicit influence of user trust and of item ratings," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, 2015no. 1.

[23] S. Purushotham, Y. Liu, and C. C. J. Kuo, "Collaborative topic regression with social matrix factorization for recommendation systems," in *Proceedings of the 24th International Conference on Machine Learning*, 2012.

[24] X. Wang, W. Lu, and M. Ester, "Social recommendation with strong and weak ties," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pp. 5–14, 2016.

[25] T. Zhao, J. McAuley, and I. King, "Leveraging social connections to improve personalized ranking for collaborative filtering," in *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, pp. 261–270, 2014.

[26] J. Tang, S. Wang, and X. Hu, "Recommendation with social dimensions," in *Thirtieth AAAI Conference on Artificial Intelligence*, pp. 251–257, 2016.

[27] H. Gao, Y. Xu, Y. Yin, W. Zhang, R. Li, and X. Wang, "Context-aware QoS prediction with neural collaborative filtering for Internet-of-Things services," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4532–4542, 2019.

[28] M. Jiang, P. Cui, F. Wang, W. Zhu, and S. Yang, "Scalable recommendation with social contextual information," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 11, pp. 2789–2802, 2014.

[29] Y. Yin, Q. Huang, H. Gao, and Y. Xu, "Personalized APIs recommendation with cognitive knowledge mining for industrial systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 9, 2021.

[30] H. Wang, N. Wang, and D. Y. Yeung, "Collaborative deep learning for recommender systems," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1235–1244, 2015.

[31] X. Wang, X. He, and L. Nie, "Item silk road: recommending items from information domains to social users," in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 185–194, 2017.

[32] S. Deng, L. Huang, G. Xu, X. Wu, and Z. Wu, "On deep learning for trust-aware recommendations in social networks," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 5, pp. 1164–1177, 2017.

[33] H. Gao, W. Huang, and Y. Duan, "The cloud-edge-based dynamic reconfiguration to service workflow for mobile ecommerce Environments," *ACM Transactions on Internet Technology (TOIT)*, vol. 21, no. 1, pp. 1–23, 2021.

[34] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[35] R. Ying, R. He, and K. Chen, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 974–983, 2018.

[36] L. Wu, P. Sun, and R. Hong, *SocialGCN: an efficient graph convolutional network based model for social recommendation*, 2018, arXiv preprint arXiv:1811.02815.

[37] W. Song, Z. Xiao, and Y. Wang, "Session-based social recommendation via dynamic graph attention networks," in *Proceedings of the Twelfth ACM international conference on web search and data mining*, pp. 555–563, 2019.

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[39] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, 2014, arXiv preprint arXiv:1409.0473.

[40] K. Ahmed, N. S. Keskar, and R. Socher, *Weighted transformer network for machine translation*, 2017, arXiv preprint arXiv:1711.02132.

[41] K. Xu, J. Ba, R. Kiros et al., *Show, attend and tell: neural image caption generation with visual attention*, 2015, https://arxiv.org/abs/1502.03044v3.

[42] H. Gao, X. Qin, and R. J. D. Barroso, "Collaborative learning-based industrial IoT API recommendation for software-defined devices: The implicit knowledge discovery perspective," in *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2020.

[43] T. Bai, J. R. Wen, and J. Zhang, "A neural collaborative filtering model with interaction-based neighborhood," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 1979–1982, 2017.

[44] L. Wu, P. Sun, and Y. Fu, "Neural influence diffusion model for social recommendation," in *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, pp. 235–244, 2019.

[45] T. Derr, Y. Ma, and J. Tang, "Signed graph convolutional networks," in *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 929–934, IEEE, 2018.

[46] C. Chen, M. Zhang, and Y. Liu, "Neural attentional rating regression with review-level explanations," in *Proceedings of the 2018 World Wide Web Conference*, pp. 1583–1592, 2018.

[47] C. Fan, H. Xu, H. Gao, X. Jiaoxiong, and W. Wei, "TRG-DAtt: the target relational graph and double attention network based sentiment analysis and prediction for supporting decision making," *ACM Transactions on Management Information Systems (TMIS)*, 2020.

[48] Y. Huang, H. Xu, H. Gao, and W. Hussain, "SSUR: an approach to optimizing virtual machine allocation strategy based on user requirements for cloud data center," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 2, pp. 670–681, 2021.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 3791859, 13 pages
https://doi.org/10.1155/2021/3791859

Research Article

Popularity-Aware In-Network Caching for Edge Named Data Network

Jiliang Yin,[1] **Congfeng Jiang**,[1] **Hidetoshi Mino**,[2] **and Christophe Cérin**[3]

[1]*School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China*
[2]*Department of Computer Science and Engineering, University of Yamanashi, Yamanashi 400-0013, Japan*
[3]*LIPN UMR CNRS 7030, Université Sorbonne Paris Nord, Villetaneuse F-93430, France*

Correspondence should be addressed to Congfeng Jiang; cjiang@hdu.edu.cn

Received 1 May 2021; Revised 20 July 2021; Accepted 16 August 2021; Published 31 August 2021

Academic Editor: Xiaoxian Yang

The traditional centralized network architecture can lead to a bandwidth bottleneck in the core network. In contrast, in the information-centric network, decentralized in-network caching can alleviate the traffic flow pressure from the network center to the edge. In this paper, a popularity-aware in-network caching policy, namely, *Pop*, is proposed to achieve an optimal caching of network contents in the resource-constrained edge networks. Specifically, *Pop* senses content popularity and distributes content caching without adding additional hardware and traffic overhead. We conduct extensive performance evaluation experiments by using *ndnSIM*. The experiments showed that the *Pop* policy achieves 54.39% cloud service hit reduction ratio and 22.76% user request average hop reduction ratio and outperforms other policies including Leave Copy Everywhere, Leave Copy Down, Probabilistic Caching, and Random choice caching. In addition, we proposed an ideal caching policy (*Ideal*) as a baseline whose popularity is known in advance; the gap of *Pop* and *Ideal* in cloud service hit reduction ratio is 4.36%, and the gap in user request average hop reduction ratio is only 1.47%. More simulation results further show the accuracy of *Pop* in perceiving popularity of contents, and *Pop* has good robustness in different request scenarios.

1. Introduction

With the rapid expansion of the Internet, diversified Internet content services are deployed based on the *publish-subscribe* [1] service model. This model enables the service provider to publish or share content in a central server, while all its subscribers can access the content independently through the distributed networking connections. However, the *publish-subscribe* model can lead to a bandwidth bottleneck in the backbone network in the traditional Internet protocol-(IP-) based network architecture. For a large-scale content service, the subscribers' considerable data transmission volume will converge to the backbone network [2]. For alleviating this problem, information-centric networking (ICN) [3, 4] is proposed. And as a crucial branch of ICN, the Named Data Networking (NDN) [5–7] uses names instead of IP to rout and forward packets. Compared with the IP network, the requested content name is shifted from the application layer to the network layer to provide packet caching and hitting. Thanks to its support of content-centric in-network caching [8, 9], NDN is promising for application in large-scale content services [10–12]. Moreover, placing content caching at the edge network can also reduce network latency.

However, the caching capacity of edge devices is around 3 to 4 orders of magnitude lower than that in the cloud. Intuitively, the native caching policy is to cache every packet that arrived at nodes. Unfortunately, this implementation will result in very high content redundancy in the service network and waste many edge caching resources. Besides, the significant magnitude difference between the number of contents and edge node caching capacity will cause any caching eviction policy to become invalid. Because the following content will continuously replace the cached content in the node, and any valuable content cannot be kept in the node. Consequently, it is essential to distinguish popular content and reduce content redundant as much as possible in the edge caching process [13–15]. Moreover, since all of the in-network caching are triggered by user behaviors, and the caching node can only passively cache arrived packets. So, the caching nodes cannot obtain explicit global knowledge about content popularity when making caching decisions. These all urge the adaptive caching decision mechanism to maximize caching utilization and reusability in resource-constrained edge networks.

For tackling the above problems, it is essential to design an in-network caching policy based on content popularity awareness. It should properly consider the characteristics of the user content requests and avoid additional overhead in the network. In addition, it is necessary to keep the caching redundancy as low as possible and achieve excellent robustness to adapt to various request scenarios. Given the above analysis, a popularity-aware in-network caching policy, namely, *Pop*, is proposed to achieve optimal in-network caching in the resource-constrained edge networks. Specifically, *Pop* allows the node to perceive the popularity of the current arrived content during the transmission process and then decide whether to cache based on the net location of the node. In the above process, there is no need to add additional hardware and traffic overhead.

Our contributions are listed as follows:

(1) We designed a popularity-aware in-network caching policy (*Pop*) for the edge network. It utilizes the existing Pending Interest Table (PIT) as a recording mechanism for current content request status. By distinguishing the request probability of content with different popularity, *Pop* naturally makes caching decisions and achieves distributed caching based on content popularity

(2) We introduced Cached Tag in the header of the returned packet to prevent the cached content from being cached again by downstream nodes, effectively reducing content redundancy in the edge network. Tag reset mechanism will be triggered by cache hit; it can effectively avoid popular content be incorrectly cached far from the edge and also can increase the utilization of edge caching space

(3) We conducted a comprehensive evaluation of *Pop* through network simulation. Compared with four existing policies and a self-designed ideal caching policy (*Ideal*), we prove that *Pop* has excellent performance. Besides, *Pop* also shows considerable advantages in terms of network-level caching distribution and content update responsiveness

(4) Complex and diverse network scenarios were simulated, which proves that *Pop* has good robustness. In large-scale and high-load scenarios, the performance of *Pop* is always better than other policies (except *Ideal*). In imbalanced-content request, *Pop* maintains its superior performance in most cases and only declines in a few extreme cases

The remainder of this paper is organized as follows. Section 2 introduces the related work of in-network caching. Section 3 introduces the system model and related assumptions. Section 4 introduces the details of the *Pop* in-network caching policy. In Section 5, performance evaluation of the *Pop* is showed. Section 6 discusses the variants of *Pop* and the real deployments in edge environments. Section 7 summarizes our work.

2. Related Work

Unlike traditional caching systems, the implementation of in-network caching can be divided into caching eviction process and caching decision process.

Caching eviction process is when the arrived content is determined to be cached, but the caching space is full, then it needs to decide which stale content should be evicted from the node. Well-known caching eviction policies include FIFO (First Input First Output), LFU (Least Frequently Used), Random, and LRU (Least Recently Used). Among them, LRU is the most widely used in ICN in-network caching research [16, 17]. In addition, LFF [18] (Least Fresh First) is used in IoT networks; it aims to use the ARMA model to predict the time of the next content update event in the sensor, so that the predicted remaining time will be set as the freshness of contents in caching nodes.

Caching decision process is when the content has arrived; the node needs to decide whether to cache it into caching space. Due to the limitation of node caching capacity, when all of the arrived contents are cached, the cache hit efficiency will be very low. As the native caching mechanism of the NDN, it allows every node to cache every arrived content, called LCE [6] (Leave Copy Everywhere). It is proven that results in high redundancy and low utilization in edge storage resources. Ber [19] (Bernoulli random caching) is a Probabilistic Caching policy that satisfies Bernoulli distribution, which allows content to be cached with a fixed probability on each node of the return path. Moreover, some researchers began to construct more complex formulas for optimizing caching probability. They introduced different parameters, which lead the caching probability to change dynamically according to node position and status. ProCache [20] (Probabilistic Caching) mainly introduces the distance and the remaining storage space, so that the higher caching probability will belong to the node close to the user and with more caching space. The pCASTING [21] (Probabilistic Caching strategy for the Internet of Things) policy applies to energy-constrained and wireless IoT networks. It introduces the remaining power, remaining storage space, and content freshness as caching probability parameters. It leads the power consumption of IoT equipment in a balanced way and effectively increases the working time of the entire IoT network. LCD [22] (Leave Copy Down) allows the returned (or cache hit) content to be cached only on its next-hop node, so the caching location of the content can be gradually moved to the user through multiple be requested. However, its upstream nodes will have many times of content eviction and redundancy in caching process.

On the other hand, other caching policies allow every content to be cached on only one node, so that to minimize network redundancy. The key to these policies is the selection of the caching node. The Ran (Random choice caching) policy does not consider any external features and randomly selects one of the downstream nodes for caching. The Betw [23] (Betweenness centrality caching) policy believes that caching the content on the node with the largest betweenness centrality can maximize future benefits. Hash [24, 25] (Hash-routing caching) policy expands the range of caching nodes from the return path to the entire network. It uses a hash function to map content to nodes in different net locations by content names, but it has high implementation complexity.

Besides, there are other policies consider in other ways. For PPCS [26] (progressive popularity-aware caching scheme), it designs from caching decision and eviction. During the first transmission, PPCS divides the larger content into several smaller packets for distributed caching. If this content is requested multiple times in the following time, it can gradually push all packets of this content to the terminal caching nodes by an intelligent caching scheme. Newberry and Zhang [27] considered the in-network caching for the Hadoop distributed file system. By comparing multiple caching eviction policies, it is found that the size of the packet capacity has a more significant impact on the performance of the caching efficiency.

The above work has done much research on improving the efficiency of the in-network caching from many aspects. However, almost none of them considers the relationship between content popularity and its request probability in the tree-like network, nor did comparison with an ideal caching policy to clearly show the gap of performance between their policy and the ideal situation. Therefore, we propose the *Pop* policy and design *Ideal* policy to compare with it. We hope our work can bring a new view to the in-network caching research and help improve its performance.

3. System Model and Assumptions

We argue that the popularity of content is the crucial factor affecting the benefits of in-network. In Section 3.1, we describe a typical network architecture that we considered. In Section 3.2, we present the assumptions and characteristics of user requests.

3.1. System Model. Typical Internet content services are deployed in hierarchical network architectures, as shown in Figure 1. In fact, all devices on the path from the cloud to user can be used as caching nodes, but we focus on the nodes in the edge environment. Obviously, it can alleviate the pressure of the backbone network and get the low-latency response from edge caching nodes.

3.2. Request Assumptions. Internet content services model include diverse request scenario. We mainly consider the following characteristics and make some assumptions based on generality.

Request generation. We suppose the request generation meets the *Poisson* distribution.

Content Popularity. Due to the content preference of consumers, most requests tend to focus on a small amount of content. Relevant research [28] has observed that the popularity of the requested contents on the web meets *Zipf-like*

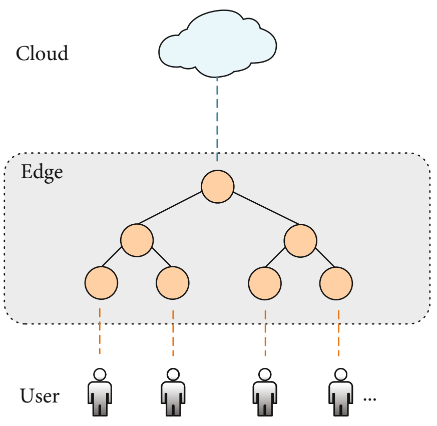


Figure 1: Hierarchical network architecture, where cloud node is the location of publishing server, and it interacts with users through edge network.

law. We suppose the content popularity distribution meets the *Zipf-Mandelbrot* law. When the total number of contents is N, the popularity ranking of the content is k; the probability that it be requested is

$$f(k, N, q, s) = \frac{1/(k+q)^s}{H_{N,q,s}}. \tag{1}$$

Here, q and s are parameters that affect the distribution, and $H_{N,q,s}$ is given as follows.

$$H_{N,q,s} = \sum_{i=1}^{N} \frac{1}{(i+q)^s}. \tag{2}$$

Request probability. In a tree-like network, requests will be collected in upstream nodes. When we consider the probability of a specific content has been requested over a period of time, its popularity will lead to very different probability distribution at every network level.

We consider the number of requests received by a node within a period of time is n, and the popularity of the content is p. Then, the probability P that this node received this content can be given as follows.

$$P = 1 - (1-p)^n. \tag{3}$$

The n is related to the user request frequency and the network topology, as shown in Table 1. If we suppose a standard z-branches tree network, the average frequency of user requests is x, and the total network level is K; the n of level k nodes can be given by (4)

$$n = z^{K-k}x. \tag{4}$$

A specific example is in Figure 2. It shows that contents with different popularity have different request probability distributions at caching nodes. Take $P = 0.9$ (thin dotted line in the figure) as a reference. 3.00% popularity content from cloud to $L4$ can receive its request more than 90%

Table 1: 6-layer standard tree network request distribution.

Level	Number of received requests n		
	Binary tree	Trinomial tree	z-branches tree
$L1$	$32x$	$243x$	z^5x
$L2$	$16x$	$81x$	z^4x
$L3$	$8x$	$27x$	z^3x
$L4$	$4x$	$9x$	z^2x
$L5$	$2x$	$3x$	zx
$L6$	x	x	x

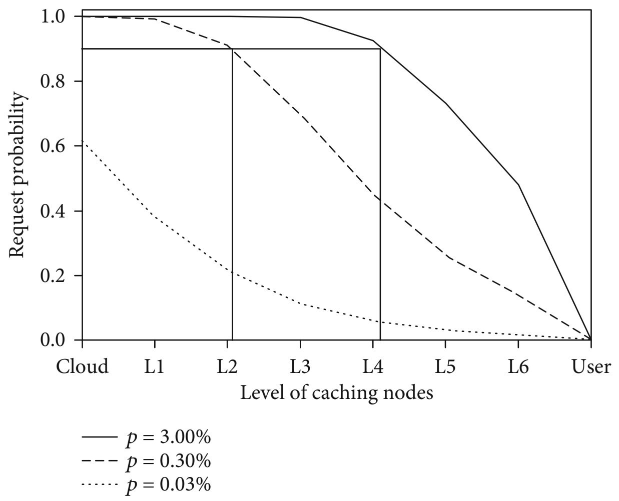


Figure 2: An example about the request probability distribution of content with different popularity in the binary tree network. The popularity of the three contents is 3.00%, 0.30%, and 0.03%; the frequency of single-user requests is 50. There is no caching mechanism in the user node, so the probability is 0 in user.

probability. In comparison, content with 0.30% popularity from cloud to $L2$ can receive its request more than 90% probability. As for content with a popularity of 0.03%, the request probability has always been lower than 62%.

4. Popularity-Aware In-Network Caching Policy

Identifying the popularity of contents and caching them to the suitable network level is crucial for the popularity-aware caching policy (*Pop*). This section will present a demonstration of *Pop* first and then introduce the implementation of the key mechanisms in detail.

According to the analysis in Section 3.2, the popularity of content will affect the request probability distribution in each level node. So, we can distinguish the popularity of contents according to the request records in caching nodes.

As a demonstration, we still use the standard binary tree network to explain main idea of the *Pop* policy. We use LRU as the eviction policy due to its excellent performance (evaluation details in Section 5.2), and the processing of caching decision is showed as Figure 3. Now, we consider a user node sends three contents requests to the cloud. They are high popularity content *Ra*, medium popularity content *Rb*, and low popularity content *Rc*. These PIT tables (NDN module, details in Section 4.1) will record three content requests (*Ra*, *Rb*, and *Rc*), and their in-records are all marked in the right sub-Face. After that, while waiting for the cloud content return, the *N1*, *N2*, and *N3* nodes will continue to receive content requests (*R* shown in Figure 3) from the upper sub-Face. According to the analysis of request probability, content popularity, and node location in Section 3.2. The *N1* node has a great chance to receive *Ra* and *Rb* content requests on the upper sub-Face. *N2* node may receive *Ra* request. The probability of the *N3* node receiving these content requests is extremely low.

Therefore, when *Ra*, *Rb*, and *Rc* are returned, each caching node will make the caching decision based on the PIT table. We set the caching condition that the returned content will be cached in the first node, which does not record its request in both sub-Faces. According to the records shown in Figure 3, node *N1* will cache *Rc*, node *N2* will cache *Rb*, and node *N3* will cache *Ra*. So far, the *Pop* policy has completed a distributed caching based on the content popularity and caching node network level.

Besides, the mechanism for reducing caching redundancy is in Section 4.2. The detail and the related algorithm extend the above caching condition to a more generalization and complex hierarchical network are in Section 4.3.

4.1. Request Recording Mechanism. We use NDN as the underlying network. NDN allows establishing communication interfaces between nodes through various underlying protocols, such as Ethernet, TCP, UDP, and Socket. They are unified and abstracted as Face. The communication of NDN is launched by the users, and their requests will be encapsulated in a packet called *Interest* and routed to the cloud. The content returned by the cloud will be encapsulated in a packet called *Data* and routed to users. The PIT (Pending Interest Table) is the native mechanism of NDN, whose function is to record the name of *Interest* that the node has forwarded but has no *Data* returned from the upstream. Therefore, we can directly use PIT as the request recording mechanism and avoiding the additional design.

As shown in Table 2, PIT mainly consists of three parts. Entry is the Interest name that has been forwarded. In-record is the downstream source Face ID. Out-record is the upstream forwarding Face ID. The records in the table indicate that the node has received *Ra* and *Rb* requests, where *Ra* has been requested on both Faces 1 and 2, and *Rb* only has requested on Face 2. Do not consider other records. According to the above caching conditions, *Ra* will be directly forwarded to Faces 1 and 2; *Rb* will be cached and then forwarded to Face 2. And then, the node will immediately clear all *Ra* and *Rb* records in the PIT. The maintenance of the PIT table is implemented by the Named Data Networking Forwarding Daemon (NFD) [29].

4.2. Cached Tag Design of the Returned Data. Only based on the above caching condition, it cannot avoid the returned

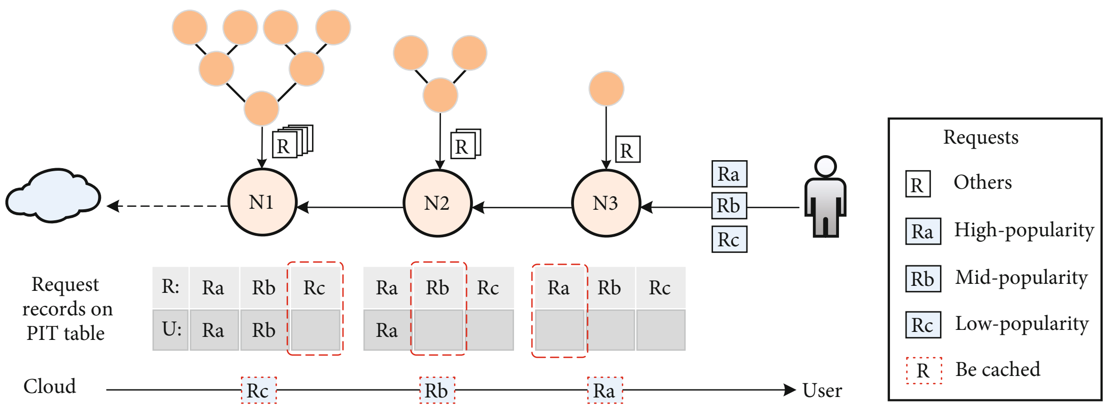


FIGURE 3: Demonstration of *Pop* in distinguishing the popularity of contents and caching them in opportune node. The user sends different popularity requests to the cloud. When contents are returned, they are cached based on their popularity and caching node network level.

TABLE 2: Record of received Interest in PIT.

Entry	In-record	Out-record
/prefix/Ra	Face 1	Face 0
/prefix/Ra	Face 2	Face 0
/prefix/Rb	Face 2	Face 0

Data be repeated caching on the return path. We designed the Cached Tag to solve this problem.

As shown in Figure 4, we denote the returned *Data* as *C* and add the Cached Tag to its header field. When *Data C* returns from cloud, its Tag is initialized to 0. As the *Data C* hops to the user node along the return path, it will go through the Tag detection process on each caching node. At node *N1*, *Data C* does not meet the caching condition, the value of Tag unchanged, and it is directly forwarded to node *N2*. At node *N2*, *Data C* meets the caching condition; the value of Tag is changed to 1, and then, it is forwarded to node *N3* after caching. After receiving *Data C*, the *N3* node checks that the Tag of *C* is 1, so it directly abandons the caching and forwards it to the user.

Besides, we also designed a Tag reset mechanism to optimize the efficiency of content caching further. Specifically, when the request triggers cache hits on the caching node, the returned *Data* is allowed to reset the Tag value to 0, thereby obtaining another caching opportunity. The Tag reset mechanism will generate benefits from two aspects.

First, solve the problem of content caching mismatch. The caching decision of *Pop* cannot be guaranteed 100% accuracy. It may occur that the cached content and the cached location do not match. As shown in Figure 5, a mismatch content (with high popularity) is incorrectly cached on a far node. Here, the Tag reset mechanism allows mismatch content to be cached at the next cache hit. The popularity-aware mechanism will smoothly cache it to the matching node (edge node).

Second, use the invalid (stale or idle) caching resources of edge nodes and optimize cache hit efficiency. The distribution of content popularity is imbalanced. The amount of high or medium popularity content is far less than the low. Reflected in the caching results, strictly caching based on content popularity will lead to a low caching utilization in middle or edge nodes. Here, the reset mechanism allows these contents on the far nodes to be cached to the middle or edge nodes, reducing the number of hops for following repeat requests.

The Tag reset mechanism will not impact the valid content in the caching nodes. The LRU caching eviction policy is deployed in each node, and it will maintain the currently popular content at the top.

4.3. Threshold-Based Adaptive Caching Decision Algorithm.

Given the complexity and heterogeneity of the real network, *Pop* policy must adaptively complete the caching decision process.

As shown in Figure 6, we introduce a threshold *T* to implement the *Pop* adaptive caching decision in the diversified nodes of the nonstandard tree network. We abstracted *Request_ratio*, which represents the ratio between *Ncur* (the number of sub-Faces with this content request) and *N* (the total number of sub-Faces). *Request_ratio* describes the popular degree of content on downstream nodes, and the value range is (0, 1]. The higher the value, the more popular the content.

In Figure 6, the threshold *T* is set to 0.6. The (a) depicts a 3-branch caching node with three sub-Faces, and the (b) depicts a 5-branch caching node with five sub-Faces. Based on the unified threshold *T*, they can make the same caching decisions on *Ra* and *Rb*.

Algorithm 1 is deployed on all caching nodes uniformly. The input includes returned *Data*, *PIT*, number of sub-Faces *N*, and the custom caching threshold *T*. When *Data* returns from the upper Face of the node, the caching decision algorithm is triggered. The algorithm first checks the *CacheTag* in the *Data* (line 2). If $ChaheTag! = 0$, it proves the upstream nodes have cached the *Data*, forward the *Data* directly (line 2), and the process finishes. Otherwise, *Data* will enter the caching decision process (lines 5-15).

For caching decision process, the algorithm decodes the *Data* and obtains the content name. It then queries the *PIT* according to the name and gets the *N_cur* (line 6). Next, calculate *Request_ratio* (line 7). By comparing *Request_ratio* and threshold *T*, the algorithm can obtain two decision

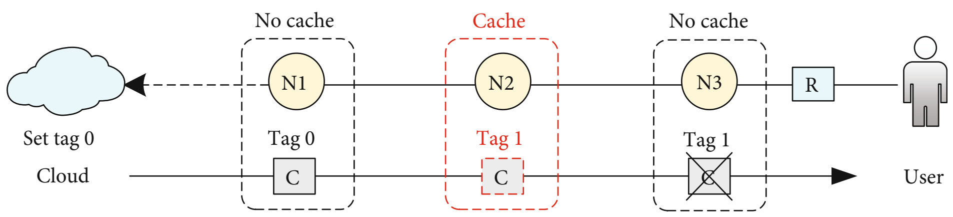


Figure 4: The process of returned *Data* with Cached Tag. The return *Data* with initial Cached Tag 0, no cache in *N1*, change Tag to 1 after caching in *N2*, give up cache in *N3*.

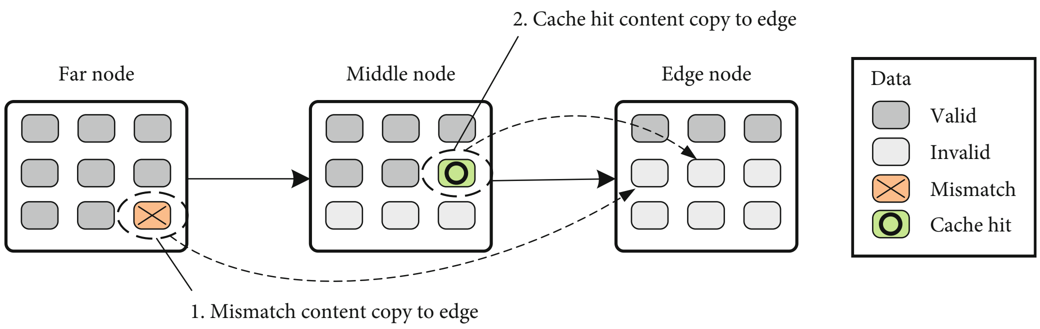

2. Cache hit content copy to edge
Far node
Middle node
Edge node
Data
Valid
Invalid
Mismatch
Cache hit
1. Mismatch content copy to edge

Figure 5: Two cases of Cached Tag reset. The mismatch content and cache hit content will have additional chance to cache close to edge.

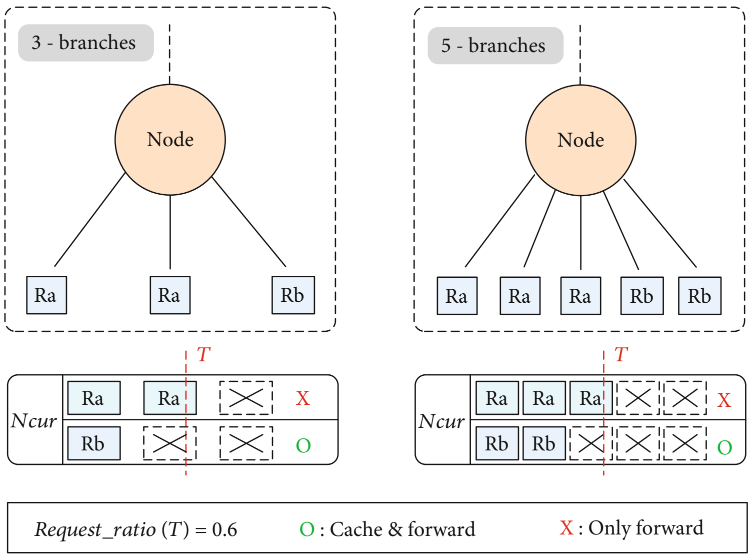


Figure 6: Adaptive content caching based on *Request_ratio*. Set $T = 0.6$. In (a), *Ra* is not cached ($2/3 > T$), and *Rb* is cached ($1/3 < T$). In (b), *Ra* is not cached ($3/5 > T$), and *Rb* is cached ($2/5 < T$).

results. If the *Request_ratio* is greater or equal to T (line 8), it proves that the content popularity is too high and does not match this caching node, and the *Data* will be forwarded directly (line 10). Otherwise, proving that the content popularity meets the current node caching requirement. The algorithm first sets the *Data.CacheTag* to 1 (line 13), then cache it (line 14), and finally forwards it (line 15) to the sub-Faces.

5. Performance Evaluation

In order to verify the effectiveness of *Pop*, we used ndnSIM [30–33] to perform a comprehensive simulation. Section 5.1 introduces the parameter configuration and indicators. Section 5.2 is the determination of the eviction policy, and Section 5.3-5 analyzes the performance in different request scenarios.

5.1. Simulation Design. The parameter configuration of ndnSIM is shown in Table 3. Please note that some parameters will change in follow simulation scenarios, and their change range is marked with "{...}."

For the evaluation of in-network cache policy, the main performance metrics are designed as follows.

```
Input: Data - Packet of Data;
       PIT - PIT table of NDN;
       N - Number of sub-Face;
       T - Threshold for caching content;
Output: Caching Decisions - (i). Cache & Forward; (ii). Only Forward.
//Deployed in all of caching nodes
1.  While ( new Data is return ) {
2.    if (Data.CacheTag != 0 ) {
3.      Forward(Data) // This Data has cached in upstream node, just forward it
4.    }else{
5.      // N_cur is number of sub-Face which has requested this Data
6.      N_cur = Search(http://Data.name, PIT)
7.      Request_ratio = N_cur / N // Get Request_ratio
8.      if (Request_ratio >= T){
9.      // Data was requested on most sub-Face, it's suitable for caching at downstream node
10.       Forward(Data) // Forward Data
11.     }else{
12.       // Data was requested on few sub-Face, it's suitable for caching at here
13.       Data.SetCachedTag(1) // Set cached Tag = 1
14.       Cache(Data) // Cache Data
15.       Forward(Data) // Forward Data to downstream nodes
16.     }
17.   }
18. }
```

Algorithm 1: *Pop* caching decision algorithm.

Table 3: Simulation parameters.

Type	Name of parameter	Value
Network	Basic network topology	Binary tree
	Network level	5-level
	Link capacity to cloud	1 Gbps
	Link capacity in edge	100 Mbps
	Propagation delay to cloud	500 ms
	Propagation delay in edge	1 ms
Publish	Total number of contents	100,000 {20,000~ 100,000}
	Content update cycle	5 s
Request	Number of user nodes	16 {16~ 81}
	Request frequency	200/s {100~ 300}
	Request generation	*Poisson*
	Content popularity	*Zipf-Mandelbrot*
	Zipf parameters (q&s)	1.0 and 1.0
Caching	Caching eviction strategy	LRU {Random, LFU, FIFO, TTL}
	Caching node capacity	50
	Payload size (content size)	1024 Byte
	Pop caching threshold *T*	0.6
Simulation	Simulation duration time	201 s

(1) *Cloud Service Hit Reduction Ratio*. This indicator measures the in-network caching policy efficiency in reducing the pressure of the cloud. We use H_C as the number of nonfirst requests cache hit on cloud node and H_E as the number of requests cache hit on edge caching nodes. The value range of α is (0,1)

$$\alpha = 1 - \frac{H_C}{H_C + H_E} = \frac{H_E}{H_C + H_E}. \tag{5}$$

(2) *User Request Ave-Hop Reduction Ratio*. This indicator measures the efficiency of in-network caching

policies in improving the quality of user requests. *Ave_hops* represent the average hops to complete the user requests under caching policies, and *Hops_to_Cloud* represents the total hops from the user to the cloud. The value range of β is (0, 1/*Hops_to_Cloud*)

$$\beta = 1 - \frac{\text{Ave\_Hops}}{\text{Hops\_to\_Cloud}}. \tag{6}$$

(3) *Single-Layer Contribution*. This indicator measures each layer caching nodes' contribution in reducing the pressure of the cloud. It can also be understood as the probability of triggering cache hit on each layer. Where l represents the level of nodes, H_{El} represents the total number of requests cache hit on layer l. $\sum H_{El}$ is the total number of requests cache hit on the entire edge network

$$\gamma(l) = \frac{H_{El}}{\sum H_{El}}, l = 1, 2, \cdots \cdots, n. \tag{7}$$

(4) *Ideal In-Network Caching Policy*. To test the *Pop* accuracy in perceiving content popularity, we design the *Ideal* policy as a comparison. In its implementation, the *Data* returned by the cloud will carry the content popularity information. It allows the caching nodes to make caching decisions based on the known popularity information and achieve the ideal caching performance. Through comparison, we can see the gap between the *Pop* and the *Ideal*

5.2. Determination of the Eviction Policy. Before the *Pop* performance evaluation, we need to determine the caching eviction policy. Under the above default configuration, we have performed the evaluation in four policies, included Random, LFU, FIFO, TTL, and LRU. As shown in Figure 7, Random and LFU all show the low reduction ratio; FIFO and TTL have similar performance, and LRU has the best performance.

Therefore, based on the above comparison results, it can find that the LRU caching eviction policy has appropriate performance. So, in the following experimental scenarios, *Pop* and other comparison policies will use LRU as the caching eviction policy.

5.3. Content Update. We thoroughly evaluated the performance of the *Pop* in content update scenarios. By comparing with other existing in-network caching policies (LCE, Prob-cache, LCD, Ran) and *Ideal* policy, we prove the significant advantages of *Pop*.

(1) *In-Network Caching Benefit.* As shown in Figure 8, excepts for the *Ideal* policy, *Pop* has the best performance in the reduction ratio of cloud service hit and user request average hops

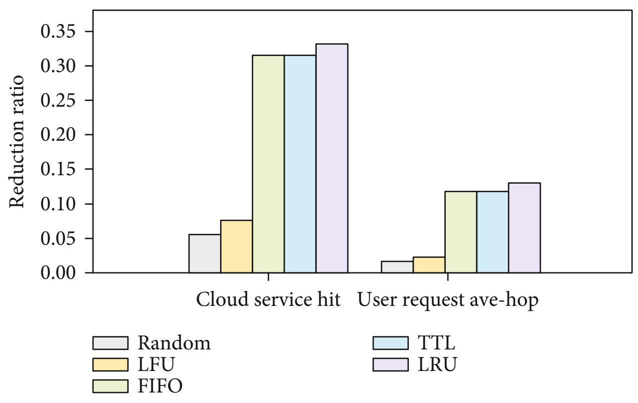


FIGURE 7: Comparison of cloud service hit and user request ave-hop reduction ratio in eviction policies. LRU has the best performance; its reduction ratio is 33.07% and 12.97%, respectively.

As Figure 8(a), the *Pop* reduced 54.39% of cloud service cache hit. Compared with the native policy LCE, the performance is improved by 21.32%. Compared with the suboptimal policy Ran, the performance is also improved by 9.7%. The performance gap between *Pop* and *Ideal* is 4.36%. As Figure 8(b), *Pop* has reduced 22.76% of user request average hops. Compared with other existing policies, the improvement range is 5.98% ~ 10.11%, and the gap with *Ideal* was only 1.47%.

It is worth noting that, due to the random selection of caching nodes, the Ran has well performance in the reduction ratio of cloud service hit, but its performance on reduction of average user request hops is lower than all of the other policies.

(2) *Contribution of Each Layer*. Furthermore, we counted the contribution of each layer. Figure 9 shows the distribution of the single-layer contribution in different in-network caching policies. Except for Ran, the contribution distribution of other caching policies decreases as the network level close to cloud ($L1$). The *Pop* caching result is very close to *Ideal*, which can reflect that *Pop* popularity awareness is accurate. Ran randomly selects the caching nodes so that all of the content is evenly cached on each layer nodes

(3) *Responsiveness to Content Updates*. Content updates will make the cached contents stale and invalid; they should be replaced by new contents quickly. In order to evaluate the *Pop* performance under dynamic scenarios, we show the responsiveness of different in-network caching policies

As shown in Figure 10, it fully proves that *Pop* responds very quickly to content updates (except for the *Ideal*). Specifically, the average hops of each caching policy change periodically. After every content update, the *Ideal* finishes new content caching at first, thereby quickly reducing the average hops. The *Pop* policy is second only to the *Ideal*. In the initial

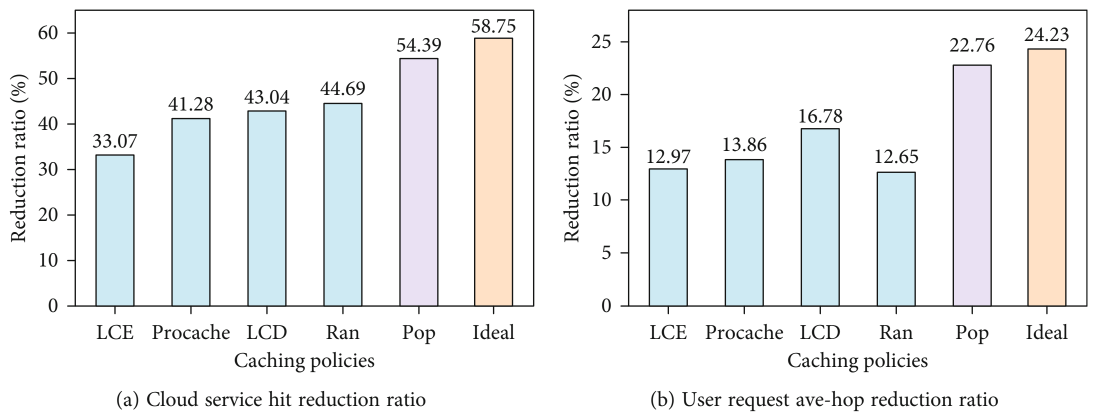


Figure 8: Comparison of cloud service hit and user request ave-hop reduction ratio in different in-network caching policies. *Pop* reduced 54.39% of cloud service cache hit and 22.76% of user request ave-hop.

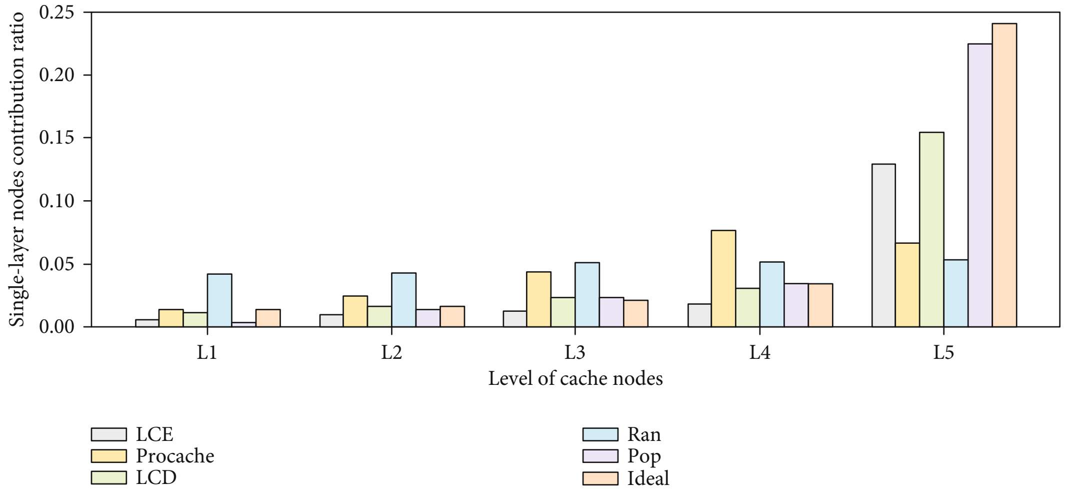


Figure 9: Comparison of single-layer contribution in different in-network caching policies. *Pop* caching result was close to *Ideal*.

stage, the *Pop* can quickly reduce the average hops at almost the same speed as *Ideal*. However, in the following stages, the reduction of the *Pop* was slightly insufficient. It shows that *Pop* can accurately perceive most high or medium popularity content and quickly cache them directly to the appropriate nodes. However, for some low popularity content, its perception ability may not be sensitive.

5.4. Large-Scale and High-Load. To test the *Pop* policy robustness, we changed different simulation parameters based on the above content update scenario. It can show the *Pop* performance from three dimensions: the number of published contents, the scale of the edge user, and the frequency of user requests.

As shown in Figure 11(a), when the number of published contents gradually increases (from 20,000 to 100,000), the cache hit reduction ratio of LCE, LCD, Procache, and Ran policies all dropped slightly. Only the *Pop* and *Ideal* policies showed an upward trend. It proves that facing the explosive growth of content volume, the in-network caching policy based on popularity-aware has significant advantages.

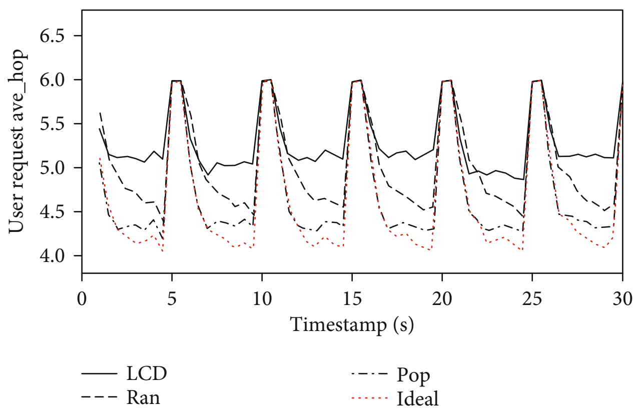
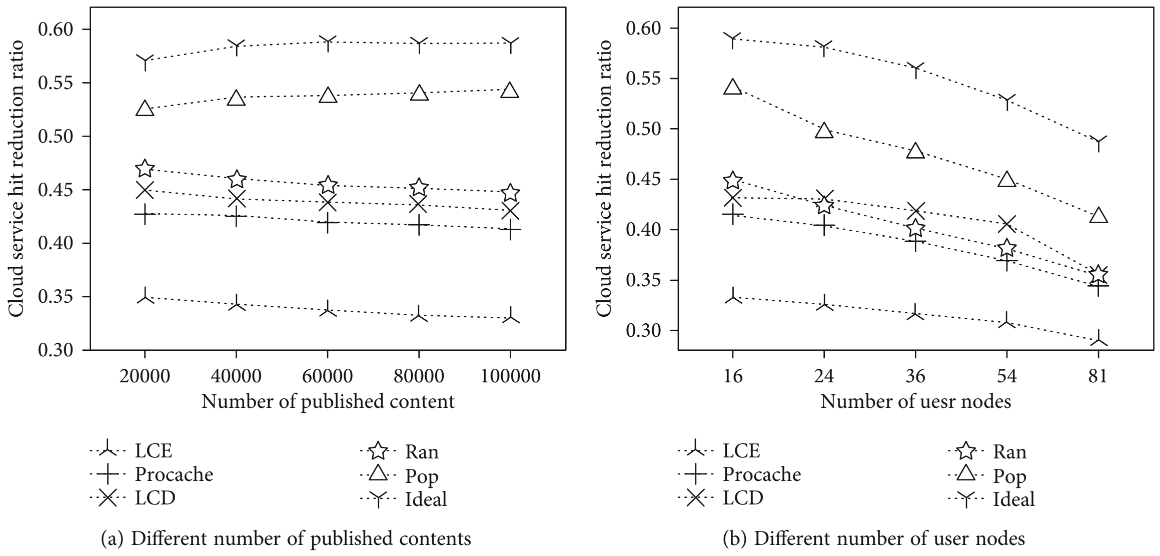


Figure 10: Comparison of content update responsiveness in different in-network caching policies. Content update cycle is 5 s; statistics period of average hops is 500 ms. Due to the delay, the line does not start at time 0, and there is a short interval after each update.



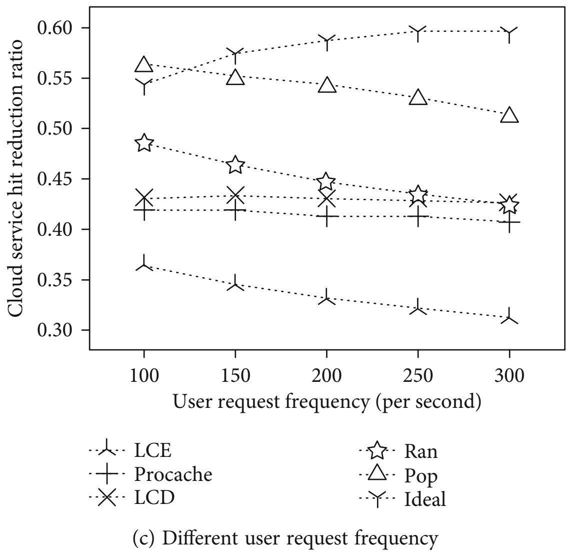


Figure 11: Cloud service hit reduction ratio in large-scale and high-load scenarios.

Figure 11(b) describes the cache hit reduction ratio in the different number of user nodes. All caching policies have the downward trend. However, within this experiment range, the *Pop* cache hit reduction ratio remains between 42% and 55%. Compared with other policies (except *Ideal*), it still has the best performance.

Figure 11(c) is an evaluation of different user request frequencies. Except the *Ideal* policy, the cache hit reduction ratio of other caching policies all meet the downward trend. Besides, it is worth noting that *Pop* performance is better than *Ideal* when the request frequency is 100.

The above three sets of simulation experiments verify the *Pop* excellent performance and robustness in large-scale and high-load content update scenarios. It also shows the considerable potential of deploying *Pop* in real networks in the future.

5.5. Imbalanced-Content Request. We considered the imbalanced-content request scenario to test the *Pop* performance degradation under extreme conditions.

In the experimental design, we divided the imbalance degree of user request into 16 levels, represented as $L0 \sim L$ 15. We still use the default network topology and parameter configuration, but some adjustments have been made to the published contents and user node request process. Specifically, the 100,000 published contents are divided into 16 groups on average. Each group has 6,250 contents and has an independent popularity distribution. Correspondingly, we also created 16 different request processes on each user node, and each process is only allowed to request one group of contents. When the imbalance level is $L0$, each user node starts 16 request processes, and the requested content pool is 100,000. When the content request level is $L15$, each user node can only start one content request process, and the requested content pool capacity is 6250. The requested content pool capacity is no overlapping part.

As shown in Figure 12, for the convenience of observation, the cache hit reduction ratio is normalized to *Ideal* policy. We can find that the *Pop* policy can still guarantee a high

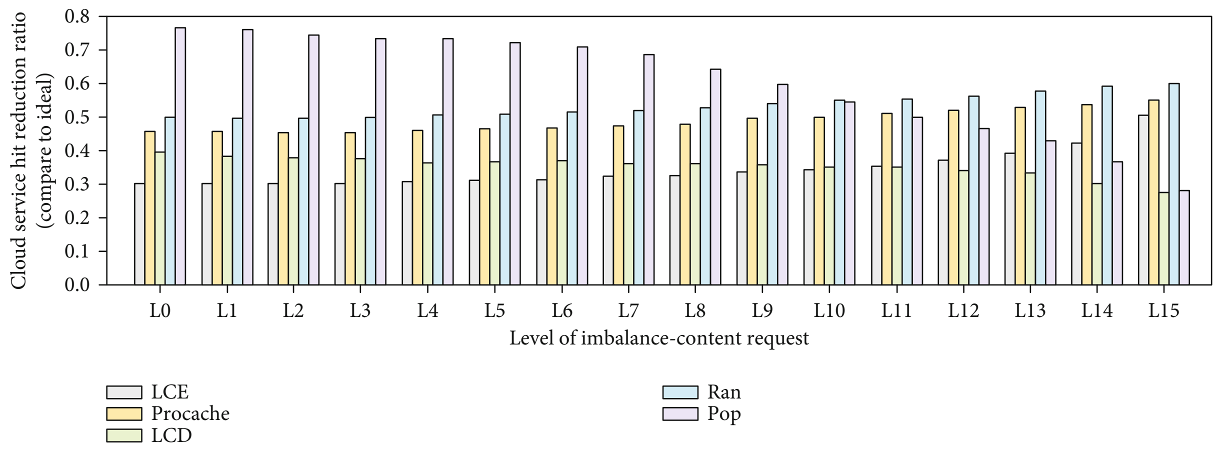


Figure 12: Cloud service hit reduction ratio (compare to *Ideal*) in imbalanced-content request.

reduction ratio in most imbalanced levels. Specifically, the *Pop* policy has the best performance at the $L0 \sim L9$ levels. At $L10 \sim L13$ levels, the hit reduction ratio remains in the top three. Nevertheless, under the extreme conditions of L 14 and $L15$, its performance drops rapidly.

In the real content request scenario, requests for high popularity content have commonalities. It generally does not happen that all users have an independent content interest. The above evaluation results show that *Pop* can be applied to most request cases.

6. Discussion

This section will discuss the content that was considered during the research but not described above.

6.1. Variant. Under the *Pop* policy, high popularity content will be forwarded to downstream nodes for caching, and low popularity content will be cached on upstream nodes. However, in a tree network, the number of upstream nodes is much smaller than that of downstream nodes. In the above mechanism, similar high popularity content will be cached in many downstream nodes. It will increase the redundancy of the edge network. An opposite variant is to cache the high popularity contents on the upstream caching nodes and store low popularity content in downstream nodes. This variant can effectively avoid the redundancy of high popularity contents, but the requests will be satisfied on the farther caching node.

Moreover, caching high popularity contents at the upstream caching nodes will making them to be another "network center." A large number of requests will be converged to the upstream nodes, and the bandwidth bottleneck that existed in the cloud will be transferred to these nodes. It goes against the original intention of in-network caching that distributes the cloud pressure to the entire network to achieve more efficient resource utilization and service response.

In fact, the in-network caching is essentially a technology that uses "space" to exchange "efficiency." The deployment of caching policy is inevitable to increase the content redundancy of the network. In the design of caching policy, we should focus on reducing the redundancy of the request path, not the redundancy of the entire network.

6.2. Real Deployment. Given much research on the IoT and edge computing [34–40], introducing containerization and serverless platform to the edge environment maybe is the best practice to implement *Pop*.

First, due to the compatibility between NDN and IP network, *Pop* can be easily deployed on traditional edge servers. Secondly, the number of requests will change over time; it requires that the resources used for caching be dynamically adjusted according to the actual workload. By using the container-based serverless platform can solve this problem very well. It can automatically control the number of containers and effectively weighing the balance between service quality and resource utilization. Finally, in the design of *Pop*, the information used in the caching decision algorithm can be obtained by itself and does not need to communicate with other external nodes. Therefore, it is possible to use stateless functions to implement *Pop* and dramatically simplifies its deployment.

In addition, the caching capacity of caching nodes and the threshold of caching decisions can be considered dynamically adjusted according to the current workload. In this way, load balancing and caching efficiency of the entire edge network node may be achieved.

7. Conclusions

We propose a popularity-aware in-network caching policy (*Pop*). The design of *Pop* makes full use of the PIT table and the distribution of request probability in the tree-like network. At the same time, the design of the Cached Tag in the *Data* header ensures low content redundancy in the edge network, and the Tag reset mechanism effectively optimizes the performance of *Pop* caching.

For the further work, we will comprehensively evaluate the performance of *Pop* and prove that it has obvious advantages over other existing in-network caching policies in many aspects. In addition, through the design of multiple scenarios, we have proven that the *Pop* policy has good robustness. We also discussed the significance of in-network caching and the possibility of introducing containerization and serverless platform to the implementation of *Pop*.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest in the work.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (No. 61972118) and the Key Research and Development Program of Zhejiang Province of China (No. 2019C01059).

References

[1] N. Fotiou, D. Trossen, and G. C. Polyzos, "Illustrating a publish-subscribe Internet architecture," *Telecommunication Systems*, vol. 51, no. 4, pp. 233–245, 2012.

[2] F. J. M. Muro, N. Skorin-Kapov, and P. Pavon-Marino, "Revisiting core traffic growth in the presence of expanding CDNs," *Computer Networks*, vol. 154, pp. 1–11, 2019.

[3] X. Qiao, H. Wang, W. Tan, A. V. Vasilakos, J. Chen, and M. B. Blake, "A survey of applications research on content-centric networking," *China Communications*, vol. 16, no. 9, pp. 122–140, 2019.

[4] D. Mars, S. Mettali Gammar, A. Lahmadi, and L. Azouz Saidane, "Using information centric networking in Internet of Things: a survey," *Wireless Personal Communications*, vol. 105, no. 1, pp. 87–103, 2019.

[5] L. Zhang, D. Estrin, J. Burke et al., "Named data networking (NDN) project," Dept. Comput. Sci., Univ. California, Los Angeles, CA, USA, 2010, Tech. Rep. NDN-001.

[6] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pp. 1–12, Rome, Italy, 2009.

[7] N. D. N. Project, "Named data networking," February 2021, http://named-data.net/.

[8] M. H. Al-Adhaileh, F. Muchtar, A. H. Abdullah, and P. K. Singh, "The Significance of Using NDN in MANET," *Proceedings of ICRIC*, , pp. 421–437, Springer, Cham, Switzerland, 2020.

[9] Z. Yan, Y. Park, Y. Leau, L. Ren-Ting, and R. Hassan, "Hybrid network mobility support in named data networking," in *2020 International Conference on Information Networking (ICOIN)*, pp. 16–19, Barcelona, Spain, 2020.

[10] D. Gupta, S. Rani, S. H. Ahmed, and R. Hussain, "Caching Policies in NDN-IoT Architecture," in *Integration of WSN and IoT for Smart Cities*, pp. 43–64, Springer, Cham, Switzerland, 2020.

[11] M. Ehsanpour, S. Bayat, and A. M. A. Hemmatyar, "An efficient and social-aware distributed in-network caching scheme in named data networks using matching theory," *Computer Networks*, vol. 158, pp. 175–183, 2019.

[12] H. Gao, X. Qin, R. J. D. Barroso, W. Hussain, Y. Xu, and Y. Yin, "Collaborative learning-based industrial IoT API recommendation for software-defined devices: the implicit knowledge discovery perspective," *IEEE Transactions on Emerging Topics in Computational Intelligence*, pp. 1–11, 2020.

[13] S. Lee, I. Yeom, and D. Kim, "T-caching: enhancing feasibility of in-network caching in ICN," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 7, pp. 1486–1498, 2020.

[14] Y. Liu, T. Zhi, H. Xi, W. Quan, and H. Zhang, "A novel cache replacement scheme against cache pollution attack in content-centric networks," in *2019 IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 207–212, Changchun, China, 2019.

[15] Y. Meng, M. A. Naeem, R. Ali, and B. S. Kim, "EHCP: an efficient hybrid content placement strategy in named data network caching," *IEEE Access*, vol. 7, pp. 155601–155611, 2019.

[16] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian et al., "Less pain, most of the gain," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 147–158, 2013.

[17] A. Sharma, A. Venkataramani, and R. K. Sitaraman, "Distributing content simplifies ISP traffic engineering," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 1, pp. 229–242, 2013.

[18] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, K. Drira, and H. Mathkour, "Least fresh first cache replacement policy for NDN-based IoT networks," *Pervasive and Mobile Computing*, vol. 52, pp. 60–70, 2019.

[19] L. Saino, I. Psaras, and G. Pavlou, "Icarus: a caching simulator for information centric networking (ICN)," in *7th International ICST Conference on Simulation Tools and Techniques*, pp. 66–75, Lisbon, Portugal, 2014.

[20] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proceedings of the second edition of the ICN workshop on Information-centric networking*, pp. 55–60, Helsinki, Finland, 2012.

[21] M. A. Hail, M. Amadeo, A. Molinaro, and S. Fischer, "Caching in named data networking for the wireless Internet of Things," in *2015 international conference on recent advances in internet of things (RIoT)*, pp. 1–6, Singapore, 2015.

[22] N. Laoutaris, H. Che, and I. Stavrakakis, "The LCD interconnection of LRU caches and its analysis," *Performance Evaluation*, vol. 63, no. 7, pp. 609–634, 2006.

[23] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache 'less for more' in information-centric networks," in *International Conference on Research in Networking*, pp. 27–40, Springer, Berlin, Germany, 2012.

[24] L. Saino, I. Psaras, and G. Pavlou, "Hash-routing schemes for information centric networking," in *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, pp. 27–32, Hong Kong, China, 2013.

[25] X. Hu, S. Zheng, L. Zhao, G. Cheng, and J. Gong, "Exploration and exploitation of off-path cached content in network coding enabled named data networking," in *2019 IEEE 27th International Conference on Network Protocols (ICNP)*, pp. 1–6, Chicago, IL, USA, 2019.

[26] Q. N. Nguyen, J. Liu, Z. Pan et al., "PPCS: a progressive popularity-aware caching scheme for edge-based cache redundancy avoidance in information-centric networks," *Sensors*, vol. 19, no. 3, p. 694, 2019.

[27] E. Newberry and B. Zhang, “On the power of in-network caching in the Hadoop distributed file system,” in *Proceedings of the 6th ACM Conference on Information-Centric Networking*, pp. 89–99, Macao, China, 2019.

[28] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, “Web caching and Zipf-like distributions: evidence and implications,” in *IEEE INFOCOM’99 Conference on Computer Communications*, pp. 126–134, New York, NY, USA, 1999.

[29] A. Afanasyev, J. Shi, B. Zhang et al., “NFD developer’s guide,” Dept. Comput. Sci., Univ. California, Los Angeles, CA, USA, 2014, Tech. Rep. NDN-0021.

[30] A. Afanasyev, I. Moiseenko, and L. Zhang, “ndnSIM: NDN simulator for NS-3,” Dept. Comput. Sci., Univ. California, Los Angeles, CA, USA, 2012, Tech. Rep. NDN-0005.

[31] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, “ndnSIM 2.0: a new version of the NDN simulator for NS-3,” Dept. Comput. Sci., Univ. California, Los Angeles, CA, USA, 2015, Technical Report NDN-0028.

[32] NS-3 based Named Data Networking (NDN) simulatorFebruary 2020, https://ndnsim.net/current/index.html.

[33] NS-3 Network SimulatorFebruary 2020, https://www.nsnam.org/.

[34] A. M. Aske, “Scheduling functions-as-a-service at the edge,” Ph.D. dissertation, Dept. Computer, Washington State University, Pullman, WA, USA, 2018.

[35] L. Baresi and D. F. Mendonça, “Towards a serverless platform for edge computing,” in *2019 IEEE International Conference on Fog Computing (ICFC)*, pp. 1–10, Prague, Czech Republic, 2019.

[36] A. Yazdinejad, R. M. Parizi, A. Dehghantanha, H. Karimipour, G. Srivastava, and M. Aledhari, “Enabling drones in the Internet of Things with decentralized blockchain-based security,” *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6406–6415, 2021.

[37] Y. Liu, J. Wang, J. Li et al., “Zero-bias deep learning for accurate identification of Internet-of-Things (IoT) devices,” *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2627–2634, 2021.

[38] Y. Wu, “Cloud-edge orchestration for the Internet of Things: architecture and AI-powered data processing,” *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12792–12805, 2021.

[39] X. Li and L. D. Xu, “A review of Internet of Things—resource allocation,” *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 8657–8666, 2021.

[40] R. Hadidi, J. Cao, M. S. Ryoo, and H. Kim, “Toward collaborative inferencing of deep neural networks on Internet-of-Things devices,” *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4950–4960, 2020.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 8902328, 16 pages
https://doi.org/10.1155/2021/8902328

Research Article

Parallel Swarm Intelligent Motion Planning with Energy-Balanced for Multirobot in Obstacle Environment

Shoubao Su,[1,2] **Wei Zhao**,[1,2] **and Chishe Wang**[1]

[1]*Jiangsu Key Laboratory of Data Science and Smart Software, Jinling Institute of Technology, Nanjing 211169, China*
[2]*School of Computer, Jiangsu University of Science and Technology, Zhenjiang 212003, China*

Correspondence should be addressed to Shoubao Su; showbo@jit.edu.cn and Chishe Wang; wangcs@jit.edu.cn

Received 28 June 2021; Revised 9 August 2021; Accepted 12 August 2021; Published 30 August 2021

Academic Editor: Xiaoxian Yang

Multirobot motion planning is always one of the critical techniques in edge intelligent systems, which involve a variety of algorithms, such as map modeling, path search, and trajectory optimization and smoothing. To overcome the slow running speed and imbalance of energy consumption, a swarm intelligence solution based on parallel computing is proposed to plan motion paths for multirobot with many task nodes in a complex scene that have multiple irregularly-shaped obstacles, which objective is to find a smooth trajectory under the constraints of the shortest total distance and the energy-balanced consumption for all robots to travel between nodes. In a practical scenario, the imbalance of task allocation will inevitably lead to some robots stopping on the way. Thus, we firstly model a gridded scene as a weighted MTSP (multitraveling salesman problem) in which the weights are the energies of obstacle constraints and path length. Then, a hybridization of particle swarm and ant colony optimization (GPSO-AC) based on a platform of Compute Unified Device Architecture (CUDA) is presented to find the optimal path for the weighted MTSPs. Next, we improve the A$*$ algorithm to generate a weighted obstacle avoidance path on the gridded map, but there are still many sharp turns on it. Therefore, an improved smooth grid path algorithm is proposed by integrating the dynamic constraints in this paper to optimize the trajectory smoothly, to be more in line with the law of robot motion, which can more realistically simulate the multirobot in a real scene. Finally, experimental comparisons with other methods on the designed platform of GPUs demonstrate the applicability of the proposed algorithm in different scenarios, and our method strikes a good balance between energy consumption and optimality, with significantly faster and better performance than other considered approaches, and the effects of the adjustment coefficient q on the performance of the algorithm are also discussed in the experiments.

1. Introduction

Robot technology is always a remarkably interdisciplinary topic of study, one that can be applied to various engineering practices as well as emerging industrial fields. The edge intelligence, on the other hand, has been known as one of the most difficult research areas to apply multirobot systems (MRSs), in which the key technologies such as path planning, robustness, fault tolerance, and stability cooperation of MRS need to be studied. Motion planning is an extension of path planning, and there exist few differences between them. Generally, path planning is to find the path between starting node and target node in specific scenes by objectives like the shortest distance or the shortest time, while motion planning is to generate interactive trajectories in the situation when robots interact with their surroundings, where there are usually many factors such as velocities, obstacles, and energies that need be considered as constraints [1]. Path planning has always been one of the important challenges, which has been widely used in engineering practices, such as underwater robot detection, UAV cruise, vehicle tracking, warehousing, robot delivery, smart edge systems, and various navigation scenarios [2]. Multirobot path planning (MRPP) is to find the shortest way that allows each robot to follow a safe and effective path to reach a goal from an initial node, optimize the motion path and avoid obstacles, and generate a smooth motion path between nodes that conforms to the objectives.

With decades of year development, there are various studies of motion planner including traditional interpolating and heuristics approaches employed effectively in multirobot navigation over different environmental conditions so far. Even though, it is still an open challenging for multirobot regarding uncertain constraints, especially dynamic adaptive, scalable, and online replanning are still open and challenge problems for practical applications in real-world with large-scale nodes [3–5], which are far from being completely solved, and many practical engineering problems are still tackled today using learning-based heuristics algorithms, for the purpose of robustness, safety, speed, and efficiency. The main advantages of these methods lie in their ease of implementation, effectiveness for specific applications, and feasibility for small-scale and low dimensional workspace. However, some of them fails to complicated time-sequential motion planning, or limited robustness, or computational expensive, or unknown environments, especially the large-scale problem still have to be faced by the exponential growth of the search space and the disaster of dimensionality.

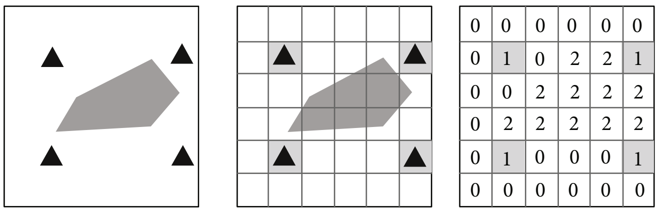
The early graphics processing unit (GPU) did not design a programming architecture for parallel computing. Programmers usually utilize special skills, such as Shader, to realize algorithm parallelism, which result in limitations and complexity of programming. With the development of GPU, especially edge intelligence, in the field of scientific computing, NVIDIA and AMD successively launched GPU-based parallel software and hardware architecture CUDA (Compute Unified Device Architecture) and ROCm (radeon open computing platform). Similarly, ROCm stack and provided a path for the parallel implementation of large-scale evolutionary computing [6]. Modern GPUs use multistream processor (SM) design and single instruction multithreading (SIMT) instruction architecture, which provides a good hardware platform for parallel algorithms. In reality, speed is essential in the applied scenes of MRSs, when the energy consumption of multirobots is not balanced, some robots may be stopped on the way, which will affect the stability and persistence of the whole multirobot system and even affect the completion of the task.

Therefore, this work is focused on motion planning for multirobot in a complex scene that has different surrounding obstacles. The goal is to design fast approaches that enable a real-time calculation of trajectories for multirobot systems which have balanced energies to finish their tasks. After the problem is to be solved modeled as a weighted MTSP, an improved RRT search combining slam construction method is proposed, which can help robots locate and map unknown areas more quickly. Then, we use particle swarm clustering and ant colony optimization methods to find the point-to-point direct path satisfying the basic constraints in the low-dimensional space and then restore the path to the actual scene through the trajectory generation algorithm. At the same time, combined with the characteristics of easy parallelization of swarm intelligence algorithm, aiming at the problems of high accuracy but long execution time of hybrid iterative algorithm, considering the energy consumption balance of multiple robots, by developing a GPU based on CUDA platform to accelerate the running speed of the algorithm, a smooth grid path algorithm combined with the minimum snap algorithm is proposed to make the final path globally optimal, safe, and fast, collision-free, energy-balanced and meet the dynamic constraints.

2. Related Works

2.1. Map Modeling. Multirobot motion planning mainly involves three stages which are building the global map model, generating a search path, and optimizing the motion trajectories. The first stage is to model a map of the given environment that is the input data for algorithms. There are algorithms commonly used to model maps that cover grid method, topology method, line of sight diagram method, tangent diagram method, and Voronoi diagram method [7, 8]. The former two are mostly utilized for global path planning, while the latter three are mostly used for local search. Among them, the grid method is simple and clear, which is recognized as an algorithm with high safety [9]. In the algorithm, the environment can be discretely gridded on a mesh map as fine as possible when the computing resources permit, to reduce the time complexity of finding both obstacles and target-nodes to $O(1)$. As shown in Figure 1, we assume that some irregular polygons are obstacles, and triangles are target nodes on the original map. After offline processing using the grid method, an original environment, the gridded mesh, and the discretized matrix are as shown in Figures 1(a)–1(c).

2.2. Solving Algorithms. Due to the characteristics and complexity of MRPP, it is a difficult and complex NP hard problem, which is essentially an optimization problem with many constraints. The shortest path planning with balanced-energy consumption for each robot is often modeled as the optimization objective. Researchers, so far, use heuristic algorithms to solve the problems, mainly including hybrid iterative method, divide and conquer method, and coevolution strategy. The A$*$ algorithm is one of the most typical graph-based in path search, and it inherits the idea of the Dijkstra and makes it different improved variants [10, 11], which can generate not only the shortest path but also the path more in line with the preference of various models, for example, it can be abstracted a variety of energies into the problems to find the optimal solution [12]. But the disadvantage is its failure to the problems of large-scale nodes, so, deep neural network (DNN) was employed to optimize the evaluation functions $g(n)$ and $H(n)$ of the A$*$ algorithm [13]. The sampling-based algorithms mainly include the probability spectrum method and RRT algorithm. The nearest sampling nodes on the map for obstacle collision detection are connected by trees, which can quickly find a reachable path in a limited time. If there are a large number of obstacles or difficult areas in the environment, the algorithm will run fast, but the algorithm has no perception of the environment. These algorithms are mostly used in local search by combining with Slam (simultaneous



(a) Original environment (b) Gridded mesh (c) Discretized matrix

Figure 1: Grid modeling.

localization and mapping), so that enhance the capabilities of local perception [14].

In recent years, there are numerous researches focused on this issue concerning intelligent methods. In many fields related to artificial intelligence, motion planning for multirobot systems (MRS) is undoubtedly one of the crucial topics that cover all kinds of applications based on swarm optimizers [15–17], including ant colony optimization (ACO) [16] and particle swarm optimization (PSO) [17], because of their effective ways to take the advantages of population information to enhance the overall solution quality and accelerate the convergence speed [15]. For example, ACO clustering with crowding mechanism and GA-based multitask scheduling methods were developed for drones safely flight in a specific airspace [18], two phases heuristic algorithm (TPHA) [19], an improved shuffled frog leaping algorithm (SFLA) [20] was embedded into the trajectory smooth path to determine an optimal subsequent position for each robot, and so on. Neural networks and reinforcement learning (RL) are never absent from routing problems, an optimal navigation strategy with energy-aware coverage designed by using RL for self-reconfigurable robots [21], while deep learning-based warm-starting optimizing motion planner [22] was employed to reduce motion time for robots picking an e-commerce warehouse. To make full use of the advantages of various methods, hybrid intelligent methods have been paid more and more attention recently, and a hybridization planning method is demonstrated [23] by integrating 0-1 optimization and EDA and GA for UAVs and UGVs to cooperative complete the coverages with minimum travel time in urban environments. When PSO planner applied to an intricate unknown environment, Shukla et al. [24] assessed its parameters including move (number of visited nodes), coverage (area explored), energy (distance traveled), and time (time elapsed). By making the best use of limited time, Salah et al. [25] proposed a path-planning approach for multidrones to conduct multiple photographic aerial wildlife surveys.

Some theoretical methodologies are introduced to plan the shortest path in static and dynamic environments. A framework regarding Satisfiability Modulo Theory (SMT) was proposed [26], which models the path as a connected network of mass-spring-damper systems and leverages the properties of Voronoi diagrams to handle complex constraints. To plan a path for the robot on the graph with edge energies, a nonmyopic path planner based on a game-theoretic framework was presented [27] by employing an infinite-horizon Markov decision process. When facing complex and changeable factors of traffic networks, the shortest path with the maximum passing probabilities based on the mobile trajectory can be effectively obtained by calculating the Markov chain and probabilistic symbolic model [28]. To minimize the path length as well as to reduce the number of handovers while sustaining the wireless connectivity of the robots, a multirobot access node association planner was presented [29] in millimeter-wave industrial scenarios. When the motion control for multirobot navigation is used in large-scale dynamic scenarios, a personalized route algorithm was presented by introducing the Polychromatic Sets (PS) for users to obtain real-time route that meet their travel preferences [4], all obstacle features can be integrated into a scalar potential field to make decisions [30], an online adaptive replanning strategy for multiple drones flying in an urban environment with a number of dynamic changes [5], and a reliable routing optimization scheme based on the Manhattan mobility model is presented [31] for UAV real-time planning in a vehicular ad hoc networks (VANETs). While a real-time tracking method was designed [32] that combines A$*$ algorithm with dynamic window to guarantee the ability of obstacle avoidance and path smoothness. Thus, it has been observed that the heuristics are more robust and conduct well in scenarios when compared to other algorithms.

3. Energy-Balanced Motion Trajectory Planning Algorithm

3.1. Problem Description and Modeling. This paper focuses on the problem of multirobot motion planning, the first stage of which can be abstracted as a single-point MTSP problem. Assume that m traveling salesmen (denoted as b_1 , $b_2 \cdots b_m$) will visit n cities (denoted as T_0, $T_1 \cdots T_n$) to sell their goods, and all traveling salesmen start from the same city T_1 and finally meet in city T_1; it is required that there must be only one salesman to visit each city, and the total energy and minimum route scheme of all salesmen are obtained [12]. The objective function is

$$\min Z = \sum_{k=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} p_{ij}^{k}, \tag{1}$$

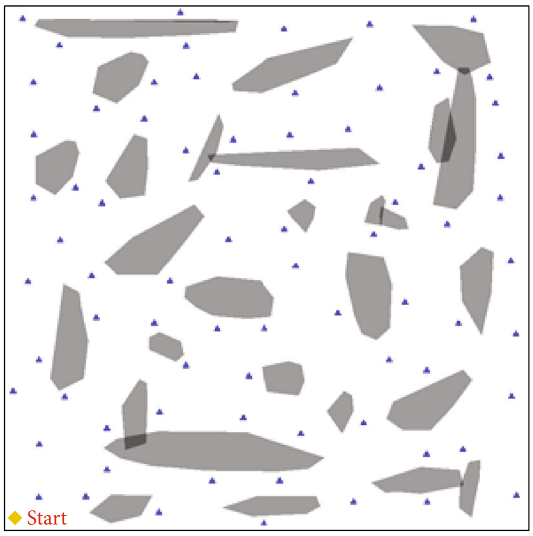


Figure 2: A schematic map of the environment with obstacles.

where p_{ij}^k is denoted as equation (2) that is the direction marking variable between cities i and j of the traveling salesman with serial number k.

$$p_{ij}^k = \left\{ \begin{cases} 1, & \text{traveling salesman } k \text{ travels from city } i \text{ to } j, \\ 2, & \text{else.} \end{cases} \right. \tag{2}$$

Then, we assume that a given environment as shown in Figure 2, in which there are a large number of obstacles (irregular polygons), m target nodes (triangles) $Q_i(i = 1, \cdots,$), a given starting node P (diamond at the bottom left). Now, n mobile robots will visit to m target nodes to perform tasks, each node is only visited by one robot, and all robots must return to the starting node after finishing their tasks.

3.2. Proposed Method. In this section, let us first explain what is energy balance and why we should balance the energy of multirobots. When each robot carries the same energy at the starting node, how to ensure that the total traveling distance of all robots is the shortest, and how to avoid the imbalance of energy consumption of each robot, that is, some robots stop on the way due to insufficient energy, while the others return to the starting node with a large amount of energy remaining. Thus, we present our motion planning scheme in this paper that has two optimization objectives, one is to minimize the total length of the multirobot traveling path, and another is to minimize the deviation of the path length of the multirobot to ensure the energy consumption balance. For these purposes, in our scheme, the environment information is simplified and merged according to the target to be optimized, the quantitative data and the model are established to solve the initial planning path in line with the optimization target, and then the initial planning path is added with obstacles constraints and dynamic constraints so that the planned path can be truly used in the actual environment. The scheme for multirobot motion planning in obstacles is first proposed as Algorithm 1, and its methodology stages of pathfinding, trajectory optimization, and smoothing can be described in details next sections.

3.2.1. Improved A ∗ Algorithm for Path Search. Based on the gridded environment, the A∗ algorithm can be selected for pathfinding with efficiency. The core evaluation function of the A∗ algorithm is as follows:

$$f(n) = g(n) + h(n), \tag{3}$$

where $f(n)$ is the total evaluation value of the current node n, $g(n)$ is the evaluation generation value from the starting node to the current node in the map, and $h(n)$ is the evaluation generation value from the current node to the endpoint. In the algorithm, Manhattan distance between nodes is usually used as the evaluation basis:

$$D(n) = |x_d - x_n| + |y_d - y_n|. \tag{4}$$

The A∗ algorithm also needs two sets to record the intermediate process: an open set and a closed set. The former is used to store the nodes that have not passed under the reachable condition, while the latter is used to store all nodes that have passed or are not reachable. Thus, the method can be presented as Algorithm 2, and the algorithm flowchart is shown in Figure 3. In addition, we also give example graph of fully connected paths with obstacle avoidance generated by the improved A∗ algorithm as shown in Figure 4.

3.2.2. Trajectory Optimization. The path search algorithm does not take into account dynamic constraints such as speed and acceleration when the robot is moving on the way. From the geometric of a view, the planned path is all splicing of line segments, which leads to the connection between the paths by turning points. It is impossible to strictly follow this path in the process, because the motion parameters such as speed and acceleration at the turning point will change unless it is assumed that the speed and acceleration at the turning point are reduced to 0, which will, of course, reduce the motion efficiency of motions. Then, a polynomial-based trajectory optimization method was presented by minimizing jerk (snap) to ensure that the generated trajectory polynomial is continuous and smooth in both speed and acceleration [32–34]. The continuous trajectories between two ends in a two-dimensional plane can be described by the analytic polynomials $x(t)$ and $Y(t)$ about time t, taking the x-axis as an example, the polynomial expression is as equation (5).

$$X(t) = p_n t^n + p_{n-1} t^{n-1} + \cdots + p_1 t + p_0 = \sum_{n=0}^{k} p_n t^n, \tag{5}$$

where k is the order of polynomial, $p_0, p_1 \cdots p_n$ is a polynomial coefficient, the corresponding velocity $V(t)$, acceleration $A(t)$, and acceleration Sanp (t) can be expressed as

```
Input: Structured 2D plan map of the environment
Output: $m$ motion trajectories
1 define the starting node, target node, and obstacles
2 parameter settings, swarm size, the adjustment coefficient $q$
3 for the structured map of the environment
4   discretize for using griding method in section 2.1
5   create a gridded mesh and discretized matrix
6 until all target nodes and obstacles are discretized on the mesh
7 initialize the open set and closed set of nodes by calling A∗ algorithm
8 while the open set is not empty
9   generate all direct path between nodes
10 end while//all nodes are reachable on the fully connected path graph
11 while two nodes is not on a direct path
12   calculate the unit energies for all not direct path
13 Endwhile modelling the problem of MTSP with energies
14 for each robot of $m$, using GPSO-AC (args)
15   generate $m$ route trajectories that met the objectives
16 Endfor
17 for $m$ route trajectories using minimum snap algorithm
18   trajectories optimization to reduce unnecessary turning points
19 Endfor
20 for $m$ route trajectories using improve A∗ and RRT algorithm
21   solving trajectory polynomial and remove all breakpoints
22 Endfor generate m smoothed motion trajectories
```

Algorithm 1: Parallel swarm intelligent motion planning scheme.

```
Input: Open set, closed set for all nodes on a map
Output: Full direct path graph
1 add all nodes with obstacles to the closed set
2 set the starting node to the current node and add it to the open set.
3 for all reachable nodes
4       collect the current node that are not in the closed set
5 Endfor
6 while (the open set is not empty)
7             calculate $f(n)$ of each reachable node, and add it to the open set
8             find a node with the smallest $f(n)$ from the open set
9             set the node as the current node, and add it to the open set
10              set its parent node as the previous current node
11              delete the previous current node from the open set and add it to the closed set
12              for all reachable nodes in the open set
14                    select the node with the smallest $g$(n) in the open combination
                      As the current node, and add it to the open set
15              End for
16 End do
```

Algorithm 2: Improved A∗ algorithm.

equations (6)–(8). Thus, the trajectory $X(t)$ can also be expressed in a matrix form as equation (9).

$$V(t) = X^{(1)}(t) = \sum_{n=0}^{k} p_n . n t^{n-1}, \quad (6)$$

$$A(t) = X^{(2)}(t) = \sum_{n=0}^{k} p_n . n(n-1) t^{n-2}, \quad (7)$$

$$Snap(t) = X^{(4)}(t) = \sum_{n=0}^{k} p_n \cdot \frac{n!}{(n-4)!} t^{n-4}, \quad (8)$$

$$X(t) = \left[1, t, t^2, \cdots, t^n\right] \cdot \left[p_0, p_1, \cdots, p_n\right]^T. \quad (9)$$

Similarly, the motion state description functions such as $X(t)$ and $A(t)$ can be expressed in matrix form. In path

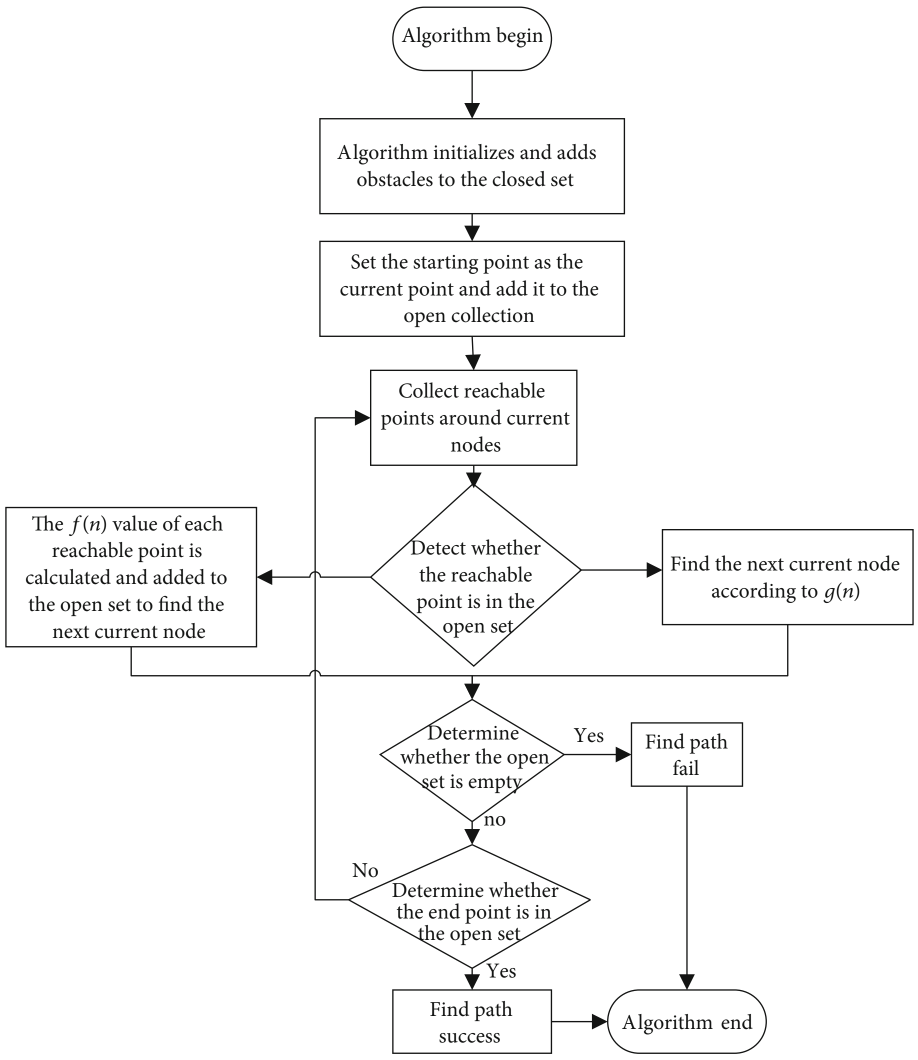

Algorithm begin
Algorithm initializes and adds obstacles to the closed set
Set the starting point as the current point and add it to the open collection
Collect reachable points around current nodes
The $f(n)$ value of each reachable point is calculated and added to the open set to find the next current node
Detect whether the reachable point is in the open set
Find the next current node according to $g(n)$
Determine whether the open set is empty
Yes
Find path fail
no
No
Determine whether the end point is in the open set
Yes
Find path success
Algorithm end

FIGURE 3: Flowchart of an improved A∗ algorithm.

planning, the multisegment trajectory between multiple target nodes can be denoted as equation (10).

$$X(t)=\begin{cases}\left[1,t,t^2,\cdots,t^n\right]\cdot P_1^T & t_0\le t<t_1,\\ \left[1,t,t^2,\cdots,t^n\right]\cdot P_2^T & t_1\le t<t_2,\\ \cdots & \\ \left[1,t,t^2,\cdots,t^n\right]\cdot P_k^T & t_{k-1}\le t<t_k.\end{cases} \tag{10}$$

To determine the trajectory of each path, it is necessary to define the value of each coefficient matrix P, minimize the $\mathrm{Snap}(t)$ function of multistage trajectories to obtain high-order continuous and smooth trajectories, that is, to ensure that the coordinates, velocities, and accelerations at the breakpoint between each trajectory are continuous, to ensure that the actual trajectory of the robot is in line with the dynamic constraints in the real world, and can maintain a steady-state continue to be in a stable state of motion. Considering that $\mathrm{Snap}(t)$ can be negative and square, the objective function of minimizing snap can be established as follows:

$$J(T)=\text{minimize}\left(\text{snap}(T)^2\right)=\int_{T_{i-1}}^{T_i}\left(X^{(4)}(t)\right)^2dt=P^TQP. \tag{11}$$

Among them, the total time of T segment trajectory, $\boldsymbol{P}$ is the parameter matrix of each section, and $\boldsymbol{Q}$ is the weight matrix. After the establishment of the objective function, the constraint conditions are specified, and the initial state values of the positions, velocities, and acceleration at both ends and internal nodes of a trajectory can be specified:

$$\sum_{n=0}^{k}p_nt^n=x_0,\ \sum_{n=0}^{k}p_n.nt^{n-1}=v_0,\ \sum_{n=0}^{k}p_n.n(n-1)t^{n-2}=a_0. \tag{12}$$

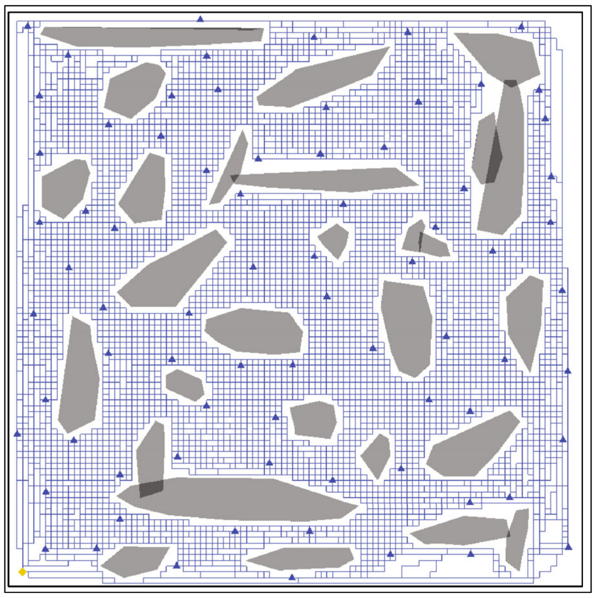


Figure 4: Fully connected path graph with obstacle avoidance.

Then, we can establish equality constraints on matrix $\boldsymbol{P}$:

$$AP_i = C, \tag{13}$$

where $\boldsymbol{A}$ is the coefficient matrix of T, and $\boldsymbol{C}$ is the initial matrix. It can be seen from equations (11)–(13) that what needs to be solved is a quadratic programming problem (QP). There are many algorithms for solving QPs omitted here. Similarly, we use the same method for y-axis to find the polynomial parameters.

3.2.3. Solve the Weighted MTSP Problem with Balanced Energy. The energy-balanced MTSP problem extends the MTSP problem modeled in Section 2.2. Considering the different traffic energies of each path section, we let the unit energy (UE) of connecting nodes i^{th} and j^{th} is v_{ij}, and the energy C_{ij} consumed by the robot traveling between nodes i and j is denoted as equation (14).

$$C_{ij} = d_{ij} \times \text{Energy}_{ij}, \tag{14}$$

where d_{ij} is the distance between nodes i and j. To find the total energy of all robots and the smallest route plan, the objective function equation (1) must be rewritten as equation (15). At the same time, it is required to minimize the variance of the energy of traveling between robots by equation (16).

$$\min Z = \sum_{k=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} v_{ij} p_{ij}^{k}, \tag{15}$$

$$\text{mini} Y = \frac{\sum_{i=1}^{m} \left(C_i - \bar{C}\right)^2}{n}. \tag{16}$$

(1) Establish the Path Weight between Nodes. The goal of this stage is to transform the problem into a weighted MTSP problem in which the weights between nodes are the energies of obstacle constraints and path length. Based on the fully connected path graph with obstacle avoidance (as shown in Figure 4 generated by the improved A∗ algorithm), considering the increase of energy consumption between nodes due to the need to avoid obstacles, the unit energy (UE) consumed by a robot traveling two nodes is approximately defined as eq. (17).

$$\text{Energy} = \frac{|x_1 - x_2| + |y_1 - y_2|}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}, \tag{17}$$

where the upper of eq. (17) actually is the Manhattan distance between two nodes, while the lower is the Euclidean distance directly connected between two nodes. If there are obstacles between two nodes, as shown in Figures 5 and 6 intercepted from the upper right corner of Figure 4, the unit energies (UEs) indicate the difficulties of moving on the direct paths between the two nodes, which are obviously more expensive than those without obstacles. The estimation matrix of UEs does not require additional information about obstacle avoidances. Therefore, when considering the obstructed MTSP problem with energies consumed by robots traveling between nodes, the objective is solved as eq. (15) that is a preparation for the optimal path of energy balance.

(2) Parallel PSO-AC Algorithm for Weighted MTSP with Energy-Balanced. In this section, we propose a hybridization of PSO and ACO algorithm to solve the weighted MTSPs. In our method, the PSO k-means algorithm is used to find clustering central nodes for each robot. Because the right initial clustering center can speed up the convergence and improve the clustering quality, this paper uses a rough classification method to classify the initial nodes by using equations (18) and (19), where the total energy R_i is consumed by a robot traveling from each node to other cities, and the total energy R is consumed by robots traveling the global path sections.

$$Ri = \sum_{j=0}^{n} d_{ij} v_{ij}, i = 0, 1, 2 \cdots, n, \tag{18}$$

$$R = \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij} v_{ij}, i \neq j. \tag{19}$$

Then, sort the distances from the samples to the clustering centers and then traverse the samples in turn. If the traversal condition satisfies equation (20), the nodes that have been traversed are classified into one route, and the clustering centers are updated using equation (21). Assume that m is the number of robots and the number of clusters, and f is a Boolean label variable. If the k^{th} robot visits i^{th} node, it will be 1, and otherwise, it will be 0. In the iterative process, the reclassification process is restricted by the condition of equation (20). The

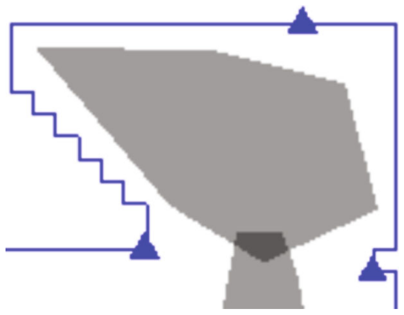


Figure 5: Path section with obstacle avoidance.

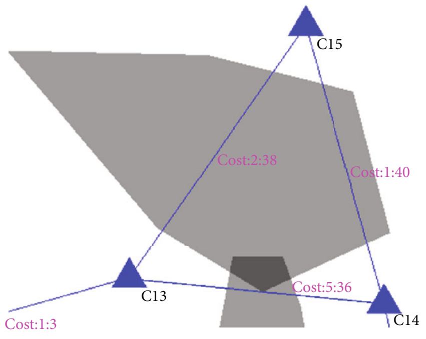


Figure 6: Sketch of UE between nodes.

purpose is to ensure that the total energy for each route is roughly balanced to m robots.

$$\sum_{i=0}^{n} f_{i^k} R_i \leq \frac{R}{m}, \tag{20}$$

$$C_{jp} = \frac{\sum_{i=1}^{n} \omega_{ij} X_{ip}}{\sum_{i=1}^{N} \omega_{ij}}, \text{ where } \omega_{ij} = \begin{cases} 1 & 1 \text{ The sample belongs to route } j, \\ 0 & \text{The sample does not belong to route } j. \end{cases} \tag{21}$$

Next, ACO is utilized to calculate the minimum energy of traveling around a node that is currently classified as a route of nodes. After that, the pheromone and taboo tables of the algorithm are updated, and the fitness of the current particle, fitness, is calculating by equation (22), where C_k is the minimum energy, $S(C_k)$ represents the node dispersion of a route by variance function S, and the smaller the variance, the more balanced approximately the traversal energy consumption for robots. The normalization method, in addition, is used to uniformly quantify their energies and the dispersions as equation (22), where the adjusting coefficient q is [0,1]. When $q = 0$, the problem is a common weighted MTSP. On the other word, the larger q, the greater impact of the energy balance on the convergence of the algorithm.

$$\text{fitness} = \frac{C_k}{\sum_{i=0}^{t} C_k} + q \frac{S(C_k)}{\sum_{i=0}^{t} S(C_k)}. \tag{22}$$

To make full use of the parallel characteristics of the swarm algorithm, we present a hybridization of PSO and ACO to solve this stage of problem. The hybrid scheme is developed on GPU parallel computing platform with CUDA parallelism and multistream processor (SM), so as to enhance the convergence accuracy and executive efficiency of the hybrid swarm intelligence method described as Algorithm 3.

3.2.4. Trajectory Optimization and Smoothing

(1) Constructing the Optimal Grid Path. From equations (14) and (17), it can be seen that the energy consumed by a robot traveling between two nodes is simplified as $C_{ij} = |x_1 - x_2| + |y_1 - y_2|$, which is actually the Manhattan distance of the two nodes on direct path after the obstacle constraint is added. Thus, the total energy of a robot traveling around is the total length of a grid path with obstacle constraints. In summary, the balanced energy in the obstructed MTSP is also the equalized length of the gridded path in path planning. For the purpose of multirobot route planning, after an optimal weighted direct path is obtained, we need to convert the direct path as shown in Figure 7 to the path on which robots travel between nodes when bypassing the obstacles. At this stage, it is necessary to use the A∗ algorithm to restore the gridded path between the target nodes, so far, the initial planned path with obstacle avoidance is completed as shown in Figure 8.

(2) Smoothing Path. Because the generated path based on the gridded environment can only be turned by a multiple of 90°, there are a large number of turning points on the path like sawtooth as shown in Figure 9, which will increase both difficulty and traveling distance of the robot movement. In addition, too many invalid turning points will also hinder the polynomial trajectory optimization process, because turning points are used as constraints for trajectory optimization, and useless turning points must be deleted before trajectory optimization. Therefore, an improved RRT∗ is used to smooth trajectories that have fewer natural turning points to facilitate the follow-up operation of the minimum snap algorithm, so as to reduce useless nodes as much as possible to optimize the subsequent polynomial trajectory. If the raster path is not smoothed, there will be too many useless interference nodes, making the trajectory like S shape generated by the minimum snap algorithm [35], as shown in Figure 10. This paper also extends the A∗ algorithm and proposes a smooth grid path algorithm to reduce useless nodes, and finally, the smoothed trajectory can be obtained as shown in Figure 11.

(3) Solving Trajectory Polynomial. After the smooth processing, all breakpoints remaining on the path can be considered as the minimum effective breakpoints that are the necessary nodes to avoid obstacles, like the target nodes. The motion parameters can be quantified as the equality constraints in the minimum snap algorithm, and finally, used to solve the polynomial coefficients. Assumes that it takes time for the robot to perform the task at the target node, then, it can be considered that the speed and acceleration of the robot are 0 at this time, and at the same time, it needs to ensure that

```
Input: Data after modeling for each node in the MTSP problem
Output: The result of the MTSP problem
1 initialize swarm size of particles and ants
2 do
3  Parallel_for particle in particle swarm
4     parallel{update position and speed}
5     parallel {//reclassification
6        for cluster center in all cluster results
7           calculate the distance between the sample and the current cluster center
8           Sort sample by distance
9           cluster center selection sample
10       end
     }
11    parallel {//calculate fitness
12       for cluster center in all clustering results
13          initialize the ant colony
14          do
15             Parallel_for ant in an ant colony
16                parallel{search path}
17                parallel{update pheromone}
18             end
19          while stop condition reached
20          calculate optimal path fitness
21       end
     }
22    parallel{update individual optimal}
23    parallel{update group optimal}
24 end
25    while stop condition reached
```

Algorithm 3: GPSO-AC.

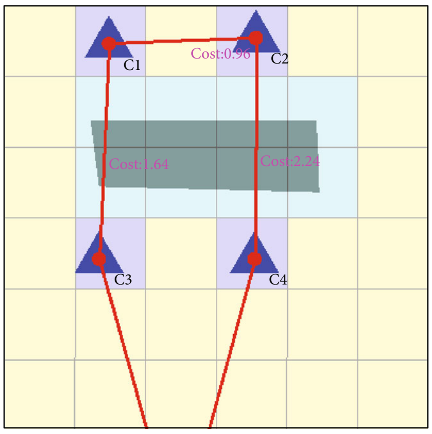


Figure 7: Gridded rough path section.

the robot does not change the motion state at the inflection node, that is, the polynomial of the speed and acceleration at the inflection node is smooth and continuous. The process of constructing equality constraints is given as follows.

Taking x-axis as an example, y-axis similarly, given a grid path L_{AB}, there are n breakpoints on it, and the coordinates of two endpoints are C_A and C_B. The L_{AB} can be divided into $n+1$ segments according to the C_n $(n=1,\cdots,n)$ coordinate at the breakpoint. First, make sure that the static acceleration of the robot at both ends is 0, described as equations (23)–(25), then, ensure that the robot runs to the breakpoint at T_n time, and at the breakpoints, the coordinates, velocities, and accelerations of the front and back trajectories should be consistent at t time.

$$\begin{cases} X(T_0)=C_A \\ X^{(1)}(T_0)=0 \\ X^{(2)}(T_0)=0 \end{cases} \Rightarrow A_{3\times 1}(T_0)P=\begin{bmatrix} C_A \\ 0 \\ 0 \end{bmatrix},$$

$$\begin{cases} X(T_{n+1})=C_B \\ X^{(1)}(T_{n+1})=0 \\ X^{(2)}(T_{n+1})=0 \end{cases} \Longrightarrow A_{3\times 1}(T_{n+1})P=\begin{bmatrix} C_B \\ 0 \\ 0 \end{bmatrix}, \tag{23}$$

$$\begin{cases} X(T_1)=C_1 \\ \cdots \\ X(T_n)=C_n \end{cases} \Longrightarrow \begin{cases} A_{1\times 1}(T_1)P=C_{1,} \\ \cdots \\ A_{1\times 1}(T_n)P=C_n, \end{cases} \tag{24}$$

$$\begin{cases} X_i(T)=X_{i+1}(T) \\ X_i^{(1)}(T)=X_{i+1}^{(1)}(T) \\ X_i^{(2)}(T)=X_{i+1}^{(2)}(T) \end{cases} \Longrightarrow [A_i(T)-A_{i+1}(T)]\begin{bmatrix} P_i \\ P_{i+1} \end{bmatrix}=0, \tag{25}$$

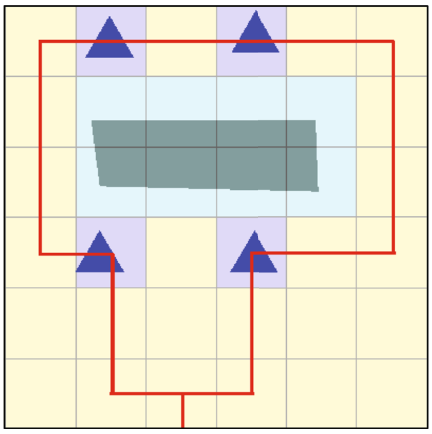


FIGURE 8: Gridded path section with obstacle avoidance.

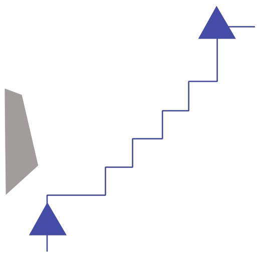


FIGURE 9: Serrated path.

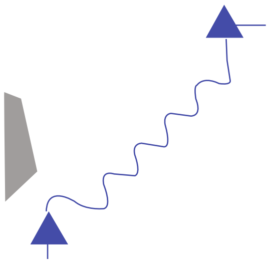


FIGURE 10: Optimized trajectory.

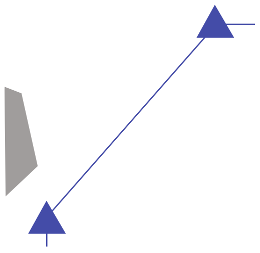


FIGURE 11: Smoothed trajectory.

where A is the weight matrix at time t, and P is the coefficient matrix to be solved. By taking these equality constraints into the QP objective function of equation (11), the P matrix can be obtained, and the polynomial analytic equation of the path planning trajectory can be obtained.

(4) Smooth Grid Path Algorithm. On the basis of the serrated path generated by the A$*$ algorithm, aiming at the shortcoming of the path, a smooth raster path algorithm is proposed in this section to solve the invalid turning point on the serrated path. The algorithm involves two processes of trajectory optimization and smooth described in pseudocode as the following Algorithm 4. If there is no obstacle between the two target nodes, the invalid turning point is deleted.

4. Simulation Experiment

4.1. Experimental Platform and Experiment Settings. In order to verify the feasibility of the proposed method, the experimental platform as shown in Figure 12 is developed using C++ on a PC with CPU Intel® Core™ i5-8400, RAM 8 GB, GPU NVIDIA GeForce GTX 1060 under Linux Ubuntu 18.04.5 LTS. This platform provides some interactive functions including algorithm testing, map grid modeling, obstacle setting, MTSP path generation and optimization, motion trajectory planning, and data saving functions. In our platform development, an open-source SDL2.0 (Simple DirectMedia Layer) is used to the graphics layer rendering library which can make full use of GPU acceleration. For the convenience of debugging, we choose the rendering UI components (ImGUI) that support almost all C++ compilers, which can be embedded in the algorithm for debugging at any time. Because the core algorithm of path planning is to solve the weighted MTSP problem, this experimental platform also provides simulation support for MTSP algorithm experiments.

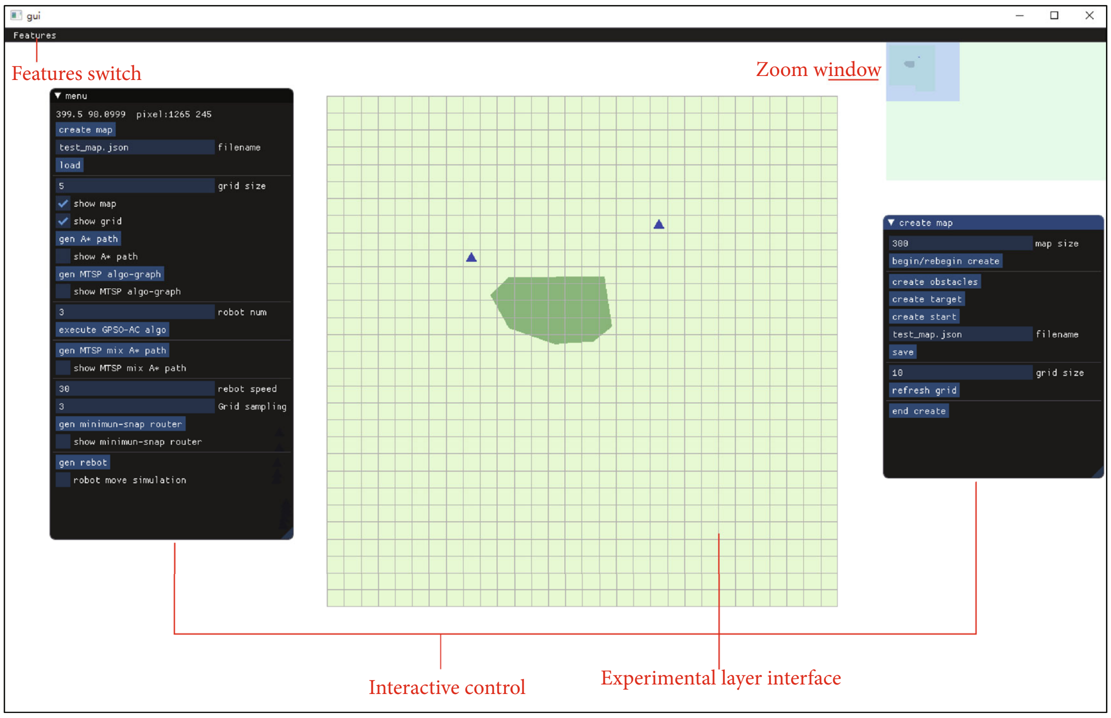
4.2. Benchmark Tests and Comparisons. The multirobot motion planning proposed in this paper is a complete solution, which involves a variety of algorithms, such as map modeling, parallel path search, and trajectory optimization and smoothing. We select several TSPLIB datasets to be used to evaluate the performance of our GPSO-AC in path search by metrics of mean running time (MeanRT) and average acceleration ratio (aveRatio).

First, for the dataset eil51 selected in TSPLIB, we assume that the number of traveling salesman is 3, the weight between nodes is set to 1 regardless of the cost balance, and the problem is a single node departure one. In this experiment, the proposed GPSO-AC algorithm and the PSO-AC algorithm [16] without GPU acceleration are run independently for 30 times, respectively, in different swarm sizes. The statistical results are shown in Table 1. From Table 1, it can be seen that PSO-AC on eil51 run much longer than the former, and its mean running time increases linearly with the swarm sizes, so it is difficult to application with high real-time requirement; in contrast, when the swarm size increases, the algorithm GPSO-AC on eil51 run much faster, and its mean running time increases less and the average acceleration ratio is greater, because the instruction alienation of clustering in GPU is greatly reduced, which enhances the parallel running efficiency.

Then, the proposed GPSO-AC algorithm is tested on 6 datasets by comparing with others PSO-AC [16], TPHA [18], and Kmeans-AC [33], assuming that the number of traveling salesman is set to 3, the swarm size is 64, the maximum iteration is 500, and the learning factors $C_1 = C_2 =$

```
Input: A continuous path on a raster map
Output: Path to remove useless kinks
1 index begin =0
2 index last =0
3 new route list record index (begin)
4 while (begin!=n) do
5     last=n
6     while(end >begin) do
7     if(no collision between connecting line and obstacles (begin, end)) then
8         new route list record index (last)
9         begin = last
10         break
11     end if
12     last = last -1
13     end while
14 end while
```

ALGORITHM 4: Smoothing grid path algorithm.



399.5 90.0999 pixel:1265 245
test_map.json filename
5 grid size
3 robot num
30 rebot speed
3 Grid sampling

300 map size
test_map.json filename
10 grid size

FIGURE 12: Multirobot path planning experimental simulation platform.

TABLE 1: Speedup ratio comparisons of runtimes.

Swarm size	Mean running time (MeanRT) GPU	CPU	Average acceleration ratio (AveRatio)
32	52.19 s	260.96 s	5
64	52.97 s	553.96 s	10.46
128	58.06 s	1145.35 s	19.73
256	59.07 s	1634.82 s	27.68

1.97, $\alpha = 1$, and $\beta = 3$ according to the previous literatures. The four algorithms run independently for 30 times for each problem, and the statistical metrics of their mean running time, worst solution, optimal solution, mean solution, and standard deviation (StD) are shown in Table 2. It can be seen from Table 2 that although the convergence result of PSO-AC is slightly better than the latter two algorithms, the mean running time is long; the algorithms follow the principle of classification before calculation, which runs faster than the first two hybrid algorithms, but the convergence accuracy is poor; the convergence accuracy of GPSO-AC is always

Table 2: Experimental results comparisons of four algorithms.

Problems	Performances	TPHA	*K*means-AC	PSO-AC	GPSO-AC
	MeanRT	0.23 s	0.42 s	461.28 s	47.11 s
	Worest	13184.9	13135.1	10706.1	10706.1
bayg29	Optimum	11791.1	11733	10403.2	10403.2
	Mean	12181	12494.5	10413.3	10413.3
	StD	538.296	375.314	55.31	55.31
	MeanRT	0.73 s	1.4 s	896.03 s	64.52 s
	Worest	9077.08	9042.33	8525.2	8591.67
berlin52	Optimum	8774.05	8809.45	8423.24	8470.35
	Mean	8974.17	8906.13	8493.73	8505.86
	StD	72.99	52.98	25.43	36.31
	MeanRT	1.38 s	2.72 s	1245.94 s	77.47 s
	Worest	1013.24	992.144	855.231	867.028
st70	Optimum	977.15	964.325	830.09	826.925
	Mean	989.44	979.85	841.94	77.47
	StD	8.71	7.24	9.52	9.41
	MeanRT	3.23 s	6.19 s	2117.81 s	158.65 s
	Worest	847.11	826.824	706.65	712.31
eil101	Optimum	780.26	776.452	695.27	681.1
	Mean	802.6	796.72	702.55	690.9
	StD	17.27	14.41	3.05	7.95
	MeanRT	8.47 s	16 s	3669.83 s	237.8 s
	Worest	9072.97	8873.1	7607.78	7022.99
ch150	Optimum	8488.81	8504.53	7347.33	6673.28
	Mean	8810.75	8719.8	7432.4	6833.43
	StD	125.1	93.6	68.43	101.06
	MeanRT	18.11 s	33.63 s	4603.84 s	292.34 s
	Worest	38417.8	37485.6	33820.9	33603.5
kroA200	Optimum	36926.4	36563.1	31940.7	31271.1
	Mean	37609.7	37076.2	32740.3	32723.6
	StD	377.98	279.49	442.56	547.5

better than others within a reasonable time, as two algorithms use clustering to effectively group nodes. TPHA only considers grouping and does not optimize the grouping in the later stage, resulting in poor convergence results.

4.3. Demonstration of Experimental Results. In the experiment, we assume that the proposed methods are used to planning the motion trajectory for 3 robots that visit the nodes to finish tasks in the environment as shown in Figure 13, and the original map is discretely gridded as a mesh scene as shown in Figure 14. Then, the following Figures 15–18 demonstrate the intermediate results of the problem solving. Figure 15 shows a solution after modeling the map information into an MTSP problem which is a rough solution without considering obstacle constraints. While a gridded path with obstacle constraints is generated by the improved A∗ algorithm as shown in Figure 16, from that, there are many useless turning points on the path at this time.

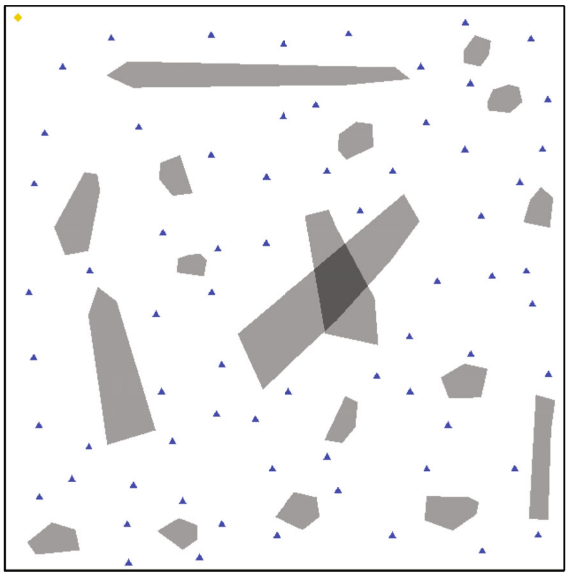


Figure 13: Original map.

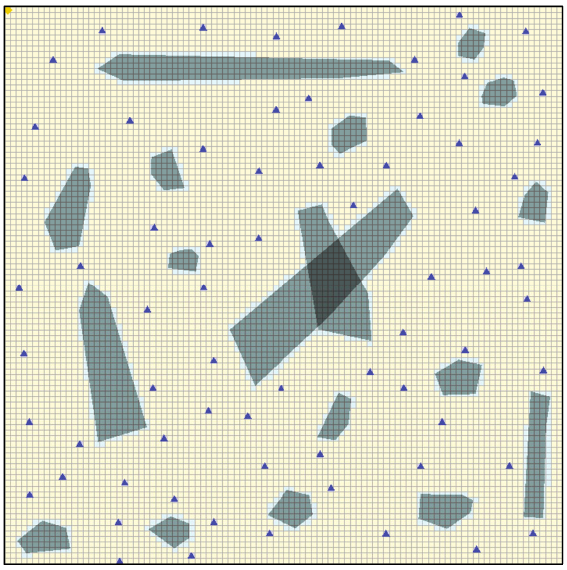


Figure 14: Gridded map.

On the basis of Figure 16, after optimizing the path using the smooth grid path algorithm, a new optimized trajectory is obtained as shown in Figure 17. The path does not have any meaningless turning points on the basis of ensuring that the path does not collide with obstacles. Finally, a path that meets the dynamic constraints is generated by using the minimized snap algorithm. As shown in Figure 18, it can be seen that at the turning point between the two target nodes, the path trajectory becomes smooth and arc-shaped, which conforms to the actual situation of the robot.

4.4. Experimental Result Analysis. In the experiment, we assume that the proposed methods with parameters the same as the above subsections are tested to planning the motion trajectory for 4 robots that visit 75 nodes to finish tasks in the environment with 28 obstacles as shown in

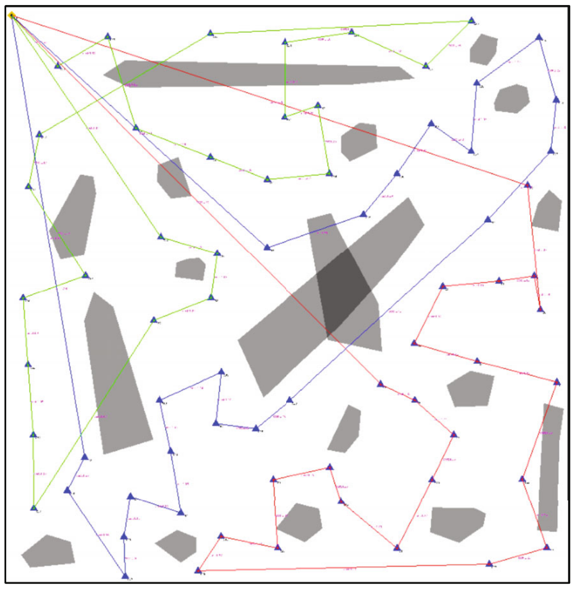


Figure 15: Results of solving MTSP problem algorithm.

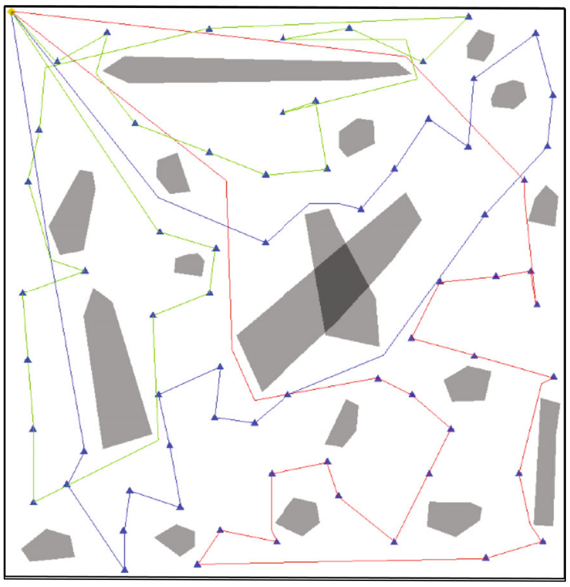


Figure 17: New path after smoothing grid path.

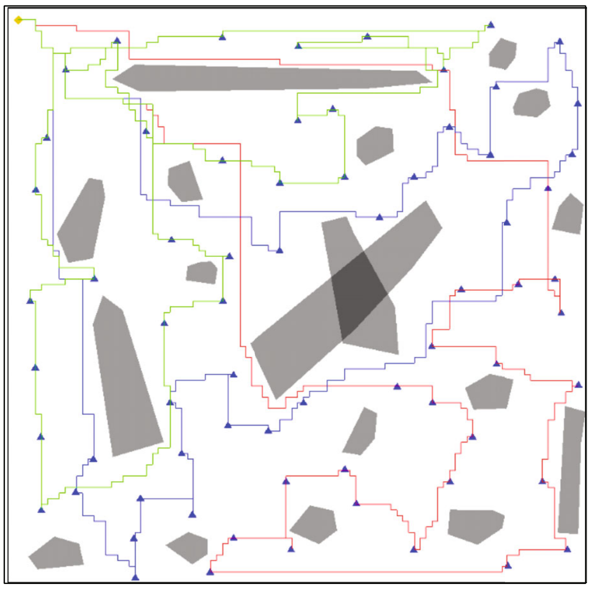


Figure 16: Building the grid path using A∗ algorithm.

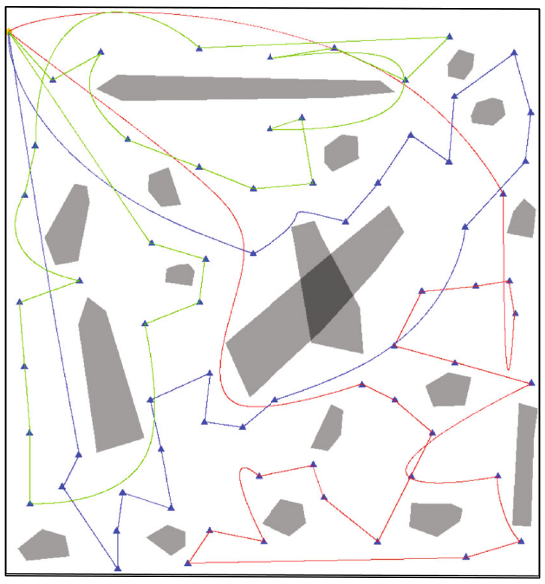


Figure 18: Optimized path with the minimum-snap algorithm.

Figure 2. The two algorithms run independently for 20 times under different adjustment coefficient q, and the statistical metrics of the total distance and standard deviation for each robot are recorded in Table 3 to investigate the adjustment coefficient of influence on generation path before and after optimization, while the relationship between the adjustment coefficient and the total length of the path and the relationship between the adjustment coefficient and the standard deviation of its total length for each robot are plotted in Figures 19 and 20, respectively.

From Table 3 and Figures 19 and 20, we can see that the total length of the path after smoothing is much shorter than that before, which shows that the proposed smooth grid path algorithm is effective and greatly reduces the energy consumption of the robot. Meanwhile, without considering the balance of energy consumption, i.e., setting the adjustment coefficient $q = 0$, the path length deviation of each robot is the largest, at this time, the total path length is the shortest. Then, gradually increasing the value of q, it is found that the standard deviation of the grid path will decrease rapidly whether it is smoothed or not, and then as the value of q continues to increase, the standard deviation tends to a smaller value. According to the proposed algorithm, we find that the energy of path smoothing increases rapidly with the increase of the standard deviation of path length, which demonstrates that the equalization algorithm is effective, and balancing the path length of each robot is the guarantee of balancing energy consumption. In addition, as the adjustment coefficient q increases, the path length shows an upward trend, because at this stage, the algorithm focuses

Table 3: Influence of different adjustment coefficients on generation path.

Adjustment coefficient q	Before and after optimization	Robot 1	Robot 2	Robot 3	Robot 4	Total distance	Standard deviation
$q = 0$	A∗ grid path	1650	2220	2380	1500	7750	428.05
	Smoothed grid path	1403.52	1893.3	1938.27	1273.68	6508.77	337.92
$q = 0.025$	A∗ grid path	2030	1540	2470	1800	7840	394.55
	Smoothed grid path	1709.26	1322.25	2142.84	1521.13	6695.48	350.31
$q = 0.05$	A∗ grid path	1690	2260	2250	1680	7880	329.14
	Smoothed grid path	1404.08	1887.27	1781.56	1442.54	6515.45	241.75
$q = 0.075$	A∗ grid path	1910	2200	1930	1960	8000	134.9
	Smoothed grid path	1608.63	1825.32	1697.47	1619.06	6750.48	99.99
$q = 0.1$	A∗ grid path	1980	2080	2000	1990	8050	45.73
	Smoothed grid path	1672.18	1742.48	1659.95	1667.45	6742.05	38.31
$q = 0.15$	A∗ grid path	2060	2050	2120	2080	8310	30.96
	Smoothed grid path	1728.8	1697.91	1754.52	1825.25	7006.48	54.27
$q = 0.2$	A∗ grid path	2140	2110	2140	2170	8560	24.49
	Smoothed grid path	1854.17	1741.49	1761.51	1909.58	7266.75	79.02
$q = 0.25$	A∗ grid path	2160	2170	2140	2130	8600	18.26
	Smoothed grid path	1855.9	1772.03	1694.56	1851.21	7173.71	76.32
$q = 0.3$	A∗ grid path	2200	2190	2190	2180	8760	8.16
	Smoothed grid path	1899.91	1804.66	1878.38	1895.44	7478.39	44.27
$q = 0.4$	A∗ grid path	2240	2230	2230	2240	8940	5.77
	Smoothed grid path	1887.13	1845.57	1867	1973.62	7573.31	56.16
$q = 0.5$	A∗ grid path	2280	2270	2270	2270	9090	5
	Smoothed grid path	1877.95	1929.97	1998.08	1865.28	7671.27	60.38
$q = 0.6$	A∗ grid path	2230	2230	2230	2240	8930	5
	Smoothed grid path	1876.77	1852.4	1947.23	1856.31	7532.72	44.02
$q = 0.7$	A∗ grid path	2190	2200	2200	2200	8790	5
	Smoothed grid path	1786.8	1847.49	1891.22	1840.25	7365.78	42.82
$q = 0.8$	A∗ grid path	2220	2230	2230	2230	8910	5
	Smoothed grid path	1840.44	1847.1	1788.94	1926.37	7402.85	56.74

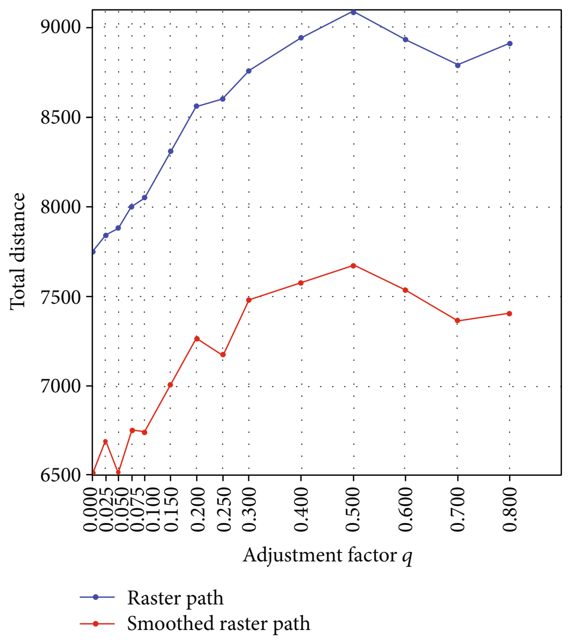


Figure 19: Effects of adjustment coefficient on the path lengths.

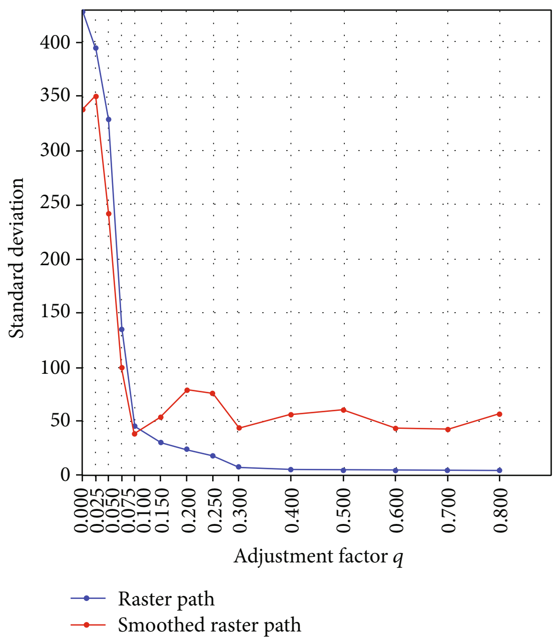


Figure 20: Influence of adjustment coefficient on the deviation of path length.

on the energy balance when calculating the fitness score and ignores the impact of the global optimal energy on the fitness. Therefore, it can be concluded that in the path standard, it is meaningless to continue to increase the value of q after the difference tends to a stable small value. Instead, it will increase the path length and increase the energy consumption of the robot.

5. Conclusion

Multirobot motion path planning is one of the hottest topics in robot technology. It is particularly difficult for multiple mobile robots in a complex obstacle environment where all robots need to execute their tasks, respectively. To speed up the running speed and guarantee the power supply of robots in a limited time, an energy-balanced parallel swarm intelligent optimizer (GPSO-AC) based on GPUs is presented for multirobot to find the shortest path in an obstacle environment where the energy consumption balance in the process of robot movement is modeled as a weighted MTSP problem. The presented method combines with grid constructing and improved the A$*$ searching, smooth gridding, and minimum shaping algorithms to find a feasible oscillation-free, sharp turn-free, and collision-free path. To test the effectiveness of the proposed method, we develop an experimental platform based on GPUs where several scenes with multiple irregularly-shaped obstacles are generated to verify the performance of the presented method. In our experiments, the proposed GPSO algorithm is test on several datasets to evaluate the performances by comparing other similar approaches, and our method on different scenes has found the shortest path in which the multirobot can faster, smoother, and energy-balanced complete the tasks in a reasonable time. In practical applications, the trade-off between the shortest path and the minimum path length should be made. According to the experimental conclusion, the q value should be increased after the path length deviation is taken to a small value within a reasonably acceptable range. However, if each robot is regarded as a dynamic obstacle, it should be further studied in the future.

Data Availability

All experimental data and platforms used to support the findings of this study are available from https://ds3.jit.edu.cn/data/8902328Datasets&Results.rar, or from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The work is partially supported by the Scientific Research Foundations and the Virtual Experimental Class Projects of Jinling Institute of Technology (nos. JIT-rcyj-201505 and D2020005), the National Natural Science Foundation of China (nos. 61375121 and 41801303), the Natural Science Foundation of Jiangsu Province (no. BK200170116), and sponsored by the funds from Jiangsu Provincial Sci-Tech Innovation Team of Swarm Computing and Smart Software led by Prof S.B. Su (corresponding author).

References

[1] A. Madridano, A. Al-Kaff, and D. Martín, “Trajectory planning for multi-robot systems: methods and applications,” *Expert Systems with Applications*, vol. 173, no. 1, p. 114660, 2021.

[2] Z. Qadir, F. Ullah, H. S. Munawar, and F. al-Turjman, “Addressing disasters in smart cities through UAVs path planning and 5G communications: a systematic review,” *Computer Communications*, vol. 168, no. 1, pp. 114–135, 2021.

[3] F. Rubio, F. Valero, and C. Llopis-Albert, “A review of mobile robots: concepts, methods, theoretical framework, and applications,” *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, article 172988141983959, 2019.

[4] P. Li, X. Wang, H. Gao, X. Xu, M. Iqbal, and K. Dahal, “A dynamic and scalable user-centric route planning algorithm based on polychromatic sets theory,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, pp. 1–11, 2021.

[5] Y. Wu and K. H. Low, “An adaptive path replanning method for coordinated operations of drone in dynamic urban environments,” *IEEE Systems Journal*, vol. 14, pp. 1–12, 2020.

[6] B. Zarebavani, F. Jafarinejad, M. Hashemi, and S. Salehkaleybar, “cuPC: CUDA-based parallel PC algorithm for causal structure learning on GPU,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 3, pp. 530–542, 2020.

[7] N. Zafar and J. C. Mohant, “Methodology for path planning and optimization of mobile robots: a review,” *Procedia Computer Science*, vol. 133, no. 3, pp. 141–152, 2018.

[8] K. Shah, G. Ballard, A. Schmidt, and M. Schwager, “Multi-drone aerial surveys of penguin colonies in Antarctica,” *Science Robotics*, vol. 5, no. 47, p. eabc3000, 2020.

[9] X. Zhong, J. Tian, H. Hu, and X. Peng, “Hybrid path planning based on safe a algorithm and adaptive window approach for mobile robot in large-scale dynamic environment,” *Journal of Intelligent & Robotic Systems*, vol. 99, no. 1, pp. 65–77, 2020.

[10] A. Candra, M. A. Budiman, and K. Hartanto, *Dijkstra's and A-Star in Finding the Shortest Path: a Tutorial Inter Conf Data Science*, Artificial Intelligence, and Business Analytics (DATABIA), IEEE CS, 2020.

[11] S. M. Bagheri, H. Taghaddos, A. Mousaei, F. Shahnavaz, and U. Hermann, “An A-Star algorithm for semi-optimization of crane location and configuration in modular construction,” *Automation in Construction*, vol. 121, no. 1, p. 103447, 2021.

[12] K. R. Jensen-Nau, T. Hermans, and K. K. Leang, “Near-optimal area-coverage path planning of energy-constrained aerial robots with application in autonomous environmental monitoring,” *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, 2021.

[13] J. Wang, N. Wu, and W. X. Zhao, “Empowering A search algorithms with neural networks for personalized route recommendation,” in *Proceedings of the 25th Inter Conf Knowledge Discovery & Data Mining (ACM SIGKDD)*, pp. 539–547, 2019.

[14] Z. Tian, C. Guo, and Y. Liu, "An improved RRT robot autonomous exploration and SLAM construction method," in *2020 5th International Conference on Automation, Control and Robotics Engineering (CACRE)*, pp. 612–619, IEEE, 2020.

[15] Y. Wu, "A survey on population-based meta-heuristic algorithms for motion planning of aircraft," *Swarm and Evolutionary Computation*, vol. 62, p. 100844, 2021.

[16] R. Uriol and A. Moran, "Mobile robot path planning in complex environments using ant colony optimization algorithm," in *3rd inter Conf control, automation and robotics (ICCAR)*, pp. 15–21, Nagoya, Japan, 2017.

[17] B. Y. Song, Z. D. Wang, and L. Zou, "An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve," *Applied Soft Computing Journal*, vol. 100, no. 1, p. 106960, 2021.

[18] Y. Wu, K. H. Low, B. Pang, and Q. Tan, "Swarm-based 4D path planning for drone operations in urban environments," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 8, pp. 7464–7479, 2021.

[19] X. L. Xu, H. Yuan, M. Liptrott, and M. Trovati, "Two phase heuristic algorithm for the multiple-travelling salesman problem," *Soft Computing*, vol. 22, no. 19, pp. 6567–6581, 2018.

[20] H. K. Paikray, P. K. Das, and S. Panda, "Improved shuffled frog leaping algorithm for path planning of multiple mobile-robot," in *2nd Int Conf Innovations in Electronics, Signal Processing and Communication (IESC)*, pp. 132–137, Shillong, India, 2019.

[21] A. V. Le, R. Parween, P. T. Kyaw, R. E. Mohan, T. H. Q. Minh, and C. S. C. S. Borusu, "Reinforcement learning-based energy-aware area coverage for reconfigurable hRombo tiling robot," *IEEE Access*, vol. 8, pp. 209750–209761, 2020.

[22] J. Ichnowski, Y. Avigal, V. Satish, and K. Goldberg, "Deep learning can accelerate grasp-optimized motion planning," *Science Robotics*, vol. 5, no. 48, p. eabd7710, 2020.

[23] Y. Wu, S. Wu, and X. Hu, "Cooperative path planning of UAVs & UGVs for a persistent surveillance task in urban environments," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4906–4919, 2021.

[24] S. Shukla, N. Shukla, and V. K. Sachan, "Multi robot path planning parameter analysis based on particle swarm optimization (PSO) in an intricate unknown environments," in *Int Conf Issues and Challenges in Intelligent Computing Techniques (ICICT)*, pp. 1–10, Ghaziabad, India, 2019.

[25] N. Salah, M. Hassen, and K. Bouallegue, "A multi-scroll chaotic system for a higher coverage path planning of a mobile robot using flatness controller," *Chaos, Solitons and Fractals*, vol. 118, no. 2, pp. 366–375, 2019.

[26] F. Imeson and S. L. Smith, "An SMT-based approach to motion planning for multiple robots with complex constraints," *IEEE Transactions on Robotics*, vol. 35, no. 3, pp. 669–684, 2019.

[27] A. Muralidharan and Y. Mostofi, "Path planning for minimizing the expected cost until success," *IEEE Transactions on Robotics*, vol. 35, no. 2, pp. 466–481, 2019.

[28] H. Gao, W. Huang, and X. Yang, "Applying probabilistic model checking to path planning in an intelligent transportation system using mobility trajectories and their statistical data," *Intelligent automation and soft computing*, vol. 25, no. 3, pp. 547–559, 2019.

[29] C. Tatino, N. Pappas, and D. Yuan, "Multi-robot association-path planning in millimeter-wave industrial scenarios," *IEEE Networking Letters*, vol. 2, no. 4, pp. 190–194, 2020.

[30] F. Bayat, S. Najafinia, and M. Aliyari, "Mobile robots path planning: electrostatic potential field approach," *Expert Systems with Applications*, vol. 100, no. 6, pp. 68–78, 2018.

[31] H. Gao, C. Liu, Y. Li, and X. Yang, "V2VR: reliable hybrid-network-oriented V2V data transmission and routing considering RSUs and connectivity probability," *IEEE Transactions on Intelligent Transportation Systems (T-ITS)*, vol. 22, no. 6, pp. 3533–3546, 2021.

[32] F. Zhang, T. Wang, Q. Li, and J. Xin, "An iterative optimization approach for multi-robot pattern formation in obstacle environment," *Robotics and Autonomous Systems*, vol. 133, no. 7, p. 103645, 2020.

[33] M. M. De Almeida, R. Moghe, and M. Akella, "Real-time minimum snap trajectory generation for quadcopters: algorithm speed-up through machine learning," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 683–689, IEEE, 2019.

[34] G. Tang, Z. P. Hou, and X. Hu, "UAV polynomial trajectory optimization based on minimizing snap method," *Computer Application Research*, vol. 38, no. 5, pp. 1455–1458, 2021.

[35] M. G. B. Atia, H. El-Hussieny, and O. Salah, "A supervisory-based collaborative obstacle-guided path refinement algorithm for path planning in wide terrains," *IEEE Access*, vol. 8, pp. 214672–214684, 2020.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 6518743, 15 pages
https://doi.org/10.1155/2021/6518743

Research Article

BSVMS: Novel Autonomous Trustworthy Scheme for Video Monitoring

Xuan Wang,[1] **Jingjing Xu**,[1] **Jiaxin Wang**,[1] **Wei Ou**,[1] **Jung Yoon Kim**,[2] **and Wenbao Han**[1]

[1]*School of Computer Science and Cyberspace Security, Hainan University, Haikou, 570228 Hainan, China*
[2]*Graduate School of Game, Gachon University, 1342 Seongnamdaero, Sujeong-gu, Seongnam, 461-701 Gyeonggi-do, Republic of Korea*

Correspondence should be addressed to Wei Ou; ouwei@hainanu.edu.cn and Jung Yoon Kim; kjyoon79@gmail.com

Received 16 June 2021; Accepted 6 August 2021; Published 26 August 2021

Academic Editor: Xiaoxian Yang

With the continuous development and application of monitoring technology, which involves increasingly more sensitive information, the global demand for video monitoring systems has surged. As a result, video monitoring technology has received widespread attention both at home and abroad. Traditional video monitoring systems experience security threats, with differing levels of severity, in terms of attack, storage, transmission, etc., which results in different degrees of damage to users' rights. Therefore, we propose a blockchain-SM-based video monitoring system called BSVMS. For the front-end device invasion risk, internal attack risk, and security storage problem of the monitoring system, we use commercial cryptography algorithms to complete the encryption processing of images through a visual change network in the imaging process, thereby ensuring the security of the video data from the source. To address the problem that the video monitoring application software and data are vulnerable to damage, we use blockchain technologies that are tamper-proof and traceable to build a trustworthy video monitoring system. In the system, no member can query the original monitoring data. To address the security issues in network transmission, we use a commercial cryptography algorithm for multilayer encryption to ensure the security of data during transmission, guarantee the confidentiality of the system, and realize domestic autonomous control. We then conduct tests and security analysis of the encryption and decryption efficiency of the SM4 algorithm used in the system, the blockchain performance, and the overall performance. The experimental results show that in this system environment, the SM4 algorithm encryption and decryption efficiency is better than other algorithms and that the blockchain used meets industry standards.

1. Introduction

With the rapid development of modern science and technology, the construction of network infrastructure is increasingly sound, and the application of network video monitoring technology is gradually increasing. Video monitoring technology is widely used, particularly in sensitive industries. In addition to the rapid development trend, the data security problem of network video monitoring systems is increasingly obvious. In particular, when monitoring technology is widely used, the monitored content may involve state secrets, military intelligence, business secrets, personal information, and other sensitive information. Once leaked, great security risks will occur, giving criminals an opportunity to harm enterprises and families through video monitoring and even threatening national and social security [1]. Therefore, the most important issue at present is to improve the security of network video monitoring systems.

The COVID-19 epidemic has promoted the development of the video monitoring industry. Thermal cameras equipped with black-body collators and facial recognition video analysis solutions are now the main solutions for screening suspected cases in public places such as railway stations, airports, hospitals, and supermarkets. Infrared thermal imaging cameras are widely used in high-temperature screening in public places [2]. The State Council has listed infrared thermal temperature measurement equipment and related products as key materials in the epidemic, which means that the production and transportation of these materials must be guaranteed and given priority. In the early stage of the epidemic, relevant news mainly focused on the latest progress and data briefing so as to meet the public's demand for authoritative information. At this stage, the public is in a state of information hunger. In special circumstances, live video monitoring without commentary, editing, or any processing has become the key to the fight against the epidemic. People's security awareness is increasing daily, and network video monitoring systems have gradually become an important way to maintain personal and social security; therefore, these systems have received extensive attention from all walks of life and have shown a rapid development trend since their inception. Intelligent video monitoring appears in the concept of distributed video monitoring [3, 4], which extends the monitoring mode to the complement of decentralized and centralized video monitoring and expands the scope of monitoring without limitations. The ubiquitous Internet and LAN are utilized to achieve plug-and-play applications within the scope of the entire network [5]. Most video monitoring systems can conduct reliable monitoring 24 hours a day and can realize unmanned monitoring modes. A large amount of electronic equipment can replace personnel and instruments to achieve the purpose of monitoring, which greatly reduces the use of human resources and financial resources and is more efficient [6, 7].

While the advent of the 5G era gives video monitoring a broader space for global development, it also brings challenges to video security technology. The current problems of video monitoring systems are as follows: (1) Front-end equipment has the risk of invasion. Front-end devices have significant advantages such as round-the-clock uninterrupted operations, strong computing power, high access bandwidth, and a large number of devices. Therefore, criminals tend to invade front-end devices. Criminals can break relatively weak passwords by force and steal high-risk sensitive data through relevant system vulnerabilities [8, 9]. (2) Application software and data are vulnerable to damage. This is an era of big data. Video monitoring focuses on application software and data mining, but it also makes application software and data the most direct targets of criminals. People use big data to obtain effective information, but at the same time, data are also vulnerable to attack and illegally exploited by criminals [10–13]. (3) In general, there are security problems in the transmission of video networks. Universal video channels have the best security, followed by VPNs, and public networks and mobile networks have the worst security [14, 15]. At present, more front-end access devices rely on the mobile Internet and public networks for transmission. Although certain security access measures will be taken, it is still difficult to avoid the intrusion of criminals, and the absolute security of the data in transmission cannot be guaranteed [16]. (4) For internal attack risk, when designing network video monitoring systems, security issues caused by internal network attacks are generally rarely considered. Therefore, most video services are in an open network environment, which makes these video services vulnerable to artificial attacks, such as data interception, information theft, data tampering, and data addition and deletion [17]. (5) For the safe storage problem, the video monitoring image data will be stored in the memory of these cameras for a considerable period of time, and traditional cameras will inevitably capture other irrelevant surrounding background information. Hackers can use the conventional cracking mode for illegal intrusion [18–22].

In order to solve the above problems, we propose a new scheme for an autonomous trustworthy video monitoring system. In view of the front-end equipment intrusion risk, internal network attack risk, and data storage security problems of video monitoring, in this paper, we modify the visual sensor directly and use a commercial cryptography algorithm to encrypt the images through the visual transformation network to realize the encryption of the images during the imaging process, which solves the root cause of the problems. In view of the vulnerability of application software and video monitoring data, we adopt blockchain technologies to manage video systems. No member is allowed to directly query the original monitoring data while all participants are allowed to query any monitoring data on the chain. Regarding the security problems during network transmissions, video data processed by key state encryption can be uploaded to the blockchain platform directly, quickly, and effectively. The blockchain network participants can use the SM4 encryption algorithm to generate a public key, which can be uploaded to the blockchain platform. In this way, we can ensure the security of the video data in the entire transmission process.

2. Background

In the development of video monitoring security, the chaotic encryption algorithm has become the mainstream method in the study of image encryption technology since Fridrich first proposed it in 1997. The chaotic encryption algorithm has developed from one-dimensional chaotic encryption to later two-dimensional and three-dimensional chaotic encryption, and even the hyperchaotic system and composite chaotic system have been proposed [23, 24]. Earlier, Microsoft used AudibleMagic's related software to extract the fingerprints of uploaded videos and protect video data by matching them with the AudibleMagic database to enable active blocking of unauthorized video distribution behavior. Kundur et al. proposed a password-based process called PICO (privacy through reversible password hiding), which applies facial recognition algorithms to images and then uses symmetric key encryption to encrypt the identified face areas. Goldstein and Osher designed a watermark-based video integrity verification method that overcomes the inability of traditional verification mechanisms such as digital signatures to distinguish between attacks and routinely modified signatures [25]. However, Kundur et al. and Goldstein and Osher still have not addressed the issues of privacy, confidentiality, and authentication in WVSNs very well.

Rahman et al. proposed a privacy-preserving mechanism that hides sensitive information from video while providing effective monitoring. This scheme is computationally efficient and suitable for real-time video transmission. However, the system is based on a chaotic encryption algorithm, which has the contradiction of achieving accuracy and confidentiality

[26]. Wang and Smith also proposed a watermarking-based framework using watermarking to embed and hide privacy information (related to the authorized person) in video with minimal perceptual changes while embedding a digital signature in the packet header to authenticate the watermarked video to address the privacy and authenticity issues in video monitoring systems [27]. Huang et al. proposed that the watermarking technique for WVSNs can be implemented using the principle of distributed source coding, which exploits the duality of data hiding and channel coding. Digital video watermarking techniques have been widely studied and implemented in various application areas [28]. However, when the watermarked video is attacked by cropping and interception, the watermark carrier is vulnerable to damage, and video security cannot be guaranteed.

In China, video monitoring security technology started late but developed rapidly. As early as the development of fingerprint technology, Youku began to try to block the spread of illegal monitoring content using fingerprint detection technology. During the COVID-19 outbreak in 2020, since people's work and studies involved video conferences, Shu-li et al. discussed the security risks associated with networking video security systems from video conferencing, video private networks, video content, and video monitoring and proposed a video protection scheme using three layers of protection: confusion scrambling, selective encryption, and selective integrity protection [29, 30].

Jia-liang et al. proposed a security audit solution for UHD video content based on visual image classification technology, target detection and recognition technology, face recognition technology, and super large-scale vector retrieval technology. However, the efficiency of the intelligent recognition and analysis of UHD video content needs to be further improved [31]. Based on the research and analysis of the private network security of public security videos, Yu-Yang et al. proposed protection methods for video private networks such as "one machine one file" docking and protocol whitelist filtering. However, these methods limit the expansion of application scenarios while ensuring data security [32]. In response to various vulnerabilities and flaws of mainstream monitoring products, Long and Haili made improvements in monitoring video transmission and storage security and proposed implementing the authentication problem of the main control side of the monitoring system with an improved RSA algorithm according to the H.265 technology framework. However, the system is not very resistant to attacks and has the problem that the application and video data are vulnerable to corruption [33]. Zheng-qiang et al. proposed a security reinforcement scheme for video monitoring system networking applications. The scheme first implements trusted operation control, access authentication, signaling reinforcement, and video data encryption in front-end devices by updating and upgrading. Second, the scheme deploys the monitoring security management platform in the central management platform to realize the security management of access devices. Finally, the scheme integrates a hardware decryption module in monitoring client computers to realize the decryption of encrypted audio and video. However, the system data are stored in a centralized manner, which requires high hardware equipment [34]. With the rapid development of 5G, Yuan-bing and Shu-li proposed a video monitoring cryptographic application framework for 5G edge computing in conjunction with domestic autonomous and controllable commercial cryptography while 5G is rapidly developing. The security of video surveillance data has been greatly improved [35]. All these results show that video monitoring security technology in China is booming, but there are still some drawbacks [36].

3. Autonomous Trustworthy Video Monitoring Scheme

3.1. Architecture. Our system is based on the SM4 algorithm, which can directly encrypt the visual transformation network in the network monitoring camera and upload the encrypted video data to the blockchain platform. After verification, the client can view the monitoring video data. The system is divided into video processing and blockchain management. The overall architecture of the system is shown in Figure 1.

(i) The source of the monitoring video is the network monitoring camera. The video is captured by the camera. The data processing adopts a camera with a low bit rate, low power consumption, and a high-quality image whose processor core is the mainstream embedded processor, a kernel processing ISP, audio and video codecs, and various types of interfaces. It has strong video graphic processing and intelligent analysis capabilities and can realize various types of video codecs

(ii) When the camera acquires video data, commercial cryptography encryption is performed in the monitoring front-end device firmware through the visual change network. First, we use the SM4 algorithm to generate the key, and then, the original image (x_0) is vectorized into the K-layer convolutional network $N(x) = N_k(N_{k\text{-}1}(\cdots N_1(x)))$. After encryption, the convolutional neural network becomes $N(x; W) = A_k W_K \cdots (A_2 W_2 A_1{}^{-1})(A_1 W_1 A_0{}^{-1}) A_0 x$, which enables the front-end device to complete the encryption of video monitoring directly in the imaging process and upload it to the blockchain platform after encryption is completed

(iii) The blockchain platform distributes the key to the terminal while receiving the data, and the terminal encrypts the locally collected video and transmits it to the counterpart device, generating the key each time it receives the data and distributing it securely to the terminal. The information is encrypted by the public key or symmetric key of the receiver and then uploaded to the chain and shared, and only the receiver with the corresponding private key can decrypt the information. The transmission, storage, and update of video monitoring data are completed on the chain, and when the original data are tampered with, the responsible person can be directly identified

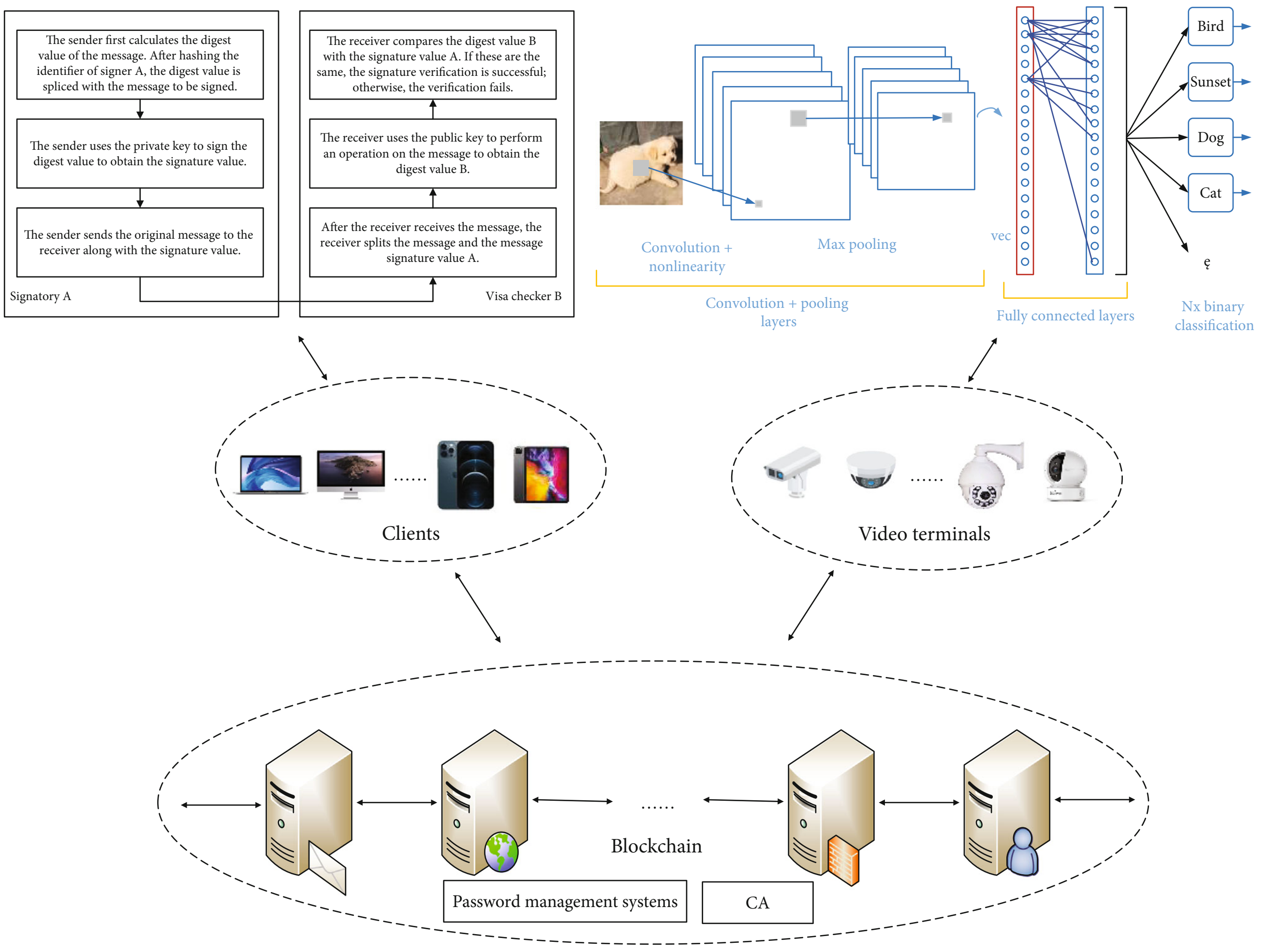

The sender first calculates the digest value of the message. After hashing the identifier of signer A, the digest value is spliced with the message to be signed.
The sender uses the private key to sign the digest value to obtain the signature value.
The sender sends the original message to the receiver along with the signature value.
Signatory A
The receiver compares the digest value B with the signature value A. If these are the same, the signature verification is successful; otherwise, the verification fails.
The receiver uses the public key to perform an operation on the message to obtain the digest value B.
After the receiver receives the message, the receiver splits the message and the message signature value A.
Visa checker B
Convolution + nonlinearity
Max pooling
vec
Convolution + pooling layers
Fully connected layers
Bird
Sunset
Dog
Cat
Nx binary classification
Clients
Video terminals
Blockchain
Password management systems
CA

FIGURE 1: Overall structure.

(iv) The receiver needs to authenticate based on the SM2 algorithm before receiving the video sent by the counterpart device and decrypt and play the video monitoring information after receiving the ciphertext. Since the digital signature only relies on the private key of the sender, which is only owned by the sender personally, this identity authentication method is more secure

3.2. Video Preprocessing. A key net is a convolutional neural network that allows inferences regarding the collected data using specially designed sensors. When processing video monitoring using key nets, visual image classification techniques must be used.

Visual image classification technology conducts visual image classification for ultra-high-definition video content. It is mainly based on deep learning-based visual image classification technology, and the image classification architecture used is a convolutional neural network (CNN). The specific principle is as follows: based on the CNN and 3D-CNN, after the key frame extraction and short video segmentation of the video to be analyzed and identified, feature extraction is performed, the feature vector is formed, and then, the sequence is recognized and analyzed by long short-term memory (LSTM). The above process is repeatedly executed for the UHD video to be analyzed and recognized until all images and short videos are recognized so that video classification can be realized more reasonably.

We describe the SM4 algorithm that encrypts the video monitoring network through the key net. Figure 2 shows the key net.

The SM4 algorithm has a packet length of 128 bits and a key length of 128 bits. Both the encryption algorithm and the key expansion algorithm use a 32-round nonlinear iterative structure to perform encryption operations in words (32 bits), and each iteration is a round of the transformation function F.

3.2.1. Wheel Functions. The key expansion and encryption process both use the wheel function $Z = F(A, B, C, D, E)$, where A, B, C, D, and E are words 4 bytes in length. Figure 3 shows the schematic structure of the wheel function.

The τ-transform is a nonlinear transformation process in the wheel function consisting of four parallel S-boxes, and the data of the S-boxes are all hexadecimal.

The L-transform is a linear transformation process in the wheel function, and for the wheel function of the encryption process, the L-transform is

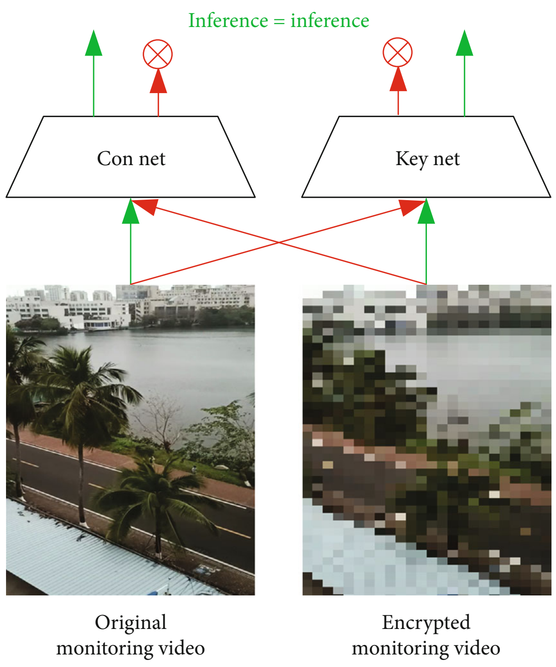


Figure 2: Key net video processing.

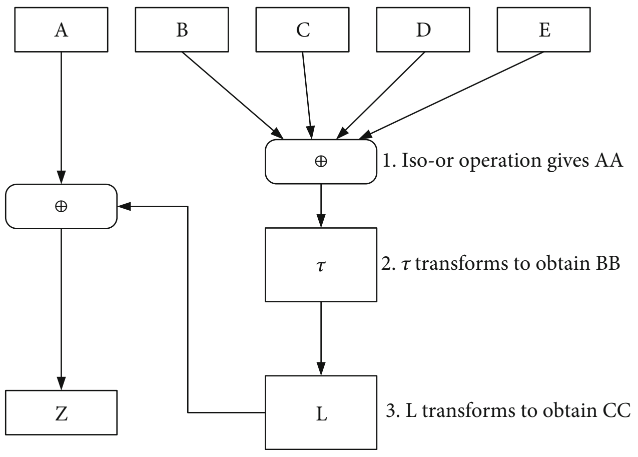

A
B
C
D
E
⊕
1. Iso-or operation gives AA
⊕
τ
2. τ transforms to obtain BB
Z
L
3. L transforms to obtain CC

Figure 3: The schematic structure of the wheel function.

$$L(BB) = BB \oplus (BB{<}{<}{<}2) \oplus (BB{<}{<}{<}10) \oplus (BB{<}{<}{<}18) \oplus (BB{<}{<}{<}24). \quad (1)$$

For the wheel function of the key expansion process, the L-transform is

$$L(BB) = BB \oplus (BB{<}{<}{<}13) \oplus (BB{<}{<}{<}23). \quad (2)$$

The overall encryption function is

$$X_{i+4} = F(X_i, X_{i+1}, X_{i+2}, X_{i+3}, rk_i) = X_i \oplus T(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus rk_i). \quad (3)$$

T is a synthetic substitution, which consists of a τ-transform and an L-transform.

3.2.2. Key Extensions. The values of some parameters are shown in Table 1.

rk_i is the wheel key, and the wheel key is generated from the encryption key.

First,

$$(K_0, K_1, K_2, K_3) = (MK_0 \oplus FK_0, MK_1 \oplus FK_1, MK_2 \oplus FK_2, MK_3 \oplus FK_3). \quad (4)$$

Then, for $i = 0, 1, 2, \ldots, 31$:

$$rk_i = K_{i+4} = K_i \oplus T'(K_{i+1} \oplus K_{i+2} \oplus K_{i+3} \oplus CK_i). \quad (5)$$

Since the system parameters and the fixed parameters are known, the wheel key can be found.

3.2.3. Encryption and Decryption. When encrypting the last round of transformations, the output is

$$(Y_0, Y_1, Y_2, Y_3) = R(X_{32}, X_{33}, X_{34}, X_{35}) = (X_{35}, X_{34}, X_{33}, X_{32}). \quad (6)$$

The decryption is simply done by reversing the order of use of the round keys.

3.3. Blockchain Management. Blockchain management is shown in Figure 4.

(i) *Node Configuration*. When building the blockchain, it is necessary to configure the information of different nodes and CA authentication and to set the consensus algorithm, block size, and other basic parameters in the blockchain configuration file. In the future, when new nodes join Fabric, the number of nodes can be dynamically added through the node configuration, allowing nodes certified by CA to join the blockchain and build a video monitoring data protection ecology.

(ii) *Create Channel*. The channel function in Fabric can be used to isolate the data of different channels in the ledger sharing of different nodes according to their needs. In this system, it is only necessary to build one channel after the nodes are configured.

(iii) *Deployment Contract*. The smart contract in the blockchain needs to be deployed at each node in the channel to take effect, so it is necessary to write video monitoring cryptographic information on the chain into the smart contract and instantiate its deployment.

(iv) *Video Monitoring Data Information Uploading*. When the monitoring end encrypts the generated monitoring data, the monitoring end will call the video monitoring data information uploading function in the smart contract to upload the ciphertext after SM4 encryption to the blockchain and generate the block.

Table 1: Parameters and values.

Parameters	Values
Encryption key	$MK = (MK0, MK1, MK2, MK3)$
System parameters	$FK = (FK0, FK1, FK2, FK3)$
Fixed parameters	$CK = (CK0, CK1, \cdots, CK31)$

When the video platform receives the request for uploading, it will release the information to the blockchain module, and the blockchain workflow is as follows.

(i) The sending node broadcasts the new video monitoring data ciphertext upload information to the entire network

(ii) Receiving nodes examine the received data record information, such as whether the recorded information is legal, and after passing the examination, the data record will be included in a block

(iii) All receiving nodes in the entire network apply a consensus algorithm to the block

(iv) The block is formally incorporated into the blockchain for storage after passing the consensus algorithm process, and all nodes of the entire network indicate acceptance of the block. The method of indicating acceptance considers the random hash value of the block as the latest blockchain hash value, and the manufacture of the new block is extended on the basis of this blockchain

3.4. Identity Authentication. We use the SM2 algorithm and the blockchain platform for identity authentication, and the processes are shown in Figure 5. The steps are as follows.

(i) The sender first calculates the digest value of the message. After hashing the identifier of signer A, the digest value is spliced with the message to be signed

(ii) The sender uses the private key to sign the digest value to obtain the signature value

(iii) The sender sends the original message to the receiver along with the signature value

(iv) After the receiver receives the message, the receiver splits the message and the message signature value A

(v) The receiver uses the public key to perform an operation on the message to obtain the digest value B

(vi) The receiver compares the digest value B with the signature value A. If these are the same, the signature verification is successful; otherwise, the verification fails

Since the digital signature depends only on the private key of the sender, which is owned only by the individual sender, our identity authentication method is relatively secure. Furthermore, the public key of the sender is open for anyone to obtain, so this method is suitable for situations with a large number of verifiers. In addition, digital signature technology can also provide peer entities, data source identity authentication, and data integrity authentication. Therefore, the identity authentication method based on the SM2 algorithm has high security and practical feasibility.

3.5. Encryption. In the autonomous and trustworthy video monitoring system, the blockchain platform distributes the key generated by the SM4 algorithm to the terminal while initiating a monitoring request. The monitoring terminal encrypts the locally collected video and transmits it to the peer device. After receiving the video ciphertext sent by the peer device, the video is decrypted and played.

The key management system distributes conference keys to monitoring terminals, and all monitoring terminals share the same conference key. As a video uploader, the monitoring terminal uses the SM4 algorithm to generate the key, which is encrypted and protected by the conference key and sent to the video receiver. Both receiving and sending terminals share the same conference key, and end-to-end encryption and decryption are performed based on the conference key.

The key management steps are as follows.

(i) The digital certificate and private key are preset when the system is initialized. After issuing the commercial cryptography double certificate (encryption certificate and signature certificate), the certificate and private key are stored

(ii) After turning on the monitoring device, all terminals share the same conference key. When monitoring starts, the conference key is issued. Conference keys are updated, new conference keys are enabled, and old conference keys are deleted according to the policies (e.g., periodically, monitor endpoints off, and new endpoints on)

(iii) The monitoring terminals use their respective query keys for end-to-end encrypted communication. At the video sending end, the monitoring key is generated to encrypt the video on its own and send the video to the receiving end. In addition, we use the monitoring key to encrypt the query key and send the key and ciphertext together to the receiving end. As the video receiver, we use the monitoring key to decrypt the video to obtain the query key from the sender and then use the query key to decrypt and play the video

The overall flow of audio and video processing between the sending end and the receiving end is shown in Figure 6. The figure shows that the scheme adopts three layers of protection: confusion scrambling, selective encryption, and selective integrity protection. The purpose of confusion scrambling is to make the stream more confusing and increase the difficulty of stream analysis and decryption. Selective encryption provides different levels of encryption strength to protect the confidentiality of audio and video data. Selective integrity protection protects the integrity of some audio and video as required to prevent illegal tampering and destruction. The three-layer protection design greatly enhances the security of the system.

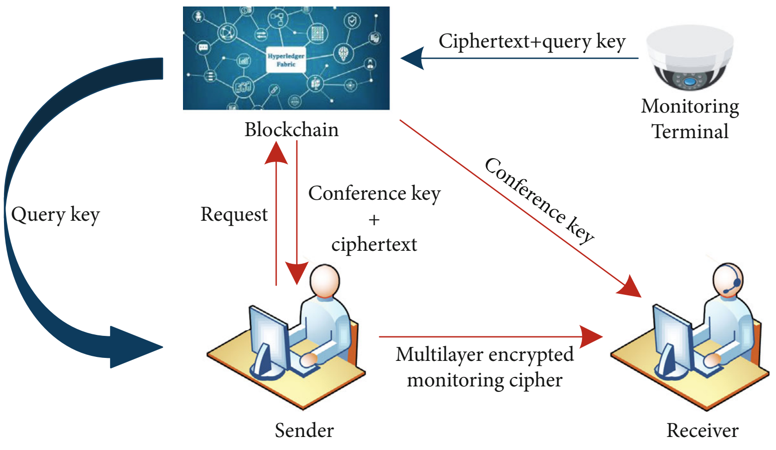


FIGURE 4: Blockchain management.

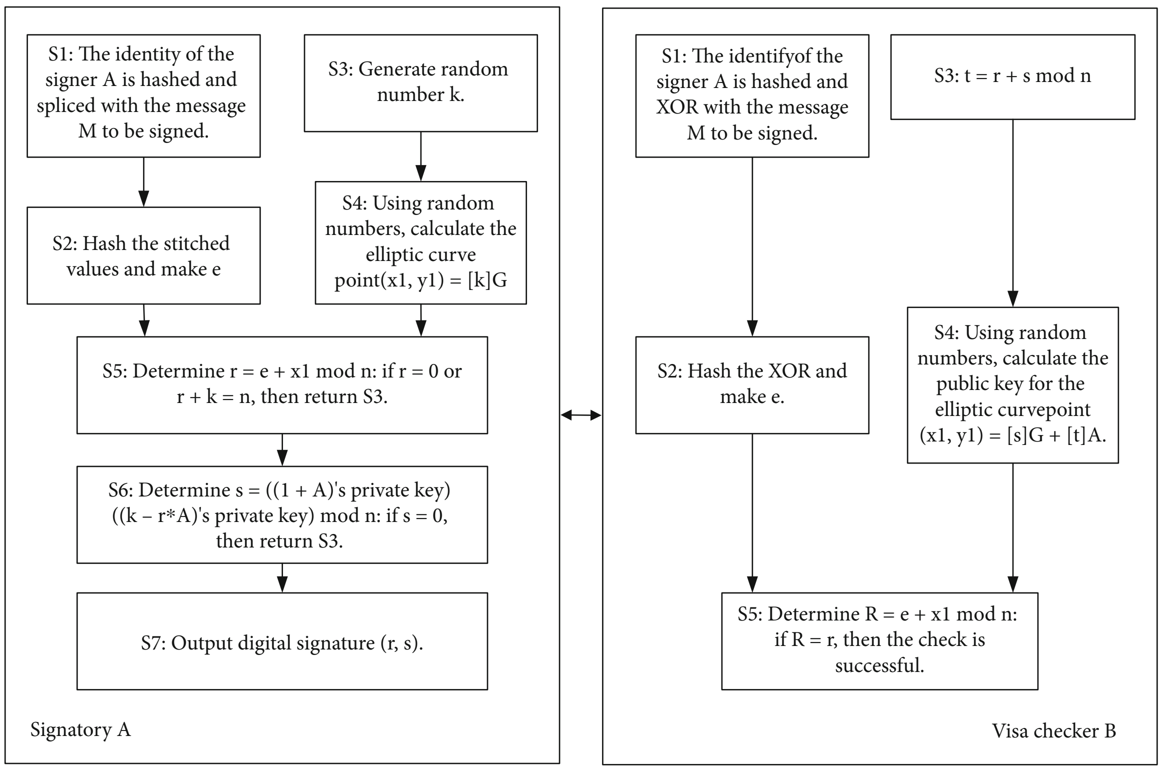

S1: The identity of the signer A is hashed and spliced with the message M to be signed.
S3: Generate random number k.
S2: Hash the stitched values and make e
S4: Using random numbers, calculate the elliptic curve point(x1, y1) = [k]G
S5: Determine r = e + x1 mod n: if r = 0 or r + k = n, then return S3.
S6: Determine s = ((1 + A)'s private key) ((k − r*A)'s private key) mod n: if s = 0, then return S3.
S7: Output digital signature (r, s).
Signatory A
S1: The identifyof the signer A is hashed and XOR with the message M to be signed.
S3: t = r + s mod n
S2: Hash the XOR and make e.
S4: Using random numbers, calculate the public key for the elliptic curvepoint (x1, y1) = [s]G + [t]A.
S5: Determine R = e + x1 mod n: if R = r, then the check is successful.
Visa checker B

FIGURE 5: SM2-based identity authentication process.

The key feature of confusion scrambling is the garbled codebook. The garbled codebook uses a one-for-one mechanism. The platform will generate the garbled codebook for each new conference and securely distribute it to the participating terminals. This book includes four parts: AudioRN (audio frame garbled code), IframeRNCoe (I frame garbled coefficient), PBframeRN (P/B frame garbled code), and ParaRN (parameter set garbled code). The related design and use criteria are as follows.

(i) AudioRN: AudioRN is a random number string with a length not less than the maximum length of a single audio frame. AudioRN is truncated to a string with the same length as the audio frame and then xor with the audio frame data.

(ii) IframeRNCoe: IframeRNCoe consists of a 1-kilobyte random number string A, a 64-byte incremental random number string B, and an incremental offset X. Equation (1) is used to obtain IframeRN, truncate IframeRN to a number string with the same length as the I frame, and then perform an iso-or operation with the I frame data.

(iii) PBframeRN: PBframeRN is a 64-byte random number string. Xor the first 32 bytes of the P/B frame with the first 32 bytes of PBframeRN, and xor the

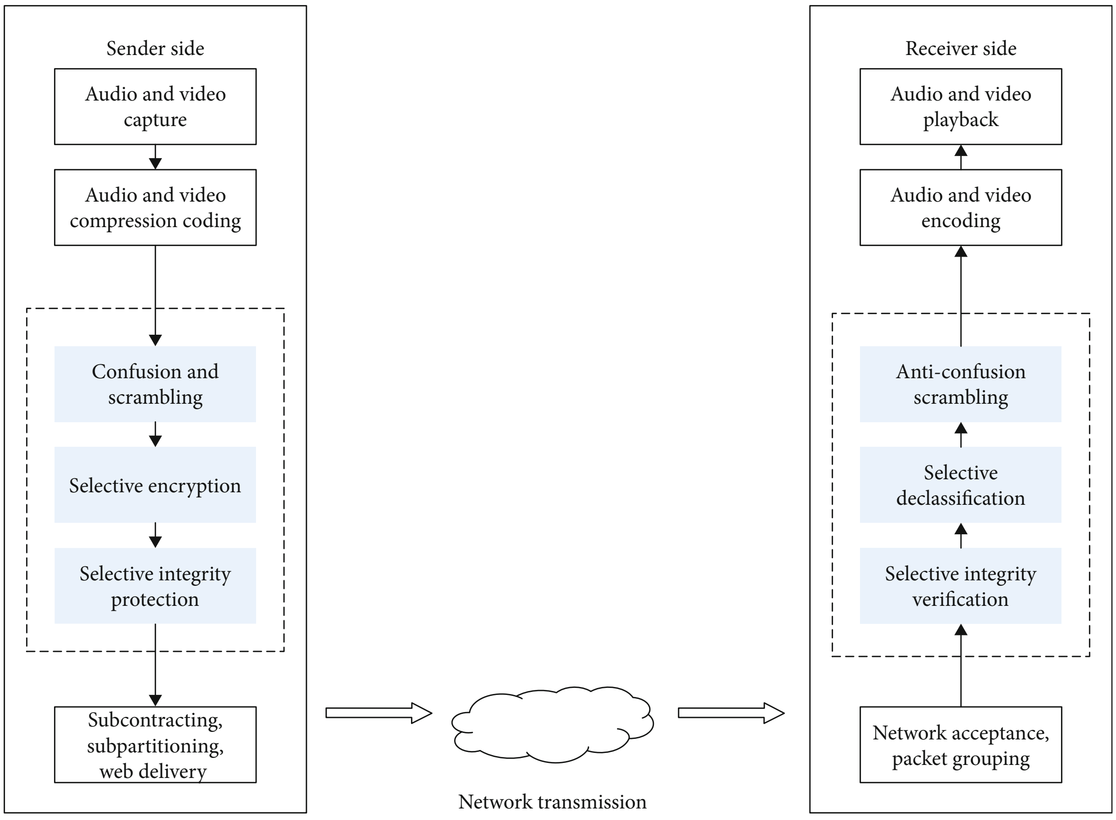

Sender side
Audio and video capture
Audio and video compression coding
Confusion and scrambling
Selective encryption
Selective integrity protection
Subcontracting, subpartitioning, web delivery
Network transmission
Receiver side
Audio and video playback
Audio and video encoding
Anti-confusion scrambling
Selective declassification
Selective integrity verification
Network acceptance, packet grouping

FIGURE 6: End-to-end encryption process.

last 32 bytes of the P/B frame with the last 32 bytes of PBframeRN. If the P/B frame is less than 32 bytes, PBframeRN is truncated into a number string with the same length as the P/B frame, and then, xor is performed with the P/B frame data.

(iv) ParaRN: ParaRN is a random string of 64 bytes. If the length of the parameter set is less than 64 bytes, ParaRN is truncated to the number string equal to the parameter set, and then, xor is performed with the parameter set. If the parameter set length is greater than 64 bytes, xor is performed only on the first 64 bytes of the parameter set.

Due to the large amount of I frame data, if IframeRNCoe is extended, the scrambled code IframeRN with a sufficient length can be obtained. Its implementation is as follows:

$$\text{IframeRN} = A\|(A \oplus (B<<X)\|(A \oplus (B<<(2X\text{-}1)))\|(A \oplus (B<<(3X\text{-}2)))\| \cdots , \tag{7}$$

where $\|$ denotes a string splice, $\oplus$ denotes a bitwise iso-or, and $<<$ denotes a circular left shift.

The selective encryption algorithm of the sender is shown in Figure 7.

The flow of the algorithm in Figure 7 involves three security levels: level I (general), level II (relatively high), and level III (high). The definitions of the three security strengths are shown in Table 2.

At different intensities, different types of data will be fully encrypted or partially encrypted. Part of the encryption process is set out below. Set the data to be processed as M and the length as L, set the grouping parameter as $T(T>4)$, and divide M into T groups of data. The length of T-1 sets of data is $\lceil L/T \rceil$, and the length of the last set of data is $L\text{-}\lceil L/T \rceil * (T\text{-}1)$. Take the first group, the $\lceil T/4 \rceil$ group, the $\lceil T/2 \rceil$ group, and the last group of data for splicing to obtain data M' of length $L\text{-}\lceil L/T \rceil * (T\text{-}4)$ and perform the SM4_OFB encryption operation on M'.

Selective integrity protection is adopted to protect the video I frame data. At level I, there is no processing. At level II, the sender calculates SM3_HMAC on the frame I data and encapsulates the HMAC value into the end of the frame I data. At level III, the sender computes the treetop summary value of frame I and its preceding $k(k>0)$ consecutive frames and encapsulates its value into the end of the frame I data.

The diagram of the treetop summary calculation is shown in Figure 8.

The process on the receiving side is the reverse process of the sending side. We need to ensure that the receiving side shares the same policy and parameter configuration as the sending side. Concrete realization forms can be realized through platform control, negotiation between the sender and the receiver, presetting, and other ways.

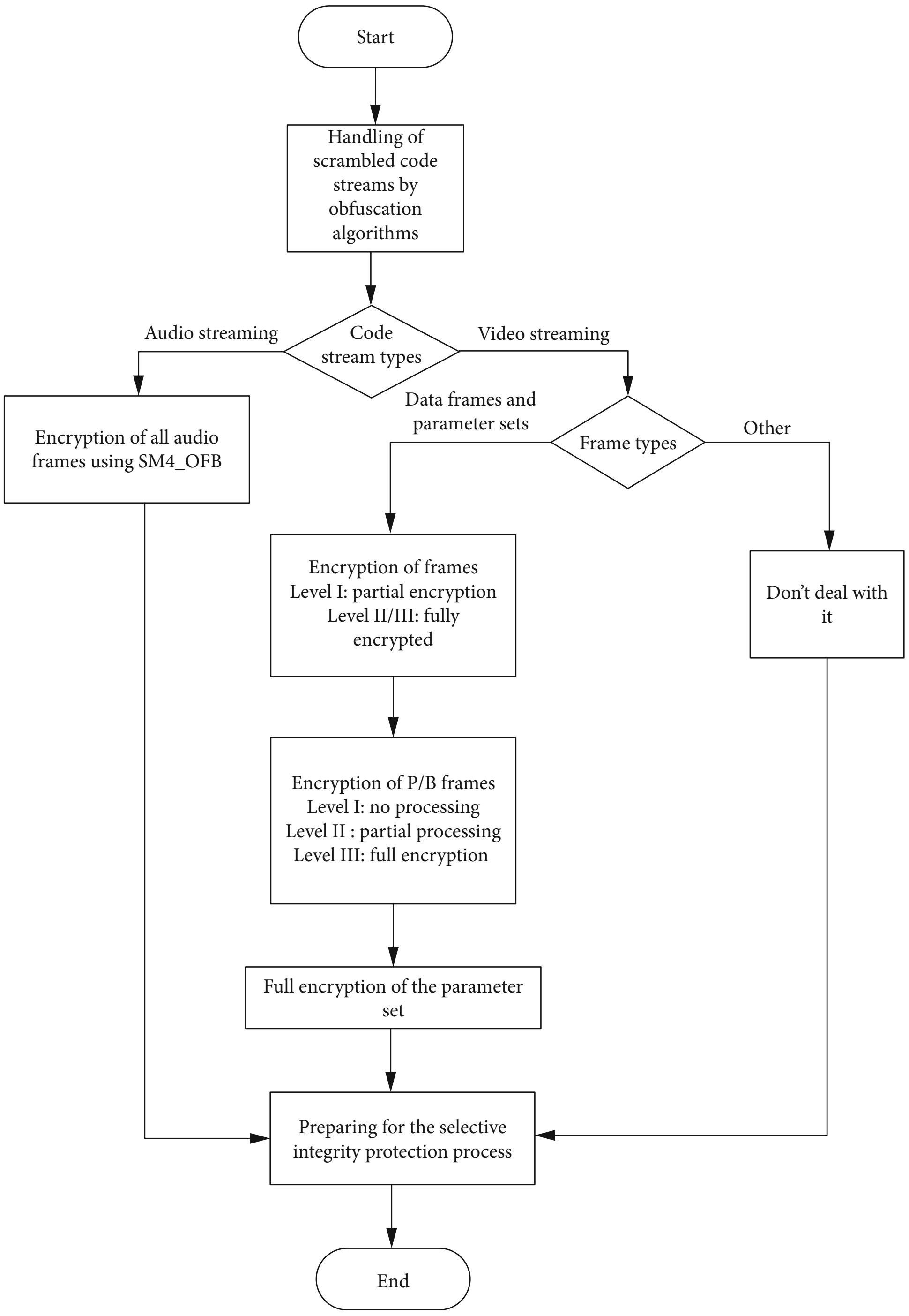

Start
Handling of scrambled code streams by obfuscation algorithms
Code stream types
Audio streaming
Video streaming
Encryption of all audio frames using SM4_OFB
Data frames and parameter sets
Frame types
Other
Encryption of frames Level I: partial encryption Level II/III: fully encrypted
Don't deal with it
Encryption of P/B frames Level I: no processing Level II : partial processing Level III: full encryption
Full encryption of the parameter set
Preparing for the selective integrity protection process
End

Figure 7: Sender side selective encryption algorithm.

4. Experiment and Analysis

4.1. Experiment. In this section, we will test the encryption and decryption time of the SM4 algorithm of the autonomous trustworthy monitoring system and other algorithms in the same environment, the encryption and decryption efficiency of the core part of data encryption, the performance of the blockchain system, and the overall performance of the system. The overall performance of the system includes the operating memory of the system, the memory occupied by the video after processing, and the response time of the system. The main purpose is to compare the overall performance of the system with the performance of the traditional system. Finally, we will analyze the experimental results and draw a conclusion. The process is as follows.

In our experiment, we use the SM4 algorithm as the core algorithm of the encryption design. In order to show that SM4 is the best algorithm of this encryption design, we compare it with AES, DES, SHA1, ECC, and RSA in encryption and decryption time in the same environment. Then, since this

Table 2: Definition of safety strength.

Level	Safety strength	Encryption range	Integrity protection range
Grade I	General	Full encryption of the parameter set, partial encryption of frame I	—
Grade II	Higher	Full encryption of frame I and parameter set, partial encryption of P/B frames	Calculation of HMAC for frame I
Grade III	Highest	Full encryption of I/P/B frames, parameter sets	Compute HMAC based on top-of-tree summary for frame I

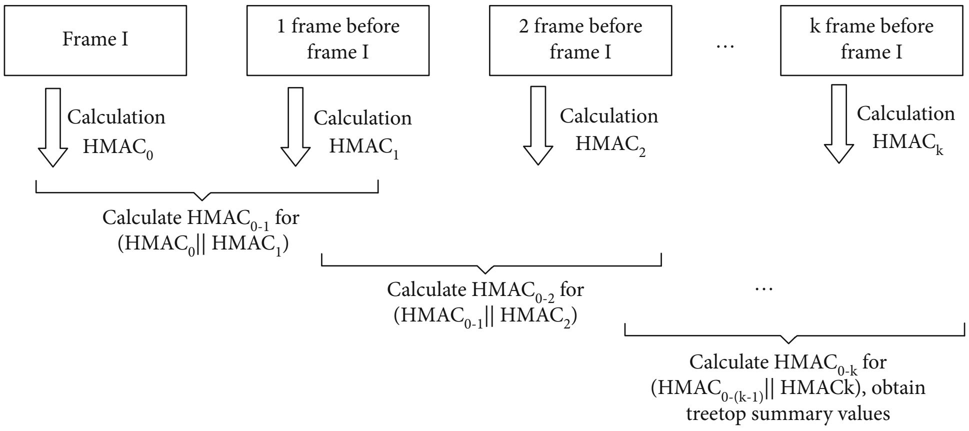

Frame I
1 frame before frame I
2 frame before frame I
…
k frame before frame I
Calculation $HMAC_0$
Calculation $HMAC_1$
Calculation $HMAC_2$
Calculation $HMAC_k$
Calculate $HMAC_{0-1}$ for ($HMAC_0$|| $HMAC_1$)
Calculate $HMAC_{0-2}$ for ($HMAC_{0-1}$|| $HMAC_2$)
…
Calculate $HMAC_{0-k}$ for ($HMAC_{0-(k-1)}$|| HMACk), obtain treetop summary values

Figure 8: Treetop summary algorithm.

system uses selective encryption, we use different encryption methods for different types of data and different security strengths. We divide the encryption strength into three levels, among which the level I security intensity is general and the level III security intensity is the highest. Because the encrypted information obtained from level I security encryption can be easily leaked, if it is used alone, it will seriously affect the security of the system. As a result, this information cannot be used alone. All things considered, we choose to compare the level III security encryption with selective encryption to analyze the execution efficiency of the encryption and decryption process. To assess blockchain performance, we adopt the caliper testing framework. First, we define the test set and then conduct performance tests against the affiliate chain we adopt. In a series of test results, we obtain the information we need to prove that the performance of the adopted blockchain meets the industry blockchain standard. Finally, we compare the overall performance of the system with that of the traditional system so as to show that our system is better than the traditional system.

For a video monitoring system, the primary goal is to ensure that the system has sufficient security, and the second goal is to achieve as high of efficiency as possible. In order to test the autonomous trustworthy video monitoring system, we use Fabric to construct our blockchain system. Fabric adopts a loosely coupled design and modularizes components such as the consensus mechanism and identity verification, which makes it easy to choose the appropriate modules according to the application scenarios during application. In addition, Fabric also adopts container technology to run Chaincode in Docker, thus enabling smart contracts to be written in almost any high-level language. We select MySQL as the offchain storage. The detailed software and hardware development environment is shown in Table 3.

A key characteristic in evaluating whether an encryption algorithm is the best for a given environment is the processing time, including both the encryption time and decryption time. Monitoring systems generally use AES, DES, RSA, SHA1, ECC, and other international cryptographic algorithms for encryption; however, SM4 is a domestic cryptographic algorithm recognized by the National Cryptography Bureau. SM4 has the advantages of a high security level and can increase the difficulty of code stream analysis and cracking; therefore, we choose to use AES, DES, RSA, SHA1, ECC, and SM4 for comparison purposes. In the first instance, we test the encryption and decryption time of the SM4, AES, DES, SHA1, ECC, and RSA algorithms in the same experimental environment.

Since the operation of the system is affected by many factors, each experiment will be conducted 5 times, and the average value is taken as the final result value. Table 4 represents the time comparison of the first 100 I frames of data (approximately 78 MB) of the same monitoring video sample with encryption and decryption.

The table shows that the encryption time is shorter than the decryption time, and the SM4 algorithm has the lowest encryption time and decryption time. Thus, it can be concluded that SM4 is the best choice among these several encryption algorithms.

Second, we test the core part of the encryption design of the system. In order to ensure the authenticity and reliability

Table 3: The experimental environment.

Name	Software and hardware environment
CPU	i7-10875H
GPU	RTX2080SMQ
Memory of a single PC	32 GB
Operating system	Ubuntu 18.04
Blockchain	Fabric 2.2
Relational database	MYSQL 5.5

Table 4: Encryption and decryption times for each algorithm.

	Encryption time (s)	Decryption time (s)
SM4	24.42	24.78
SHA1	25.69	25.96
ECC	25.45	26.10
AES	26.87	27.07
DES	24.56	24.97
RSA	25.74	26.23

of the results, we encrypt and decrypt the different-sized monitoring video data 50 times each using the SM4 algorithm with level III encryption and selective encryption and obtain the relationship between the volume of encryption and decryption data and the running time in the strength of level III encryption and selective encryption. The results are shown in Figure 9.

As shown in Figure 9, selective encryption has a short runtime, and the system performs efficiently. We can conclude that at different intensities, the full encryption or partial encryption of different types of data can not only guarantee the security of the system but also improve the performance of the system.

We then use Caliper to test the performance of our blockchain network. Caliper is a blockchain performance testing framework that tests the performance of a blockchain network with a defined test set and generates the results in a test report. Caliper supports the testing of transaction success rate, throughput (TPS), transaction latency, and resource consumption performance metrics. The data after many experiments are shown in Figure 10.

TPS is the number of transactions a system can process per second. It is calculated as TPS = number of concurrencies /average response time. It is not always the case that a higher TPS is better; instead, this should be evaluated on a case-by-case basis. Public chains have a large number of users and nodes, which require a high concurrent TPS to meet the underlying technical requirements; however, private chains and alliance chains do not need a particularly high TPS and only need to meet the industry demand. As seen in Figure 10, the throughput peaks at 5000 transactions as the number of transactions increases and then starts to decrease. The transaction latency, however, increases as the number of transactions increases, but the magnitude of the increase decreases as the number of transactions increases.

Finally, we test the main performance of the system, including the running memory, the memory occupied by the processed video, and the response time of the system; and compare the main performance of this system with the traditional system.

First, we test the running memory of the system, which refers to the memory when the system is running, the results of which are shown in Figure 11.

Figure 11 shows that, in general, the runtime memory size shows an increasing trend as the number of transactions increases. The running memory of this system is slightly smaller than that of the traditional method as it takes less memory from the user when running; therefore, the user can have more memory to run other programs at the same time.

Figure 12 is the result of the memory occupied by the video after system processing. In order to ensure the accuracy of the experiment, the system and the traditional system process the same monitoring sample.

Figure 12 shows that there is little difference between this system and the traditional system for sample processing, but the sample capacity after this system is applied is still slightly smaller; therefore, we can see that encryption has little effect on the capacity size of the sample itself.

The response times of both systems are then tested. The response time is the time it takes for a computer to respond to a user's input or request. Again using uniform variables, this system is processed on the same monitoring sample as the conventional system, but since there are many determinants of the system response time, we conduct several experiments and average the results after removing the highest and lowest values.

Figure 13 shows that the response times of both this system and the conventional system increase as the size of the monitored samples increases, but the response time of this system is shorter than that of the conventional system for the same sample size.

In summary, this system outperforms the conventional system in terms of operating memory and system response time. The system itself does not have much impact on the memory occupied by the samples.

4.2. Analysis

4.2.1. Performance. The blockchain standards we use are referenced in the Technical Reference Framework for a Trusted Blockchain, Core Requirements and Evaluation Metrics for a Trusted Blockchain, Functional Evaluation Methodology for a Trusted Blockchain, Performance Benchmarking Methodology for a Trusted Blockchain, and Blockchain Security Evaluation Metrics and Testing Methodology for a Trusted Blockchain, which are shown in Table 5.

The network test results of the system are shown in Table 6. The transaction success rate is 100%, the average transaction latency is 0.67 s (the maximum transaction latency is 2.53 s and the minimum transaction latency is 0.06 s), and the throughput is 214.8 TPS.

The experiments indicate that our blockchain performance meets industry standards.

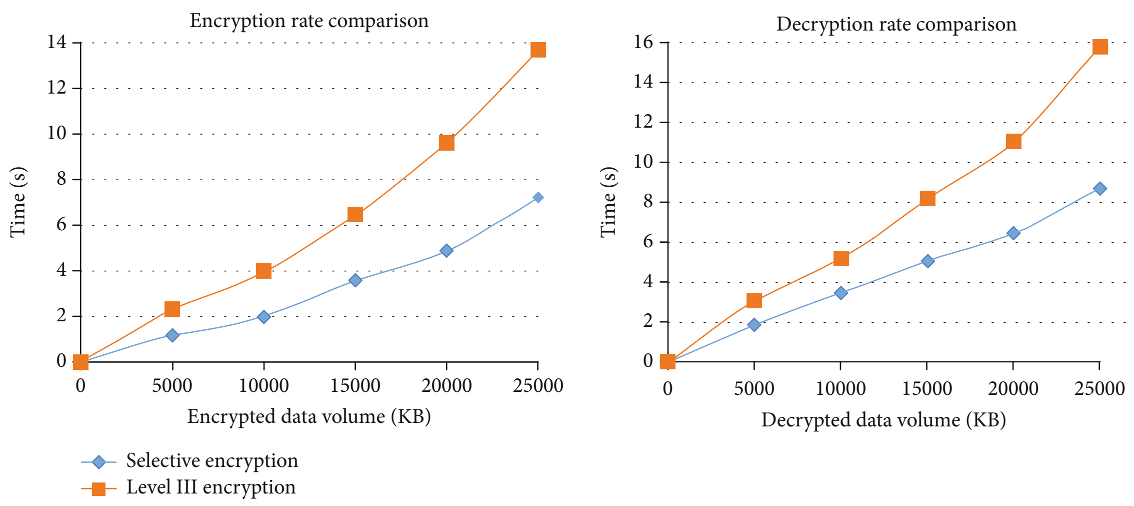


Figure 9: Encryption and decryption rate comparison.

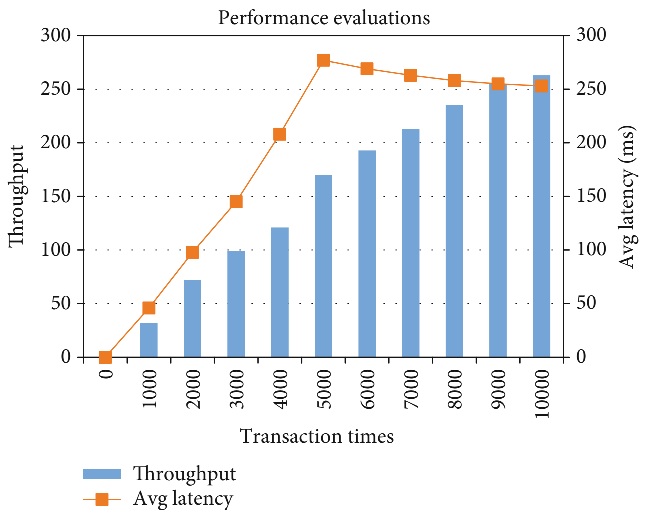


Figure 10: Performance evaluations.

4.2.2. Security

(1) Anonymity. In the video data sharing process, we protect the privacy of our users via data anonymization. Users join social groups and share video information as members of the group. Only the administrator, the users in the local cloud, and the certification center can obtain the users' private information. The blockchain management platform provides authentication functions. In addition, the authentication function can be moved forward to achieve access network authentication, or a secondary authentication system can be constructed to achieve reauthentication. For secondary authentication, according to the SM4 algorithm and the characteristics of the blockchain platform, authentication can reside in different locations such as on-chain and in business data centers to meet the actual authentication requirements in terms of performance and latency. In addition, except for the administrator and the authentication center, it is impossible to identify the users' data information through a certain video record. The ownership of video data cannot be identified by the index number. Since random numbers are added during the storage index generation process, the index of each piece of video data is different. In addition, the identity-related information is stored in a data card, thereby protecting the video data using hardware, and the information in the smart card cannot be read without authorization. For the transmission of monitoring video, similar to the sharing of data information, users use group member IDs to protect privacy during transmission.

(2) Confidentiality. In this system, the strength of the cryptographic algorithm used is high. Commonly used video encryption systems often adopt international cryptographic algorithms such as AES, RSA, and SHA1. This system uses the independently designed SM4 commercial cryptographic algorithm, which is designed to possess a high security level using a multiround cyclic operating mode, and the operating overhead is high, increasing the difficulty of analyzing and cracking the code stream, protecting the confidentiality of video data and preventing illegal tampering and destruction. The encryption design of SM4 greatly enhances the security of the system. Its security strength is no less than that of international algorithms and is independently controllable, making it suitable for deployment in government, finance, energy, and other industries. In the blockchain data sharing platform, the blockchain platform generates a key every time it receives data and distributes the key to the terminal securely. The information is encrypted by the recipient's public key or symmetric key and then chained and shared. Therefore, only the receiver with the corresponding private key can decrypt the information. Each symmetric key is different. Each symmetric key is only responsible for encrypting one piece of video information, which improves the confidentiality of video information. Regarding key distribution, it is necessary not only to protect the confidentiality of the key but also to ensure the integrity, verifiability, correctness, and controllability of the key. In special cases, the key also guarantees the nonrepudiation of both parties in

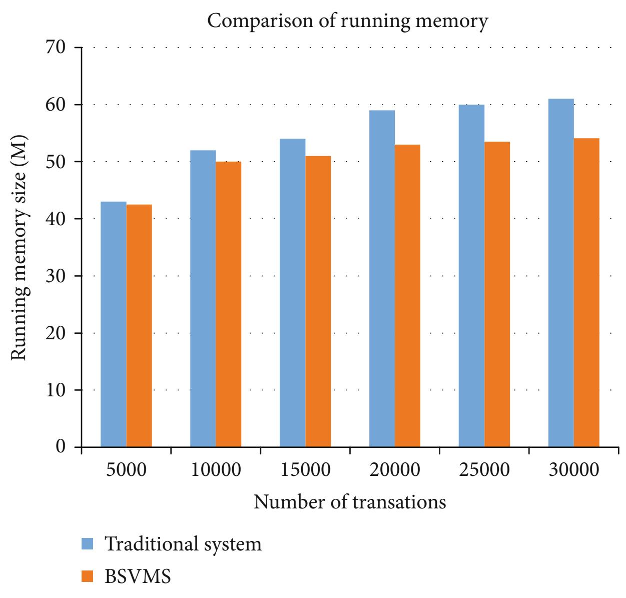


Figure 11: Running memory comparison.

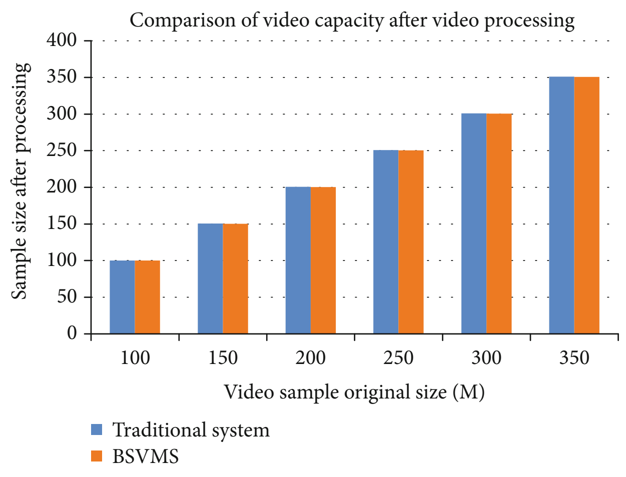


Figure 12: Video capacity comparison after video processing.

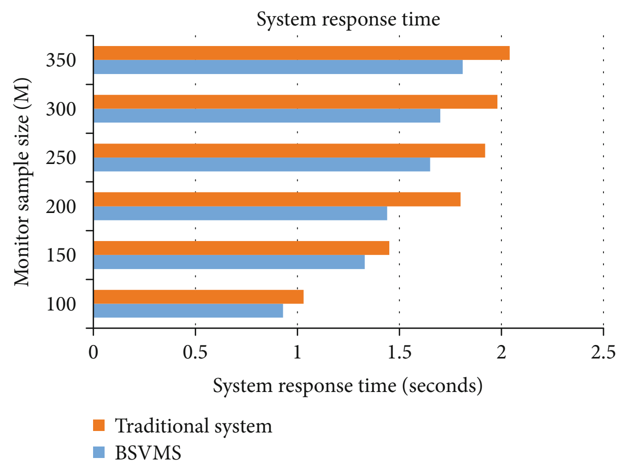


Figure 13: System response time comparison.

communication. According to the exchange method of key information, the key distribution methods are center-based key distribution and authentication-based key distribution. In terms of key storage, the key should be stored in a secret form, and the operating password should be strictly protected to ensure the confidentiality, authentication, and integrity of the key and prevent the key from being leaked and tampered with. In terms of key update and destruction, a new key must be generated after the key has reached its expiration date. The key can be updated using a batch key mode; that is, multiple keys can be injected at a time, and the update can be performed in the form of one key being effective and another key being invalid. Regarding the transmission of monitoring video, the monitoring video is encrypted by the receiver's public key, ensuring that only the corresponding receiver can read the information.

(3) Access Control. The blockchain platform provides authentication functions. In addition, the authentication function can be moved forward to achieve access authentication, or a secondary authentication system can be built to achieve reauthentication. For secondary authentication, according to the SM4 algorithm and the characteristics of the blockchain platform, the authentication can reside in different locations on the chain, in a business data center, etc., to meet the actual authentication requirements in terms of performance and delay. Through blockchain authority management, users can not only view their own monitoring information but also share monitoring information with other authorized entities. Before video information can be used, the identity of both the administrator and the user needs to be verified. Access to different entities is password-controlled and user-defined. Without the user's authorization, no entity can access the user's video monitoring information.

(4) Integrity and Authenticability. The administrator must generate a digital signature for video monitoring after verification. The user verifies the confirmation of the digital signature and the monitoring video and further generates a digital signature, i.e., a double signature. Before data transmission, users will also digitally sign their video information. Without the signer's private key, no entity can forge the entity's digital signature. Since the digital signature is only generated by a specific signer, any information with a digital signature can be authenticated. If a piece of video information is tampered with, the receiver can find it through verification.

(5) Scalable Security Levels. The traditional encryption system only encrypts audio and video without distinguishing the security level. The system fully considers the structural characteristics of audio and video data; differentiates between video data I frames, video data P/B frames, video parameter sets, and audio data frames; and integrates various security techniques such as confusion scrambling, selective encryption, and selective integrity protection to provide different levels of security protection capabilities to meet the needs of a wide range of encrypted video conferencing scenarios.

Table 5: Blockchain industry standards.

Rule requirements	Rule items	Requirements
Test scenarios	Stress test	The number of transactions received per second is basically the same as the number of uploads, and the success rate of uploads is higher than 95%
	Spike test	The number of transactions received per second is significantly higher than the number of uploads, and the success rate of uploads is higher than 75%
	Stability test	Low-load operation with no system crashes
Result items	Performance indicators	Uplink success rate (>95%) TPS (200-300) Average transaction latency (<1 s)

Table 6: Blockchain performance test results.

Name	Requirement
Success rate	100%
Average latency	0.67 s
Throughput (TPS)	214.8 TPS

5. Conclusions

In this paper, based on the commercial cryptography algorithm and blockchain technology, we modify the visual sensor and realize the encryption processing of an image during the imaging process of a camera by encrypting the visual transformation network to solve the video monitoring security problem from the root. We first elaborate on the importance and advantages of video monitoring and analyze the security threats faced in video monitoring systems and related solutions. In the second part, we introduce the current status of domestic and foreign research on video monitoring security. In the third part, we propose an autonomous trustworthy video monitoring system solution based on the commercial cryptography algorithm for the security problems faced by a video monitoring system and design a new visual sensor to replace the traditional lens imaging device to complete the encryption of an image during the imaging process. We research blockchain technologies in depth and design a complete solution for video monitoring to complete identity authentication and encryption design on the chain to realize the interaction between the key network and the blockchain part. Finally, we test the encryption and decryption efficiency of the core part of the data encryption, the performance of the blockchain system, and the overall performance of the system. We then analyze the performance of the blockchain and several aspects of the system such as its anonymity, confidentiality, access control, integrity, and authenticability to ensure that the video monitoring system is secure and trustworthy.

However, the system efficiency is not efficient enough and there are "information silos" between blockchains. The aims of further works are outlined as follows.

(i) To improve the efficiency of the encryption and decryption of video data on the side of mobile terminals using lightweight cryptography algorithms

(ii) To construct a cross-domain trustworthy monitoring system based on blockchain technologies and commercial cryptography algorithms

Data Availability

The monitoring video data used to support the findings of this study have not been made available because the data for this experiment was collected for real and involves numerous private information, such as faces and address information.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] T. Wang, "Network video monitoring system status and security issues," *Network Security Technology and Application*, no. 9, pp. 146-147, 2020.

[2] C. Jiang, "Application of thermal imaging body temperature measurement camera in epidemic prevention and control period," *Electronic production*, no. 8, pp. 31-32, 2020.

[3] H. Gao, X. Qin, R. J. Duran Barroso, W. Hussain, Y. Xu, and Y. Yin, "Collaborative learning-based industrial IoT API recommendation for software-defined devices: the implicit knowledge discovery perspective," *IEEE Transactions on Emerging Topics in Computational Intelligence (TETCI)*, pp. 1–11, 2020.

[4] Y. Huang, H. Xu, H. Gao, and W. Hussain, "SSUR: an approach to optimizing virtual machine allocation strategy based on user requirements for cloud data center," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 2, pp. 670–681, 2021.

[5] K. Huang, X. Chen, Y. Tang, and T. Tan, "An overview of intelligent video monitoring technology," *Journal of Computer Science*, vol. 38, no. 6, pp. 1093–1118, 2015.

[6] L. Zhang, Y. Zou, W. Wang, Z. Jin, Y. Su, and H. Chen, "Resource allocation and trust computing for blockchain-enabled edge computing system," *Computers & Security*, vol. 105, 2021.

[7] L. Zhang, Z. Zhang, W. Wang, Z. Jin, Y. Su, and H. Chen, "Research on a covert communication model realized by using smart contracts in blockchain environment," *IEEE Systems Journal*, pp. 1–12, 2021.

[8] Q. Xie, “Application and development trend of intelligent video monitoring system,” *Shanxi Coal Mine*, vol. 38, no. S1, pp. 31–33, 2019.

[9] W. Gao, “The current situation and development trend of network video monitoring system,” *Information and Computer (Theory Edition)*, vol. 375, no. 5, pp. 178-179, 2017.

[10] S. Li, “Analysis of the key points of the application of new technologies in the security industry in the construction of safe cities,” *Information Communication*, no. 10, pp. 130-131, 2020.

[11] S. Ma, “Smart cities: connotation, dimensions, evaluation and practice,” *Frontiers of Foreign Social Sciences*, no. 11, pp. 64–72+96, 2020.

[12] X. Xu and X. Liu, “A brief description of the organization and implementation plan for the construction of "snow bright" project,” *China Cable TV*, no. 11, pp. 1296–1298, 2020.

[13] H. Gao, L. Kuang, Y. Yin, B. Guo, and K. Dou, “Mining consuming behaviors with temporal evolution for personalized recommendation in mobile marketing apps,” *Mobile Networks and Applications (MONET)*, vol. 25, no. 4, pp. 1233–1248, 2020.

[14] C. Bo, “Research on the status and development trend of network video monitoring system,” *China New Communication*, vol. 20, no. 13, p. 78, 2018.

[15] S. Zhang, “Video monitoring system development status and trends,” *Electronic Paradise*, no. 9, p. 0005, 2018.

[16] Y. Duan, “The current situation and development trend of video monitoring system,” *Technology Wind*, no. 29, p. 87, 2019.

[17] S. Wang, “The application and development direction of video monitoring image detection under video monitoring system,” *China High-Tech*, no. 3, pp. 106–108, 2019.

[18] K. Al-Marashda, “Video security assurance framework based on efficient Joint Cryptography Compression approach,” in *Electronics, Circuits, and Systems (ICECS), IEEE 20th International Conference on*, pp. 76-77, Abu Dhabi, 2013.

[19] M. Meng, “Research on network security assessment technology based on penetration testing,” *Automation and Instrumentation*, no. 10, pp. 182–184, 2018.

[20] X. Wang, “Network video monitoring system status and development,” *Modern Industrial Economy and Informatization*, vol. 9, no. 8, pp. 112-113, 2019.

[21] H. Gao, C. Liu, Y. Li, and X. Yang, “V2VR: reliable hybrid-network-oriented V2V data transmission and routing considering RSUs and connectivity probability,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3533–3546, 2021.

[22] H. Gao, W. Huang, and Y. Duan, “The cloud-edge based dynamic reconfiguration to service workflow for mobile ecommerce environments: a QoS prediction perspective,” *ACM Transactions on Internet Technology*, vol. 21, no. 1, pp. 1–23, 2021.

[23] G. Giusseppe and M. Saverio, “Synchronizing high dimensional chaotic systems via eigenvalue placement with application to cellular neural networks,” *International Journal of Bifurcation and Chaos*, vol. 9, no. 4, pp. 705–711, 1999.

[24] G. Giusseppe and M. Saverio, “Synchronization of high-dimensional chaos generators by observer design,” *International Journal of Bifurcation and Chaos*, vol. 9, no. 6, pp. 1175–1180, 1999.

[25] T. Goldstein and S. Osher, “The split bregman method for L1regularized problems,” *SIAM Journal on Imaging Sciences*, vol. 2, pp. 323–343, 2009.

[26] S. M. Rahman, M. A. Hossain, H. Mouftah, A. El Saddik, and E. Okamoto, “Chaos-cryptography based privacy preservation technique for video surveillance,” *Multimedia Systems*, vol. 18, no. 2, pp. 145–155, 2012.

[27] J. Wang and G. L. Smith, “A cross-layer authentication design for secure video transportation in wireless sensor network,” *International Journal of Security and Networks*, vol. 5, no. 1, pp. 63–76, 2010.

[28] H.-Y. Huang, C.-H. Yang, and W.-H. Hsu, “A video watermarking algorithm based on pseudo 3D DCT,” in *IEEE Symposium on Computational Intelligence for Image Processing*, pp. 76–81, Nashville, TN, USA, 2009.

[29] S. Zhang, Y. Shi, X. Ren, B. Dou, M. Chao, and Z. Zhou, “Research on video conference encryption technology,” *Communications Technology*, vol. 53, no. 10, pp. 2520–2527, 2020.

[30] X. Ma, H. Gao, H. Xu, and M. Bian, “An IoT-based task scheduling optimization scheme considering the deadline and cost-aware scientific workflow for cloud computing,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, 2019.

[31] J. Zhang, B. Zeng, Y. Shen, M. Zhang, and Z. Chen, “Security audit technology of ultra high definition video content,” *Communications Technology*, vol. 53, no. 8, pp. 2049–2054, 2020.

[32] Y. Y. Chun, S. Guo, H. Xing, and P. Yang, “Research and analysis on security of public security video private network,” *Communications Technology*, vol. 53, no. 9, pp. 2292–2296, 2020.

[33] T. Long and W. Haili, “Security risks and technical solutions of video monitoring system,” *Journal of Xi'an College of Arts and Sciences (Natural Science Edition)*, vol. 23, no. 2, pp. 68–71, 2020.

[34] Z. Zhang, Z. Wu, B. Zeng, Y. Shen, and B. Li, “Security enforced solution for video monitoring system networking application,” *Communications Technology*, vol. 52, no. 1, pp. 207–212, 2019.

[35] S. Yuan-bing and Z. Shu-li, “Research of cryptographic application on video monitoring based on 5G edge computing,” *Communication Technology*, vol. 53, no. 5, pp. 1224–1230, 2020.

[36] X. Yang, S. Zhou, and C. Min, “An approach to alleviate the sparsity problem of hybrid collaborative filtering based recommendations: the product-attribute perspective from user reviews,” *Mobile Networks & Applications*, vol. 25, no. 2, pp. 376–390, 2020.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 5750646, 17 pages
https://doi.org/10.1155/2021/5750646

Research Article

An Approach to Modeling and Analyzing Reliability for Microservice-Oriented Cloud Applications

Zheng Liu[1], **Guisheng Fan**[1], **Huiqun Yu**[1], **and Liqiong Chen**[2]

[1]*Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai, China*
[2]*School of Computer Science and Information Engineering, Shanghai Institute of Technology, Shanghai, Shanghai, China*

Correspondence should be addressed to Guisheng Fan; gsfan@ecust.edu.cn

Received 4 May 2021; Accepted 21 July 2021; Published 26 August 2021

Academic Editor: Xiaoxian Yang

Microservice architecture is a cloud-native architectural style, which has attracted extensive attention from the scientific research and industry communities to benefit independent development and deployment. However, due to the complexity of cloud-based platforms, the design of fault-tolerant strategies for microservice-oriented cloud applications becomes challenging. In order to improve the quality of service, it is essential to focus on the microservice with more criticality and maximize the reliability of the entire cloud application. This paper studies the modeling and analysis of service reliability in the cloud environment. Firstly, a formal description language is defined to model microservice, user request, and container accurately. Secondly, the reliability analysis is conducted to measure a critical microservice's fluctuation and vibration attributes within a period, and the related properties of the constructed model are analyzed. Thirdly, a fault-tolerant strategy with redundancy operation has been proposed to optimize cloud application reliability. Finally, the effectiveness of the method is verified by experiments. The simulation results show that the algorithm obtains the maximum benefits and has high performance through several experiments.

1. Introduction

Cloud computing is becoming the mainstream of information technology, which usually involves providing dynamic, scalable, and virtual resources through the Internet. Cloud applications are usually large in scale, have complex structures and contain many distributed components. With the extensive deployment of information systems in critical industries and the continuous expansion of cloud computing applications, the demand for high reliability and cloud computing availability is becoming more urgent. Fault-tolerant technology is an effective way to ensure reliability using the redundancy method to eliminate the influence of fault. A new Fault-Tolerant Elastic Scheduling Algorithm FESTAL for real-time tasks is designed to promote resource utilization [1]. Research on fault-tolerant cloud computing technology can improve the performance so that the task being processed will not terminate abnormally or complete the constraint time functions.

Many of the existing studies on cloud workflow systems focus on reducing the budget and ensuring effectiveness by identifying the critical services in the cloud environment. Determining the potential costs of cloud computing can be complex considering the cloud application reliability. The heuristic search determines the most potential mobile edge computing nodes for each IoT service to meet the reliability requirement based on priorities [2]. There have been attempts to build neural network models for Quality of Service (QoS) prediction. The deep neural networks have achieved exciting results in improving QoS in cloud applications by alleviating overfitting problems. It is necessary to identify the components in cloud applications with high failure risk and reduce the adverse effects [3]. However, the existing methods are limited due to the lack of modeling and analysis of fluctuation reliability time series. Hence, providing highly reliable cloud applications is a challenging and critical research issue.

Microservice architecture offers an alternative solution to realizing software modularization, which has the enormous advantages of solid substitutability, sustainable development, independent scalability, and sustainable delivery. Due to the complexity of applications in the cloud-based system, services are often completed by multiple microservices in complex networks. Microservices bring benefits for implementation flexibility, smooth scaling, improved delivery speed, and flexibility. It overcomes the shortcomings of the high deployment cost with traditional monomer and inflexible response to environmental changes. For Quality of Experience (QoE) requirements, the microservice-oriented application needs long-term stable implementation due to environmental changes [4]. Hence, how to guarantee the reliability of microservice-based cloud applications is considered a challenging problem.

To solve these problems systematically, we propose a reliability sensitivity and vibration measurement method based on disturbance perception to limit the changing reliability amplitude and frequency. The Fault-Tolerant Cloud Application Reliability Enhancement (FTCARE) strategy using a redundancy mechanism for Microservice-Oriented Cloud Application (MOCA) are proposed to improve QoS. However, the redundant nodes are limited, so distributing the constraint resources to optimize the reliability of cloud applications is a problem answered in this paper.

In this paper, our contributions are as follows:

(i) Firstly, the predicated Petri nets are used to model different components of MOCA, such as microservice request, microservice, microservice composition, and container. Then, we identify the critical microservice in complex cloud applications

(ii) Secondly, a time-homogeneous reliability analysis method based on microservice is proposed. The criticality of each microservice is evaluated through request frequency. The reliability sensitivity is calculated by analyzing the impact of disturbance of the microservice on the whole cloud application. The reliability time series of microservice conforms to the 1st-order Markov Chain evolution rule

(iii) Finally, an active adaptation method to reduce the impact of degradation and fluctuation on microservice composition has been proposed to ensure the reliability of cloud applications. The extensive experiments evaluate the effectiveness of improving MOCA reliability

The rest is organized as follows. First, Section 2 summarizes the existing works. Next, Section 3 and Section 4 introduce the research methodology and modeling cloud application with PrT net, respectively. Then, we propose a ranking algorithm for evaluation and apply the FTCARE strategy in Section 5. Section 6 shows experiments, and Section 7 concludes the paper.

2. Related Work

Many types of research have been done in the cloud environment in recent years to improve the availability and reliability of cloud services. Vincenzo and Dragi [5] study and propose a novel method of task unloading with Pareto optimization, which considers three objectives: response time, reliability, and cost. Clab et al. [6] propose a delay adaptive replica synchronization strategy and a replica recovery strategy based on load balancing to solve replica synchronization and recovery of failed nodes in the cloud application. Some indicators have been proposed to measure software reliability, such as Mean Time To Failure (MTTF), Rate of Failure Occurrence (RoCoF), hazard rate, and Probability of Failure on Demand (PoFoD). MTTF and hazard rate are more suitable for system failure with statistical regularity in time. In Ref. [7], the theoretical prediction of the overall MTTF is provided to meet the QoS requirements by fine-tuning the redundancy granularity. Chen et al. [8] improved the cloud service performance by minimizing the propagation of uncertainty in the scheduling workflow application. In Ref. [9], a redundancy minimization algorithm using RIR calculates workflow reliability without calculating the reliability of each task. In order to solve the problems of network instability and limited bandwidth, a reliable VANET routing decision-making scheme based on the Manhattan mobile model is proposed by Gao et al. [10] who consider integrating roadside units (RSU) into wireless and wired modes for data transmission and routing optimization.

A considerable amount of literature focuses on designing new methods of modeling services in the cloud computing environment. Firas and Mazlina [11] offer a literature survey towards agent-based Petri net decision-making modeling for cloud service composition. In Ref. [12], a reliability prediction model based on Petri nets is proposed. For an atomic service, a phased reliability model is proposed to predict the reliability from four aspects: network environment availability, hermit equipment availability, discovery reliability, and critical reliability. A method of constructing reliable service composition is proposed in Ref. [13] with the Petri net, which provides a method to observe the essential behavior of components and describe their relationships. Huang et al. [14] try to solve these challenges by designing a simulation-based optimization method for reliability-aware service composition. A composite stochastic Petri net model of multilayer edge computing system dynamics is proposed and analyzed quantitatively. Zang et al. [15] propose a scheme to automatically generate a service dependency graph using a service registry and improve the service fault tree with a reliability model. In Ref. [16], a Finite State Machine (FSM) model-driven architecture is established, and a typical implementation of the architecture is discussed, including two actual use cases and related evaluations.

Microservices are increasingly regarded as a promising architecture style, building large-scale cloud-based applications within and across organizational boundaries. This microservice-based architecture dramatically improves application scalability, but it also brings expensive performance overhead, which requires careful design of modeling and task scheduling [17]. The proposed methods in this paper are different from the related works that we considered microservice composition with workflow scheduling. Yao et al. [18] propose a fault-tolerant multiprocessor scheduling algorithm using time redundancy. Zhao et al. [19] combine the resubmission and replication with meeting the soft time limit of the workflow. Setlur et al. [20] study the scheduling problem with less resource redundancy to optimize the reliability and minimize the completion time. In Ref. [21], a replication heuristic algorithm in an unsupervised way has been studied. Safari et al. [22] consider the number of copies, frequency, and reliability and uses the reserved processor to replicate and redistribute copies on the reservation processor.

In the field of cloud-based software stable operation, fault-tolerant technology has been widely concerned with solving the reliability problem under a dynamic and uncertain operating environment. Gao et al. [23] propose a dynamic reconfiguration method of mobile e-commerce service workflow based on cloud edge. Using task replication, Sharif et al. [24] meet the reliability requirements by improving the service quality. Yao et al. [25] propose a nonreplicating model to take over the failed fog node to normalize the fog node by using an operable fog node. Ray et al. [26] propose a novel workflow scheduling algorithm for a cloud application, which combines resubmission and replication. In Ref. [27], an active fault-tolerant system based on CPU temperature is proposed, preempting the federation faults. Shi et al. [28] propose a strategy with mechanism, execution mode, and required resource, ensuring real-time applications. In Ref. [29], the temporal-perturbation aware reliability sensitivity, a disturbance sensing method, is proposed to measure the reliability sensitivity of cloud service components for adaptive cloud service selection.

Although there are many kinds of research on reliability, few current research works consider the reliability of microservice-based cloud application. In particular, we focus on modeling the reliability of cloud applications based on microservice through a redundancy operation. One of the main goals is to reduce the importance of microservices so that the microservices with high request frequency can be backed up. Thus, the reliability of cloud applications is modeled by the fluctuation frequency and amplitude of reliability in a cycle. In this way, the reliability of microservice composition is maximized.

3. Framework for Reliability Modeling and Analysis

This section discusses how to model the problem and build the whole framework, covering the reliability analysis approach in the following research. As shown in Figure 1, three blocks with solid lines represent three different modules, and three dotted boxes are the analysis process.

3.1. User Module. A user requests the service from the cloud application, in which user requests can be quantitatively described by request frequency. That is, there are n requests from a user per unit time. In each unit time, the failure rate of a microservice is variable due to the change of environment. Besides, in the SaaS (Software-as-a-Service) environment, the microservice composition required by users is usually composed of multiple microservices, which serve with different request frequencies.

3.2. Microservice Composition Module. By arranging the microservices in the workflow, the tasks are distributed as a whole cloud application, in which the microservice has an independent failure rate. However, there are dependency and cascade relationships between microservices in practical applications, which will affect the whole application. When a microservice calls the task in the other microservice, the workflow process is regarded as a combination of microservices.

3.3. Redundancy Module. The microservices deployed using containers are independent of one another. Each microservice in a container implements multiple tasks such as catalog microservice, ordering microservice, and user profile microservice. An FTCARE strategy with Cloud Application Reliability Optimization (CARO) algorithm for cloud application is proposed to achieve more reliable, more effective and more robust design. Furthermore, the execution model depends on the deadline of microservice and execution types.

3.4. Reliability Analysis Module. The formal model of microservice composition with a reliability attribute and its time series is defined. The reliability evaluation method is proposed based on failure probability and exponential reliability equation. The concept of reliability perturbation is defined to describe the reliability changes in continuous time series. The reliability in the evaluation period is calculated with sampled period and failure rate. The microservice sensitivity helps to measure the degradation attribute. The reliability sensitivity based on interference perception and the negative effect of microservice reliability interference is analyzed.

3.5. Critical Identification Module. We propose a microservice ranking framework based on the PageRank algorithm for fault-tolerant cloud applications. Since MOCA involves many microservices, it is expensive to provide a redundant replacement for all microservices in cloud applications. Therefore, it is necessary to identify a small number of critical microservices to reduce costs and develop highly reliable cloud applications within a limited budget.

3.6. Reliability Optimization Module. To calculate cloud application reliability, we combine the time-homogeneous reliability disturbance analysis to identify the critical microservices. Since the reliability value for each microservice is calculated, the microservices are sorted based on their reliability. The ranking algorithm generates the redundancy scheme. For the FTCARE strategy with redundancies, the most critical microservice determines the optimal solution.

4. Reliability Model Based on PrT Net

This section describes the definition of the basic concept and the reliability calculation with predicated Petri net (PrT net), which includes three modeling modules in the framework. Then, the reliability model is analyzed, and the PrT net variables are applied as the parameters in the calculation.

4.1. PrT Net Syntax and Semantics. As a graphical modeling tool and formal method with a rich mathematical foundation, PrT net has been widely used to represent the

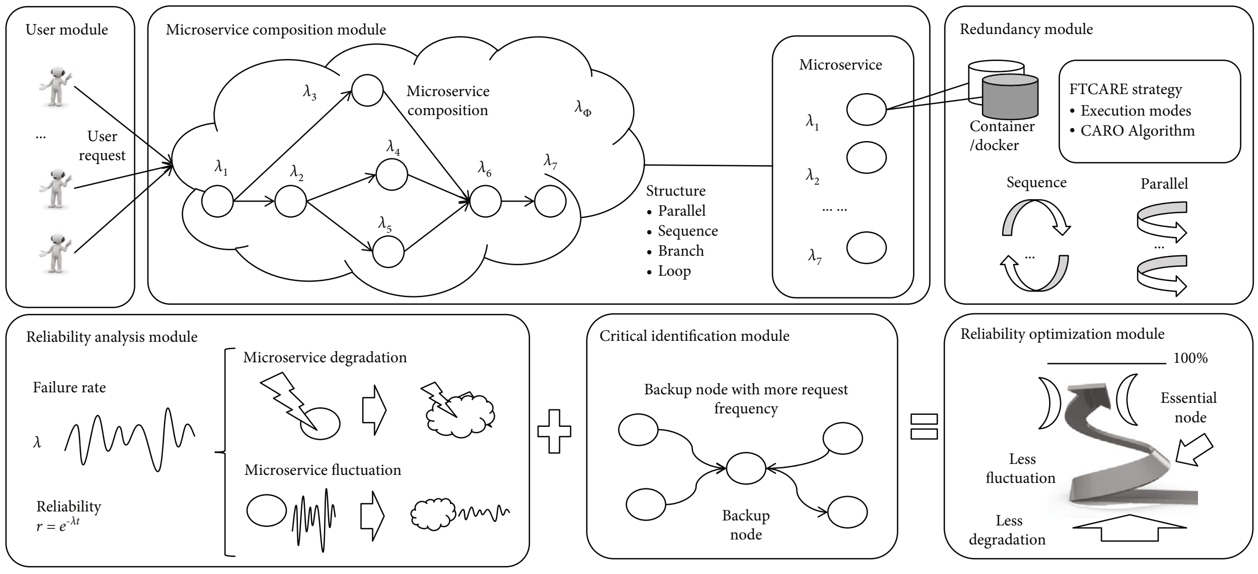


FIGURE 1: Framework for reliability modeling and analysis.

microservice and influencing factors, and we give the basic definitions as follows:

Definition 1. A triple $N = (P, T, F)$ is defined as a Petri net.

(1) $P = \{p_1, p_2, \cdots, p_{|p|}\}$ is a finite set of places

(2) $T = \{t_1, t_2, \cdots, t_{|t|}\}$ is a finite set of transitions and $P \cup T \neq \emptyset$, $P \cap T \neq \emptyset$

(3) $F \subset (P \bigtimes T) \cup (T \bigtimes P)$ is a set of directed arcs

The PrT net is used to model the execution process of microservice instances. The state, possible operation, and parameter of microservice instances are described by repository, transformation, and interface, respectively. For any $x \in (P \cup T)$, set $x = \{y \mid y \in (P \cup T) \wedge (y, x) \in F\}$ and $x = \{y \mid y \in (P \cup T) \wedge (x, y) \in F\}$ correspond to the input and output of x, respectively.

Definition 2. A 6-ary tuple $\Sigma = (N, IO, D, A_T, A_F, M_0)$ is called PrT net and used as the primary service net (PSN), where we have the following:

(1) $N = (P, T, F)$ is a basic Petri net, where P, T, and F, denote the finite set of place, transition, and arc, respectively

(2) $IO \subset P$ is a special set, which is the interface of Σ

(3) D is a nonempty finite individual set, and f_D and f_S are given as predicate sets and symbol sets, respectively

(4) $A_T : T \longrightarrow f_D$, for $t \in T$, the free variable in $A_T(t)$ must be a directed arc free variable with t as one end

(5) $A_F : F \longrightarrow f_S$, if $(p, t) \in F$ or $(t, p) \in F$, then $A_F(p, t)$ or $A_F(t, p)$ is the sum of $n$ary symbols and is null by default

(6) $M : P \longrightarrow f_S$ is the marking of Σ, $M_0(p)$ is the initial marking

For $\forall t \in T$, if $FV(t_i) = \{x_1, x_2, \cdots, x_n\}$, the individuality set $\{d_1, d_2, \cdots, d_n\}$ meets $d_i \in \{M(p) \mid p \in {}^\bullet t \cup t^\bullet\}$ and d_i corresponds to the variable x_i, denoted by $t\langle d_1, d_2, \cdots, d_n\rangle$. For any $t_i \in T$, the input/output arc of transition t_i and the set of free variables in $A_T(t_i)$are denoted as$FV(t_i)$.$A_T(t)\langle d_1, d_2, \cdots, d_n\rangle$and$A_F(p, t)\langle d_1, d_2, \cdots, d_n\rangle$describe that the individuals $d_1, d_2, \cdots, d_n$ replace the formula $A_T(t)$ and the predicate $A_F(p, t)$, respectively. If $A_T(t)\langle d_1, d_2, \cdots, d_n\rangle$ = true, then $t\langle d_1, d_2, \cdots, d_n\rangle$ is called a feasible replacement of transition t under M. All feasible replacements of transition t under M are denoted by set $\mathrm{VP}(M, t)$.

If $\mathrm{VP}(M, t) \neq \emptyset$, then transition t is enabled under M, denoted by $M[t >$. Transition t is enabled under M if and only if there is a feasible replacement under M. All the enabled transitions under M are denoted by set $\mathrm{ET}(M)$. For transition $t_i \in \mathrm{ET}(M)$, if there is no transition $t_j \in \mathrm{ET}(M)$, whose priority is higher than t_i, the firing of transition t_i under M is effective. All the effective firing transitions under M are denoted by set $\mathrm{FT}(M)$. The process that M reaches a new marking M' by firing a feasible replacement $t_i\langle d_1, d_2, \cdots, d_n\rangle$ of transition t_i is denoted by $M[t_i\langle d_1, d_2, \cdots, d_n\rangle > M'$.

We model the microservice MS with PSN, where P is the place in the Petri net and represents the different states of the microservice $\mathrm{MS}(i)$. T is a set of firing microservice, representing the transitions in Petri nets. F is the set of arcs between p and t. W is a function defined on t, which represents the weight of transitions. M_0 represents the initial state of the microservice $\mathrm{MS}(i)$. $M_0 = \{m_{01}, m_{02}, \cdots, m_{0n}\}$, where$m_{0n}$is the number of tokens at state n, and M is the number of tokens at n.

The microservice composition with many microservices can be defined as the hierarchical framework. On this basis, the HMCN definition for microservice composition with different structures is given.

Definition 3. A 6-ary tuple $\Omega = (\Sigma, \Gamma, TI, TA, PI, PA)$ is called hierarchical microservice composition net (HMCN).

(1) Σ is a PSN, which describes the basic structure of Ω

(2) $\Gamma = \{\Gamma_i \mid i \in N*\}$ is a finite set of pages, PSN or HMCN. If $N_i = (P_i, T_i, F_i)$, then $\forall \Gamma_i, \Gamma_j \in \Gamma$, $(P_i \cup T_i \cup F)_i \cap (P_j \cup T_j \cup F_j) = \emptyset$

(3) TI $\subset T$ is a collection of alternative nodes, where each page corresponds to a replacement node

(4) TA : TI $\longrightarrow \Gamma$ is the page allocation function, which is used to assign specific pages for a replacement node

(5) PI $\subset P$ is a set of port nodes that describe the input and output locations of the substitute nodes

(6) PA is a port mapping function. The purpose is to map the port node of the replacement node to the interface of the corresponding page

PSN is a special case of HMCN, namely HMCN with Γ, which is equal to null. HMCN is mainly used for modeling the components in MOCA or the whole microservice composition. Individual set D is mainly used to describe microservice set or available instance set.

According to the microservice relations and behavior characteristics of microservices, PSN and HMCN are mapping with a double circle and a dotted circle, respectively. The page allocation function is to assign specific pages for each replacement node. In order to distinguish between the input interface and output interface, the input interface is marked with superscript I, and the output interface is marked with superscript O. In particular, only some basic concepts are introduced, and other concepts can refer to Ref. [30].

The microservice request is modeled as HMCN shown in Figure 2, which has the specific operation process: (1) Place p_s is used to store the execution task for request. Transition t_{req} fires if others request the microservice MS(i). (2) After obtaining the input parameter $p^I_{\text{ser\_s}}$, the task individual is ready to put into the execution places p_{exe} and p_{ctrl} to control the deadline. If the task cannot realize the function within the deadline ($M(p_{\text{to}}) \neq \emptyset$), time-out transition t_{to} is fired. Otherwise, the execution time is insufficient before task deadline $t > T_{\text{deadline}}$, where $t = T_{\text{arrival}} + t_{\text{execution}}$. The parameters T_{arrival} and $t_{\text{execution}}$ represent arrival time and consuming execution time in the container, respectively. Next, the task results are fed back from the container ($M(p^O_{\text{containero}}) \neq \emptyset$), then output after firing t_{exe} is transferred to place $p^O_{\text{ser\_e}}$. Finally, place p_e is the termination of the microservice request.

Places $p^I_{\text{ser\_s}}$ and $p^O_{\text{ser\_e}}$ with dotted circles are mapping to places $p_{\text{ser\_s}(i)}$ or $p_{\text{ser\_s}(j)}$ and $p_{\text{ser\_e}(i)}$ or $p_{\text{ser\_e}(j)}$ with double circles, respectively, which define the fundamental structures of microservice composition in Definition 4. Places $p^I_{\text{containeri}}$ and $p^O_{\text{containero}}$ with dotted circles are mapping to places $p_{\text{containeri}}$ and $p_{\text{containero}}$ with double circles, respectively, which formulates the redundancy modes in Definition 5.

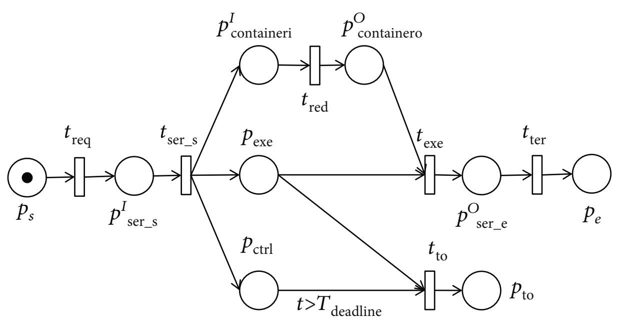


Figure 2: Modeling microservice request.

Property 1. The state of request service model can be either executed in the container or out of time.

Proof. Given a token as user request in place p_{exe}, $M(p_{\text{exe}}) \neq \emptyset$. When the condition $t > T_{\text{deadline}}$ satisfies, then transition t_{to} is fired, $M(p_{\text{exe}}) = \emptyset$. The prepositional place p_{exe} of transition t_{to} has no token anymore, so the token cannot reach place p_e. On the contrary, it can be proved too. □ □

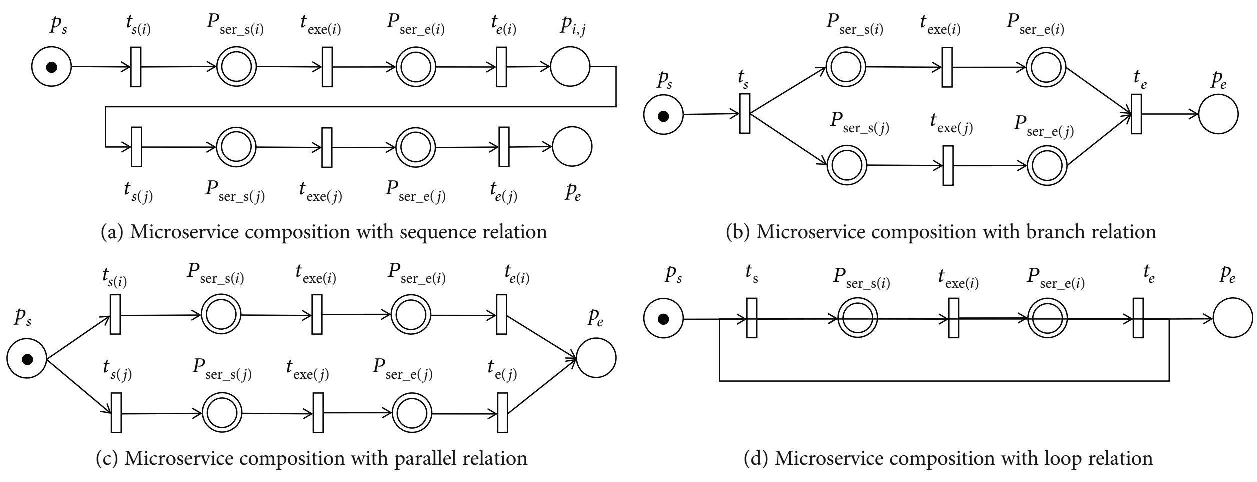
In this HMCN model, places $p^O_{\text{ser\_e}}$ and place p_{to} represent two states (executed in container successfully and running out of time) in the microservice request model, respectively. They are proved to be mutually exclusive.

Definition 4 (microservice relation). The symbols MS$(i) >$ MS(j), MS(i) + MS(j), MS$(i) \parallel$ MS(j), and MS(i)©MS(j), represent the sequence, branch, parallel, and loop relations between MS(i) and MS(j), respectively.

The microservice relation of microservice composition is modeled with RL(i, j), which represents the relation between MS(i) and MS(j). RL(i, j): MS$(i) \times$ MS$(j) \longrightarrow \{>, +, \parallel, ©\}$ is the relationship between tasks, as depicted in Figure 3.

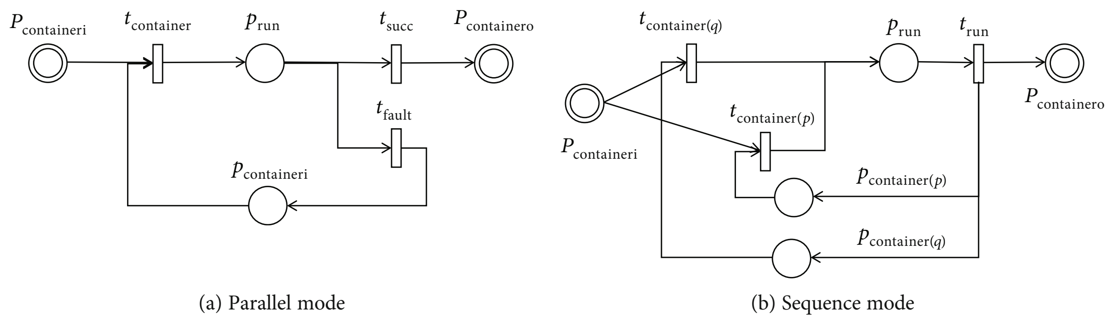
Definition 5 (redundant fault-tolerant mechanism). A pair RFTM = (FT, MODE) is used to describe the redundant mechanism for fault tolerance. The redundant microservice is assigned in light of the CARO algorithm. FT(MS(i)) = {MS$(i, 1)$, MS$(i, 2)$, ⋯, MS(i, r), ⋯, MS(i, R)}. MS(i) is the primary microservice, and MS(i, r) is the rth backup of microservice MS(i). When the primary microservice MS(i) fails, the standby microservices MS(i, r) starts, thus improving the reliability. MODE = {SEQ, PAR}. Under the deadline of the microservice T_{deadline}, the redundancy mode can be sequence or parallel, in which the backups are executed actively or passively. For active backup execution, the primary and secondary copies are executed at the same time. When the primary copy runs successfully, the secondary copy is terminated. For passive backup execution, after the primary copy fails, the secondary copy starts to run. If the task execution is complete, the container is released.

The container model realizes the redundant fault-tolerant mechanism, as shown in Figure 4. Place $P_{\text{containeri}}$ stores the queuing tasks of the container as tokens. If the



(a) Microservice composition with sequence relation
(b) Microservice composition with branch relation
(c) Microservice composition with parallel relation
(d) Microservice composition with loop relation

Figure 3: Modeling basic structures of microservice composition.



(a) Parallel mode
(b) Sequence mode

Figure 4: Modeling container with two modes.

processing capacity of the container is faster than the request frequency, then transition $t_{container}$ is fired. Finally, place p_{run} represents the execution state of tasks.

Property 2. The designed PrT net can effectively describe the microservice backup redundancy in the container.

Proof. According to the backup execution mode, there are two redundancy modes: sequential and parallel. Therefore, this property is proven from the following two parts:

(i) When the condition $(\mathrm{T}_{deadline} - \mathrm{T}_{arrival}) > 2 * t_{execution}$ is satisfied, then the sequence redundancy mode works. In case of failure, the transition t_{fault} fires, and the token is transited to $p_{container}$, which is ready for the next iteration

(ii) When the condition $(T_{deadline} - T_{arrival}) < 2 * t_{execution}$ is satisfied, then the parallel mode is selected for redundancy. The synchronized execution of microservices is realized by t_{run}. Transitions $t_{container(p)}$ and $t_{container(q)}$ fire independently, representing the parallel redundancy mode

$T_{deadline}$ is a given variable, and $T_{arrival}$ depends on the actual deployment of the container and the schema of the microservice composition. □ □

The microservice MS in microservice composition MC performs on specific nodes and links. In case of failure from the node or link, the whole composition will also be affected. The reliability of microservice composition is analyzed based on the PrT net. The reliability requirement for MOCA is defined below.

Definition 6 (cloud application reliability requirement). The reliability of cloud application is a five-tuple (MS, MC, RD, RP, V). MS and MC are the microservice and the microservice compositions defined above, respectively. For the microservices $\{\mathrm{MS}(1), \mathrm{MS}(2), \cdots, \mathrm{MS}(i), \cdots, \mathrm{MS}(n)\}$, the parameters $\mathrm{RD} = \{\mathrm{rd}_{(1,\mathrm{MC})}, \mathrm{rd}_{(2,\mathrm{MC})}, \cdots, \mathrm{rd}_{(n,\mathrm{MC})}\}$ and $\mathrm{RP} = \{\mathrm{rp}_{(1,\mathrm{MC})}, \mathrm{rp}_{(2,\mathrm{MC})}, \cdots, \mathrm{rp}_{(n,\mathrm{MC})}\}$ represent reliability degradation and perturbation, which are two basic reliability indicators for the measurement. The criticality value $V = \{V_1, V_2, \cdots, V_n\}$ of each microservice can be calculated with microservice relation and request frequency.

In this paper, RD and RP are the microservice reliability quality parameters analyzed in Section 5 and the criticality value V helps to measure the role of a microservice.

4.2. Reliability Calculation. Each microservice has a special attribute failure rate, which describes the number of failures over a while. The failure rate can be defined as the expected number of project failures in a specified period [31]. The failure rate of a microservice conforms to the normal distribution $N(\mu, \sigma^2)$ in this paper.

In the exponential reliability calculation, the failure rate parameter (i.e., λ_i for the ith microservice) determines the reliability evaluation indicator and time t is measured as the execution time of a microservice.

$$r_i = e^{-\lambda_i * t}. \quad (1)$$

Let MC be the microservice composition, and $\{\lambda_1, \lambda_2, \cdots, \lambda_n\}$ be the failure rate of n microservices. The failure rate of the microservice composition is calculated based on the workflow execution. MOCA may contain more than one microservice relation. Wang et al. [29, 30] provides the aggregation functions in Table 1 to calculate the λ_{MC} of the sequential, parallel, branching, and loop relations of microservice composition.

The parameter b_i represents the execution probability of the ith branch in microservice composition, and l_i is used to represent the execution probability of i times in a cycle.

Due to the uncertainty of the internal working state and the instability of the cloud operation environment, the time interval of fault occurrence is uncertain. For a specific microservice or the microservice composition in cloud application within a specific time interval ΔT in a lifetime, the reliability is as follows:

$$r^{\Delta T} = e^{-\lambda * \Delta T}. \quad (2)$$

Unstable links in microservice composition will affect the response time of microservices in the lifetime intervals $\Delta T(0), \Delta T(2), \cdots, \Delta T(m), \cdots, \Delta T(M-1)$, which may lead to execution out of time or unexpected failure during processing tasks. $\Delta T(m)$ represents the mth period ΔT in the time series. Each period has a reliability evaluation value $r^{\Delta T(m)}$.

According to Ref. [32], the evolution in reliability time series satisfies the 1st-order Markov Chain rule. Moreover, since uncertain events trigger a single one-step transition, the statistics of multiple one-step transitions in advance can reflect the long-term evolution law of reliability.

5. Reliability Analysis and Algorithm

This section analyzes the cloud application reliability and proposes a redundancy mechanism, three analysis modules in the framework.

5.1. Reliability Perturbation Analysis. This section describes the dynamic influence of microservice perturbation on microservice composition.

The microservice is running under a specific status during a period. For any $m \in [1, M-1]$, the mth reliability perturbation of the ith microservice within $\Delta T(m)$ can be defined as follows:

$$p_i^{\Delta T(m)} = r_i^{\Delta T(m+1)} - r_i^{\Delta T(m)}, \quad (3)$$

Table 1: The calculation for the failure rate of microservice composition according to structure.

Structure	Failure rate calculation
Sequence	$\lambda_{\mathrm{MC}} = 1 - \prod_{i=1}^{n}(1-\lambda_i)$
Parallel	$\lambda_{\mathrm{MC}} = 1 - \prod_{i=1}^{n}(1-\lambda_i)$
Branch	$\lambda_{\mathrm{MC}} = 1 - \prod_{i=1}^{n} b_i * (1-\lambda_i),\ \sum_{i=1}^{n} b_i = 1$
Loop	$\lambda_{\mathrm{MC}} = 1 - \prod_{i=1}^{n} l_i * (1-\lambda_i)^i,\ \sum_{i=1}^{n} l_i = 1$

where $r_i^{\Delta T(m)}$ represents the reliability of microservice s_i within $\Delta T(m)$. Similarly, $r_{\mathrm{MC}}^{\Delta T}$ represents the reliability of microservice composition MC within $\Delta T(m)$.

In order to eliminate the incompatibility between the different parameters, the Min-Max normalization technique widely used in formula (4) is used to normalize the original value. After performing Min-Max normalization for perturbations, the reliability perturbation rate of the ith microservice within $\Delta T(m)$ is defined as follows:

$$pr_i^{\Delta T(m)} = \begin{cases} \dfrac{p_i^{\max} - p^{\Delta T(m)}}{p_i^{\max} - p_i^{\min}}, & \text{if } p_i^{\max} \neq p_i^{\min}, \\ 1, \quad \text{if } p_i^{\max} = p_i^{\min}, \end{cases} \quad (4)$$

where

$$p_i^{\max} = \max_{m \in [1, M-1]} p_i^{\Delta T(m)}, \quad (5)$$

$$p_i^{\min} = \min_{m \in [1, M-1]} p_i^{\Delta T(m)}. \quad (6)$$

Similar to formula (3) and formula (4), $p_{\mathrm{MC}}^{\Delta T(m)}$ and $pr_{\mathrm{MC}}^{\Delta T(m)}$ are perturbation and perturbation rate of microservice composition MC within $\Delta T(m)$.

The perturbation function for the single-step transition can be regarded as a 1st-order Markov Chain evolution, which describes the evolution rules of the latest reliability time series of a microservice. The multiple times of perturbations in the 1st-order Markov evolutionary reliability time series can reflect the cumulative effects. The closer the disturbance is, the more accurate the execution state is.

A reliability sensitivity measurement method based on perturbation perception is proposed based on reliability perturbation, and the cumulative negative impact of perturbation is analyzed. Assuming $(M-1)$ times of perturbations in time series, the cumulative effect defines an influencing factor parameter $\alpha \in [0, 1]$ and the influencing factor for the mth perturbation is α^{M-m}. Through calculating the reliability sensitivity, the perturbation based on the disturbance is measured by

$$rp_{(i,\mathrm{MC})} = \frac{1}{M-1} * \sum_{m=1}^{M-1} \alpha^{M-m} \pi^{\Delta T(m)}, \quad (7)$$

where

$$\pi^{\Delta T(m)} = \begin{cases} \dfrac{pr_{\mathrm{MC}}^{\Delta T(m)}}{pr_i^{\Delta T(m)}}, p_{\mathrm{MC}}^{\Delta T(m)} > 0, \\ 0, p_{\mathrm{MC}}^{\Delta T(m)} \le 0. \end{cases} \quad (8)$$

The reliability sensitivity $\pi^{\Delta T(m)}$ in formula (7) is equal to zero when the perturbation of microservice composition MC within $\Delta T(m)$ is negative. The more significant the positive impact on the composition of microservices, the higher the sensitivity of microservices. If the microservice has a negative effect, the sensitivity is 0.

5.2. Reliability Degradation Analysis. The degradation caused by microservice runtime exceptions may lead to cascading effects in cloud applications. This section describes the static impact of microservice degradation on microservice composition.

In the dynamic cloud environment, the changes lead to the upgrading or degradation of a microservice, measured by microservice sensitivity.

The fluctuation rate of microservice s_i at continuous time series points is calculated with formula (9):

$$fr_{(i,\mathrm{MC})} = \frac{1}{M-1} * \sum_{m=0}^{M-1} \pi^{\Delta T(m)}, \quad (9)$$

where

$$\pi^{\Delta T(m)} = \frac{pr_{\mathrm{MC}}^{\Delta T(m)}}{pr_i^{\Delta T(m)}}. \quad (10)$$

The reliability fluctuation rate with M time intervals ΔT is represented by the ratio of the reliability of microservice composition to the reliability of microservice s_i in unit time.

The reliability degradation of microservice composition MC under influence of microservice s_i is calculated as follows:

$$rd_{(i,\mathrm{MC})} = \frac{fr_{(i,\mathrm{MC})}}{\sum_{i=1}^{n} fr_{(i,\mathrm{MC})}}. \quad (11)$$

5.3. Critical Microservice Identification. This section describes the importance of user requests to each microservice from the structure level of microservice composition.

In the cloud application, some microservices are often requested by others. These microservices are considered more critical because their failure has more impact on the microservice composition than other normal ones. Intuitively, critical elements in MOCA are those that have many requests from other critical ones. Inspired by the PageRank algorithm in Ref. [33, 34], we propose an algorithm to measure the criticality of cloud microservices.

A microservice composition can be modeled as a weighted directed graph G, where a node s_i in the graph represents a microservice and a directed edge $e_{j,i}$ from node s_i to node s_j represents the invocation relationship, i.e., s_i invokes s_j. Each node s_i in graph G has a nonnegative criticality value V_i, which is in the range of $(0, 1)$. Each edge $e_{j,i}$ in the graph has a nonnegative weight value $W(e_{j,i})$, which is in the range of $[0, 1]$. The weight value of an edge $e_{j,i}$ based on request frequency can be calculated by

$$W(e_{j,i}) = \frac{\mathrm{freq}_{j,i}}{\sum_{j=0}^{p} \mathrm{freq}_{j,i}}, \quad (12)$$

where $\mathrm{freq}_{j,i}$ is the invocation frequency of microservice s_j by other microservice s_i and p is the number of microservices invoked by s_i. In this way, the edge $e_{j,i}$ has a more considerable weight value if microservice s_j is invoked more frequently by microservice s_i compared with other microservices invoked by s_i.

The measurement of the microservice importance is based on invocation relationship and request frequency [35]. The microservices are considered more critical if many other essential microservices frequently request them. This microservice failure has more impact on the whole cloud application than the normal microservices. The criticality value for the microservice s_i is as follows:

$$V_i = \frac{1-d}{n} + d * \sum_{j=1}^{p} V_i * W(e_{j,i}), \quad (13)$$

where n is the number of microservices and p is the number of microservices that invoke microservice s_i. The parameter $d (0 \le \mathrm{d} \le 1)$ in equation (13) is employed to adjust the criticality values derived from others so that V_i is composed of the fundamental value of itself (i.e., $(1-d)/n$) and the derived values from the microservices that invoked s_i.

By formula (13), microservice s_i has a more considerable criticality value if the number of predecessors microservices p and their weight value $W(e_{j,i})$ are large, indicating that microservice s_i is invoked by the other critical microservices frequently.

Reliability degradation measures the degree of microservice composition affected when microservice is degraded. The critical microservice identification indicates the importance of a microservice in the microservice composition with request frequency.

5.4. CARO Algorithm. This section will analyze the reliability of cloud applications and explain the CARO algorithm based on redundancy technology.

```
Input: $\lambda_i^{\Delta T(m)}$, $\gamma$, number of redundant microservices $R$, microservice composition schema $S$, and $M$ time series from initial time interval $\Delta T(0)$ to final time interval $\Delta T(M-1)$ number of microservices $n$
Output: index list of redundant microservices $ReS$
1 calculate $r_i^{\Delta T(m)}$ with formula (2);
2 calculate $\lambda_{MC}^{\Delta T(m)}$ and $r_{MC}^{\Delta T(m)}$ with formula (2) and formulas in Table I;
3 for $r = 1$ to $R$ do
4   for $i = 1$ to $n$ do
5     calculate $rd_{(i,MC)}$ with formula (7);
6     calculate $rp_{(i,MC)}$ with formula (11);
7     calculate $V_i$ with formula (15);
8   if $r_{MC} \le r_{MC}^E$ then
9       calculate $CAR_{(i,MC)}$ with formula (11) and formula (15);
10    else
11        calculate $CAR_{(i,MC)}$ with formula (7);
12 ranking $CAR_{(i,MC)}$ and get max index $i$;
13 add index $i$ to $ReS$;
14 update microservice composition schema $S$;
15 return $ReS$;
```

Algorithm 1: CARO.

The microservice fails only if all the backups fail, so formula (14) can calculate the failure rate after R backups the redundancy process:

$$\lambda_i^R = 1 - \prod_{r=1}^{R} (1 - \lambda_i^r). \tag{14}$$

Let R be the number of redundant microservices and r_i^r be the reliability of the rth redundancy of microservice s_i.

The criticality value is updated because of changing the microservice schema.

$$V'_i = \frac{1-d}{n} + d * \sum_{j=1}^{p'} V'_i * W(e_{j,i}), \tag{15}$$

where

$$p' = p + |s_i^r|. \tag{16}$$

In formula (16), $|s_i^r|$ is the redundancy number of microservice s_i and p' is the number of microservices that invoke microservice s_i after redundancies.

To measure the negative influences of microservice reliability degradation and perturbation, we combine the degradation and perturbation, considering microservice criticality, and present a novel approach for cloud application reliability measurement. The cloud application reliability of microservice composition MC under the influence of microservice s_i is calculated as follows:

$$\mathrm{CAR}_{(i,\mathrm{MC})} = \begin{cases} \gamma * rd_{(i,\mathrm{MC})} + (1-\gamma) * V'_i, & r_{\mathrm{MC}} < r_{\mathrm{MC}}^E, \\ rp_{(i,\mathrm{MC})}, & r_{\mathrm{MC}} \ge r_{\mathrm{MC}}^E. \end{cases} \tag{17}$$

In the case of lower reliability, $r_{\mathrm{MC}} < r_{\mathrm{MC}}^E$ in formula (17), the purpose is to improve the microservice reliability and reduce its degradation to the cloud application. r_{MC}^E is the expected value of reliability for microservice composition MC. After achieving the desired value, the main objective is to stabilize the perturbation, and $rp_{(i,\mathrm{MC})}$ measures the reliability quality dominantly.

The CARO algorithm includes two parts: (1) ranking the influence degree from microservice reliability to cloud application and (2) selecting the optimal fault-tolerant strategy. The development procedures are as follows:

(1) The initial architecture design of MOCA is provided with a microservice composition graph. The microservices can be sorted based on the cloud application reliability, and the most critical microservices can be identified. The ranking results determine the allocation of redundancies (line 12 in Algorithm 1).

(2) Since the most critical microservices are identified, each candidate performance is calculated, and the most suitable redundancy resource is selected for each important microservice (lines 5-11 in Algorithm 1) after iteration. The improved design and ranking results are updated to the redundancy scheme again (line 14 in Algorithm 1) after iteration

The output value ReS in the CARO algorithm represents the redundancies for each microservice in the schema. After the reliability optimization process, we can obtain the updated microservice composition schema with redundancies.

6. Simulation

We conducted a series of experiments to evaluate the effectiveness of the proposed FTCARE strategy. This paper focuses on the reliability guarantee of cloud applications based on a microservice. In conclusion, the experiment is aimed at answering the following Research Questions (RQ):

(i) Proving the effectiveness of the FTCARE strategy with different workflow examples. According to the analysis, the cloud application reliability based on a microservice increases with redundancy operation, and minor perturbation is expected

(ii) Comparing the cloud application reliability with and without considering degradation. Considering the degradation, the improvement of microservice reliability is faster for saving time to stabilize reliability perturbation

6.1. Simulation Environment. The reliability improvement approach is implemented in Java with Python 3.6.0 and PyCharm 2020.1.2. A program package for analysis and visualization of cloud application reliability is used to simulate microservice composition workflows. Microservice reliability data sets are obtained from papers of other researchers, and other data are generated in simulation.

Take ten workflows in Ref. [36] as examples, as depicted in Figure 5. These ten commonly used workflow models are based on the basic structure of microservice composition, including serial and parallel. By mixing the basic microservice composition structure, we can get individual examples. In ten groups of workflows, each microservice has an independent failure rate. The microservice composition composed of microservices also has its failure rate and reliability. The failure rate of microservice composition is determined according to the microservice relations modeling and analysis.

Fourteen microservices are selected as the essential simulation elements, which run for a period of 100 s. In order to simplify the actual scene, in the simulation experiment, each workflow node only runs one microservice. That is, each task is mapping one microservice. During the runtime, the failure rate varies in the range of $(0, 0.1]$, and we observe 99 intervals for simulation. The microservice failure rate dataset is generated randomly because the failure in an actual situation occurs by accident or under rapidly changing conditions. Moreover, in the observed execution time, the microservice failure rate conforms to a normal distribution in formula (18) with a mathematical expectation of $\mu = 0.05$ and variance $\delta = 0.015$.

$$\lambda_i^{\Delta T(m)} = \text{random•normal}(\mu, \delta, N), \tag{18}$$

where m represents the time interval [1,100] and $i \in [1, 14]$. N is the sample number.

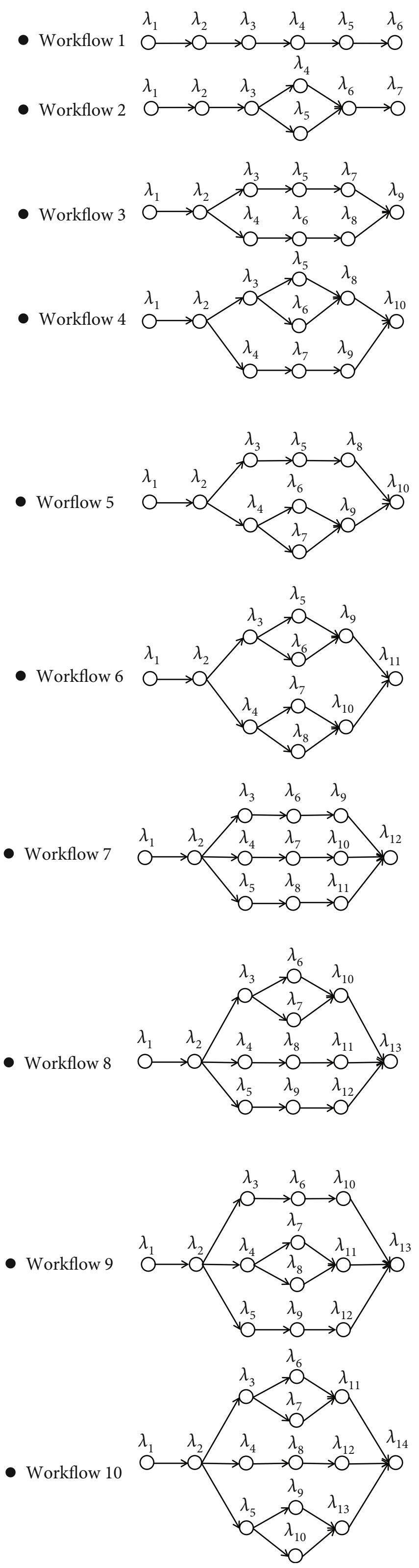


Figure 5: Ten workflows of microservice composition for simulation.

The request frequency between its processor or successor is generated randomly in the interval [80,100] with formula (19).

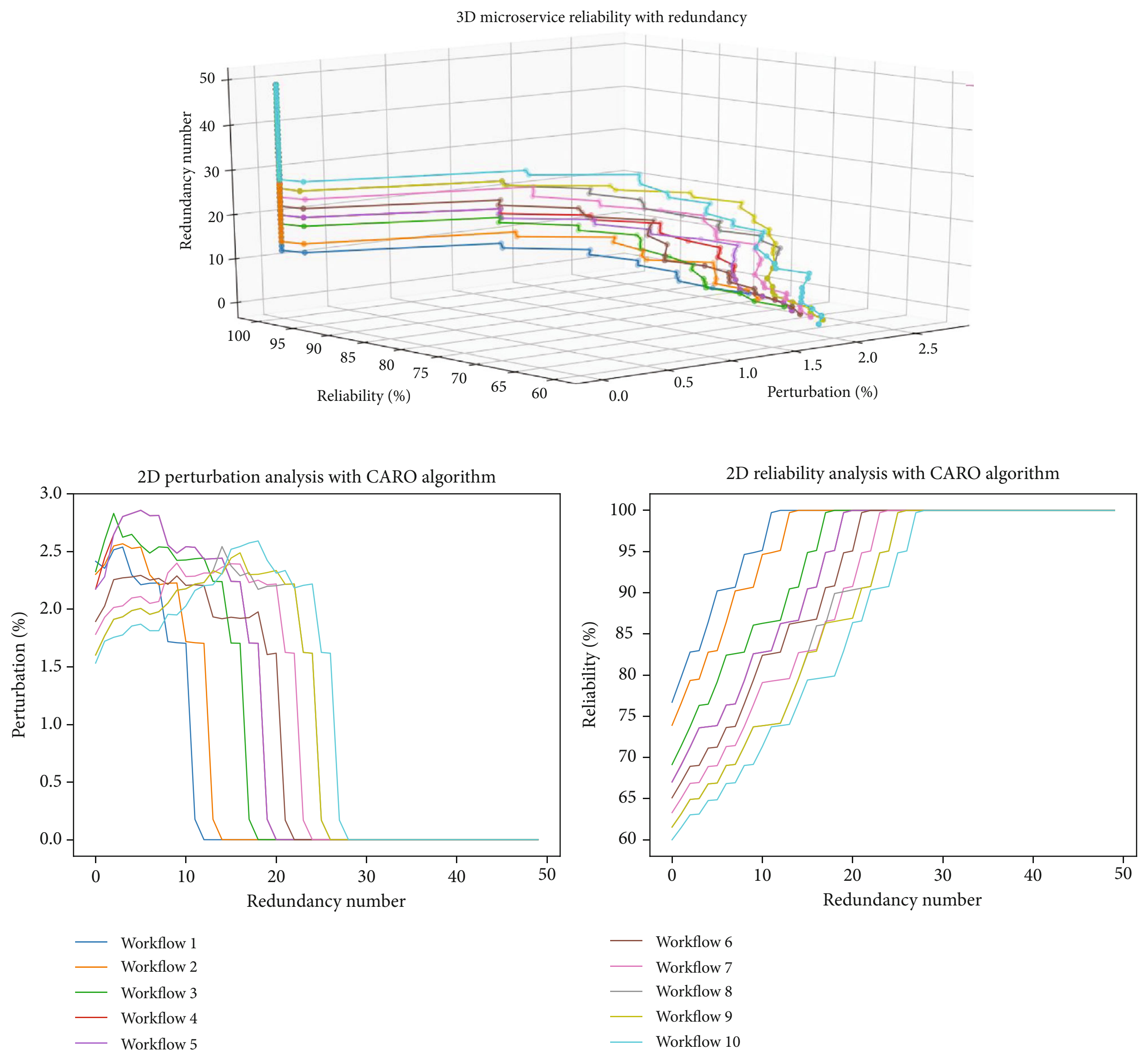


Figure 6: Reliability analysis for ten workflows.

$$\mathrm{freq}_{j,i} = \mathrm{random}() * \mathrm{scale} + \mathrm{min}, \quad (19)$$

where $\mathrm{scale} = 20$ and $\mathrm{min} = 80$.

The purpose of the experiment is to explore the reliability of microservice composition under dynamic disturbance. In order to simplify the simulated experiment, the request frequency is constant during the whole execution time in the experiment.

The redundancy scheme is designed as an array with repeating elements, representing the redundant microservice, and the failure rate will update by replicating critical elements in Section 5. For example, the scheme array [1, 3, 5] means the first microservice has two redundancies, and the third and fifth microservice has only one backup.

The details about the sequence and parallel modes are not considered in simulation related to the execution time and subdeadline of microservices. Instead, we only consider the redundancy number, and the execution mechanism is viewed as a black box.

6.2. Simulation Results. Through the simulation of ten data groups, we can calculate the reliability with redundancies quantitatively. Figure 6 shows the analysis of cloud application reliability after simulating ten workflows. The observed variables are reliability and disturbance. We can find that the algorithm proposed in this paper helps improve the QoS performance of microservice composition based on the fluctuation of existing microservice reliability through these ten sets of data. The trend shows that with the increase of

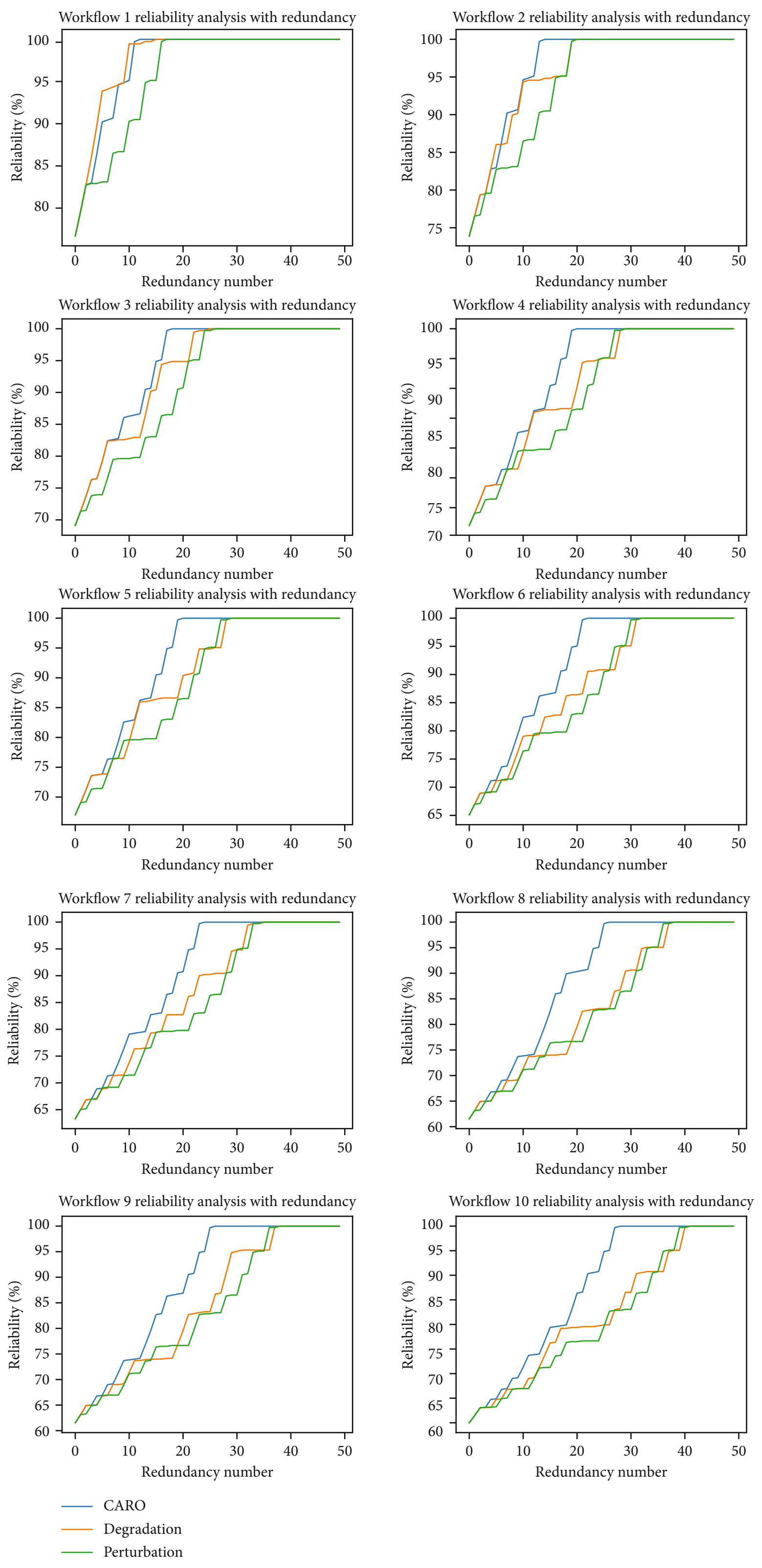


Figure 7: Reliability analysis with various algorithms for 10 workflows.

redundancies, the reliability of microservice composition improves, and its vibration decreases.

As shown in Figure 6, the reliability disturbance decreases from 6% to nearly 0% as redundancies increase. However, this change is not linear because the disturbance is focused on the difference in microservice reliability between adjacent sampling periods in time series. Thus, by improving the reliability of microservices, the reliability of

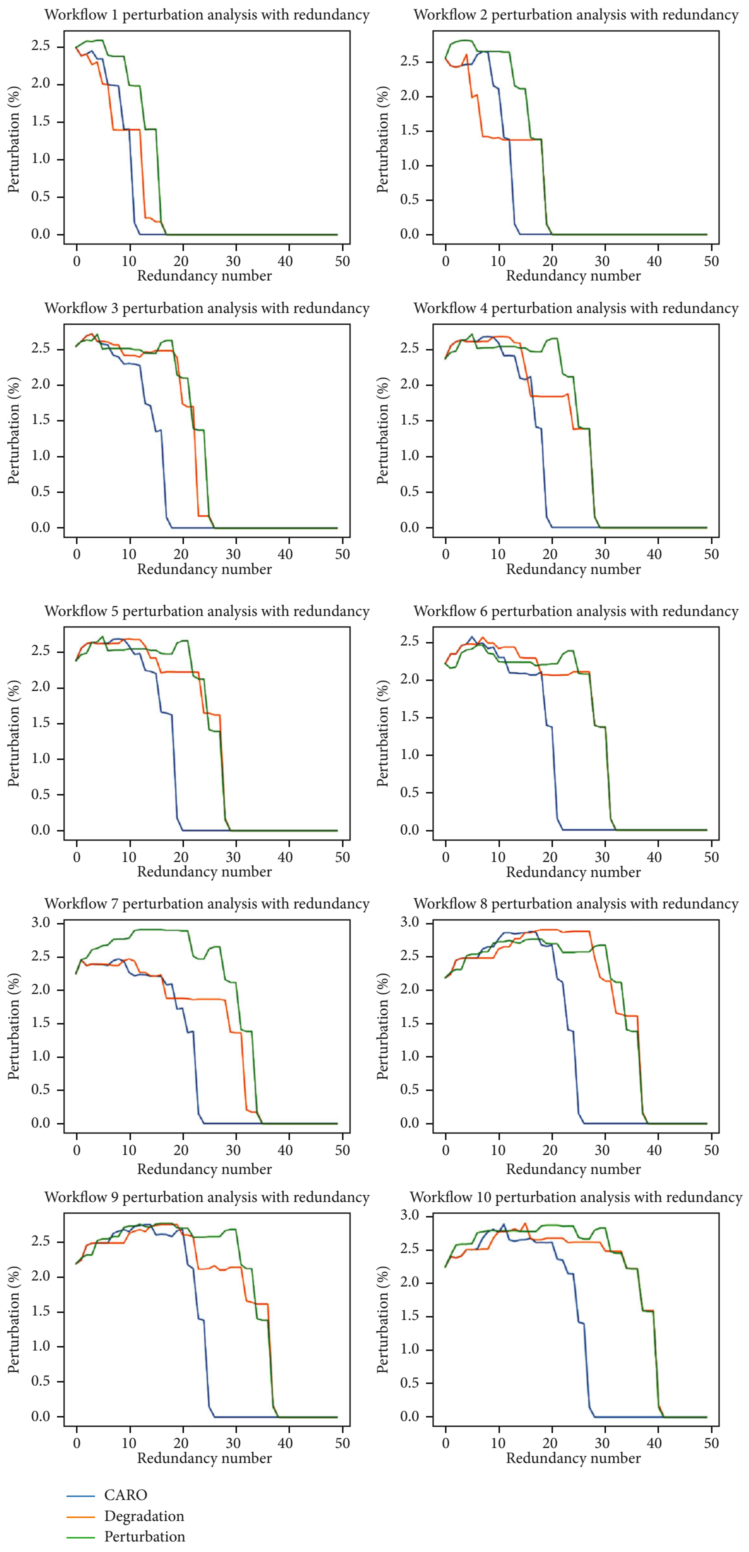


FIGURE 8: Perturbation analysis with various algorithms for ten workflows.

microservice composition is also improved. However, the ratio between them is uncertain in the short term considering the influence of timing.

Besides, our algorithm is designed to discuss the reliability fluctuation under the premise of high reliability. Therefore, it does not make sense if the microservice composition remains stable at low reliability. Therefore, with the improvement of redundancy operation, it is expected that the overall reliability will be improved, and the fluctuation will tend to be stable.

6.3. Algorithm Comparison and Result Analysis. The emphasis of this section is to compare the influence of different redundancy strategies. We use the CARO algorithm without considering degradation and perturbation, selecting three algorithms for comparison by simulating ten workflows.

The algorithm without considering degradation is only for microservice disturbance redundancy. Therefore, we optimize reliability, as shown in the high-reliability condition. In Figure 7, the redundancy helps improve reliability, but it will take a long time to achieve high reliability if only for the fluctuation of the redundancy operation. The iterative steps will be wasted on maintaining the stability of microservice composition and ignoring the reliability parameters.

Figure 7 shows that the proposed CARO algorithm performs better than the others. The microservice composition reliability can achieve a higher value because of the ranking process. For each iteration, the microservices with the top reliability value are replicated. To improve the effectiveness of the FTCARE strategy, we distinguish the levels so that the application reliability is not confused with perturbation at a low level.

The CARO algorithm with ranking selection performs not as well as the other two algorithms but can also achieve high reliability with minor vibration by increasing the redundancy number finally.

Comparing our ten sets of trials, each set of data fits the rules of the above analysis. Hence, there is a wide range of applicability, not for an individual microservice composition. From the diagram data, we can see that as the number of microservices increases and their complexity increases, it takes longer to reach an expected state.

Figure 8 compares the CARO algorithm perturbation performance with degradation and perturbation analysis for workflow simulations. Again, the CARO algorithm can reach the lowest perturbation much faster than the other two.

The reliability perturbation reduces not continuously because it offers dynamic vibration during execution time. For example, the reliability of the ith microservice at $\Delta T(m-1)$, $\Delta T(m)$, and $\Delta T(m+1)$ time intervals are 60%, 60%, and 65%, and the maximum perturbation is 5%. However, in the redundancy process, the reliability is 90%, 80%, and 95%, and the perturbation is 10%, which means the reliability perturbation performs terribly at this time point. However, with the improvement of reliability, the vibration space is narrower. For example, if overall reliability achieves more than 95%, the maximum reliability perturbation is only 5% because it cannot exceed 100% in reality.

Besides that, we can find that the impact of redundancy optimization strategy on stability is more prominent with the growth of the number of microservices and the improvement of schema complexity. For workflow 10, the earlier stability of microservice composition is achieved by using fewer redundancies, and the optimization efficiency is much higher. In workflow 1, the efficiency difference of the three algorithms is not particularly obvious. As the number of microservices increases, the amount of redundancies required also increases to achieve the stability goal.

Similarly, ten trials found that, as the number of microservices and the complexity of microservice composition structure increases, it takes longer to reach a stable state. In other words, with the improvement of workflow complexity, the efficiency of our optimization CARO algorithm is higher.

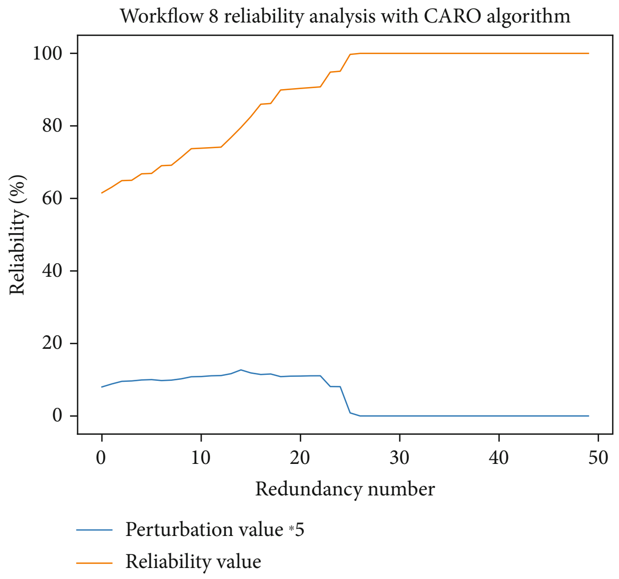


Figure 9: Analysis of the expected reliability r_{MC}^{E} for workflow 8.

Our proposed CARO algorithm takes high reliability as a priority and fluctuation as the second optimization factor. Therefore, the fluctuation is relatively large in the early stage of the optimization disturbance process, reaching an expected state. In contrast, without degradation, which takes the influence of optimization fluctuation as the only factor, it has high efficiency of disturbance optimization but may produce low reliability for a while.

As we analyzed, in low-reliability, degradation ranking is used to update the redundancy scheme. Thus, the redundancy operation improves the microservice composition reliability on one side. On the other side, it reduces or narrows the degradation. Both effects help speed up the optimization effectiveness, and after that, perturbation plays a role instead.

6.4. Parameter Analysis. We have used some variables in the calculation formula to analyze the reliability of cloud applications based on microservices in Section 5. In the following part, we will discuss the influence of these variables on reliability optimization.

CARO optimization focuses on steady improvement by computing the reliability of cloud applications. We have defined the variables of expected reliability r_{MC}^{E}. When the reliability is lower than the threshold $r_{\mathrm{MC}} < r_{\mathrm{MC}}^{E}$, the main objective of the optimization is to improve the reliability of the application. When the reliability reaches the threshold $r_{\mathrm{MC}} \geq r_{\mathrm{MC}}^{E}$, the purpose of optimization is to eliminate the fluctuation. Therefore, we take workflow eight as an example to verify whether the optimization goal is achieved. In order to make the comparison more apparent, the fluctuation of reliability is expressed by multiplying its value by five. In Figure 9, the threshold of cloud applications is 80%. When the reliability value is

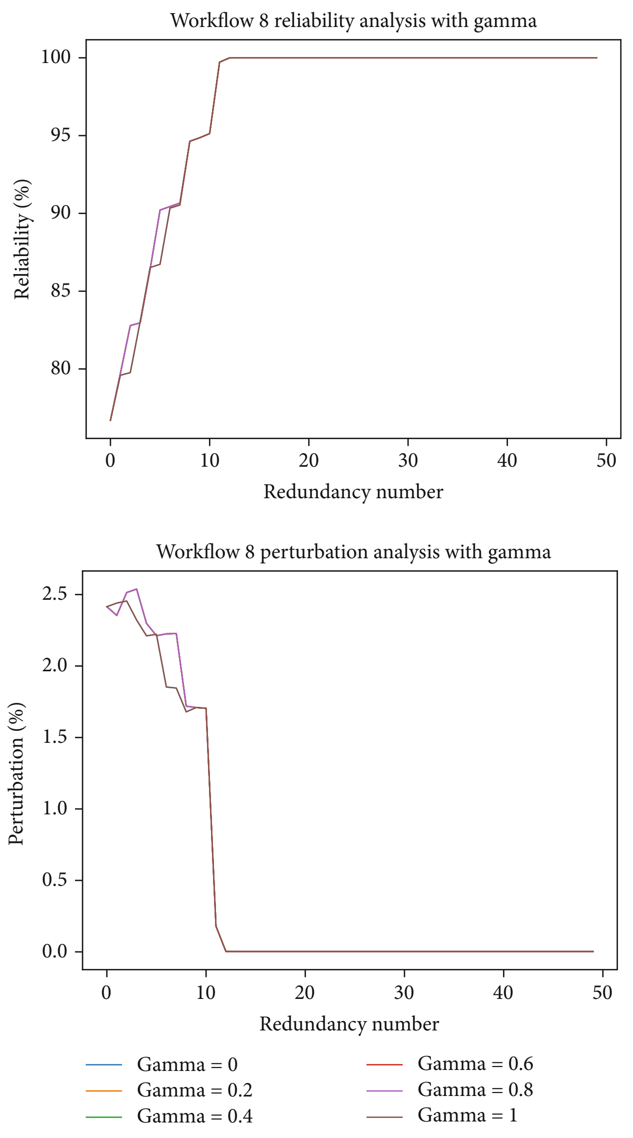


FIGURE 10: Reliability and perturbation analysis of parameter γ for workflow 8.

lower than it, the overall reliability is improved, but the fluctuation is relatively high. When the reliability reaches 80%, the overall reliability is improved, but the fluctuation also tends to be gentle.

We can find that other workflow mentioned in this paper can also meet this optimization goal through our experimental comparison. Similarly, when we modify the expected value of reliability r_{MC}^{E}, the figures of reliability analysis follow the same rule. However, because of the length of the article, we do not demonstrate all the pictures here. In the practical MOCA applications, both positive and negative fluctuations of reliability are not expected. However, the positive impact of microservice reliability on service composition is pursued, while the negative impact is suppressed through a reliability improvement strategy. That is where we optimize the difference between reliability degradation and fluctuation. The goal of fluctuation is to be stable, and the goal of degradation is to enhance.

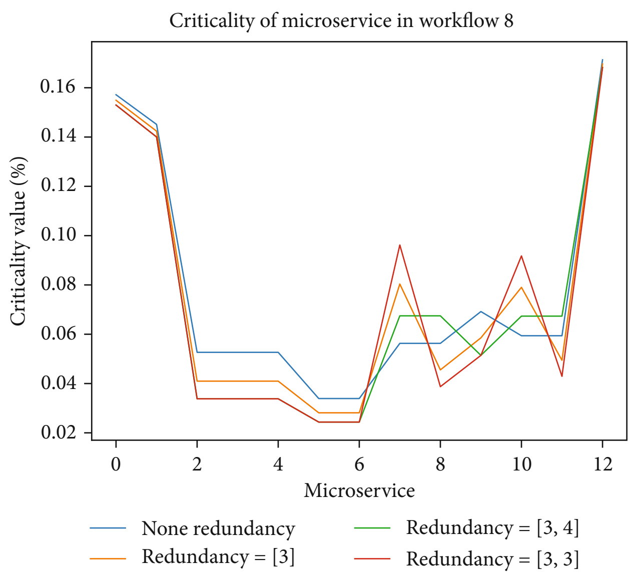


FIGURE 11: Experimental result with critical value with and without redundancy.

When computing the reliability of cloud services, the parameter γ divides the importance and degradation of microservices. With experience, we take γ as a number from 0 to 1. On the one hand, we need to consider the importance of a microservice in microservice composition schema. On the other hand, the impact of microservice reliability on microservice composition also needs to be studied. Therefore, the value of γ needs to be discussed.

In order to analyze the influence of γ on reliability optimization, we use workflow eight as a simulation example. When $\gamma = 1$, the impact of reliability degradation on microservice composition is considered first. When $\gamma = 0$, the criticality of a microservice in microservice composition plays a decisive role. The value of γ between 0 and 1 indicates the trade-off between the both indicators.

We can see from Figure 10 that the value of γ does not affect the reliability and fluctuation. In other words, if reliability degradation has a significant impact on microservice composition, it is equally essential in the schema. Thus, we can use parameters $rd_{(i,\mathrm{MC})}$ and V'_i to analyze the reliability characteristics when the threshold is at a low level identically.

Finally, the critical value of a microservice after updating redundancy schema is influenced. Here, we take an example with workflow 8. In Figure 11, none redundancy means all the microservices in the workflow is without redundancy. Redundancy = [3] and Redundancy = [3, 3] mean the third microservice has once and twice the redundancy. Redundancy = [3, 4] means the third and the fourth microservice in the workflow has redundancy.

As shown in Figure 11, with redundancy, the importance of microservices in service composition decreases. On the one hand, the objective of the optimization is to reduce the importance of a microservice. On the other hand, its objective is to reduce the negative impact of a microservice on microservice composition. From this simulation, we can conclude that the importance of a microservice decreases with the increase of redundancy.

7. Conclusion

In the cloud computing environment, the reliability dynamically occurs at runtime, which also has a dynamic impact on the whole application. It leads to the concept drift of probability distribution of reliability time series of a microservice. The evolution of the reliability time series for a microservice follows a continuous time-homogeneous 1st-order Markov Chain evolution rule.

The fluctuation of cloud applications at a time series is uncertain at neighboring time points. So, the effective reliability fluctuation measurement method must evaluate the impact of the uncertainty.

This paper proposes a ranking-based framework to build fault-tolerant cloud applications.

(i) We first model the components in MOCA and cloud application reliability requirement with PrT net. In order to identify the critical microservices, an improved PageRank algorithm has been proposed with request frequency and microservice relation through three steps, i.e., graph building, criticality ranking, and microservice determination

(ii) Then, the multiple temporal transitions of microservice reliability influence the microservice composition in the cloud environment, and the perturbation function describes the single-step transition of microservice reliability in a cloud application. The reliability degradation of a microservice may pose a more significant potential threat to the cloud application. The impact of microservice failures on the MOCA reliability is measured

(iii) Finally, we present a novel FTCARE strategy to determine the most suitable redundancy for a microservice after analyzing the perturbation and degradation properties of a microservice. The microservice reliability properties in microservice composition are iteratively updated by ranking after redundancy. The reliability of the redundancy scheme is the probability that at least one microservice completes successfully

We plan to further expand our work, mainly around the following two directions. Firstly, different fault-tolerant strategies are studied so that each key microservice can be deployed more suitably. Secondly, the microservice based on environmental diversity is combined with the reliability prediction method to improve effectiveness further.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

This work is partially supported by the NSF of China under no. 61772200.

References

[1] A. Samanta and J. Tang, "Dyme: dynamic microservice scheduling in edge computing enabled IoT," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6164–6174, 2020.

[2] G. Xie, Y. H. Wei, Y. Le, and R. Li, "Redundancy minimization and cost reduction for workflows with reliability requirements in cloud-based services," *IEEE Transactions on Cloud Computing*, vol. 4, no. 8, pp. 2351–2369, 2020.

[3] Y. Yin, Z. Cao, Y. Xu, H. Gao, R. Li, and Z. Mai, "QoS prediction for service recommendation with features learning in mobile edge computing environment," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 4, pp. 1136–1145, 2020.

[4] Y. Z. Huang, H. H. Xu, H. H. Gao, and W. Hussain, "SSUR: an approach to optimizing virtual machine allocation strategy based on user requirements for cloud data center," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 2, pp. 670–681, 2021.

[5] D. M. Vincenzo and K. Dragi, "Multi-objective scheduling of extreme data scientific workflows in Fog," *Future Generation Computer Systems*, vol. 106, pp. 171–184, 2020.

[6] C. Clab, A. Ms, Z. B. Min, and Y. L. Luo, "Effective replica management for improving reliability and availability in edge-cloud computing environment," *Journal of Parallel and Distributed Computing*, vol. 143, pp. 107–128, 2020.

[7] R. Florin, A. Ghazizadeh, P. Ghazizadeh, S. Olariu, and D. C. Marinescu, "Enhancing reliability and availability through redundancy in vehicular clouds," *IEEE Transactions on Cloud Computing*, vol. 15, no. 4, pp. 2654–2674, 2019.

[8] H. Chen, X. Zhu, G. Liu, and W. Pedrycz, "Uncertainty-aware online scheduling for real-time workflows in cloud service environment," *IEEE Transactions on Services Computing*, vol. 4, no. 5, pp. 1311–1334, 2019.

[9] N. Kherraf, S. Sharafeddine, C. M. Assi, and A. Ghrayeb, "Latency and reliability-aware workload assignment in IoT networks with mobile edge clouds," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1435–1449, 2019.

[10] H. Gao, W. Huang, and Y. Duan, "The cloud-edge based dynamic reconfiguration to service workflow for mobile ecommerce environments: a QoS prediction perspective," *ACM Transactions on Internet Technology*, vol. 13, no. 4, pp. 3469–3484, 2021.

[11] D. A. Firas and A. M. Mazlina, "Towards agent-based petri net decision making modelling for cloud service composition: a literature survey," *Journal of Network and Computer Applications*, vol. 130, pp. 14–38, 2019.

[12] W. Ha, "Reliability prediction for Web service composition," in *2017 13th International Conference on Computational Intelligence and Security (CIS)*, pp. 570–573, Hong Kong, China, 2017.

[13] G. S. Fan, H. Q. Yu, L. Q. Chen, and D. M. Liu, “Petri net based techniques for constructing reliable service composition,” *Journal of Systems and Software*, vol. 86, no. 4, pp. 1089–1106, 2013.

[14] J. Huang, J. Liang, and S. Ali, “A simulation-based optimization approach for reliability-aware service composition in edge computing,” *IEEE Access*, vol. 8, pp. 50355–50366, 2020.

[15] Z. G. Zang, Q. Wen, and K. Xu, “A fault tree based microservice reliability evaluation model,” *IOP Conference Series: Materials Science and Engineering*, vol. 569, pp. 032–069, 2019.

[16] R. W. Xiao, Z. W. Wu, and D. Y. Wang, “A finite-state-machine model driven service composition architecture for internet of things rapid prototyping,” *Future Generation Computer Systems*, vol. 99, pp. 473–488, 2019.

[17] R. M. Pathan and J. Jonsson, “FTGS: fault-tolerant fixed-priority scheduling on multiprocessors,” in *2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 1164–1175, Changsha, China, 2011.

[18] G. Yao, Y. Ding, and K. Hao, “Using imbalance characteristic for fault-tolerant workflow scheduling in cloud systems,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 12, pp. 3671–3683, 2017.

[19] L. P. Zhao, Y. Ren, and K. Sakurai, “Reliable workflow scheduling with less resource redundancy,” *Parallel Computing*, vol. 39, no. 10, pp. 567–585, 2013.

[20] A. R. Setlur, S. J. Nirmala, H. S. Singh, and S. Khoriya, “An efficient fault tolerant workflow scheduling approach using replication heuristics and checkpointing in the cloud,” *Journal of Parallel and Distributed Computing*, vol. 136, pp. 14–28, 2020.

[21] N. Kumar, J. Mayank, and A. Mondal, “Reliability aware energy optimized scheduling of non-preemptive periodic real-time tasks on heterogeneous multiprocessor system,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 4, pp. 871–885, 2020.

[22] S. Safari, M. Ansari, G. Ershadi, and S. Hessabi, “On the scheduling of energy-aware fault-tolerant mixed-criticality multicore systems with service guarantee exploration,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 10, pp. 2338–2354, 2019.

[23] H. Gao, C. Liu, Y. Li, and X. Yang, “V2VR: reliable hybrid-network-oriented V2V data transmission and routing considering RSUs and connectivity probability,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3533–3546, 2021.

[24] A. Sharif, M. Nickray, and A. Shahidinejad, “Energy-efficient fault-tolerant scheduling in a fog-based smart monitoring application,” *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 36, no. 1, 2020.

[25] J. Yao, Q. Lu, H. Jacobsen, and H. Guan, “Robust multi-resource allocation with demand uncertainties in cloud scheduler,” in *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, pp. 34–43, Hong Kong, China, 2017.

[26] B. Ray, A. Saha, S. Khatua, and S. Roy, “Proactive fault-tolerance technique to enhance reliability of cloud service in cloud federation environment,” *IEEE Transactions on Cloud Computing*, vol. 6, no. 8, pp. 2247–2264, 2020.

[27] G. Fan, L. Chen, H. Yu, and D. Liu, “Modeling and analyzing dynamic fault-tolerant strategy for deadline constrained task scheduling in cloud computing,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 4, pp. 1260–1274, 2020.

[28] T. Shi, H. Ma, and G. Chen, “A genetic-based approach to location-aware cloud service brokering in multi-cloud environment,” in *2019 IEEE International Conference on Services Computing (SCC)*, pp. 146–153, Milan, Italy, 2019.

[29] L. Wang, Q. He, D. Gao, J. Wan, and Y. Zhang, “Temporal-perturbation aware reliability sensitivity measurement for adaptive cloud service selection,” *IEEE Transactions on Services Computing*, vol. 16, no. 2, pp. 1254–1279, 2020.

[30] L. Wang, “Architecture-based reliability-sensitive criticality measure for fault-tolerance cloud applications,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 11, pp. 2408–2421, 2019.

[31] Z. Zheng, T. C. Zhou, M. R. Lyu, and I. King, “FTCloud: a component ranking framework for fault-tolerant cloud applications,” in *2010 IEEE 21st International Symposium on Software Reliability Engineering*, pp. 398–407, San Jose, CA, USA, 2010.

[32] Z. Zheng, T. C. Zhou, M. R. Lyu, and I. King, “Component ranking for fault-tolerant cloud applications,” *IEEE Transactions on Services Computing*, vol. 5, no. 4, pp. 540–550, 2012.

[33] K. Inoue, R. Yokomori, T. Yamamoto, M. Matsushita, and S. Kusumoto, “Ranking significance of software components based on use relations,” *IEEE Transactions on Software Engineering*, vol. 31, no. 3, pp. 213–225, 2005.

[34] T. Shi, H. Ma, G. Chen, and S. Hartmann, “Location-aware and budget-constrained application replication and deployment in multi-cloud environment,” in *2020 IEEE International Conference on Web Services (ICWS)*, pp. 110–117, Beijing, China, 2020.

[35] T. Shi, H. Ma, G. Chen, and S. Hartmann, “Location-aware and budget-constrained service deployment for composite applications in multi-cloud environment,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 8, pp. 1954–1969, 2020.

[36] Z. Zheng, Y. Zhang, and M. R. Lyu, “Cloudrank: a QoS-driven component ranking framework for cloud computing,” in *2010 29th IEEE Symposium on Reliable Distributed Systems*, pp. 184–193, New Delhi, India, 2010.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 6378139, 14 pages
https://doi.org/10.1155/2021/6378139

Research Article

A Combining Method for Wireless Protocol Conformance Testing: A Empirical Case

Lin Wei-Wei,[1,2] Zeng Hong-Wei,[1] and Jung Yoon Kim[3]

[1]*School of Computer Engineering and Science, Shanghai University, Shanghai, China*
[2]*Shanghai Key Laboratory of Computer Software Evaluating & Testing, Shanghai, China*
[3]*Graduate School of Game, Gachon University, Seongnam, Republic of Korea*

Correspondence should be addressed to Lin Wei-Wei; newstart@shu.edu.cn and Jung Yoon Kim; kjyoon@gachon.ac.kr

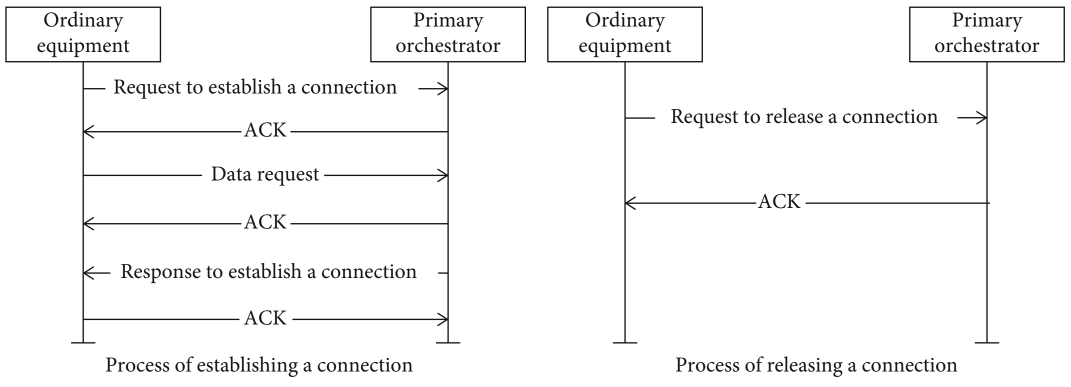
Received 29 May 2021; Revised 6 June 2021; Accepted 18 June 2021; Published 15 July 2021

Academic Editor: Xiaoxian Yang

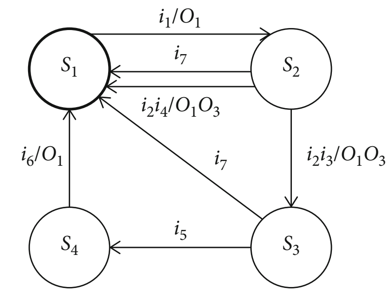
Ensuring the consistency of protocol implementation and protocol specification is the basic premise of wireless communication. With the application of wireless communication in more and more fields, the wireless communication environment becomes more and more complex, and the fault coverage of wireless protocol conformance testing is also facing more and more challenges. To solve this problem, this paper uses Finite State Machine (FSM) as a formal description tool for wireless protocols and presents a combining test method based on two test methods with complementary characteristics in the test technologies based on structural coverage and state identification. Then, the paper evaluates the effectiveness of the method based on 14 empirical cases. The experimental results show that the fault coverage of each empirical case can be effectively improved to 100% when the average test cost is only increased by 17.99%.

1. Introduction

Wireless protocol is the rule that must be obeyed by the communication entities and processes in wireless communication [1]. It is also the basic guarantee to realize the stability of wireless communication and the competitive advantage in the low-power wireless communication market [2, 3]. Conformance testing, performance testing, interoperability testing, and robustness testing are all essential to verify the correctness of wireless protocol implementation [4]. Conformance testing is the basis of the other three kinds of testing, which is used to ensure the consistency of protocol implementation and protocol specification [1, 5, 6]. Finite State Machine (FSM) [7–10] is an effective formal modeling tool [11, 12] in wireless protocol conformance testing since the operations implemented on the entities in wireless protocols (such as button click or window input in GUI and message request in Web service) can be mapped to the inputs in FSMs, and the responses to the operations according to the wireless protocols (such as message response in GUI or web service) can be mapped to the outputs in FSMs [13–16]. To facilitate the description, we abstract these events into alphabetic or numeric symbols in the examples and experiments of this paper. ZigBee is an emerging short-range, low-rate, and low-power wireless network technology. Taking the connection and release process of MAC layer in ZigBee protocol as an example [17], the interaction process between communication entities is shown in Figure 1(a) and the corresponding FSM is shown in Figure 1(b). The relationships between specific events in ZigBee protocol and abstract symbol definitions in FSM are described in Tables 1, 2, and 3, respectively. In this way, the problem of judging whether the implementation of ZigBee protocol is consistent with the ZigBee protocol specification can be transformed into the problem of judging whether the implementation of ZigBee protocol is consistent with the specification FSM of ZigBee protocol. Specifically, a test set is generated based on the specification FSM and the test set is applied to the wireless protocol implementation. When the actual output is consistent with the expected output, the implementation is consistent with the protocol. Otherwise, it means that there is a fault in the implementation.


Ordinary equipment
Primary orchestrator
Request to establish a connection
ACK
Data request
ACK
Response to establish a connection
ACK
Process of establishing a connection
Ordinary equipment
Primary orchestrator
Request to release a connection
ACK
Process of releasing a connection

(a)



(b)

FIGURE 1: Connection and release process in ZigBee.

TABLE 1: Abstract definition of state.

State signal	State
s_1	Unconnected state (initial state)
s_2	Received connection request data frame and waiting for data request command frame
s_3	Agree to the connection request, and wait for the ACK
s_4	Connection status

TABLE 2: Abstract definition of input event.

Input signal	Input event
i_1	Connection request frame
i_2	Data request frame
i_3	Normal network status and network resources are not exhausted
i_4	Malicious connection request frames, exhausting network resources
i_5	ACK
i_6	Release request frame
i_7	No input data, waiting timeout

Both the transition-pair coverage (TPC) method [18–20] based on structure coverage and the unique input output (UIO) method [7, 13] based on state identification can be used in wireless protocol conformance testing. However, neither TPC method nor UIO method can achieve 100% fault coverage. To meet the high reliability requirements of wireless protocol conformance testing, this paper presents a combined method based on the complementary characteristics of the TPC method and UIO method. And the combined method is studied by 14 established empirical cases including ZigBee protocol and the WM2RP protocol both of which are widely used in wireless communication. The contribution of this paper is mainly as follows:

TABLE 3: Abstract definition of output event.

Output signal	Output event
o_1	ACK
o_2	Response data frame of allowing nodes to join the network
o_3	Response data frame of refusing nodes to join the network

(1) The fault coverage of wireless protocol conformance testing is effectively improved

(2) Since the experimental objects are all FSM empirical cases designed and generated by testers according to professional experience and domain knowledge, the experimental data and results have more practical significance, which can also promote the application of FSM conformance testing in industry

The remainder of the paper is organized as follows. Section 2 introduces the concepts and terms about FSM. Section 3 presents a combined method for wireless protocol conformance testing. Section 4 describes the case study and evaluates the experimental results. Section 5 overviews the

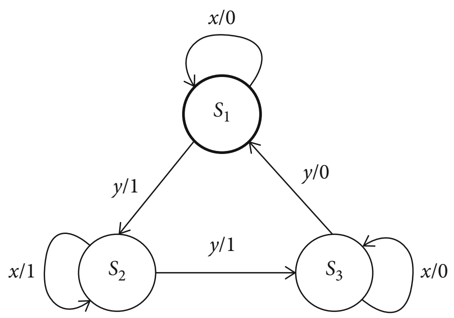


Figure 2: FSM M_1.

related work. Finally, Section 6 summarizes the full text and prospects of the future work.

2. Preliminaries

An FSM is formally defined as a 6-tuple $M = (I, O, S, s_0, \delta, \lambda)$ [21, 22] where

(i) I and O are finite and nonempty sets of input symbols and output symbols, respectively

(ii) S is a finite and nonempty set of states

(iii) $s_0 \in S$ represents the initial state of M

(iv) $\delta : S \times I \longrightarrow S$ denotes the state transition function

(v) $\lambda : S \times I \longrightarrow O$ is the output function

According to this definition, when an input symbol i is delivered to the current state s, M moves to state $s' = \delta(s, i)$ with an output produced by $\lambda(s, i)$. A transition t is defined by a tuple $(s, s', i/o)$ where s is the start state of t, i is an input of s, $o = \lambda(s, i)$ is the associated output, and $s' = \delta(s, i)$ is the end state of t.

An FSM M is initially connected if for any state s there is at least one input sequence from s_0 to s, i.e., $\forall s \in S, \exists \alpha \in I^* \cdot (\delta(s_0, \alpha) = s)$ (where I^* is the set of finite sequences of input symbols).

If $s \neq s', \alpha \in I^* \cdot (\lambda(s, \alpha) \neq \lambda(s', \alpha))$, then s and s' are a pair of distinguishable states, and α is called a separating sequence of s and s'. If all the states in FSM M are pairwise distinguishable, i.e., $\forall s, s' \in S, s \neq s', \exists \alpha \in I^* \cdot (\lambda(s, \alpha) \neq \lambda(s', \alpha))$, then M is said to be minimal or reduced.

An FSM M is deterministic if at any state s and for any input i, there is at most one transition leading to the next state. Otherwise, M is nondeterministic.

An FSM is not initially connected means there are some states that cannot be reached from the initial state. In other words, the states that cannot be reached from the initial state are meaningless to the FSM. Therefore, an FSM that is not initially connected can be converted to an initially connected FSM by deleting the states that cannot be reached from the initial state. An FSM with equivalent states can be transformed into a reduced FSM by joining equivalent states. A nondeterministic FSM can be converted to a deterministic FSM by combining states, i.e., raising the level of abstraction. Since all the above definitions can be satisfied, only initially connected, reduced and deterministic FSMs are considered in this paper. In Figure 2, M_1 is an initially connected, reduced, and deterministic FSM where $S = \{s_1, s_2, s_3\}, I = \{x, y\}, O = \{0, 1\}$ and s_1 highlighted in bold defaults to the initial state.

The connection of input sequences in FSM is represented by the juxtaposition of input sequences; that is, if $\alpha_1, \alpha_2 \in I^*$, then $\alpha_1\alpha_2$ represents the connection of α_1 and α_2. If $\beta = \alpha_1\alpha_2$, then α_1 is the prefix of β, expressed as $\alpha_1 \leq \beta$. If $\alpha_1 \neq \varepsilon$ (ε is an empty string), then α_1 is the true prefix of β, expressed as $\alpha_1 < \beta$. Given a set of input sequences T, pref(T) represents the set of all prefixes of all sequences in T; that is, pref(T) $= \{\alpha | \exists\beta \in T \text{ and } \alpha \leq \beta\}$; if $T = \text{pref}(T)$ then, T is a set of prefix closures.

In this work, we assume that FSM M has a reset operation r which can bring M to its initial state from any state without any output. A string $x_1x_2, x_{k\in}I^*$ is said to be a defined input sequence at $s \in S$ if there exists $s_1, \cdots, s_{k,}s_{k+1}$, where $s_1 = s$, such that $(s_i, x_i) \in (S \times I)$ and $\delta(s_i, x_i) = s_{i+1}$ for all $i = 1, \cdots, k$. We use $\Omega_M(s)$ to denote the set of all defined input sequences at state s and Ω_M as an abbreviation for $\Omega_M(s_0)$, i.e., the input sequences defined at the initial state of M, and hence for M itself. A test case of M is an input sequence $\alpha \in \Omega_M$ starting with r. A test suite of M is a finite set of test cases, and there are not two test cases α and β such that $\alpha < \beta$. The number of symbols of a sequence α is represented by $|\alpha|$. For instance, given a test case *ryyy* for M_1, the length of *ryyy* is 4, since it contains four symbols, that is, $|ryyy| = 4$. This notation is extended to a test suite T; i.e., $|T|$ means the sum of all the length of sequences in T. For instance, given a test suite $T = \{ryx, ryyy, ryyx, rx\}$ for M_1, the length of T is 13, i.e., $|T| = |\{ryx, ryyy, ryyx, rx\}| = 13$.

An initialization fault occurs when FSM starts from an incorrect initial state s_i ($s_i \neq s_0$). An output fault occurs when the correct output of a transition $(s, s', i/o)$ is changed, and a transition $(s, s', i/o')$ in which the output is different from the expected one is generated. An end state fault occurs when the end state of a transition $(s, s', i/o)$ is changed, and a transition $(s, s'', i/o)$ in which the end state is different from the expected one is generated. Figure 3 demonstrates the three types of faults based on M_1. There is an initialization fault starting from an incorrect initial state s_2 in Figure 3(a). There is an output fault in Figure 3(b). The transition $(s_3, s_1, y/0)$ is changed to the transition $(s_3, s_1, y/1)$ with a wrong output. There is an end state fault in Figure 3(c). The transition $(s_1, s_1, x/0)$ is changed to the transition $(s_1, s_2, x/0)$ with a wrong end state.

3. TPC-UIO Test Method

3.1. Motivation Example. Figures 4 and 5 are two examples to present the complementarity between the TPC method and UIO method.

The FSM in Figure 4(a) is a wireless protocol specification, and the FSM in Figure 4(b) is an implementation, where the dotted line represents a transition with an end state fault. $\{rcbbdb$, $rcbbecda$, $rcbbecbbac\}$ is the test set of TPC method

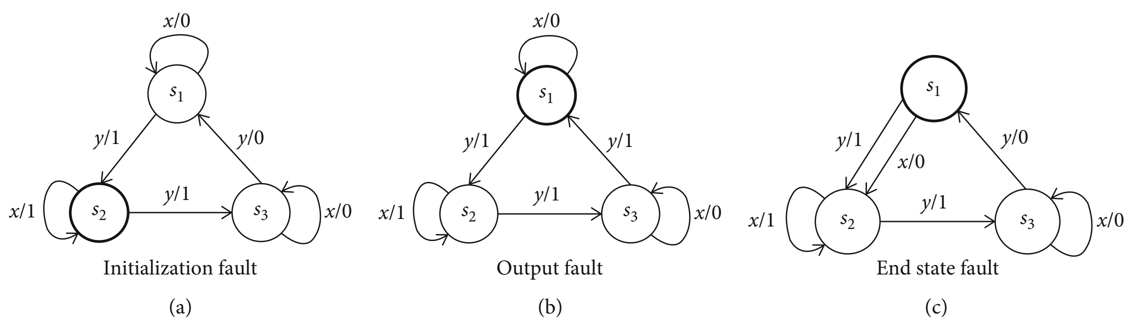


Figure 3: Three types of faults in M_1.

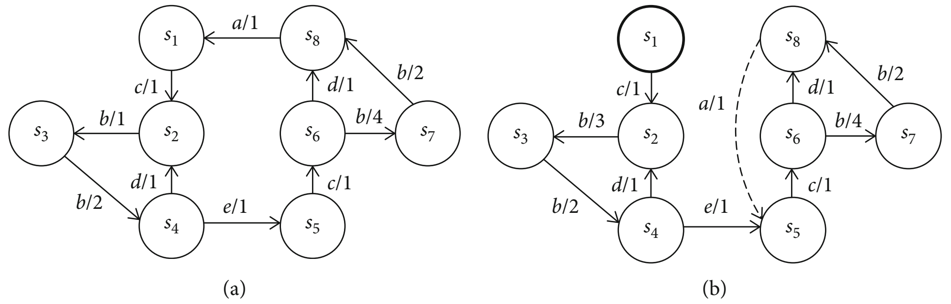


Figure 4: Example 1 of relationship between the TPC method and UIO method.

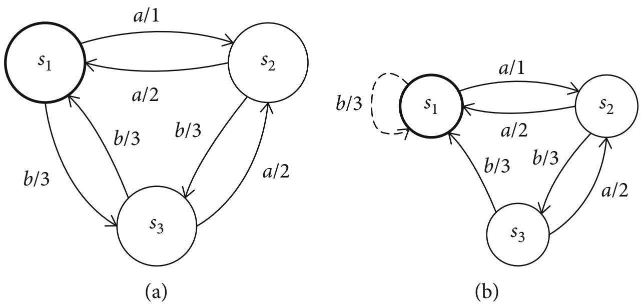


Figure 5: Example 2 of relationship between the TPC method and UIO method.

generated based on Figure 4(a), and {*rcbba*, *rcbbdb*, *rcbbecb-bacb*, *rcbbecda*} is the test set of UIO method generated based on Figure 4(a).

The FSM in Figure 5(a) is a wireless protocol specification, and the FSM in Figure 5(b) is an implementation, where the dotted line also represents a transition with an end state fault. {*rba*, *rbb*, *raaa*, *raab*, *rabaa*, *rabab*, *rabba*, *rabbb*} is the test set of the TPC method generated based on Figure 5(a), and {*raaa*, *rbba*, *rabba*, *rabaaa*} is the test set of the UIO method generated based on Figure 5(a).

Test results indicate the following:

(1) The test set of the TPC method can test the fault in Figure 5(b) but cannot test the fault in Figure 4(b)

(2) The test set of the UIO method can test the fault in Figure 4(b) but cannot test the fault in Figure 5(b)

It can be concluded from the above examples that there are obvious differences in the performance of TPC method and UIO method in different examples. Specifically, the two methods show strong complementarity. When the number of transitions outgoing from the correct end state and the wrong end state is equal, furthermore, the input/output of the transitions outgoing from the correct end state and the wrong end state are all the same, fault coverage ability of UIO method is superior to TPC method. When there are differences about the number of transitions outgoing from the correct end state and the wrong end state or there are differences about the input/output of transitions outgoing from the correct end state and the wrong end state and the UIO sequence of the correct end state is also available in the wrong end state, fault coverage ability of the TPC method is superior to that of the UIO method.

3.2. Test Generation of TPC-UIO Method. Based on the above analysis, we introduce a combined TPC-UIO test method to realize the complementarity of the TPC method and UIO method.

```
Input: FSM M with $n$ states $\{s_1, s_2,\dots s_n\}$ and $s_1$ is the initial state, transition tree T and Root(T)=$s_1$
Output: State covering set SC(M)
  1: Initialize SC(M)=$\phi$
  2: for each transition starting with $s_1$
  3: Add a branch in T and mark the branch with the input/output of the transition
  4: end for
  5: do
  6: for each leaf node appears for the first time in T
  7: Add a branch in T for each transition starting with the leaf node and mark the branch with the input/output of the transition
  8: end for
  9: until leaf node appears for the first time cannot be found in T
  10: for ($i$=1; $i \le n$; $i$++)
  11: Get an input/output sequence $\beta_i$ from $s_1$ to $s_i$ that appears for the first time in T
  12: SC(M)= SC(M)$\cup\beta_i$
  13: end for
```

Algorithm 1: State covering set generation

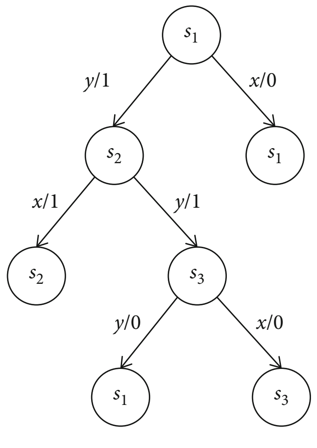


Figure 6: Transition tree of M_1.

3.2.1. State Covering Set Generation

Definition 1 (State covering set). Given an FSM M and an input sequence set $Q = \{\alpha_0, \alpha_1, \cdots, \alpha_{n-1}\}$, if for any state s_i, $\exists \alpha_i \in Q\ (0 \le i \le n-1)$ satisfies $\delta(s_0, \alpha_i) = s_i$, where s_0 is the initial state, then Q is a state covering set of M.

There must be a state covering set in initially connected FSM, and an empty sequence ε is included to reach the initial state s_0. In addition, only state covering set of prefix closure is considered in this paper. As described in Algorithm 1, given an FSM with n states $\{s_1, s_2, \cdots, s_n\}$ in which s_1 is the initial state and a transition tree T with only a root node s_1, state covering set SC(M) is first initialized to ϕ. Then, for each transition starting with s_1, a branch annotated with the input/output of the transition is added in T. Next, for each transition starting with a leaf node appears for the first time in T, a branch annotated with the input/output of the transition is added in T iteratively until there is no leaf node appearing for the first time that can be found. So far, the complete transition tree is constructed. Finally, for each state s_i $(1 \le i \le n)$, an input/output sequence β_i from s_1 to s_i that appears for the first time in T is obtained and β_i is thrown to SC(M).

According to Algorithm 1, the transition tree of M_1 is shown in Figure 6, and $\{\varepsilon, y/1, (y/1)(y/1)\}$ is a state covering set obtained from the transition tree.

3.2.2. TPC Test Set Generation

Definition 2 (adjacent transition). If the end state of a transition t_1=(s_1, s, i_1/o_1) is the start state of a transition t_2=(s, s_2, i_2/o_2), t_1 and t_2 are a pair of adjacent transition.

The test set of TPC method covers each pair of adjacent transitions in FSM at least once. As shown in Algorithm 2, given an FSM with n states $\{s_1, s_2, \cdots, s_n\}$ in which s_1 is the initial state and a state covering set SC$(M) = \{\alpha_1, \alpha_2, \cdots, \alpha_n\}$, TPC test set TPC$(M)$ is first initialized to ϕ. Then, for each state s_i $(1 \le i \le n)$, each transition sequence β of length 2 starting from s_i is obtained and $\alpha_i\beta$ is thrown to TPC(M).

Consider M_1 in Figure 2, {*rxx*, *rxy*, *ryy*, *ryx*, *ryxx*, *ryxy*, *ryyx*, *ryyy*, *ryyxx*, *ry*, *ryyxy*, *ryyyx*, *ryyyy*} is the test set of M_1 under the TPC method.

3.2.3. UIO Generation

Definition 3 (UIO sequence). An input/output sequence β is a UIO sequence of state s_i if and only if for any state $s_j \in S(s_j \ne s_i)$, $\lambda(s_i, \beta_{\text{in}}) \ne \lambda(s_j, \beta_{\text{in}})$, where β_{in} is the input part of β.

Definition 4 (uncertainty of the start states). Given an FSM M and an input sequence α, a partition $\pi(\alpha)$ of the state set S can be obtained according to the output generated by each state for α. For any state s_i and s_j, they will be in the same partition block if and only if they cannot be distinguished by α, i.e., $\lambda(s_i, \alpha) = \lambda(s_j, \alpha)$, and this partition is called the uncertainty of the start states.

Definition 5 (uncertainty of the current states). The current states of M after delivering α can be expressed as $\delta(B, \alpha) \mid B$

```
Input: FSM M with $n$ states $\{s_1, s_2,\dots s_n\}$ and $s_1$ is the initial state, state covering set SC(M)=$\{\alpha_1, \alpha_2,\dots\alpha_n\}$
Output: TPC test set TPC(M)
  1: Initialize TPC(M)=$\phi$
  2: for ($i$=1; $i\le n$; $i$++)
  3: for each transition sequence $\beta$ of length 2 starting from $s_i$
  4: TPC(M)= TPC(M) $\cup\alpha_i\beta$
  5: end for
  6: end for
```

Algorithm 2: TPC test set generation

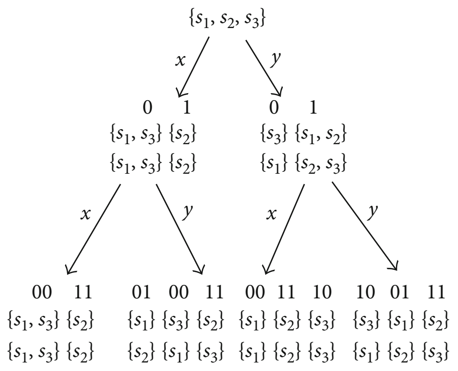


Figure 7: Successor tree of M_1.

$\in \pi(\alpha)$, and this partition is called the uncertainty of the current states.

In this way, the scope of the state before and after the input α can be learned according to the output corresponding to α. The idea of state partition can be applied to the construction of successor tree.

Definition 6 (successor tree). The successor tree of an FSM is a tree showing the behavior of the FSM at any state and upon any input sequence.

For each possible input sequence, there is a path starting from the root node in the successor tree and each node in the successor tree is annotated with the uncertainty of start states as well as the uncertainty of current states.

The successor tree of M_1 is shown in Figure 7. When x is delivered to the set of states $\{s_1, s_2, s_3\}$, according to the different output, the partition blocks of start state uncertainty are $0\{s_1, s_3\}$ and $1\{s_2\}$, respectively, where 0 and 1 represent the output generated by the state in the partition block. The corresponding partition blocks of current state uncertainty are $\{s_1, s_3\}$ and $\{s_2\}$, respectively. When x is again delivered to the current state uncertainty $\{s_1, s_3\}$ and $\{s_2\}$ according to the different output, the partition blocks of start state uncertainty are $00\{s_1, s_3\}$ and $11\{s_2\}$, respectively. The corresponding partition blocks of current state uncertainty are $\{s_1, s_3\}$ and $\{s_2\}$. Since the set $\{s_1, s_3\}$ can never be further divided by x, this branch can terminate the decomposition about input x. Next, y is delivered to the partition blocks of the current state uncertainty $\{s_1, s_3\}$ and$\{s_2\}$; according to the different output, the partition blocks of start state uncertainty are $01\{s_1\}$, $00\{s_3\}$, and $11\{s_2\}$, respectively. The corresponding current state uncertainty partition blocks are $\{s_2\}$, $\{s_1\}$, and $\{s_3\}$, respectively. Since there is only one element in each partition block, the division of this branch ends here. In the same way, we can construct other branches in the successor tree until a successor tree meeting the requirement is generated.

In this paper, UIO sequence is obtained by successor tree. Given an FSM and its successor tree, if there is only one state s_i in a partition block of the start state uncertainty, we will get a UIO sequence of s_i. Here, we only consider the shortest UIO sequence; that is, the UIO sequence we first obtained in the top-down search of the successor tree. As shown in Figure 7, the UIO sequence of s_1 is $(x/0)(y/1)$, $(y/1)(x/1)$, or $(y/1)(y/1)$, the UIO sequence of s_2 is $x/1$, and the UIO sequence of s_3 is $y/0$.

3.2.4. Test Set Generation of TPC-UIO Method

Definition 7 (TPC-UIO method). The test set of the TPC-UIO method should satisfy two conditions. (1) Cover each pair of adjacent transition in an FSM at least once. (2) Identify the end state of each pair of adjacent transition using the shortest UIO sequence.

As shown in Algorithm 3, given TPC test set TPC(M) of an FSM and UIO sequence of each state, for each test case in TPC(M), identify the end state of the test case by the corresponding UIO sequence. Then, the test cases which are true prefixes of others are deleted and TPC-UIO test set TPC-UIO (M) is constructed.

Taking M_1 for example, {*rxxxy*, *rxyx*, *ryxxx*, *ryxyy*, *ryyyxy*, *ryyxyxy*, *ryyyxxy*, *ryyyyx*} is the final test set of the TPC-UIO method.

4. Case Study

4.1. Experimental Subject. In existing literatures on FSM conformance testing, most of researchers conduct experiment using randomly generated FSMs. Generate FSMs randomly can obtain a large amount of FSMs in a short time, however, whether the experimental conclusions obtained from randomly generated FSMs are completely applicable on FSM empirical cases are still uncertain. This paper selects 14 established FSM empirical cases which are designed and generated by testers with professional experience and domain knowledge [17, 22–29]. The information of all FSM empirical cases

```
Input: TPC test set TPC(M), UIO sequence of each state
Output: TPC-UIO test set TPC-UIO(M)
  1: for each test case in TPC(M)
  2: Identify the end state of the test case by the corresponding UIO sequence
  3: end for
  4: delete the test cases which are true prefixes of others
```

Algorithm 3: Test set generation of TPC-UIO method

Table 4: Experimental subject.

FSM	$\|S\|$	$\|I\|$	$\|T\|$
Connection and release process of MAC layer in ZigBee protocol	4	7	7
Base station model in WM2RP protocol	3	3	3
Intermediate node model in WM2RP protocol	4	4	4
Leaf node model in WM2RP protocol	3	3	3
Railway signal system communication protocol	9	15	20
Conference protocol	5	11	28
INRES communication protocol	4	5	9
Transfer Protocol	4	10	18
ISDN BRI network layer protocol	8	13	25
NBS TP4 protocol	8	12	26
Alternating Bit Protocol	8	5	10
RPL host protocol	5	6	10
RPL router protocol	8	14	29
LLN Border Router Protocol	4	5	6

is shown in Table 4. $|S|$, $|I|$, and $|T|$ denote the number of states, the number of inputs, and the number of transitions, respectively.

4.2. Mutation Generation. In this paper, we use mutations of the specification to model faults in the implementation. Mutation operators [30] are introduced to obtain the three most typical faults in FSM conformance testing.

Definition 8 (change initial state (CIS)). This operator represents the initialization fault: change initial state s_0 to $s_k (s_k \neq s_0)$ such that the implementation starts from an incorrect initial state, that is, $\text{CIS}(M) = \{N = (I, O, S, s_k, \delta, \lambda) \mid s_k \in S \cdot s_k \neq s_0\}$.

Definition 9 (change output (CO)). This operator represents the output fault such that an unexpected output is produced in an implementation. Formally, $\text{CO}(M) = \{N = (I, O, S, s_0, \delta, \Lambda) \mid \exists x \in I, s_i \in S \cdot \Lambda(s_i, x) \neq \lambda(s_i, x)\}$.

Definition 10 (change end state (CES)). This operator represents the end state fault such that the implementation moves to an incorrect state. Formally, $\text{CES}(M) = \{N = (I, O, S, s_0, \Delta, \lambda) \mid \exists x \in I, s_i \in S \cdot \Delta(s_i, x) \neq \delta(s_i, x)\}$.

4.3. Experimental Setup. In order to improve the credibility of the experimental results, experimental replication is adopted in this paper, and the number of replications in each round is not the same. For every FSM empirical case, each type of mutation is randomly operated 10 times, 100 times, 1000 times, 10000 times, 20000 times, 30000 times, 40000 times, 50000 times, 60000 times, 70000 times, 80000 times, 90000 times, and 100000 times; that is, a total of three groups including 551110 mutations in each group are obtained.

4.4. Result Analysis

4.4.1. Comparison of Test Cost. The test set information of all FSM empirical cases is shown in Table 5. L represents the length of the test set and N represents the number of the test cases in the test set, that is, the number of reset operations.

(i) Only consider the length of the test set. For all FSM empirical cases, when considering the average length of the test sets of each test method, the average length of the UIO method is 59.64, the average length of the TPC method is 186.57, and the average length of the TPC-UIO method is 227.93. Figure 8 indicates that the length of the test set for TPC-UIO method is larger than that for the TPC method and UIO method for each empirical case. Compared with that of the TPC method, the average increase of the length of the TPC-UIO method is 24.32%

(ii) Only consider the number of the test cases. For all FSM empirical cases, when considering the average number of the test cases of each test method, the

Table 5: Collection of test set information.

FSM	UIO		TPC		TPC-UIO	
	L	*N*	*L*	*N*	*L*	*N*
Connection and release process of MAC layer in ZigBee protocol	19	4	19	4	23	4
Base station model in WM2RP protocol	5	1	5	1	6	1
Intermediate node model in WM2RP protocol	6	1	6	1	7	1
Leaf node model in WM2RP protocol	5	1	5	1	6	1
Railway signal system communication protocol	66	12	260	48	308	48
Conference protocol	135	25	762	138	922	142
INRES communication protocol	32	6	72	13	85	13
Transfer Protocol	70	16	204	45	249	45
ISDN BRI network layer protocol	98	18	223	41	264	41
NBS TP4 protocol	153	22	218	40	318	45
Alternating Bit Protocol	36	5	24	3	40	5
RPL host protocol	37	6	95	15	110	15
RPL router protocol	156	22	691	96	820	98
LLN Border Router Protocol	17	3	28	5	33	5

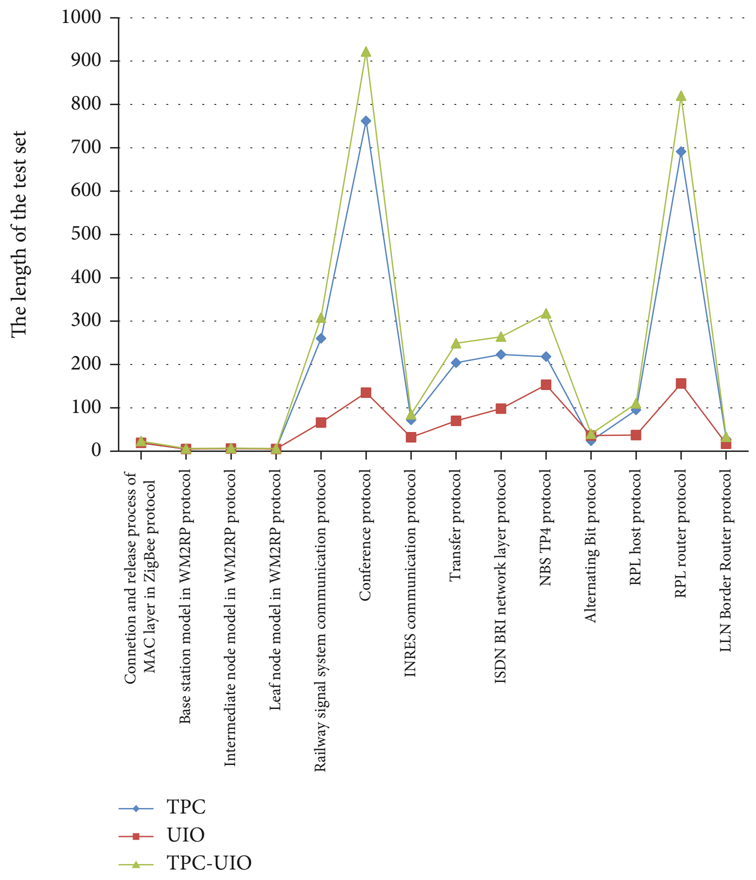


Figure 8: Relationship among the length of test sets for TPC, UIO, and TPC-UIO.

average number of the test cases for the UIO method is 9.93, the average number of the test cases for the TCP method is 32.07, and the average number of the test cases for the TPC-UIO method is 33.14. Figure 9 indicates that the number of the test cases for TPC-UIO method is more than or equal to that for the TPC method and UIO method for each empirical case. Compared with the TPC method, the average increase of the number of the test cases of the TPC-UIO method is 6.43%

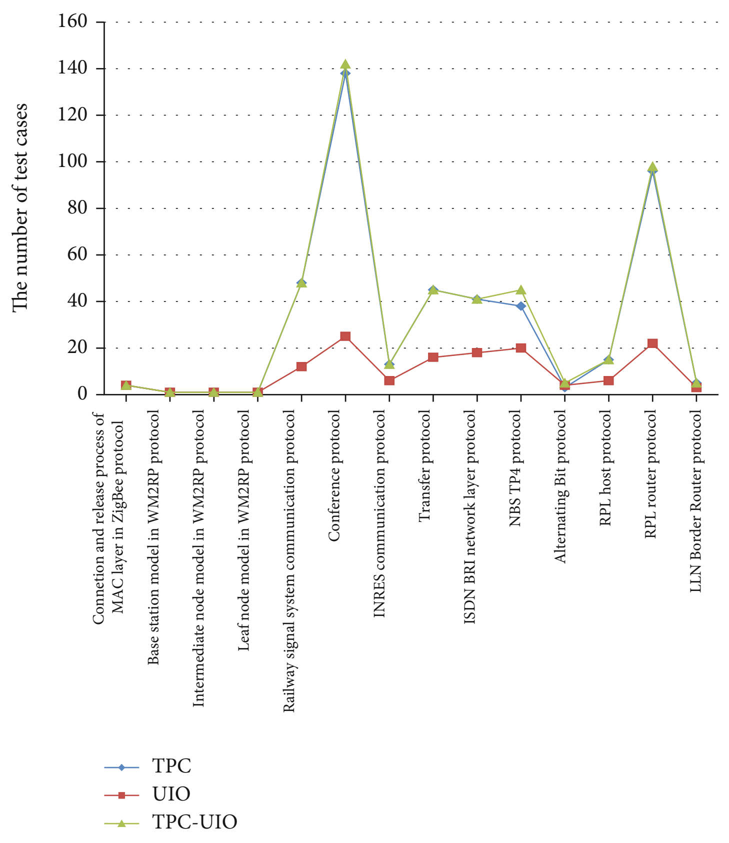
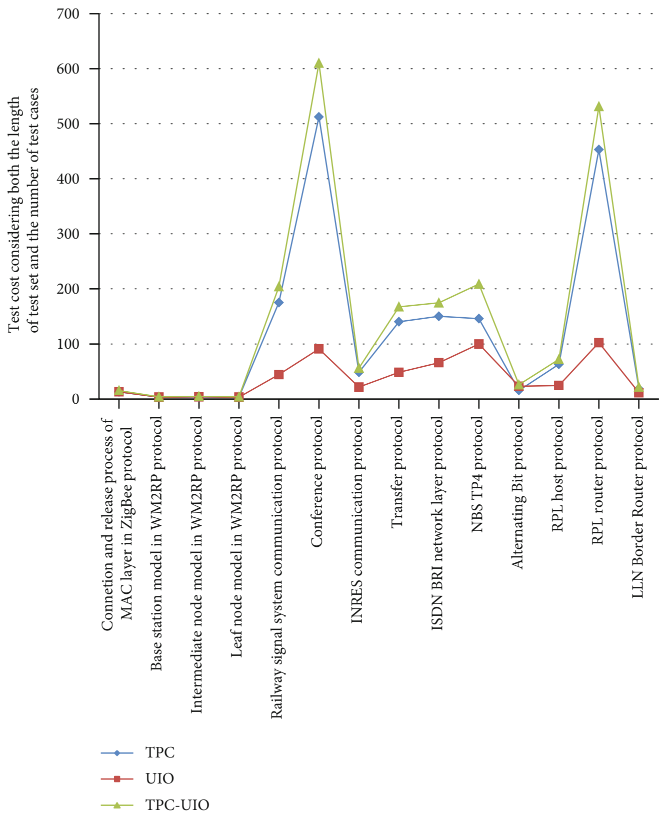


Figure 9: Relationship among the number of test cases for TPC, UIO, and TPC-UIO.

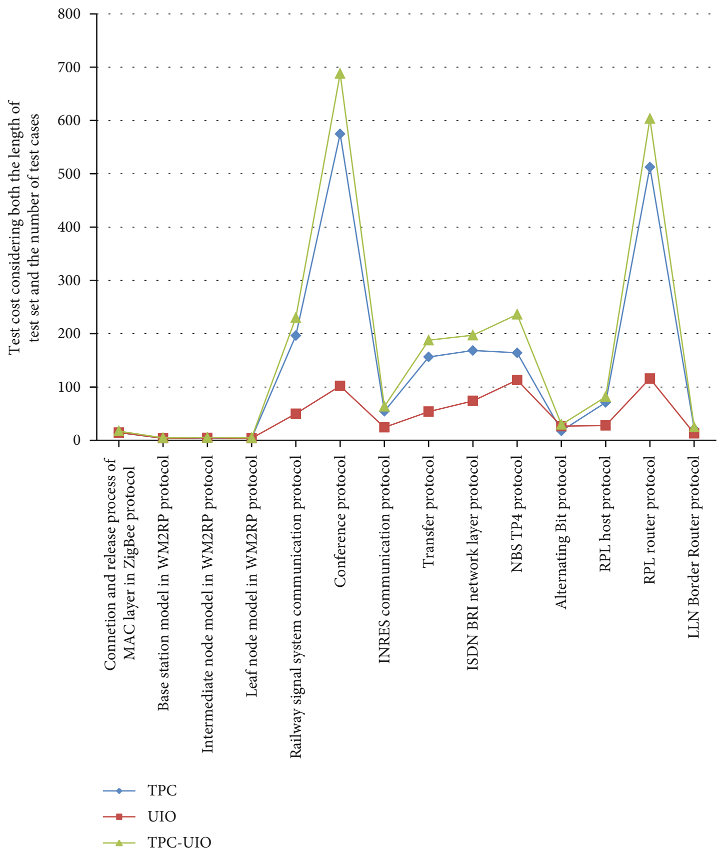
(iii) Consider both the length of the test set and the number of the test cases. In order to quantitatively analyze the cost considering both the length of the test set and the number of the test cases, L is used to denote the length of the test set and the cost generated by the length of the test sets is expressed as a function $C_1 = f(L)$, and N is used to denote the number of the test cases and the cost generated by the number of the test cases is expressed as a function $C_2 = g(N)$. The total test cost considering both the length of the test set and the number of the test cases can be expressed as $C = C_1 + C_2$. Assuming that the test cost of a test set containing only one test case of length 1 is 1, then $f(1) + g(1) = 1$ $(f(1) < 1, g(1) < 1)$. Assuming $\alpha = f(1)$, then $g(1) = 1 - \alpha$, $f(L) = \alpha \times L$, $g(N) = (1 - \alpha) \times N$, and $C = \alpha \times \sum_{i=1}^{N} L_i + (1 - \alpha) \times N$ where L_i represents the length of the ith test case, so that the cost of each test method can be quantified by assigning a value to α. Generally, the impact of the length of the test set is greater than the impact of the number of the test cases, so it is usually $\alpha > 0.5$. Figure 10 shows the relationship among the comprehensive test costs of each empirical case under different test methods when α is assigned 0.6, 0.7, 0.8, and 0.9, respectively. From the examples of different values of α, it can be seen that when α takes different values greater than 0.5, the comprehensive test cost of each empirical case under different test methods changes, and the larger the value of α, the greater the test cost. However, the relative relationship among test costs of different empirical cases has not changed. The comprehensive test cost of the TPC-UIO method is always larger than that of the TPC method and UIO method for each empirical case. Compared with the TPC method, the average increase of the comprehensive test cost of the TPC-UIO method is 23.22%.

4.4.2. Comparison of Fault Coverage. As shown in Table 6, the TPC method, UIO method, and TPC-UIO method can achieve 100% fault coverage for initialization fault and output fault. Neither the TPC method nor the UIO method can reach 100% fault coverage for end state fault. The TPC-UIO method can effectively improve the fault coverage rate and achieve 100% fault coverage.

Although the differences of data in the experiment seem to be subtle, these differences are still worthy of attention. (1) When the fault domain is very large, even a small increase of fault coverage represents a large number of faults. (2) In a safety-critical environment, there is a tiny optimization even if an optimization of only one fault has high practical value. It is noted that the end state faults in the experiment are generated randomly. In fact, the higher the proportion of the analyzed end state faults in Section 4, the better the fault coverage of the combined method.

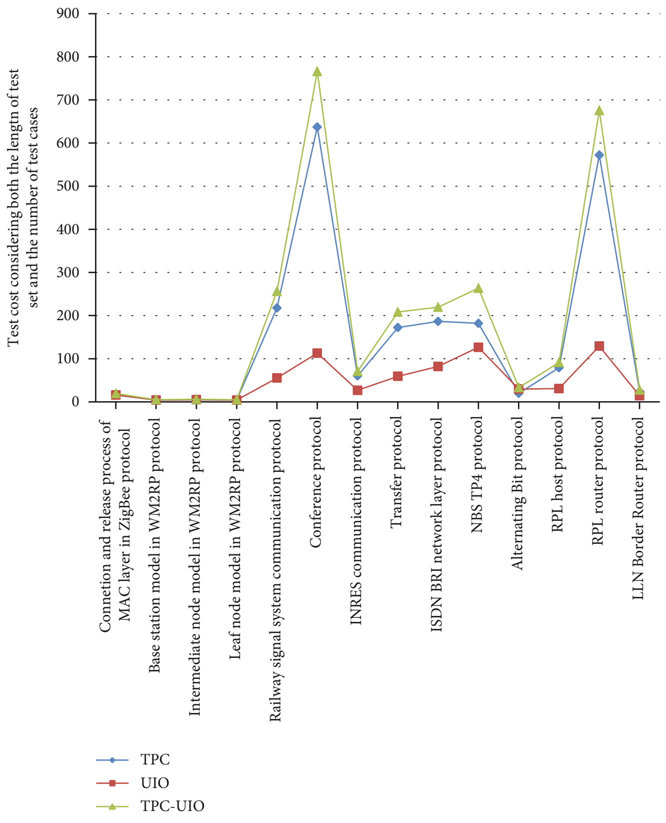


(a) $\alpha = 0.6$

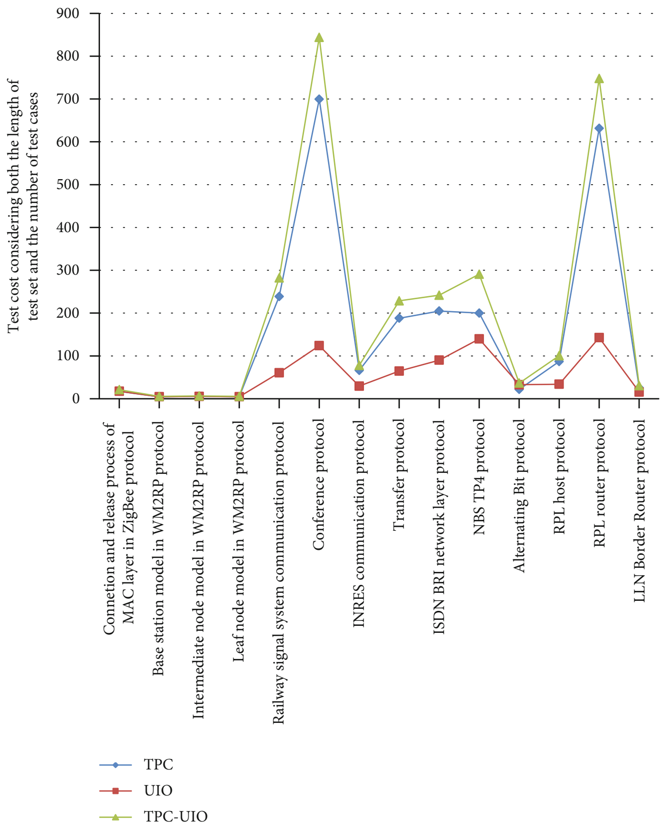


(b) $\alpha = 0.7$

Figure 10: Continued.



(c) $\alpha = 0.8$



(d) $\alpha = 0.9$

FIGURE 10: Relationship among TPC, UIO, and TPC-UIO considering both the length of test sets and the number of test cases.

Table 6: Fault coverage ranking of TPC, UIO, and TPC-UIO.

Ranking	Initialization fault	Output fault	End state fault
1	TPC (100%) UIO (100%) TPC-UIO (100%)	TPC (100%) UIO (100%) TPC-UIO (100%)	TPC-UIO (100%)
2			UIO (99.8%)
3			TPC (99.06%)

5. Related Work

Conformance testing based on FSM has always been an active research direction in software testing [31–39]. In the existing literature, there are precedents for improving fault coverage in conformance testing by combining different testing methods. Mouchawrab et al. [40] investigated the complementarity of the test method of round trip path based on UML state machine and the test method of edge coverage based on source code and then extended round-trip paths testing with edge coverage testing. Results show that the two test strategies are significantly more effective when combined by augmenting state machine testing with structural testing. Hutchins et al. [41] combined the all-use criterion in a white-box test and the edge coverage criterion in a black-box test to improve fault coverage with higher test cost. Results show that significant improvements in the effectiveness of coverage-based tests usually occurred as coverage increased from 90% to 100% and the complementarity of the two methods in their effectiveness was proved. Briand et al. [42] extended the round trip path criterion in the white-box test with the classification partition in the black-box test and obtained better fault detection performance.

This paper also analyzes the complementary characteristic of different test methods and improves the fault coverage at the expense of an acceptable increase of test cost, so as to meet the high requirements of fault coverage in wireless communications.

6. Conclusion and Future Work

There are some differences about the test requirements of the TPC method and UIO method in wireless protocol conformance testing. The TPC method focuses on covering all consecutive operations of length two between entities or processes in wireless communication. UIO method focuses on identifying each state of entities or processes and confirming the input and output information between any two states in wireless communication. In order to improve the effectiveness of wireless protocol testing, a combined method is presented and the method is evaluated by 14 established empirical cases. A unified method based on state covering set is used to generate test sets for each test method and three types of classic faults are generated by mutating the specification. Fault coverage has been effectively improved by the combining method. The experimental results show that the fault coverage of each empirical case can be improved to 100% when the average test cost is only increased by 17.99%. Since what we considered is the specifications modeled by FSMs and the faults generated based on FSMs, although the fault coverage has been effectively improved, these results cannot be popularized to the specifications modeled by other formal models or faults generated based on code.

In a future work, we will try to enrich the experimental data with more publicly available FSM empirical cases. To better describe the behavior characteristics of wireless protocol testing, the future work will also consider extended semantic FSMs to improve the expression ability of the model.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

None of the authors have any conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 61572306.

References

[1] W. W. Lin, H. W. Zeng, H. H. Gao, H. K. Miao, and X. L. Wang, “Test sequence reduction of wireless protocol conformance testing to Internet of Things,” *Security and Communication Networks*, vol. 2018, Article ID 3723691, 13 pages, 2018.

[2] H. H. Gao, W. Q. Huang, and X. X. Yang, “Applying probabilistic model checking to path planning in an intelligent transportation system using mobility trajectories and their statistical data,” *Intelligent Automation and Soft Computing*, vol. 25, no. 3, pp. 547–559, 2019.

[3] X. L. Wang, H. W. Zeng, H. H. Gao, H. K. Miao, and W. W. Lin, “Location-based test case prioritization for software embedded in mobile devices using the law of gravitation,” *Mobile Information Systems*, vol. 2019, Article ID 9083956, 14 pages, 2019.

[4] H. H. Gao, W. Q. Huang, X. X. Yang, Y. C. Duan, and Y. Y. Yin, “Toward service selection for workflow reconfiguration: an interface-based computing solution,” *Future Generation Computer Systems*, vol. 87, pp. 298–311, 2018.

[5] H. L. Yang, K. Ma, C. Deng, H. S. Liao, J. Yan, and J. Zhang, “Towards conformance testing of choreography based on scenario2013 International Symposium on Theoretical Aspects of Software Engineering,” in pp. 59–62, Birmingham, UK, 2013.

[6] M. S. Ramada, T. W. De Lima, A. Simao, and A. D. Soares, “Generating reduced tests for FSMs using a search-based testing approach,” in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 400–407, Portland, OR, USA, November 4-6, 2019.

[7] P. Liu, H. K. Miao, H. W. Zeng, and Y. Liu, "FSM-based testing: theory, method and evaluation," *Chinese Journal of Computers*, vol. 34, no. 6, pp. 965–984, 2011.

[8] A. Jaffari, C. Yoo, and J. Lee, "Automatic test data generation using the activity diagram and search-based technique," *Applied Sciences*, vol. 10, no. 10, p. 3397, 2020.

[9] G. Z. Lu and H. K. Miao, "Optimized test cases generation for EFSM model combining abstraction refinement and satisfiability," *Chinese Journal of Computers*, vol. 39, no. 11, pp. 2236–2252, 2016.

[10] A. Saeed, S. H. Ab Hamid, and A. A. Sani, "Cost and effectiveness of search-based techniques for model-based testing: an empirical analysis," *International Journal of Software Engineering and Knowledge Engineering*, vol. 27, no. 4, pp. 601–622, 2017.

[11] H. H. Gao, S. Y. Mao, W. Q. Huang, and X. X. Yang, "Applying probabilistic model checking to financial production risk evaluation and control: a case study of Alibaba's Yu'e Bao," *IEEE Transactions on Computational Social Systems*, vol. 5, no. 3, pp. 785–795, 2018.

[12] H. H. Gao, H. K. Miao, L. L. Liu, J. Y. Kai, and K. Zhao, "Automated quantitative verification for service-based system design: a visualization transform tool perspective," *International Journal of Software Engineering and Knowledge Engineering*, vol. 28, no. 10, pp. 1369–1397, 2018.

[13] D. Lee and M. Yannakakis, "Principles and methods of testing finite state machines-a survey," *Proceedings of the IEEE*, vol. 84, no. 8, pp. 1090–1123, 1996.

[14] F. R. Porto, A. T. Endo, and A. Simao, "Generation of checking sequences using identification sets," in *In 15th International Conference on Formal Engineering Methods (ICFEM)*, pp. 115–130, Berlin, Heidelberg, October 29-November 1, 2013.

[15] K. Hoda and L. Yvan, "On FSM-based testing: An empirical study: complete round-trip versus transition trees," in *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 305–315, Toulouse, France, October 23-26, 2017.

[16] M. C. Gaudel, "Formal methods for software testing," in *In International Symposium on Theoretical Aspects of Software Engineering (TASE)*, pp. 1–3, Sophia Antipolis, France, September 13-15, 2017.

[17] S. R. Liang, *Fuzzy Test Algorithm of ZigBee Protocol Based on FSM [M.S. Thesis]*, BUPT, Beijing, China, 2014.

[18] A. Simão, A. Petrenko, and J. C. Maldonado, "Experimental evaluation of coverage criteria for FSM-based testing," in *In the 21st Simpósio Brasileiro de Engenharia de Software (SBES)*, pp. 359–374, Joao Pessoa, Brazil, January 13-15, 2007.

[19] P. Ammann and J. Offutt, *Introduction to Software Testing*, China Machine Press, BeiJing, 2018.

[20] J. Lee, S. Kang, and P. Jung, "Test coverage criteria for software product-line testing: systematic literature review," *Information and Software Technology*, vol. 122, no. 2020, article 106272, 2020.

[21] K. EI-Fakih, N. Yevtushenko, and G. Bochmann, "FSM-based incremental conformance testing methods," *IEEE Transactions on Software Engineering*, vol. 30, no. 7, pp. 425–436, 2004.

[22] J. F. Cutigi, A. Simao, and S. R. Souza, "Reducing FSM-based test suites with guaranteed fault coverage," *The Computer Journal*, vol. 59, no. 8, pp. 1129–1143, 2016.

[23] C. C. Liu and H. L. Yang, "Testing of data gathering protocol for wireless sensor networks based on FSM model," *Software Guide*, vol. 16, no. 9, pp. 14–18, 2017.

[24] J. D. Lee, J. I. Jung, J. H. Lee, J. G. Hwang, J. H. Hwang, and S. U. Kim, "Verification and conformance test generation of communication protocol for railway signaling systems," *Computer Standards & Interfaces*, vol. 29, no. 2, pp. 143–151, 2007.

[25] B. Sarikaya and G. V. Bochmann, "Synchronization and specification issues in protocol testing," *IEEE Transactions on Communications*, vol. 32, no. 4, pp. 389–395, 1984.

[26] Z. B. Wang, H. Zhou, and B. H. Zhao, "Path overlapped method for protocol conformance test generation," *Computer Systems & Applications*, vol. 20, no. 7, pp. 47–52, 2011.

[27] R. E. Miller and S. Paul, "On the generation of minimal-length conformance tests for communication protocols," *IEEE/ACM Transactions on Networking*, vol. 1, no. 1, pp. 116–129, 1993.

[28] K. Sabnani and A. Dahbura, "A new technique for generating protocol test," *ACM SIGCOMM Computer Communication Review*, vol. 15, no. 4, pp. 36–43, 1985.

[29] J. Tang, X. Huang, J. Qian, and C. Viho, "A FSM-based test sequence generation method for RPL conformance testing," in *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, pp. 591–597, Beijing, China, August 20-23, 2013.

[30] X. Y. Dang, X. J. Yao, D. W. Gong, and T. Tian, "Efficiently generating test data to kill stubborn mutants by dynamically reducing the search domain," *IEEE Transactions on Reliability*, vol. 69, no. 1, pp. 334–348, 2020.

[31] V. H. Fragal, A. Simao, A. T. Endo, and M. R. Mousavi, "Reducing the concretization effort in FSM-based testing of software product lines," in *2017 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pp. 329–336, Tokyo, Japan, March 13-17, 2017.

[32] P. Liu, Y. D. Li, and Z. J. Li, "Some thoughts on model-based test optimization," in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 268–274, Sofia, Bulgaria, July 22-26, 2019.

[33] S. Pradhan, M. Ray, and S. K. Swain, "Transition coverage based test case generation from state chart diagram," *Journal of King Saud University-Computer and Information Sciences*, vol. 5, no. 5, pp. 1–10, 2019.

[34] T. L. Zhou, H. Y. Sun, J. Liu, X. H. Chen, and D. H. Du, "Improving testing coverage for safety-critical system by mutated specification," in *2014 21st Asia-Pacific Software Engineering Conference*, pp. 43–46, Jeju, South Korea, 2014.

[35] A. Denise, M. C. Gaudel, S. D. Gouraud, R. Lassaigne, J. Oudinet, and S. Peyronnet, "Coverage-biased random exploration of large models and application to testing," *International Journal on Software Tools for Technology Transfer*, vol. 14, no. 1, pp. 73–93, 2012.

[36] R. M. Hierons and H. Ural, "Optimizing the length of checking sequences," *IEEE Transactions on Computers*, vol. 55, no. 5, pp. 618–629, 2006.

[37] H. Ural and F. Zhang, "Reducing the lengths of checking sequences by overlapping," in *In International Conference on Testing of Communicating Systems*, pp. 274–288, New York, NY, USA, May 16-18, 2006.

[38] L. H. Duan and J. Chen, "Reducing test sequence length using invertible sequences," in *In 9th International Conference on*

Formal Engineering Methods (ICFEM), pp. 171–190, Boca Raton, FL, USA, November 14-15, 2007.

[39] G. V. Jourdan, H. Ural, and H. Yenigün, "Reduced checking sequences using unreliable reset," *Information Processing Letters*, vol. 115, no. 5, pp. 532–535, 2015.

[40] S. Mouchawrab, L. C. Briand, Y. Labiche, and M. Di Penta, "Assessing, comparing, and combining state machine-based testing and structural testing: a series of experiments," *IEEE Transactions on Software Engineering*, vol. 37, no. 2, pp. 161–187, 2011.

[41] M. Hutchins, H. Foster, T. Goradia, and T. Ostrand, "Experiments on the effectiveness of data-flow and control-flow-based test adequacy criteria," in *In Proceedings of 16th International conference on Software engineering (ICSE)*, pp. 191–200, Sorrento, Italy, 1994.

[42] L. C. Briand, M. Di Penta, and Y. Labiche, "Assessing and improving state-based class testing: a series of experiments," *IEEE Transactions on Software Engineering*, vol. 30, no. 11, pp. 770–783, 2004.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 1997212, 8 pages
https://doi.org/10.1155/2021/1997212

Research Article

A Graph Convolutional Method for Traffic Flow Prediction in Highway Network

Tianpu Zhang,[1,2] **Weilong Ding**,[1,2] **Tao Chen**,[3] **Zhe Wang**,[1,2] **and Jun Chen**[1,2]

[1]*School of Information Science and Technology, North China University of Technology, Beijing 100144, China*
[2]*Beijing Key Laboratory on Integration and Analysis of Large-Scale Stream Data, Beijing 100144, China*
[3]*Cloud Computing Business Unit, Beijing China-Power Information Technology Co., Ltd., Beijing 100096, China*

Correspondence should be addressed to Weilong Ding; dingweilong@ncut.edu.cn

Received 25 April 2021; Revised 18 May 2021; Accepted 22 June 2021; Published 5 July 2021

Academic Editor: Xiaoxian Yang

As a transportation way in people's daily life, highway has become indispensable and extremely important. Traffic flow prediction is one of the important issues for highway management. Affected by many factors, including temporal, spatial, and other external ones, traffic flow is difficult to accurately predict. In this paper, we propose a graph convolutional method. And the name of our model proposed is the hybrid graph convolutional network (HGCN), which comprehensively considers time, space, weather conditions and date type to achieve better predicted results of traffic flow at highway stations. Compared with baselines implemented by various machine learning models, all metrics of our model are reduced dramatically.

1. Introduction

With economic growth and the infrastructure improvement, people's lives and work are no longer limited to a specific city. The social activities interact with multiple cities have become the most basic requirement of people's daily life. As a bridge between two cities, highway undoubtedly plays a leading role in people's lives and work. If a road is closed occasionally or traffic jam happens owing to an accident on the highway or the bad weather conditions, respectively, people's travel or work plans will be seriously affected [1]. As a mean of traffic management, highway traffic flow prediction can help the government and relevant departments to make road planning and to dispatch vehicle to prevent traffic congestion [2]. Because highway traffic prediction not only has the temporal and spatial characteristics but also is affected by external factors, insufficient consideration of influencing factors is one of the important challenges faced by the traffic flow prediction of highway toll stations. The combination of temporal and spatial can be termed as spatiotemporal characteristics. External factors include weather conditions and date type. For example, people are more willing to take a road trip in fine weather. Especially in weekend or holiday, the demands of folks enjoying travel sharply increase. So, how to effectively use these factors on historical data has become the focus of highway traffic flow prediction.

In this paper, based on deep learning, a graph convolutional method is proposed for highway traffic flow prediction. It that name hybrid graph convolutional network (HGCN) is a combination of the graph convolutional network (GCN) and feedforward neural network (FNN). HGCN can effectively extract the non-Euclidean spatial features of the highway network and comprehensive consideration of the weather type and date type of the toll stations to get better prediction. The rest of this paper is organized as follows. At Section 2, we describe related work with our research. Then, at Section 3, we demonstrate the process of our model building. Next, at Section 4, we introduce several experiments to describe that our model has more accurate prediction effects. Finally, at Section 5, we summarize our work and discuss the direction of future work.

2. Related Work

The traffic flow of highway toll stations is a vital metric for people's daily travel or government decision-making. With the development of Internet of Things (IOT) and big data, people can obtain more types of data for analysis and research on highway traffic tasks [3–5]. Traffic flow prediction is one of the most important tasks in the Intelligent Transportation System (ITS) [6–11]. Methods commonly used in traffic flow prediction can be roughly divided into three categories.

The first type of methods is based on traditional statistical perspective. autoregressive integrated moving average model (ARIMA) [12, 13] and vector autoregressive models (VAR) [14] are the representative model of traditional statistical methods. At the same time, KARIMA [15] and SARIMA [16] based on the ARIMA algorithm are also such statistical learning methods. However, these methods have certain limitations. Because these algorithms cannot consider nonlinear factors of traffic flow, they are impossible to accurately estimate the traffic flow of highway toll stations.

The second type of methods is based on machine learning algorithms. By acquiring nonlinear features from the data, these models more conform to the problem of traffic flow prediction of highway toll stations under the real condition. Compared with statistical methods, models based on machine learning can get more accurate prediction results. Many representative models for traffic flow prediction of such methods are proposed like support vector regression (SVR) [17], Bayesian model [18], boosting technology [19], and K-nearest neighbor model (KNN) [20, 21]. Although learning nonlinear features, these models have a strong dependence on the processing of input raw data. Good feature processing can bring ideal prediction results and vice versa. Therefore, shallow machine learning models cannot obtain better prediction results.

The third type of methods is based on deep learning. Recently, recurrent neural networks (RNNs) and convolutional neural networks (CNN) are two main adopted models of deep learning algorithms. Among [22–27], RNN models are generally used to deal with temporal series, because RNN models can extract temporal feature from data very well. In particular, long short-term memory (LSTM), as a variant of RNN models, effectively avoids the problems of gradient explosion and gradient disappearance during the process of training by adding gating mechanism and is widely used in traffic flow prediction. Works [28–33] employ variant RNN models to predict traffic flow. In addition, the CNN network is good at obtaining the spatial characteristics of a two-dimensional matrix in the Euclidean space. So, many studies such as [34–36] use CNN models to obtain spatial feature of the highway network to get better prediction for traffic flow. In order to consider more comprehensively, combined CNN and RNN models like [37–39] are proposed considering the spatiotemporal characteristics of traffic prediction. On the other hand, because the limitation of the graph convolutional network (CNN) that is not good at dealing with the problems in non-Euclidean space, GCN has become the most popular technology for traffic flow prediction problems. [1, 37] are the typical example of using GCN to learn spatial correlation features to predict traffic flow.

However, in fact, the factors they considered were insufficient. Temporal characteristics and external factors, including weather conditions and date types, have a great influence on the prediction accuracy of traffic flow. Therefore, we propose a graph convolutional method, which belongs to the third type of methods. Our model is a combination of the graph convolutional network (GCN) and feedforward neural network (FNN) and is named hybrid graph convolutional network (HGCN). Our model not only can effectively extract the non-Euclidean spatial features of the highway network but also learn temporal feature of traffic flow and external features of toll stations, and then our model gets a better prediction of traffic flow.

3. HGCN Model and Its Construction

3.1. Feature Engineering. Because the traffic flow prediction problem of highway toll stations is affected by many factors, it is necessary to consider multiple factors extracted from input raw data. These features are defined as follows.

Definition 1(highway network): In this paper, we use $G = (V, E)$ to represent the highway network structure. Here, $V = [V_1, V_2, \cdots, V_N]$ represents highway toll stations, N represents the total number of toll stations, $E = [(V_1, V_2), (V_2, V_3), \cdots, (V_i, V_j), \cdots]$, $V_i, V_j \in V$, indicates whether there is an edge between two toll stations, and i and j are a positive integer.

In our work, we regard highway network as an undirected graph according to actual bidirectional highway roads. In addition, the prediction of highway toll station traffic flow is not only related to the spatial relationship of toll stations but also related to temporal characteristics. For example, the traffic flow of a station has a certain periodicity. In weekdays, it will have relatively similar volumes, and in weekends or holidays, it will increase. Therefore, it is necessary to consider the influence of the temporal features on the traffic flow at the highway toll station. The temporal features can be defined as follows.

Definition 2 (historical traffic flow of highway toll stations): On the day t, the temporal features of a certain toll station can be represented by the vector $S_t^{V_i}$:

$$S_t^{V_i} = \begin{bmatrix} T_1 \\ T_2 \\ \cdots \\ T_k \\ \cdots \\ T_d \end{bmatrix}, V_i \in V, \tag{1}$$

where $S_t^{V_i}$ represents the features of highway toll station V_i on the day t, d represents the length of the historical time window, $k \le d$ represents a day in historical time window, and T_k represents the traffic flow of the highway toll station on day $t - k$.

In addition to spatiotemporal characteristics, the prediction of traffic flow at highway toll stations also faces external factors. According to the analysis for the traffic flow data of highway toll stations, the external factors include weather conditions and date type [19]. For example, in extreme weather conditions such as heavy snow, heavy fog, and downpour, the highway may be closed. Besides, people are more willing to travel or go on a road trip on sunny days. Especially on weekends and holidays, the number of people traveling increases obviously. Therefore, we take external characteristics of traffic flow into the model. Then, the external characteristics can be defined as follows.

Definition 3 (external factors): On the day t, external factors can be represented by the vector $P_t^{V_i}$:

$$P_t^{V_i} = \left(W_t^{V_i}, D_t, Vol_t^{V_i}\right)^T, V_i \in V. \tag{2}$$

$W_t^{V_i}$ represents the weather conditions at the V_i toll station on day t, D_t represents the date type of day t, and $Vol_t^{V_i}$ represents the traffic flow of the highway toll station adjacent to the V_i on the t day. The calculation of these parameters can refer to our previous work [21]. In order to explain the weather condition type and date type characteristics in more detail, we define them as formula (3) and formula (4).

$$W_t^{V_i} = \begin{cases} 1, & \text{extrem weather} \\ 0, & \text{otherwise} \end{cases}, \tag{3}$$

where $W_t^{V_i}$ represents the weather conditions at the V_i toll station on day t, and there are two values for this parameter. When this parameter equals 1, it represents extreme weather condition including heavy snow, heavy fog, downpour, and hurricane. When this parameter equals 0, it represents other weather conditions. In the other hand, we divide the date into three types include holidays, weekends, and others. The specific definition is as follows.

$$D_t = \begin{cases} 0, & \text{otherwise} \\ 1, & \text{if t is a holiday} \\ 2, & \text{if t is a weekend} \end{cases}. \tag{4}$$

3.2. Graph Construction. How to effectively construct a highway network graph has a great influence on extracting the spatial characteristics of the associated toll stations. Neighborhood will directly affect the prediction effect of traffic flow at highway toll stations. Therefore, we constructed our highway network graph based on the following three principles.

3.2.1. Connectivity Principle. This principle guarantees the completeness and correctness of the highway network graph. It means that there will be no isolated nodes or subgraphs in the highway network graph. That is, there is always a path from any node V_i to any node V_j in the graph. We set this principle because in actual situations, any two toll stations on the highway are always reachable, and there is no isolated toll station.

3.2.2. Neighborhood Principle. This principle explains how to select neighbor nodes. The edge in the graph taken from any toll station V_i to its neighbor toll station V_j does not pass through other toll stations. Because we believe that adjacent toll stations in the highway network have higher spatial correlation, and at the same time, it will have a greater impact on the traffic flow of its adjacent toll stations, we look for the nearest toll stations on the highway network as the neighbor node of toll station V_i.

3.2.3. Bidirection Principle. This principle stipulates that the highway network graph we construct is an undirected graph. That means, toll stations V_i and V_j are connected, and vice versa. This principle is set to conform to the real situation of the highway network.

According to the above three rules, we use the adjacency list to construct our highway network, as shown in Figure 1.

In the example of Figure 1, the arrow represents starting toll station V_i, and its neighboring toll station V_j is connected. For example, Anyang is selected as the starting toll station, and its neighboring toll stations include Anyangbei, Anyangdong, Aanyangnan, and Anyangkaifaqu. However, it is worth noting that the arrow only represents the connection relationship between the starting toll station and its neighboring toll stations.

3.3. Model Structure. After processing features, the traffic flow prediction problem of highway toll stations can be expressed by formula (5).

$$T_t^V = \begin{bmatrix} T_t^{V_1} \\ T_t^{V_2} \\ \cdots \\ T_t^{V_i} \\ \cdots \end{bmatrix} = F\left(G, S_t^V, P_t^V\right). \tag{5}$$

T_t^V represents the traffic flow of all highway toll stations on day t. S_t^V represents the historical traffic flow of all highway toll stations on day t. P_t^V represents the external factors on the day t. F represents the method including feature engineering and graph convolutional method proposed in this paper. The specific structure of the model is shown in Figure 2.

For this method F, input is the raw data, and output is the prediction result of traffic flow at highway toll stations. The detailed process of F can be divided into three parts, including feature engineering, GCN, and FNN.

In the feature engineering part, raw input data including highway toll stations network and traffic flow of highway toll stations are needed for extracting spatiotemporal and external factors. First, in order to extract spatial features, we construct an undirected graph G by using the highway toll station network. Then, we obtain historical traffic flow based on traffic data of highway toll stations and add weather conditions and date type factors. Through the above processing,

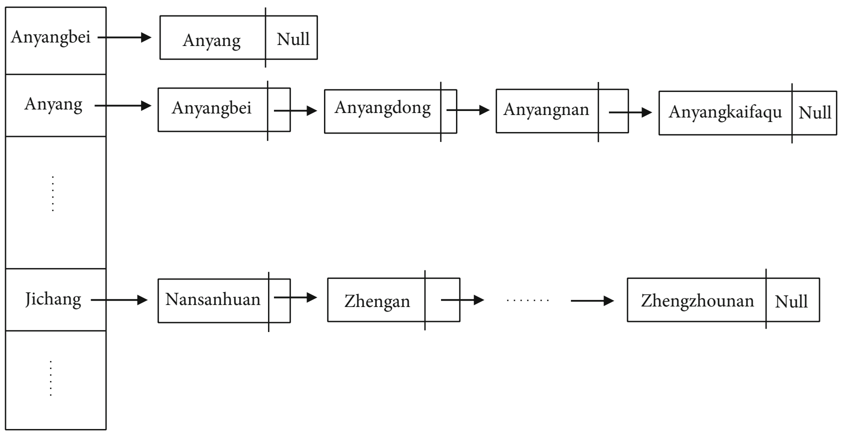


Figure 1: Adjacency list of the highway network.

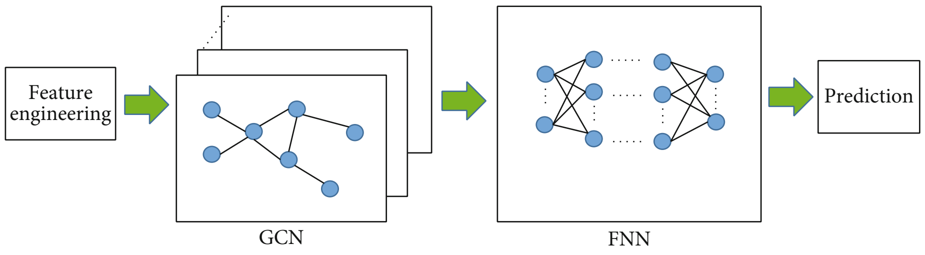


Figure 2: Network traffic flow prediction method.

our feature engineering part finally output the spatial feature G and the temporal and external factor matrix X.

In the GCN part, we use the GCN model to extract the spatial characteristics of the highway network and to obtain the influence factors of spatial relationship between toll stations. Using the approximated 1st order Chebyshev polynomial expansion, GCN proposed by [40] not only reduces the number of parameters but also increases the prediction accuracy. Using GCN, we can obtain the relationship among neighboring nodes and learn the impact of different nodes on the toll station to be predicted. Considering the influence of neighboring nodes, our model can make more accurate predictions. Therefore, in this part, two layers of GCN are used to extract spatial features. The input feature dimension of the first layer of GCN is the same as that of the feature matrix X. The specific formulas of the GCN layer can be defined as follows.

$$\begin{aligned} X^{(1)} &= \mathrm{Relu}\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}XW\right), \\ X^{(2)} &= \mathrm{Relu}\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X^{(1)}W\right). \end{aligned} \tag{6}$$

X represents feature matrix including the temporal feature of traffic flow at highway toll stations and external factors, $\tilde{A} = A + I_N$, A is the adjacency matrix of the highway network G, I_N represents identity matrix of size N, and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, $i, j \le N$. The weight matrix W is a learnable parameter. $X^{(1)}$ and $X^{(2)}$ represent the output of the first layer and the second layer, respectively, and the dimension of their feature is 64.

In the FNN part, the feedforward neural network (FNN) is used to comprehensively consider three influencing factors. Through nonlinear transformation and weighted summation on the input feature matrix $X^{(2)}$ extracted by GCN, the FNN part gets the predicted result of highway toll stations. In this part, we use a three-layer neural network structure, corresponding to the input layer, hidden layer, and output layer. The specific structure of the network is shown in Figure 3.

In order to extract more features, the number of neurons in the input layer is 128, and the number of neurons in the hidden layer is the same with the input layer. Finally, the number of neurons in the output layer is one, and the output of the output layer represents prediction result of traffic flow at highway toll stations. In order to clearly illustrate process in the FNN part, we defined some formulas as follows.

$$\begin{aligned} M^{(1)} &= \mathrm{Relu}\left(X^{(2)}W_1 + b_1\right), \\ M^{(2)} &= \mathrm{Relu}\left(M^{(1)}W_2 + b_2\right), \\ M^{(3)} &= M^{(2)}W_3 + b_3. \end{aligned} \tag{7}$$

Here, M^* represents the output of each layer in FNN part, and $M^{(3)}$ represents the result of prediction of traffic flow at highway toll stations. W_* and b_* represent weighted matrix and bias, respectively, and both are learnable parameters.

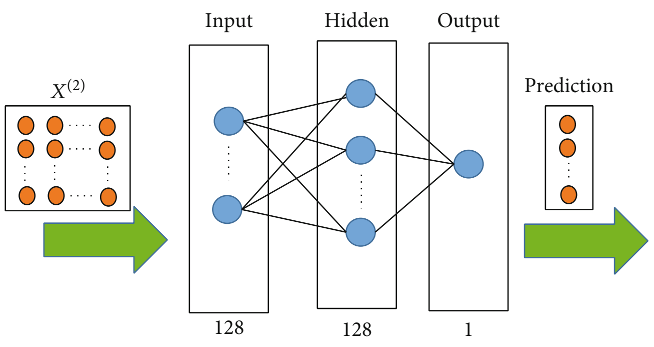


Figure 3: FNN model structure.

4. Evaluation

4.1. Settings. The experimental data comes from a real application system, which is the Henan Highway Management System [13]. In the experiment, we used weather and traffic flow data on highway toll stations from May 2017 to September 2017. The total amount of data is approximately more than 20000. From September 16, 2017 to September 30, 2017 as our testing set and the rest of data as our training set, we choose $d = 7$, that means we use 7 days of historical traffic flows of highway toll stations as the temporal feature. Due to the large amount of traffic flow, the system stores the traffic flow in the HBase database. Our system uses three servers to build the HBase database that its version is 1.6.0, each of which server have 4 cores CPU, 22 GB RAM, and 700 GB storage. Meanwhile, the machine configuration used for model training is Intel (R) Core (TM) CPU i7-9750 2.59GHz, 16GB RAM, 1 TB storage, and one NVIDIA GeForce GTX 1660 Ti.

In addition, our experiment is supported by common machine learning software, including python 3.6, pytorch 1.7.0, torch-geometric 1.6.1, and sklearn 0.23.2. In the experiment, we compared HGCN with several other models, including gradient boost regression tree (GBRT) [19] and KNN [21] model on the prediction accuracy of highway toll stations. To evaluate the prediction effect of our model, three metrics, root mean square error (RMSE), mean absolute percentage error (MAPE), and mean absolute error (MAE), are used in our experiments. Their formulas are defined as follows.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y\wedge_i - y_i)^2},$$
$$\mathrm{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right|, \qquad (8)$$
$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|\hat{y}_i - y_i\right|.$$

Here, $\hat{y}_i$ represents the predicted value of traffic flow, y_i represents the ground truth value of traffic flow, $\bar{y}$ represents average predicted value, and n represents the number of samples.

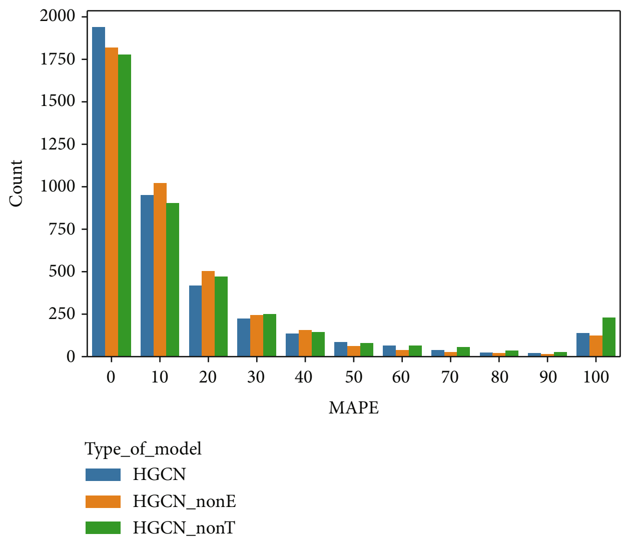


Figure 4: MAPE in different factors.

Table 1: prediction performances of different parameters in a model.

	HGCN	$\mathrm{HGCN}_{\mathrm{noE}}$	$\mathrm{HGCN}_{\mathrm{noT}}$
RMSE	682.393	894.616	948.247
MAE	403.861	446.829	500.227
MAPE (%)	23.185	22.232	29.218

4.2. Experiments. Two experiments are designed to verify the effectiveness of our feature engineering and the prediction effect of the HGCN model for predicting traffic flow at highway toll stations.

4.2.1. Experiment 1:Feature Engineering Effects. In this experiment, we compare two variant models based on HGCN, $\mathrm{HGCN}_{\mathrm{noE}}$, and $\mathrm{HGCN}_{\mathrm{noT}}$. Here, $\mathrm{HGCN}_{\mathrm{noE}}$ means that in the feature Engineering part, the model does not consider external factors (abbr. for *noE*). In addition, $\mathrm{HGCN}_{\mathrm{noT}}$ ignores the temporal features in the feature engineering part (abbr. for noT). In this case, we use three different models to predict the traffic flow of 269 highway toll stations, and the comparison result of models can refer to Figure 4.

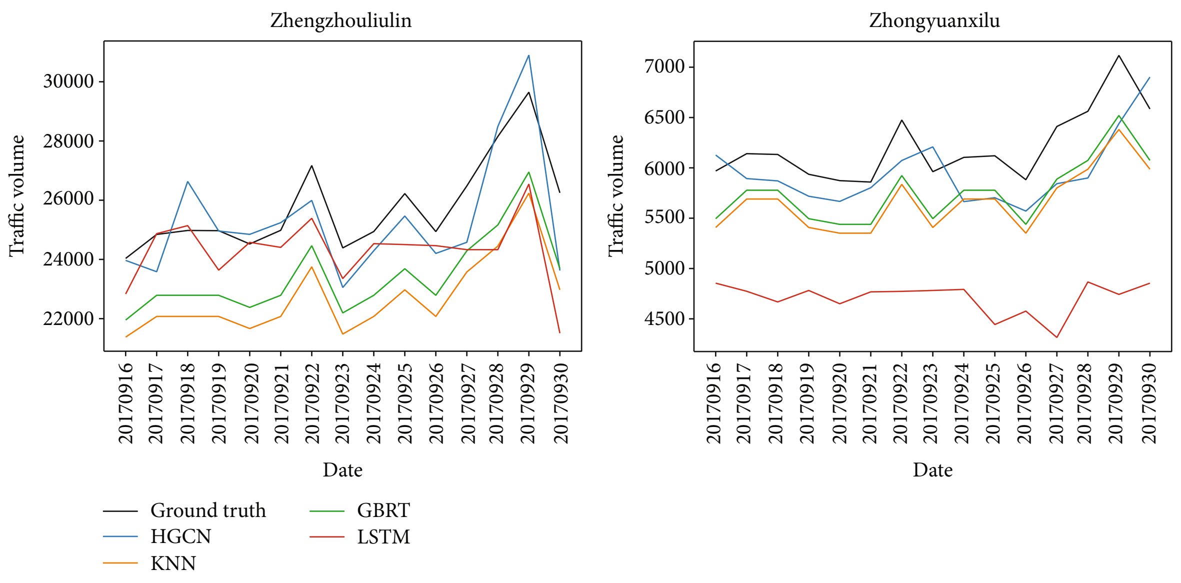


Figure 5: The comparison of prediction accuracy with other models.

Table 2: Prediction performances of different models.

	HGCN	GBRT	KNN	LSTM	HGCN	GBRT	KNN	LSTM
		Zhengzhouliulin				Zhongyuanxilu		
RMSE	1194.719	387.864	387.864	387.864	387.864	998.137	1231.851	895.941
MAE	956.928	346.011	346.011	346.011	346.011	993.786	1226.910	724.597
MAPE (%)	3.683	5.469	5.469	5.469	5.469	8.545	10.549	6.110

In Figure 4, we use a MAPE distribution chart to compare the prediction effect among different models. First of all, we can see that the distribution chart has the right-skewed attribute as we expect. In the chart, the more concentrated the density is on the left of the chart, the better the prediction accuracy would be. It can be seen from the figure that the density of the HGCN model on the left is more concentrated than other models. This phenomenon indicates that our HGCN model has obtained better prediction results on the problem of highway traffic flow prediction at toll stations; so, weather conditions and date types will affect the prediction accuracy. Therefore, it is very necessary to add weather conditions and date type features to the model to improve the prediction accuracy.

We have carried out traffic predictions on all stations on the highway and constructed Table 1 on the experiment results. From Table 1, we can see that the HGCN model is better than the other two models in RMSE and MAE. While MAPE of HGCN is slightly lower than HGCN_{noE}, the difference between the two models is less than 1%, and we can regard them as being at the same level. It may be the cause that weather factors mislead our model. Especially for toll stations with a small amount of traffic flow, weather conditions have little effect on the prediction of traffic flow. By comprehensively analyzing RMSE, MAE, and MAPE, our model HGCN brings more accurate predictions.

The result of three metrics also confirmed that it is necessary to consider temporal and external factors to predict the traffic flow of highway toll stations. By considering temporal characteristics and external factors, our model can better learn the historical relationship of traffic flow and the influence of external factors to obtain better prediction results.

4.2.2. Experiment 2: Prediction Effect Comparison with Machine Learning and Deep Learning Models. In this experiment, we use HGCN to predict the traffic flow of all 269 highway toll stations in Henan Province and compare the accuracy with other models. We have selected 2 typical toll stations to show the prediction results of the traffic flow. The traffic flow of the first toll station is about more than 10,000 vehicles per day, and the traffic flow of another toll station is about 4,000 to 8,000 vehicles per day. The experimental comparison results are shown in Figure 5.

From Figure 5, we can clearly see that HGCN can better fit the ground truth of the traffic flow at those highway toll stations than other models. Obviously, by using GCN to obtain the spatial factors of the highway network, our model has a higher prediction accuracy. At the same time, we can see from Table 2 that, in the toll station of Zhengzhouliulin, compared with the LSTM model, our model HGCN has greatly reduced from 5.651 to 3.683 in MAPE metric. And on another toll station, our model is also better than other models including GBRT, KNN, and LSTM in metrics RMSE, MAE, and MAPE. This experiment clearly shows, because our model uses GCN to consider the spatial relationship of toll stations and uses the FNN model to comprehensively consider the characteristics of temporal-spatial and external factors; so, the HGCN model we proposed can get more accurate prediction result of traffic flow.

5. Conclusion

In this paper, we propose a convolutional method HGCN based on a deep learning algorithm to predict the traffic flow of highway toll stations. For solving insufficient consideration of influencing factors of traffic flow, our work can be divided into two parts. First, in order to better fit the traffic flow prediction problem of highway toll stations, we considered spatiotemporal and external factors including weather conditions and date type. Then, by using GCN to extract spatial features, our model fully considers the non-Euclidean attributes of the highway network. Our model obtains a more accurate prediction of the traffic flow at highway toll stations. In the future, we will dynamically adjust the graph structure of GCN and take the distance between toll stations into the highway network structure as an edge attribute in order to comprehensively consider the influence of different distance among toll stations for traffic flow prediction.

Data Availability

The traffic flow data used to support the findings of this study have not been made available because the data were supplied by local management Henan Transport Department under license with certain confidentiality level and so cannot be made freely available. Requests for access to these data should be made to the corresponding author for an application of joint research.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

We would like to thank Mengda Xing, Jindon Xu, Bo Lv, and Ji Liu who are classmates in our laboratory to optimize our model and improve the quality of this paper. This work was supported by the National Natural Science Foundation of China (No. 61702014) and Beijing Municipal Natural Science Foundation (No. 4192020).

References

[1] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 922–929, Honolulu, Hawaii, USA, 2019.

[2] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive graph convolutional recurrent network for traffic forecasting," 2020, https://arxiv.org/abs/2007.02842.

[3] L. Zhu, F. R. Yu, Y. Wang, B. Ning, and T. Tang, "Big data analytics in intelligent transportation systems: a survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 383–398, 2019.

[4] F. Belletti, D. Haziza, G. Gomes, and A. M. Bayen, "Expert level control of ramp metering based on multi-task deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1198–1207, 2018.

[5] H. Gao, X. Qin, R. J. D. Barroso, W. Hussain, Y. Xu, and Y. Yin, "Collaborative learning-based industrial IoT API recommendation for software-defined devices: the implicit knowledge discovery perspective," *IEEE Transactions on Emerging Topics in Computational Intelligence*, pp. 1–11, 2020.

[6] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *2016 IEEE 16th international conference on data mining (ICDM)*, pp. 316–324, Barcelona, Spain, 2011.

[7] L. Kuang, T. Gong, S. OuYang, H. Gao, and S. Deng, "Offloading decision methods for multiple users with structured tasks in edge computing for smart cities," *Future Generation Computer Systems(FGCS)*, vol. 105, pp. 717–729, 2020.

[8] H. Gao, C. Liu, Y. Li, and X. Yang, "V2VR: reliable hybrid-network-oriented V2V data transmission and routing considering RSUs and connectivity probability," *IEEE Transactions on Intelligent Transportation Systems(T-ITS)*, vol. 22, no. 6, pp. 3533–3546, 2020.

[9] H. Gao, W. Huang, and Y. Duan, "The cloud-edge-based dynamic reconfiguration to service workflow for mobile ecommerce environments," *ACM Transactions on Internet Technology*, vol. 21, no. 1, pp. 1–23, 2021.

[10] X. Yang, Z. Sijing, and C. Min, "An approach to alleviate the sparsity problem of hybrid collaborative filtering based recommendations: the product-attribute perspective from user reviews," *ACM/Springer Mobile Networks and Applications (MONET)*, vol. 25, no. 2, pp. 376–390, 2020.

[11] Y. Yin, Z. Cao, Y. Xu, H. Gao, R. Li, and Z. Mai, "QoS prediction for service recommendation with features learning in mobile edge computing environment," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 4, pp. 1136–1145, 2020.

[12] M. M. Hamed, H. R. Al-Masaeid, and Z. M. B. Said, "Short-term prediction of traffic volume in urban arterials," *Journal of Transportation Engineering*, vol. 121, no. 3, pp. 249–254, 1995.

[13] W. Ding and Z. Zhao, "DS-harmonizer: a harmonization service on spatiotemporal data stream in edge computing environment," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 9354273, 12 pages, 2018.

[14] E. Zivot and J. Wang, "Vector autoregressive models for multivariate time series," in *Modeling Financial Time Series with S-PLUS®*, pp. 385–429, Springer Nature, 2006.

[15] M. Van Der Voort, M. Dougherty, and S. Watson, "Combining kohonen maps with arima time series models to forecast traffic flow," *Transportation Research Part C: Emerging Technologies*, vol. 4, no. 5, pp. 307–318, 1996.

[16] B. M. Williams and L. A. Hoel, "Modeling and forecasting vehicular traffic flow as a seasonal arima process: theoretical basis and empirical results," *Journal of Transportation Engineering*, vol. 129, no. 6, pp. 664–672, 2003.

[17] C.-H. Wu, J.-M. Ho, and D. T. Lee, "Travel-time prediction with support vector regression," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 4, pp. 276–281, 2004.

[18] S. Sun, C. Zhang, and G. Yu, "A Bayesian network approach to traffic flow forecasting," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 1, pp. 124–132, 2006.

[19] W. Ding, Y. Xia, Z. Wang, Z. Chen, and X. Gao, "An ensemble-learning method for potential traffic hotspots detection on heterogeneous spatio-temporal data in highway domain," *Journal of Cloud Computing Advances Systems and Applications*, vol. 9, no. 1, 2020.

[20] X. Zhang, G. He, and H. Lu, "Short-term traffic flow forecasting based on K-nearest neighbors non-parametric regression," *Journal of Systems Engineering*, vol. 24, no. 2, pp. 178–183, 2009.

[21] W. Ding, X. Wang, and Z. Zhao, "CO-STAR: a collaborative prediction service for short-term trends on continuous spatio-temporal data," *Future Generation Computer Systems*, vol. 102, pp. 481–493, 2020.

[22] Z. Diao, X. Wang, D. Zhang, Y. Liu, K. Xie, and S. He, "Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 890–897, 2019.

[23] S. Fang, Q. Zhang, G. Meng, S. Xiang, and C. Pan, "Gstnet: global spatial-temporal network for traffic flow prediction," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, Maco, China, 2019.

[24] S. Guo, Y. Lin, S. Li, Z. Chen, and H. Wan, "Deep spatial–temporal 3d convolutional neural networks for traffic data forecasting," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 10, pp. 3913–3926, 2019.

[25] D. Chai, L. Wang, and Q. Yang, "Bike flow prediction with multigraph convolutional networks," in *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 397–400, New York, NY, USA, 2018.

[26] C. Chen, K. Li, S. G. Teo et al., "Gated residual recurrent graph neural networks for traffic prediction," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 485–492, 2019.

[27] W. Chen, L. Chen, Y. Xie, W. Cao, Y. Gao, and X. Feng, "Multirange attentive bicomponent graph convolutional network for traffic forecasting," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 4, pp. 3529–3536, 2020.

[28] Y. Lee and O. Min, "Long short-term memory recurrent neural network for urban traffic prediction: a case study of Seoul," in *2018 21st international conference on intelligent transportation systems (ITSC)*, pp. 1279–1284, Maui, HI, USA, 2018.

[29] Z. Zhao, W. Chen, X. Wu, P. C. Chen, and J. Liu, "LSTM network: a deep learning approach for short-term traffic forecast," *IET Intelligent Transport Systems*, vol. 11, no. 2, pp. 68–75, 2017.

[30] L. Bai, L. Yao, S. S. Kanhere, Z. Yang, J. Chu, and X. Wang, "Passenger demand forecasting with multi-task convolutional recurrent neural networks," in *Advances in Knowledge Discovery and Data Mining*, Springer, 2019.

[31] X. Tang, H. Yao, Y. Sun, C. Aggarwal, P. Mitra, and S. Wang, "Joint modeling of local and global temporal dynamics for multivariate time series forecasting with missing values," 2019, https://arxiv.org/abs/1911.10273.

[32] H. Yao, F. Wu, J. Ke et al., "Deep multi-view spatial-temporal network for taxi demand prediction," in *Thirty-Second AAAI Conference on Artificial Intelligence*, China, 2018.

[33] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[34] J. Zhang, Y. Zheng, D. Qi, R. Li, X. Yi, and T. Li, "Predicting citywide crowd flows using deep spatio-temporal residual networks," *Artificial Intelligence*, vol. 259, pp. 147–166, 2018.

[35] Y. Wu and H. Tan, "Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework," 2016, http://arxiv.org/abs/1612.01022.

[36] J. Wang, Q. Gu, J. Wu, G. Liu, and Z. Xiong, "Traffic speed prediction and congestion source exploration: a deep learning method," in *2016 IEEE 16th international conference on data mining (ICDM)*, pp. 499–508, Barcelona, Spain, Dec. 2016.

[37] L. Cai, K. Janowicz, G. Mai, B. Yan, and R. Zhu, "Traffic transformer: capturing the continuity and periodicity of time series for traffic forecasting," *Transactions in GIS*, vol. 24, no. 3, pp. 736–755, 2020.

[38] C. Chen, K. Li, S. G. Teo et al., "Gated residual recurrent graph neural networks for traffic prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, Honolulu, Hawaii, USA, 2019.

[39] Z. Wang, W. Ding, and H. Wang, "A hybrid deep learning approach for traffic flow prediction in highway domain," in *Proc. 16th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2020)*, pp. 253–267, Shanghai, China, 2020.

[40] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, https://arxiv.org/abs/1609.02907.