Journal on
Information Technologies & Communications
https://www.ictmag.vn
Volume 2024, No.1
March

TABLE OF CONTENTS

Network Traffic Attention Features Fusion for Unauthorized IoT Devices Detection

Duong Van Dan, Ta Minh Thanh
Le Quy Don Technical University, 236 Hoang Quoc Viet, Ha Noi, Viet Nam

Correspondence: Ta Minh Thanh, thanhtm@lqdtu.edu.vn
Communication: received 25 Apr 2023, revised 17 May 2023, accepted 29 June 2023
DOI: 10.32913/mic-ict-research.v2024.n1.1205

***Abstract*: The demand for using IoT devices is increasing because of its usefulness. IoT devices are present in all areas of life. They make life easier and more comfortable. Moreover, many companies and households are tending to increase in the use of IoT devices to turn on/off home, school, office into smart home, smart school, smart office and so on. Although IoT devices are very useful, they have major security problems because they are not interested, leading to a large number of security holes and creating opportunities for hackers to attack the systems. In this paper, we develop an attention features fusion (AFF) method for the purpose of classifying and detecting IoT devices that are not in the pre-registered white list. Our recommended technique finds the features with the best classifier ability, then converts them into images and uses ensemble learning based on the baseline models which are deep learning models to detect and classify. Our proposed solution helps to detect strange devices that are exchanging data in the IoT network, reducing the risk of the system being attacked by hackers. According to the experimental results, our method achieves very good results when we detect ten IoT devices with up to 97.85% accuracy.**

Keywords: *Internet of Things (IoT), deep learning, IoT device identification, attention features fusion method (AFF), IoT device detection, feature fusion; image processing.*

I. INTRODUCTION

1. Overview

In the world, more and more IoT systems are deployed. IoT devices are increasingly useful and almost everyone knows about it. It includes devices connected to the internet system such as computers, cameras, sensors, and many other smart devices. The number of IoT devices worldwide by the end of 2022 is about 14 billion devices and this number will increase even more. It is expected to reach threshold of 30 billion devices by the end of 2030 [1]. IoT devices are used in many different fields, with about 60% of devices connected by 2020 and this number is increasing [1]. According to Cisco's prediction [2], this number may increase to 29 billion as soon as 2023.

However, apart from the strong development of IoT systems, the field of information security for IoT networks has received little attention. Leading to the existence of a large number of security holes in the system, creating opportunities for hackers to attack the system and steal important resources. The security vulnerabilities of IoT devices come from the manufacturer's lack of information security knowledge as well as the lack of national common safety standards for IoT devices [3].

In 2016, a renowned attack on the IoT systems called Mirai Botnet [4]. In earlier times, IoT systems as well as IoT devices were set up and deployed with almost default accounts and passwords, leading to systems being quickly attacked by hackers if they had a set of default accounts and passwords of companies. In addition, when IoT devices are connected to the Internet, hackers can easily use search engines like Shodan [5] to find the physical location of the devices and other information of the devices. The top 10 critical security vulnerabilities of IoT devices published by OWASP [6] in 2018, including very basic vulnerabilities but which are weak points of IoT devices such as weak passwords, lack of authentication mechanisms, hierarchical, lack of coding element, and so on.

Collecting and managing data from all devices in a large organization is also a challenge. If there is an error in the system, it is likely that the devices connected to the IoT network will be disconnected and need time to fix. At present, there are no international standards for IoT devices, when participating in data exchange in the Internet, it is easy for bad actors to take advantage of weaknesses to attack these devices.

In order to ensure that the devices in the IoT network are the devices of the organization, have been registered for use in advance and prevent the strange devices of hackers from entering the IoT network, the first step we need to do is to identify/detect which are the organization's IoT devices and which are potentially bad IoT devices. Therefore, we need

to build solutions to classify IoT devices that are exchanging data in the IoT network.

2. Our contributions

In this paper, we propose a method to detect IoT devices that are exchanging data in the network by using network traffic to analyze and detect those IoT devices based on the our Attention Features Fusion method (AFF). The suggested approach achieves wholly high results when we apply it to the test dataset, obtaining up to 97.85% accuracy comparing with another conventional algorithms.

Our contributions of this paper are explained as follows:

- In fact, the dataset collected about the network traffic of IoT devices has many data attributes. This causes noise when training deep learning models, leading to low accuracy. Therefore, our first contribution is to find the 14 best features for each device and generated an image dataset from 14 best features.
- Propose an attention features fusion (AFF) method by using 14 features with the best classification ability then convert them into images and use Deep Learning models for training. Finally, we use ensemble learning with voting type with deep learning models.
- Discover a collection of models that provide the most accurate classification results for each device.

3. Roadmap

In the remainder of this paper, we organize the content as follows: In Section II, we present the relevant knowledge and some methods proposed by previous researchers. Our suggested approach is presented in Section III. The experimental environment and experimental results are explained in Section IV. Section V concludes this paper and shows the future research works.

II. Related Work

Many researchers have done on unauthorized IoT devices detection before and have obtained extremely positive results. The field of information security on IoT networks is increasingly interested and developed.

The research problems of identifying IoT devices also have different specialized ways. In the paper [7], the author has synthesized and analyzed how to use machine learning in detecting and classifying IoT devices in a very detailed and relatively complete manner. In the paper [7], more specialized, the authors have divided the problem into small aspects such as device-specific pattern recognition, deep learning-enabled device identification, unsupervised device identification, and abnormal device detection. Some

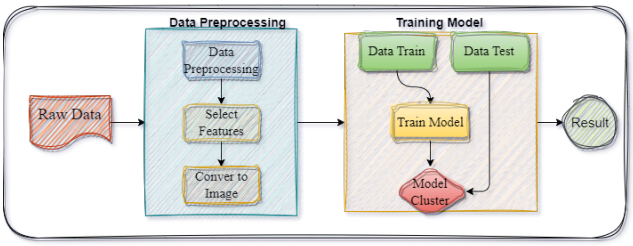


Figure 1. Pipeline for Solving Problem

others, they can use GAN [8] algorithms to extract features, then they use Machine Learning algorithms like Decision Tree, Support Vector Machine (SVM) classification [9, 10]. The results they obtained were relatively good, reaching 95% [11] with the test data set.

In another approach, Jmila *et al*. [12] provides an overview of the use of machine learning-based network traffic analysis for the classification of smart home IoT devices. The paper discusses the challenges associated with smart home IoT device classification and presents various machine learning techniques that have been used for this purpose, including decision trees, support vector machines, and deep learning. The authors also discuss the different types of network traffic that can be used for classification, such as packet header information, packet payload, and flow-based information. The paper concludes with a discussion of future research directions in the area of smart home IoT device classification using machine learning-based network traffic analysis.

Some other studies in transforming data into images and using powerful models of deep learning give better results. This type is widely used in both medical and digital signal processing. Bashivan *et al*. [13] performed the experiment on the data set of electroencephalograms (EEG). They perform transformations to convert the signal to an image form and use Deep Learning for classification. In the article titled:"From ECG signals to images: a transformation based approach for deep learning" [14], the authors have converted the electrocardiogram data to binary data and used classical deep learning networks for classification such as VGG16 [15], AlexNet [16].

With the results of previous researchers, we also expect IoT data after converting into images and using deep learning models to give better results for IoT devices classification.

III. Attention Features Fusion Method (AFF)

In this section, we focus on finding important attributes and converting them to images for unauthorized IoT devices

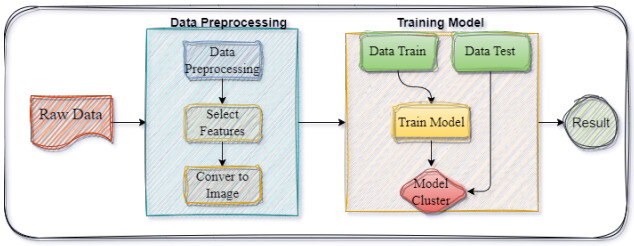


Figure 2. Pipeline for Solving Problem

detection. We collect data in pcap form (pcap file) from IoT devices then pre-process the data to convert the data into csv file format with each record being a data exchange session of each device, including data fields such as ack, ack_A,ack_B, bytes, bytes_ A, bytes_A_B_ratio, bytes_B, ds_field_A, ds_field_B, duration, and many more data fields.

For other methods, researchers use the approach of selecting highly categorical attributes, then feeding into machine learning algorithms or simple Deep Learning models for classification. In this paper, we use all the attributes of the dataset because each attribute has its own classification and will all be effective when we perform the classification based on several machine learning methods.

Our problem-solving approach is to find the best categorical attributes, then use those attributes to convert to images and use deep learning models for classification.

The method we use to find the best attributes through finding the most important attributes for machine learning algorithms like XGBoost, Random Forest, LightGBM [17]. By surveying of the best feature for each algorithm, we combine them to extract the best features for classification.

IV. Attention Features Fusion Method (AFF)

In this section, we focus on finding important attributes and converting them to images for unauthorized IoT devices detection. We collect data in pcap form (pcap file) from IoT devices then pre-process the data to convert the data into csv file format with each record being a data exchange session of each device, including data fields such as ack, ack_A,ack_B, bytes, bytes_ A, bytes_A_B_ratio, bytes_B, ds_field_A, ds_field_B, duration, and many more data fields.

For other methods, researchers use the approach of selecting highly categorical attributes, then feeding into machine learning algorithms or simple Deep Learning models for classification. In this paper, we use all the attributes of the dataset because each attribute has its own classification and will all be effective when we perform the classification based on several machine learning methods.

Our problem-solving approach is to find the best categorical attributes, then use those attributes to convert to images and use deep learning models for classification.

The method we use to find the best attributes through finding the most important attributes for machine learning algorithms like XGBoost, Random Forest, LightGBM [17]. By surveying of the best feature for each algorithm, we combine them to extract the best features for classification.

1. Approach

Well-known algorithms like Random Forest, XGBoost are all based on the classic famous algorithm, Decision Tree as the foundation. To calculate which attribute has the most influence on the final decision, we just need to look at the tree and the weight values in that tree, the larger the value and the closer the node is to the root, the better the attribute, important, the more categorical.

In this paper, we use Scikit-learn[1] library which is a powerful support library for Machine Learning (ML) algorithms. This library supports the Random Forest algorithm and can display the important properties of this algorithm. Similar to the algorithms XGBoost and LightGBM [17] have open source libraries supported, we use *xgboost* and *lightgbm* libraries, then, find the best categorical importance attributes for each algorithm.

To find those important features we used a function in the libraries called feature_importances_. This function is created by the library based on the mathematical formula of the authors Trevor Hastie, Robert Tibshirani, and Jerome Friedman. They presented it in the book **The Elements of Statistical Learning: Data Mining, Inference, and Prediction**. They define the calculation of feature importance using the following equation:

$$I_l^2(T) = \Sigma_{t=1}^{J-1} i_t^2 \mathbb{I}[v(t) = l]$$

In there, T is the whole decision tree, l feature in question, J number of internal nodes in the decision tree, i squared the reduction in the metric used for splitting, II indicator function, v(t) a feature used in splitting of the node t used in splitting of the node. This math formula will sum up all the decreases in the metric for all the features across the tree (Random Forest, Xgboost, LightGBM algorithms are all based on Decision Tree algorithm).

We take the overlap within the 50 best attributes of all 3 algorithms to find the 14 features best classifiers.

[1]https://scikit-learn.org/stable/

Converting those 14 attributes to the image data from 0 to 255, we get the new form data. Such generated data can be fed into classifier algorithms to recognize the unauthorized IoT devices.

We use that image data to train basic deep learning models such as Multilayer Perceptron (MLP), LeNet [18, 19], AlexNet [20], to find the model that best fits the dataset and gives the most accurate prediction results. Then, the traffic will go through a list of models, and return it belongs to which device in the network. If not it belongs to the device outside, then system and we need to issue a warning to the administrator.

2. Data Pre-processing

The dataset that we have is a relatively large dataset, and there are many data fields (features) in it, leading to the training process of many data fields that will cause noise, making the efficiency of the obtained model not high.

Firstly, in order to use this dataset, we need to clean the data, because in the data collection process, it is likely that there will be many missing data columns, we need to remove those missing data. Therefore, we use marking it in this dataset as "?" and during data pre-processing we convert "?" then to "zero" to facilitate data processing, and the model can learn from that data.

As described above, we select the best attributes that each algorithm considers important and highly categorical, convert to images, and use deep learning models for classification.

We proceed to create sub-datasets to train each device by taking all the data of that device and assigning it to "1" at random and randomly assigning the data in the remaining devices to "0" so that the number of data of the device and not of the device are equal.

Data after each train dataset for each device is created, we divide the dataset into train dataset and the dataset to test the model is generated at the rate of 80 to 20, $i.e$, 80% data for train model and 20% data for model testing.

To better understand the whole process we analyze and approach the problem, we can see Figure 2, which describes our steps and solution, from data preprocessing, to data transformation, model training and testing with test dataset.

3. Data Transformation

The first task after we have selected the important attributes with the data set is that we convert the data into images. Some of the attributes we found include ttl_min, B_is_system_port, ttl_A_sum, ttl_max, ssl_dom_server_name_alexaRank and some other attributes, are suitable for image-based classification.

After selecting the important attributes by taking overlap within the 50 most important attributes of machine learning algorithms such as XGBoost, Random Forest, LightGBM. We concatenate the selected features as new features, then convert it to between 0 and 255.

Our purpose is to use AI models that can run similar to the Mnist dataset [21], we try to convert the selected data to the same form as the Mnist dataset which is $28 \times 28 \times 1$. We have tested on two methods. Firstly, we take those 14 important features and padding add zero to the dataset so that there are 784 columns. However, we see that in terms of images, the images representing the data exchange sessions of different devices do not have a certain difference, so we made the next test. In this next test, we do not add zero padding to the dataset, but instead, we padding the dataset itself, ie. 14 existing features, into the dataset until 784 columns are reached. After converting it to an image, the difference between sessions of different devices can be seen.

As seen on Figure 3, we can easily see the difference between the IoT devices that are exchanging data in the network. The results show that the devices produce relatively different images, and the device sessions are relatively similar. Because the sessions of the same device have negligible difference, we cannot see the difference of those sessions with the naked eye. With the above results, we expect the AI models to be able to distinguish well between devices, which is the basis for detecting unauthorized devices in the IoT network.

4. Model Architecture

After converting the data into images, we use well-known deep learning models to train and predict the classification of IoT devices. The models that we use are Multilayer Perceptron(MLP) [22], LeNet, AlexNet, these are models with very simple and primitive architecture of deep learning models.

We use Ensemble Learning [23, 24] with voting type. After converting all 14 features into images, we feed them to the training with a Voting model. The data is put through training the MLP, LeNet, AlexNet models. After training, we get the prediction models with different accuracy. Based on our experience, we weight the MLP, LeNet, and AlexNet models at 1, 2, and 2 respectively.

When a session of a certain IoT device passes, the data is passed through the prediction model. The model will then give that session data to each submodels one by one, and the submodels will predict different values. Based on the predicted value of each model, the voting model multiplies the weighted values and averages them all. The label that

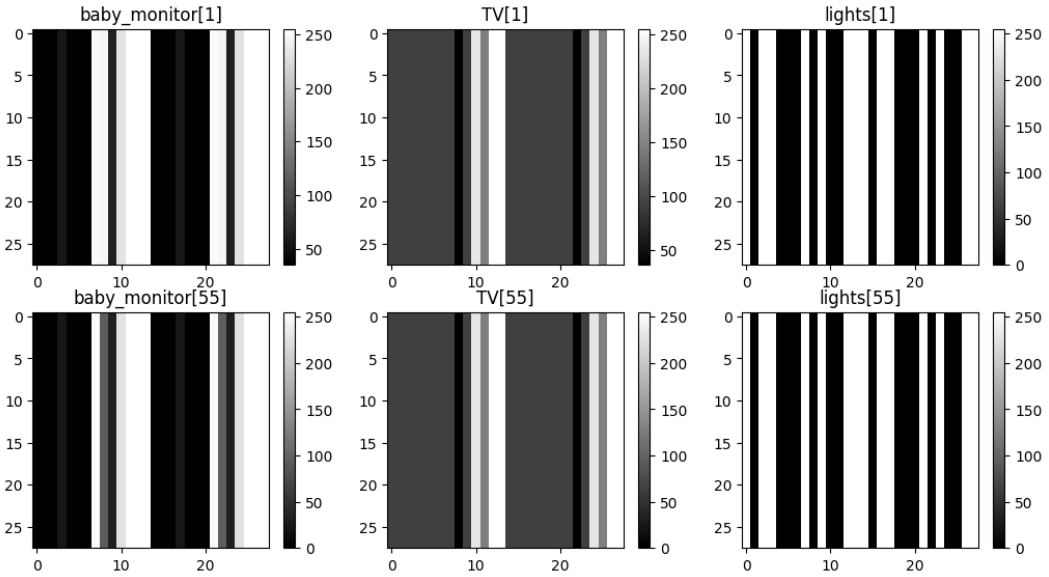


Figure 3. Several Sessions of the IoT Devices after Converting to Images

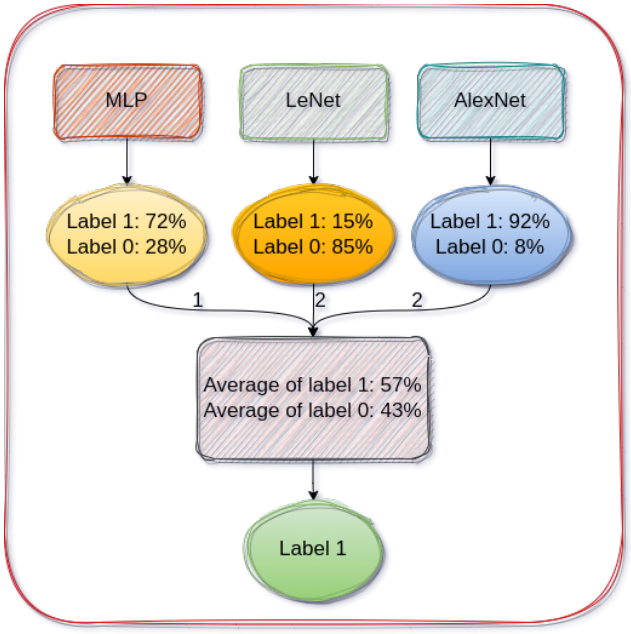


Figure 4. Predictive Voting Model

gives the highest prediction result will be taken as the output of the Voting model. We describe the operation and prediction of the voting model in Figure 4.

V. Experimental Results

1. Experimental environment and Parameter setting

Our experimental environment is on a PC with the following configuration: Intel i7-10700f CPU, 32 GB RAM, and 16GB VRAM RTX A4000 VGA card. We choose python language and jupyter notebook platform to perform the experiments. When we train the model, we use python version 3.6.13, scikit learn 0.24.2, tensorflow 2.6.2, numpy 1.19.2, pandas 1.1.5 and some other tools to analyze and train the model in the experiment.

In this paper, our proposed solution uses basic deep learning models such as MLP, LeNet, and AlexNet. These models have been around for a long time and are relatively suitable for the data set we have. Based on our experience in training each submodel, we found that LeNet and AlexNet models have relatively better accuracy than MLP models. The accuracy of the MLP is about 15% to 20% lower than the rest of the models.So we have set the parameters for model voting with weights for submodels(MLP, LeNet, AlexNet) of 1, 2, 2 respectively.

After we apply that weight to the voting model we get the final model which gives better accuracy in most devices with prediction accuracy increasing from 3% to 12%.

2. Datasets

The dataset we use consists of ten IoT devices such as: security camera, TV, smoke detector, thermostat, water sensor, watch, baby monitor, motion sensor, lights, socket, which are common devices commonly used to build smart homes. Our dataset consists of 298 columns and 399544 rows. This is a relatively large data set and has many different attributes, in which there are also many data attributes that do not have much classification effect.

TABLE I
RESULT USING ML WITH ORIGINAL DATASET

Device \ Algor	DT	RF	XGB	LoRC	LGBM
baby monitor	**97.70**	96.65	94.95	49.21	91.88
lights	92.45	94.80	**97.52**	50.21	93.40
motion sensor	93.75	91.90	89.95	49.58	**96.95**
security camera	94.65	94.91	**97.95**	49.75	90.52
smoke detector	**98.52**	96.01	93.42	89.75	94.53
socket	91.15	92.57	**93.76**	51.09	91.01
thermostat	91.08	91.68	92.52	50.85	**97.56**
TV	**97.65**	95.83	94.86	49.96	93.85
watch	94.62	97.66	**98.64**	50.88	96.64
water sensor	82.99	86.64	**86.86**	51.33	86.61

During operation, the system has collected session data of IoT devices, but this raw data set is difficult for models to learn and classify between devices. The reason is that the excessive amount of noise exists in the data set, exists in attributes that have no categorical value, leading to wrong learning models and inaccurate predictions.

To solve that problem, we propose the AFF method to select the features with the best classifier ability, convert it into an image and pass it through deep learning models for prediction.

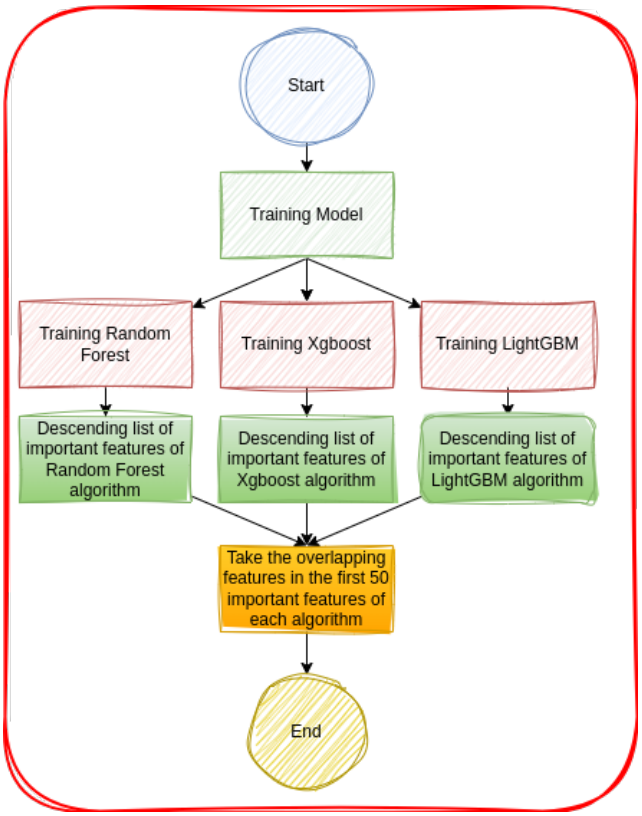


Figure 5. The process to take important features

3. Results of Each Device Classification with The Original Dataset

In this part of the experiment, we tried training the original dataset with machine learning algorithms like Decision Tree (DT), Random Forest (RF), XGBoost (XGB), Logistic (LoRC), LightGBM (LGBM). The obtained results are the basis for us to perform the steps of selecting features and comparing with our method later.

For this method, we simply put all the features into the algorithm and expect that the model will be able to filter out the noise and give the best results. Based on Table I we see, Logistic Regression algorithm gives the lowest accuracy with 49.21% in baby monitor devices, and Xgboost algorithm has the highest accuracy in many devices with 98.64% in watch devices. The table above shows that using the original features is also a relatively good problem-solving approach, but it is not the optimal solution.

4. Results of Each Device Classification with AFF

In this part of the experiment, we find the features with the best classifier, then do experiment with basic Machine Learning algorithms. Then, we continue the experiment by converting the 14 features with the best classifier into an image and using Deep Learning models to classify the generated image.

Finally, we experiment with the voting model belonging to ensemble learning. With the weights of basic deep learning models such as MLP [22], LeNet[2], AlexNet[3] are 1, 2, 2, respectively.

a) The best features of the classification model

After training the machine learning model with all the features, we get the features that are the most important for each algorithm. These features have the best classifier ability with each of those algorithms.

There are many ways to find the most important features of each algorithm. This can include methods such as based on mean decrease in impurity or feature permutation. In this article, we used based on mean decrease in impurity to perform the experiment. This function is used in scikit-learn, xgboost, lightgbm libraries under the function name feature_importances_. It will calculate and rate the importance of each feature to the algorithm. Based on that we find the most important features for each algorithm.

[2]https://paperswithcode.com/method/lenet
[3]https://paperswithcode.com/method/alexnet

TABLE II
THE MOST IMPORTANT FEATURES FOR EACH ALGORITHM

Feature \ Score	XGB	LGBM	RD
ttl_min	**0.57899**	**147**	**0.09298**
B_is_system_port	0.01973	61	0.06732
ttl_A_sum	0.00751	82	0.00327
ttl_max	0.00673	32	0.00554
ssl_dom_server _name_alexaRank	0.00415	78	0.00215
ttl_thirdQ	0.00210	73	0.00728
bytes_A_B_ratio	0.00201	87	0.03479
ttl_B_stdev	0.00070	62	0.00825
reset	0.00065	42	0.03416
ttl_firstQ	0.00064	49	0.00728
ttl_stdev	0.00056	94	0.06376
packet_inter_arrivel_B_min	0.00055	74	0.00914
ttl_B_firstQ	0.00049	30	0.00727
ttl_B_avg	0.00048	39	0.00772

Within the 50 most important features for each of those algorithms, we take overlap and get the important features with all 3 algorithms: Random Forest, XGBoost, and LightGBM. We obtained the 14 most important features with all 3 algorithms within the 50 most important features of the algorithms.

Table II gives us the weight scores of the most important attributes for each algorithm. In all 3 algorithms ttl_min(time to live min) feature is always the most important feature.

b) Results using Machine Learning with 14 selected features

We use classic machine learning algorithms such as Decision Tree (DT), Random Forest (RF), XGBoost (XGB), Logistic (LoRC), LightGBM (LGBM) [17]. These algorithms give relatively good results for the individual training of each device.

After finding the 14 most important features, we proceed to select those 14 features out of a total of 297 features of the original dataset to obtain a new dataset. In this part of training each device, once we have training data for each device, we divide it according to the ratio 80 and 20 to generate training data and testing data to calculate the quality of the model after being trained for each device.

As we can see in Table III, after using the most important features and retraining with machine learning models, significantly better accuracy was obtained in some algorithms for some devices.

c) Results Using Attention Features Fusion Method (AFF)

We use basic deep learning models such as MLP [22], LeNet, AlexNet and Attention Features Fusion Method (AFF) proposed by us to solve the problem. This is the next step after we have selected the best attributes to classify IoT devices. We convert those 14 features into grayscale images with range from 0 to 255, then, train with MLP, LeNet, AlexNet models.

TABLE III
RESULT USING ML WITH 14 SELECTED FEATURES

Device \ Algor	DT	RF	XGB	LoRC	LGBM
baby monitor	95.70	95.65	**97.95**	50.21	96.88
lights	93.46	96.80	**97.80**	49.51	97.40
motion sensor	91.75	95.90	**96.95**	47.58	**96.95**
security camera	95.65	96.81	**97.95**	49.75	97.52
smoke detector	93.52	95.01	95.42	44.75	**96.53**
socket	93.25	**95.87**	95.76	50.99	95.01
thermostat	93.38	95.38	96.52	51.35	**96.56**
TV	93.65	**96.83**	95.86	49.96	94.85
watch	97.62	**97.66**	98.64	50.88	96.64
water sensor	83.07	85.40	86.62	47.97	**87.11**

TABLE IV
RESULT USING DL WITH IMAGE DATASET

Device \ Model	MLP	LeNet	AlexNet	AFF
baby monitor	98.70	99.03	99.15	**99.19**
lights	65.89	86.80	87.80	**88.09**
motion sensor	96.75	98.90	98.95	**99.01**
security camera	97.65	98.15	99.05	**99.22**
smoke detector	89.52	94.01	93.42	**97.32**
socket	66.25	**87.87**	81.76	83.51
thermostat	95.38	98.38	**99.02**	99.12
TV	94.15	96.30	**97.62**	97.53
watch	90.62	95.60	97.64	**98.02**
water sensor	81.09	65.64	91.86	**95.06**

For the AFF method, we conduct training with the voting model belonging to the ensemble learning. We use the basic models are MLP, LeNet, AlexNet and the weights are 1, 2, 2 respectively.

Based on our experimental results in Table IV. We find a set of models that have the best classification for each type of device. Based on the algorithms and methods that we have tested, we can easily choose that model set.

Looking at the chart (Figure 6), we can see that the AlexNet model trained by the image converted from 14 features gives very good results with the baby monitor.

Table IV shows that, our proposed method gives relatively good results and gives better accuracy than using all the features of the dataset. Models like MLP [22], LeNet, AlexNet are also very suitable for classifying the images we provide.

After performing the experiments, we found that some models give the highest accuracy with each device as shown

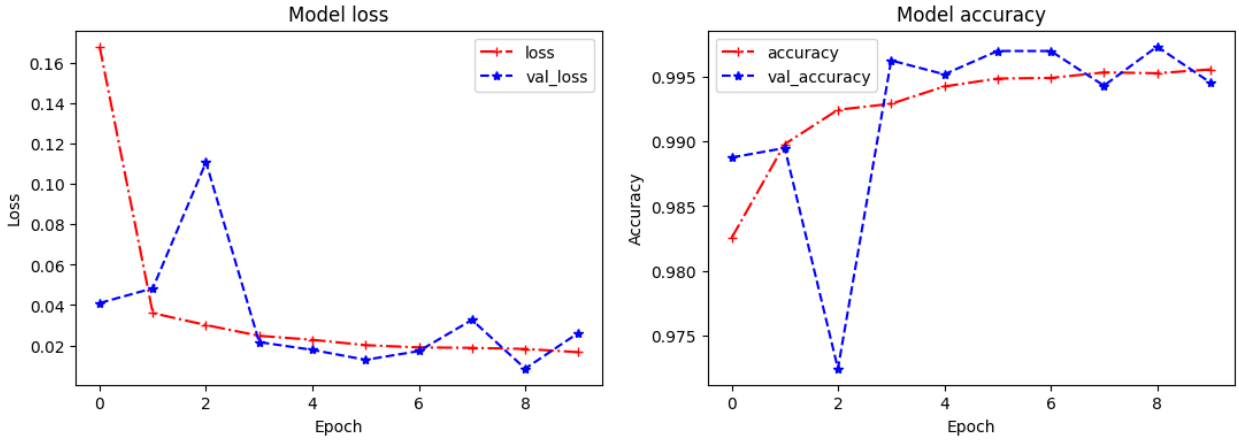


Figure 6. Baby Monitor Device Accuracy and Loss

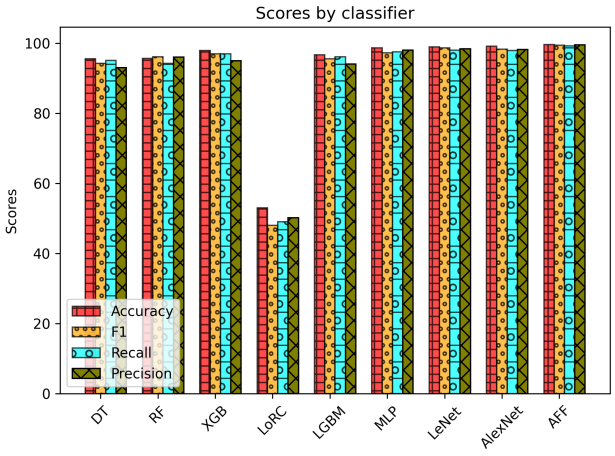


Figure 7. Scores of baby monitor's Classification Models

in Table V.

TABLE V
BEST MODELS FOR EACH DEVICE

Device	Model	Accuracy
baby monitor	AFF	99.19
lights	AFF	88.09
motion sensor	AFF	99.01
security camera	AFF	99.22
smoke detector	AFF	97.32
socket	LeNet	87.87
thermostat	AlexNet	99.02
TV	AlexNet	97.62
watch	AFF	98.02
water sensor	AFF	95.06

From here, we get the model set that gives the best prediction results for each device as listed in the table above.

Based on the Table V we can see that the AFF model runs the best results in most devices, and runs relatively well (but not the best) in a few remaining devices.

The graph of Accuracy, F1, Recall, Precision (Figure 7) shows that the ML algorithms and the DL model both give relatively good results when running with 14 important features selected, except for the Logistic algorithm. Model AFF proved to be superior to other DL models when giving Accuracy, F1, Recall, Precision results a bit higher and more balanced than other models.

5. IoT Device Detection System Results

When we get the models that give the best classification results, we use it to test with the original test dataset. We pass each record through the models in turn. The models will check and return the results that the record belongs to that device or not. If yes, end the test process, otherwise the data will be pushed to the next model in the system for testing. When we have checked all the trained models in the system, but still can not find any device in the system, we conclude that this is a strange device that is not in the pre-registered whitelist of the system, then alert the system administrator.

Performing that test method, we obtained a very high accuracy for the system to detect strange IoT devices accessing the system, the detection results were up to 97.85%. We tested it with a separate dataset that was not seen in any individual device training, and the results were very encouraging as the system reached a good alert level for the system. This can help system administrators better detect strange IoT devices of objects that attack the IoT system.

Our unregistered device detection system works according to Figure 8: Conduct a test of the network traffic of the devices, we will put that network traffic through

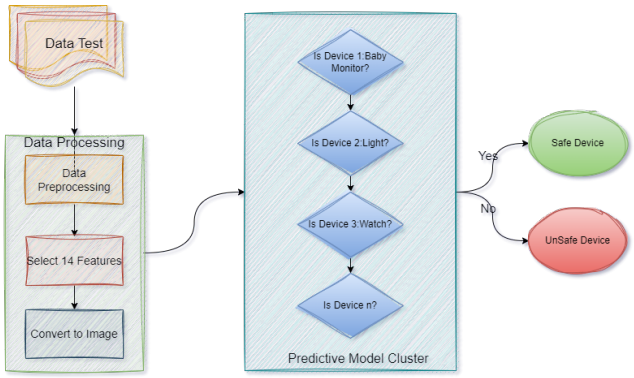


Figure 8. Predictive Model of The System

each device classification model, if it is an organization's device and has been registered, it will return a safe device result. On the contrary, if you have gone through all the classification models and still do not recognize the device of the system, return the device as unsafe and alert the system administrator.

VI. Conclusion

We have proposed and proven that it is possible to build a system to classify IoT devices based on the use of network traffic, and propose a plan to be able to classify that IoT device based on the application of technology deep learning technique.

Our recommended technique (AFF) achieves very good results when conducting tests on our test augmented dataset. The obtained results are the basis for practical application of IoT device classification methods.

In the coming time, we will conduct in-depth research into ways to extract information from network traffic data, such as classifying and detecting the operating systems of IoT devices that are exchanging data in the network system.

References

[1] L. S. Vailshery, "Number of internet of things (iot) connected devices worldwide from 2019 to 2021, with forecasts from 2022 to 2030," Available at https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/ (2022/27/12).

[2] Cisco, "Cisco annual internet report (2018–2023)," Available at https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf (2022/12/06).

[3] T. Robinson, "Interpol warns iot devices at risk," Available at https://www.scmagazine.com/news/architecture/interpol-warns-iot-devices-at-risk (2022/27/12).

[4] J. Margolis, T. T. Oh, S. Jadhav, Y. H. Kim, and J. N. Kim, "An in-depth analysis of the mirai botnet," in *2017 International Conference on Software Security and Assurance (ICSSA)*, 2017, pp. 6–12.

[5] "Shodan," Available at https://www.shodan.io/ (2022/27/12).

[6] "Owasp iot top 10 2018," Available at https://owasp.org/www-pdf-archive/OWASP-IoT-Top-10-2018-final.pdf (2022/27/12).

[7] Y. Liu, J. Wang, J. Li, S. Niu, and H. Song, "Machine learning for the detection and identification of internet of things devices: A survey," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 298–320, 2022.

[8] S. Sivanandam and S. Deepa, *Genetic Algorithms*. Springer Berlin Heidelberg, 2008, pp. 15–37.

[9] F. Nie, W. Zhu, and X. Li, "Decision tree svm: An extension of linear svm for non-linear classification," *Neurocomputing*, vol. 401, pp. 153–159, 2020.

[10] M. Arun Kumar and M. Gopal, "A hybrid svm based decision tree," *Pattern Recognition*, vol. 43, no. 12, pp. 3977–3987, 2010.

[11] A. Aksoy and M. H. Gunes, "Automated iot device identification using network traffic," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–7.

[12] H. Jmila, G. Blanc, M. R. Shahid, and M. Lazrag, "A survey of smart home iot device classification using machine learning-based network traffic analysis," *IEEE Access*, vol. 10, pp. 97 117–97 141, 2022.

[13] P. Bashivan, I. Rish, M. Yeasin, and N. Codella, "Learning representations from eeg with deep recurrent-convolutional neural networks," 11 2015.

[14] M. Naz, J. H. Shah, M. A. Khan, M. Sharif, M. Raza, and R. Damaševičius, "From ECG signals to images: a transformation based approach for deep learning," *PeerJ Comput Sci*, vol. 7, p. e386, Feb. 2021.

[15] H. Qassim, A. Verma, and D. Feinzimer, "Compressed residual-vgg16 cnn model for big data places image recognition," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, 2018, pp. 169–175.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25. Curran Associates, Inc., 2012.

[17] S. Ray, "A quick review of machine learning algorithms," in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, 2019, pp. 35–39.

[18] L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, Y. LeCun, U. Muller, E. Sackinger, P. Simard, and V. Vapnik, "Comparison of classifier methods: a case study in handwritten digit recognition," in *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3 - Conference C: Signal Processing (Cat. No.94CH3440-5)*, vol. 2, 1994, pp. 77–82 vol.2.

[19] T. Li, D. Jin, C. Du, X. Cao, H. Chen, J. Yan, N. Chen, Z. Chen, Z. Feng, and S. Liu, "The image-based analysis and classification of urine sediments using a lenet-5 neural network," *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, vol. 8, no. 1, pp. 109–114, 2020.

[20] S. Lu, Z. Lu, and Y.-D. Zhang, "Pathological brain detection based on alexnet and transfer learning," *Journal of Computational Science*, vol. 30, pp. 41–47, 2019.

[21] A. Baldominos, Y. Saez, and P. Isasi, "A survey of handwritten character recognition with mnist and emnist," *Applied Sciences*, vol. 9, no. 15, 2019.

[22] I. García-Magariño, R. Muttukrishnan, and J. Lloret, "Human-centric ai for trustworthy iot systems with explainable multilayer perceptrons," *IEEE Access*, vol. 7, pp. 125 562–125 574, 2019.

[23] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on

ensemble learning," *Frontiers of Computer Science*, vol. 14, no. 2, pp. 241–258, Apr 2020.

[24] O. Sagi and L. Rokach, "Ensemble learning: A survey," *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.

Duong Van Dan received a Bachelor's degree from Le Quy Don technical university, Hanoi, Vietnam, in 2023. His research interests lie in the area of computer science, network security.
Email: duongvandan2806@gmail.com

Ta Minh Thanh is currently an associate professor and vice dean of Institute of information technology and Communication (IITC), Le Quy Don Technical University, Vietnam. He is also a Postdoctoral Fellow of the Department of Mathematical and Computing Sciences at Tokyo Institute of Technology. He received his B.S. and M.S in Computer Science from National Defense Academy, Japan, in 2005 and 2008 and his Ph.D. from Tokyo Institute of Technology, Japan, in 2015, respectively. He is a member of IPSJ Japan and IEEE. His research interests lie in the area of watermarking, network security, and computer vision.
Email: thanhtm@lqdtu.edu.vn

Vietnamese Scene Text Detection Method based on Deep Learning

Huynh Van Huy [1], Nguyen Thi Thanh Tan[2], Ngo Quoc Tao [3]
[1] Lac Hong University, Vietnam
[2] Electric Power University, Vietnam
[3] Institute of Information Technology - Vietnam Academy of Science and Technology, Vietnam
Correspondence: Nguyen Thi Thanh Tan, tanntt@epu.edu.vn
Communication: received 18 Aug 2023, revised 28 Sep 2023, accepted 23 Nov 2023
Digital Object Identifier: 10.32913/mic-ict-research.v2024.n1.1242

Abstract: This paper proposes an efficient method for detecting Vietnamese text in outdoor scene images. Essentially, the text detection method presented here is based on the idea of utilizing deep learning network architectures to learn various geometric properties in order to reconstruct polygonal representations of text regions. The effectiveness of the method has been evaluated on four real-world outdoor scene image datasets, including the ICDAR 2015, Total-Text, VinText, and VnSceneText datasets. Experimental results show that the proposed method can detect text of various shapes and sizes with high and consistent accuracy. Specifically, the method achieved Precision, Recall, and Hmean scores of 87.53%, 86.94%, and 87.23%, respectively, on the test datasets, 84.32%, 88.17%, and 86.20% on a different dataset, 85.63%, 87.94%, and 86.77% on yet another dataset, and 85.14%, 87.23%, and 86.17% on the last dataset. The experimental results indicate that this approach is feasible for detecting Vietnamese text in outdoor scene images.

Keywords: *Scene text, scene images, text regions, segmentation, detection, features, mapping, accuracy, recall, harmonic mean, convolution, scale, batch, batch normalization.*

I. Introduction

The traditional problem of text or document recognition (OCR) has been the subject of research investment for several decades. The input for this problem mainly consists of document images acquired from scanning devices (scanners). It can be said that the traditional text recognition problem has been effectively addressed, and OCR systems commercially available on the market now meet and satisfy user requirements.

In recent times, stemming from practical needs, the application of text recognition has expanded into various fields [1, 2] exemplified by: Detecting and retrieving information (origin, composition, user recommendations, etc.) from signs, product packaging, and product labels to enhance customers' understanding of products or services in the commercial sector. Employing OCR to assist machines in reading and recognizing labels, and technical specifications on components in the industrial sector. Reading information on road signs, directional signs, and street names to support visually impaired individuals in navigating or developing autonomous vehicle systems. Additionally, integrating advanced technologies like text recognition, machine translation, and speech recognition for translating comprehensible content from signs, menus, and instructions into the user's (customer's) language is becoming increasingly popular in the tourism industry as well as daily life. The common characteristic of these expanded applications is that the input images are typically irregular in shape and captured or recorded using various handheld devices (smartphones, webcams, tablets, iPads, cameras, etc.) under entirely natural conditions (with no constraints on lighting, angles, shooting distances, as well as the area, tilt/tilt of the text-containing image, etc.). This class of input images is referred to by a general term called "scene images." Detecting text in scene images or videos is also known as scene text detection.

Compared to traditional OCR, text detection in scene images often involves significant complexity and faces numerous challenges due to the following factors: i) The images are highly complex due to containing various objects and different types of noise; ii) Text in images is typically non-standard (irregular) with diverse shapes, sizes, orientations, colors, and different font styles; iii) There is no uniformity among the images as they are captured from various angles, distances, lighting conditions, and different resolutions; iv) Text can be occluded by other objects in the image or by various factors such as blurriness, darkness, glare, or noise; v) Scene text detection tasks often require processing a large volume of images or videos, demanding

algorithms capable of handling data quickly and efficiently. This paper proposes an effective method for detecting Vietnamese scene text based on a deep learning approach. The main idea of the method is to use deep neural network architectures to learn various geometric properties for reconstructing polygonal representations of text regions. The proposed model is composed of four main components, including: (1) A backbone network for extracting features from the input image; (2) A feature fusion model constructed as a Feature Pyramid Network (FPN) to integrate and represent features at multiple scales, aiding in the detection of text regions of various sizes; (3) A context attention model structured as a deep convolutional network, trained to capture context information (long-range dependencies) of pixels to obtain more representative features and (4) A text instance segmentation algorithm in the final step, which is responsible for identifying and grouping boundary points into text regions. Our main contributions in this paper include:

- First, we have proposed an effective method for detecting Vietnamese text in scene images or videos with high accuracy, that is less affected by noise and the orientation of the text. The method is capable of adapting to and performing well with text of various shapes, including curved text.

- Second, we have incorporated a contextual attention mechanism that allows the model to learn how to create a weight matrix (referred to as the attention matrix). This matrix indicates the importance of each pixel in the image for text detection. Regions with higher attention values will be given more consideration by the model. This helps reduce the algorithm's processing time and diminishes the impact of unimportant factors in the image. As a result, it simplifies and enhances the efficiency of subsequent text segmentation and post-processing stages.

- Third, we have collected and constructed a dataset comprising 3000 Vietnamese scene images to facilitate research and evaluate the testing of algorithms for text detection and recognition in Vietnamese scene images. In addition, we have applied advanced image processing techniques such as transformations, synthesis, and image augmentation to enrich the training dataset for building and testing deep learning algorithms.

The remaining contents of the paper are presented as follows: Section 2 provides a summary of related approaches. The main idea and the implementation steps of the proposed method are detailed in Section 3. Section 4 presents the experimentation and evaluation process of the method's effectiveness. Some conclusions and avenues for further development are discussed in Section 5.

II. Related works

Numerous methods have been proposed to address the problem of scene text detection. Traditional approaches have focused on image processing techniques and basic machine learning algorithms, such as sliding window methods and connected component analysis [1]–[8]. In practice, these methods have exhibited several limitations and achieved low detection performance on real-world scene images. They can particularly fail in challenging scenarios, such as detecting curved or skewed text, text with many characters joined together, or text captured under non-uniform lighting conditions [1], [9].

For this reason, most methods proposed in recent years have adopted a deep learning approach. Fundamentally, these methods can be categorized into three main approaches: bounding-box regression-based methods, part-based methods, and segmentation-based methods. The first approach, bounding-box regression-based, utilizes regression models to predict bounding boxes for text regions. M. Liao and colleagues [10] adjusted the anchors and aspect ratios of the convolutional layers based on SSD [11] for text detection. In [12]-[13], authors applied quadrilateral regression to detect multi-oriented text. P. He and colleagues [14] proposed an attention mechanism to preliminary identify text regions. In [15], the authors introduced the RRD method, separating classification and regression. In this method, rotation-invariant features are used for classification, while rotation-sensitive features are employed for regression to achieve better performance, particularly for long and multi-oriented text. In [16], [17], authors presented anchor-free methods, applying pixel-level regression for multi-oriented text regions. L. Xie and colleagues [18] introduced an RPN (Region Proposal Network) architecture to address the scaling issue in scene text detection. Generally, regression-based methods often employ simple post-processing algorithms like Non-Maximum Suppression (NMS). However, most of these methods do not achieve the expected accuracy for input text with irregular shapes, such as curved or covered text.

For part-based text detection methods, the approach initially involves detecting small components of text regions and then linking them to form words' bounding boxes or text line bounding boxes. In [19], the authors discovered bounding boxes for text segments and predicted their connections to handle long text regions effectively. J. Tang et al. [20] proposed an instance-aware component grouping algorithm to separate text regions more efficiently and improve the linking algorithm to accommodate text regions with arbitrary shapes. These methods generally achieve good performance in detecting long text lines. However, the linking algorithms are relatively complex with

manually tuned hyperparameters, making it challenging to adjust the algorithms and adapt to the diversity of input data effectively. Segmentation-based methods often combine pixel-level predictions with post-processing algorithms to determine bounding boxes. Zhang et al. [21] detected multi-oriented text using semantic segmentation and MSER-based algorithms. Xue et al. [22] used contour lines to separate text regions. In [23] and [24], the authors detected arbitrarily-shaped text regions using an instance segmentation approach based on Mask R-CNN [25]. W. Wang et al. [26] proposed progressive scale expansion by segmenting text regions with different scale kernels. Tian et al. [27] introduced a pixel embedding mechanism to cluster pixels from segmentation results. In [10] and [27], the authors presented new post-processing algorithms for segmentation results, leading to the lower inference speed. Fast methods for text detection in scene images focus on accuracy and inference speed. Recently, Pengfei Wang and colleagues [28] proposed the SAST method for detecting arbitrary-shaped text in scene images. This method uses a context-aware multi-task training framework based on the Fully Convolutional Network (FCN) architecture to learn various geometric properties for polygon representation of text regions. Experimental results show that SAST can accurately detecting text with arbitrary shapes and quickly process image. However, this method has limitations in detecting extremely large or small characters and text in highly curved images. Zhu and colleagues [29] introduced the FCE (Fourier Contour Embedding) method to represent the outline of arbitrarily shaped text as compact Fourier series symbols. They built the FCENet architecture with a backbone network, feature pyramid networks (FPN), and a simple post-processing model using Inverse Fourier Transform (IFT) and the Non-Maximum Suppression (NMS) algorithm. Experimental results demonstrate that this method achieves high accuracy on the CTW1500 and Total-Text datasets. However, it may encounter challenges when the text and background lack clear contrast or are oversized. M. Liao and colleagues [30] proposed a text detection method for arbitrary-shaped text using the Differentiable Binarization algorithm and the Adaptive Scale Fusion (ASF) mechanism to combine information from different scales of images flexibly. . This method can effectively address text detection in distorted text, and text of various shapes and sizes, achieving high accuracy and fast processing. However, the method's performance may significantly decrease in the presence of heavy noise. Additionally, the method's effectiveness depends on the selected scale threshold, as it performs well when the chosen threshold is suitable for the input image.

For Vietnamese scene text recognition, apart from facing the challenges of scene text recognition in general as mentioned above, there are additional difficulties related to diacritics and tone marks in the Vietnamese language. These diacritics can lead to issues like character and line concatenation within the text, reducing accuracy in detection and recognition. To ensure accurate recognition, it's necessary to integrate specialized processing mechanisms suitable for the specific characteristics of the Vietnamese language. Practical surveys reveal that research outcomes of Vietnamese text detection and retrieval in scene images are still comparatively limited. Some recent notable research results in Vietnamese text recognition have been published [31], [32]. In [31], the authors primarily focused on the text recognition stage. They proposed an approach that incorporates a dictionary into both the training and inference phases of the text recognition system. The dictionary is used to generate a list of possible results and find the most suitable result for the text's display format. This method helps to handle ambiguous cases encountered in practice and improves the overall performance of text recognition frameworks. The authors have contributed to the research community by providing the VinText scene text dataset, which serves research and algorithm quality improvement. In [32], the authors presented experimental results of various methods including DBN, PMTD, PAN, FCENet, RCNN, SATRN, NRTR, and RS on the VinText dataset. Inheriting from existing approaches (EAST [16], SAST[28], DB [29]), our Vietnamese Scene Text Detector (VNSTD) method is proposed to address challenging issues in recognizing Vietnamese scene text in images. It focuses on two main objectives: the capacity of accurately detecting scene text of any shape or size with high precision and fast processing speed and performs well in handling Vietnamese diacritics and tone marks.

III. VIETNAMESE SCENE TEXT DETECTION METHOD

The basic steps in the scene text detection process of the proposed method are described in Figure 1 From any given input image, it first passes through a backbone network to automatically extract features. The extracted features are fed into a fusion model. The fusion model is specially designed to integrate and represent the features at various scales. This processing mechanism aims to address text regions of different sizes.

The output results obtained from the fusion model will continue to be passed to the context attention module to integrate distant dependencies of pixels in order to obtain features with better representational qualities, thereby enhancing the accuracy of the segmentation algorithm in the subsequent step. The results obtained from the context

Figure 1. Vietnamese Scene Text Detection method (VNSTD)

attention module are used as input for the text instance segmentation task. The text instance segmentation step is responsible for locating text regions (identifying the position and shape) in the input image. In the following section, we will describe the implementation steps of the proposed method in detail and with more specificity.

1. Backbone network architecture

The backbone network is responsible for automatically extracting features from the input image. In the proposed model, we use the ResNet-50 network architecture Figure 2(a)

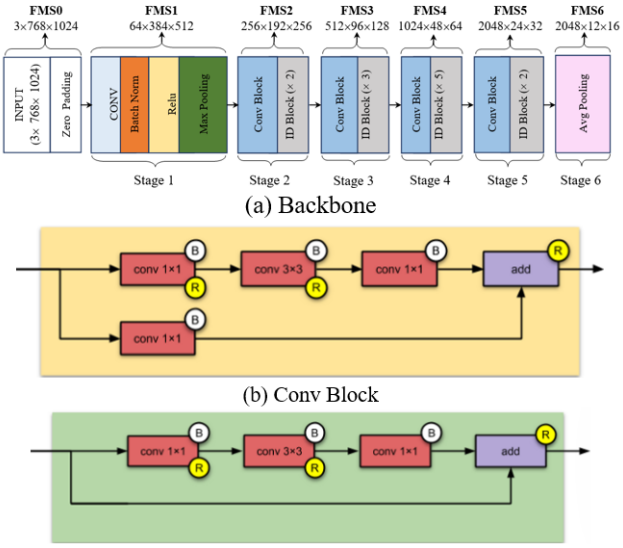


Figure 2. The backbone network architecture

This is the best-rated network architecture at the current time. The remarkable difference between ResNet-50 and traditional CNN models is the use of shortcut connections. This mechanism not only makes training deeper models easier (eliminating the vanishing gradient problem) but also reduces the dependence on the number of layers. This architecture consists of 50 layers: Zero Padding, Convolution, Max Pooling, Convolution Blocks, Identity Blocks, Average Pooling, and Flattening. In this context, the multiplication symbol (×) in the ID block implies multiple stacked ID blocks (for example, ID block (×2) means two stacked ID blocks). Each convolution layer in ResNet is applied with Batch Normalization.

The network execution process is divided into 6 stages, denoted from stage 1 to stage 6. The result of each stage is a set of feature maps. Specifically, with 6 stages, we obtain 6 sets of feature maps (denoted as FMS1 to FMS6). In the proposed method, the input image consists of 3 channels and is normalized to a size of 768×1024. Each channel is considered a feature map in FMS0. Zero padding is of size 3×3. Stage 1 takes the original input as a 3×768×1024 image. The architecture of stage 1 consists of 3 convolution layers (CONV) with 64 filters (size 7×7) and a stride of 2×2, BatchNorm normalization, and MaxPooling (3×3). The output of stage 1 (FMS1) comprises 64 feature maps with a size of 384×512. The feature maps' output from Stage 1 continues to be provided as input for Stage 2. The architecture of stage 2 includes 01 Convolutional block and 02 identity blocks, with 256 filters (size 2×2) and a stride of 2×2. The output of stage 2 (FMS2) consists of 256 feature maps (size 192×256). The feature maps from FMS2 continue to be provided as input for stage 3. The architecture of stage 3 consists of 01 convolutional block and 03 Identity blocks, with 1024 filters (size 2×2) and a stride of 2×2. The output of stage 3 (FMS3) comprises 512 feature maps, with a size of 96×128. The feature maps from FMS3 continue to be provided as input for stage 4. The architecture of stage 4 consists of 01 convolutional block and 05 identity blocks, using 1024 filters (size 2×2) with a stride of 2×2. The output of stage 4 (FMS4) comprises 1024 feature maps, with a size of 48×64. The feature maps from FMS4 are further provided as input to stage 5. The architecture of stage 5 consists of 01 convolutional block and 2 Identity blocks, using 2048 filters (size 2×2) with a stride of 2×2. The output of Stage 5 (FMS5) comprises 2048 feature maps, with a size of 24×32. The feature maps from FMS5 are further provided as input to Stage 6. The architecture of stage 6 consists of 01 Convolutional block and 02 identity blocks, using 2048 filters (size 2×2) with a stride of 2×2. The output of stage 6 (FMS6) comprises 2048 feature maps, with a size of 12×16.

Some representative output results (randomly selected) of the stages are illustrated in Figure 3. In this figure, the input image is depicted, and the notation (FMSi: C m/n) corresponds to the order of the stage and the order of the channel and the total channels (the number of output feature maps) of each stage (for example, the notation FMS1: C 39/64 means the feature map 39 (channel 39) out of a total

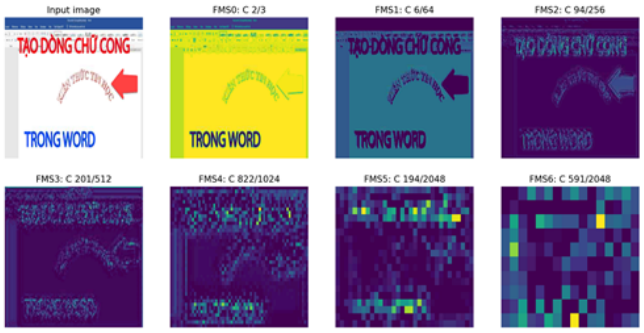


Figure 3. Some representative results of the stages

of 64 output feature maps of stage 1).

The structure of each Conv Block Figure 2(b) consists of four convolutional layers (Conv) and one merge layer (Add). In this structure, the symbol B stands for Batch normalization (applying the Batch normalization technique), and R stands for Relu (using the Relu activation function). Similarly, the structure of each ID block consists of three normalized convolutional layers (Batch normalization) and one merge layer (Add).

2. Fusion model

The Fusion model is described in detail in Figure 4. This model takes as input the 07 feature maps (from FMS0 to FMS6) obtained from the backbone network. This model

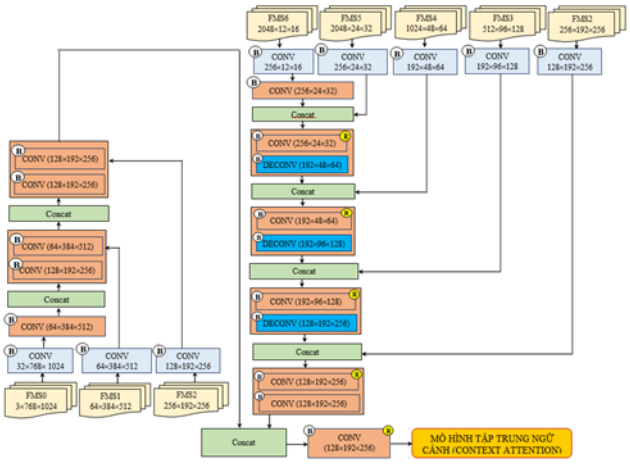


Figure 4. Fusion model

integrates and represents features at multiple scales to detect text regions of varying sizes. Essentially, the architecture of the model is constructed as a Feature Pyramid Network (FPN). This network is structured by two pathways: the bottom-up and the top-down pathways. In each layer of the network, three values correspond to the number of channels, height, and width of the output feature maps.

The bottom-up pathway serves as an encoder network to extract features. As the resolution and feature map size increase, the number of filters (corresponding to the number of output feature maps) and contextual information also increase. The bottom-up pathway yields 128 feature maps (corresponding to 128 output channels). Figure 5(a) illustrates an output result of this pathway (C: 100/128 means feature map 100 out of a total of 128 feature maps - displayed randomly). The top-down pathway acts as a

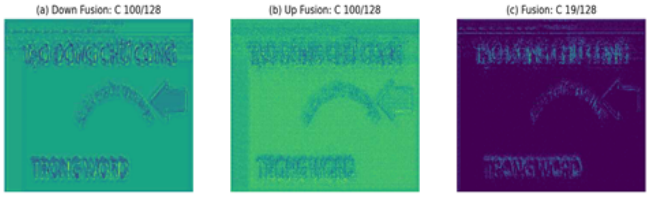


Figure 5. Fusion model

decoder network to construct the object. The execution process of the top-down pathway is the reverse of the bottom-up pathway: as it descends, the resolution increases, the feature map size grows, and the number of filters (number of output feature maps) decreases. Figure 5(b) illustrates an output result of this pathway. The final processing step involves concatenating the bottom-up and top-down pathways using a convolutional layer to produce an output consisting of 128 feature maps, each with a size of 192×256×128 (H = 192, W = 256, C = 128, where H is the height, W is the width, and C is the number of channels) - Figure 5(c). These feature maps are further provided to the context attention model to obtain better representative features, thereby enhancing the accuracy of text detection/segmentation in the subsequent stage.

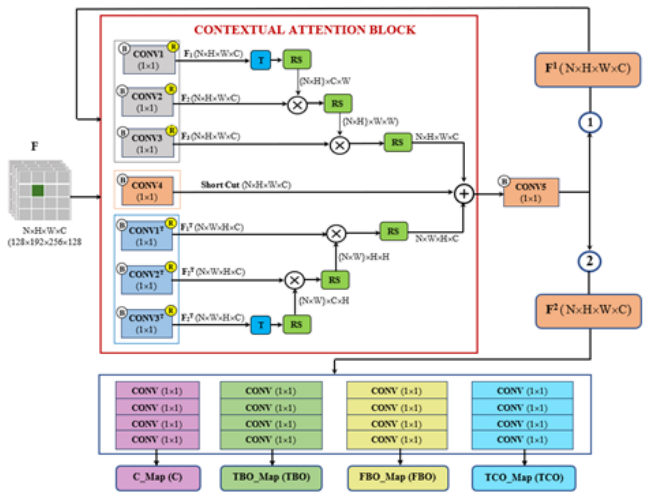


Figure 6. The context attention model

3. Contextual attention

The contextual attention model (Figure 6) inherits the self-attention mechanism from [28], [33]. The aim of the contextual attention model is to aggregate contextual information (at a higher level) to enhance the representation of feature maps, thereby improving the accuracy of text detection/segmentation in the subsequent step. The specific steps of the contextual attention model are described in Figure 6. The Contextual Attention Block computes attention matrices from horizontal or vertical features and integrates contextual information through matrix multiplication between attention matrices and the original features. The transformations used in the contextual attention block include:

Transpose operation of the feature map	T
Reshape the feature map	RS
Multiply two feature maps	⊗
Concatenate two feature maps	⊕

To reduce the computational cost, the contextual attention mechanism here only considers the similarity of each position in the feature map with other positions in the same column (vertical context) or row (horizontal context). The flowchart shown in Figure 6 demonstrates that the contextual attention block is applied twice consecutively to capture both near and far dependencies for each pixel. The specific process is as follows: Starting from the input feature map F (obtained from the fusion model), with a size of N×H×W×C, where N, H, W, and C represent the number of input feature maps, height, width, and the number of channels in each feature map (specifically, N = 128, H = 192, W = 256, C = 128), the steps for collecting horizontal contextual information include

- B1 Perform 3 parallel 1×1 convolution layers (CONV1, CONV2, CONV3) to obtain 3 feature maps,F_1, F_2 and F_3
- B2. Transpose F_1 and reshape the result to size $N \times H \times W \times C$.
- B3. Multiply F_2 with the transposed feature map obtained in B2 and reshape the result to size $N \times H \times W \times W$
- B4 Multiply F_3 with the product feature map obtained in B3 and reshape the result to size $N \times H \times W \times C$. The collection of vertical contextual information is performed based on the transposed feature maps. Specifically, three parallel convolution layers (CONV1T, CONV2T, CONV3T) are executed to obtain three corresponding transposed feature maps (F_1^T, F_2^T, F_3^T) with a shape of $N \times W \times H \times C$. The specific steps are as follows:
- B5. Transpose F_3^T and reshape the result to size $N \times W \times C \times H$.
- B6 Multiply F_2^T with the transposed feature map obtained in B5 and reshape the result to size N×W×H ×H.
- B7 Multiply F_1^T with the product feature map obtained in B6 and reshape the result to size $N \times W \times H \times C$.

The shortcut path is utilized to preserve local features. The final step involves concatenating the vertical contextual feature maps, horizontal contextual feature maps, and the short/cut path feature maps to obtain the output feature map F_1 and reduce the number of output channels using a 1×1 convolution layer.

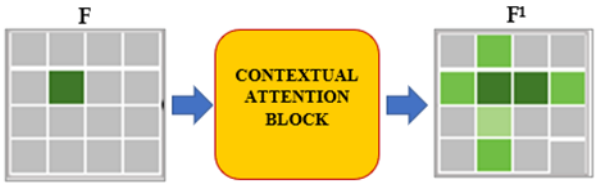


Figure 7. Apply the contextual attention on the feature map F

This mechanism allows the contextual attention block to aggregate contextual information in the vertical and horizontal directions for each pixel. The result of applying the first round of contextual attention on the input feature map F is depicted in Figure 7.
Since the convolution layers (CONV1, CONV2, CONV3) and $(CONV1^T, CONV2^T, CONV3^T)$ share weights, continuing to apply the contextual attention block on the feature map F1 will enable each pixel to capture more distant dependencies from all other pixels. Figure 8 shows that after applying the contextual attention block for the second time, the long-range dependencies for each pixel have been enhanced. In other words, the feature map F^2 is richer in contextual information than to F^1.

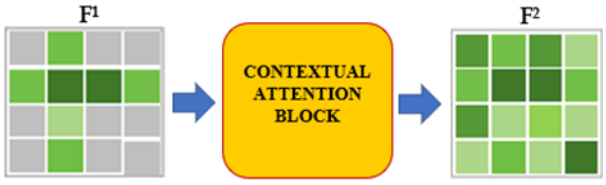


Figure 8. Apply the contextual attention on the feature map F^1

This processing mechanism also helps alleviate issues arising from limited receptive capabilities when handling more challenging text, such as longer text. Figure 9 illustrates the result of applying the contextual attention block to a specific input image. The feature map F^2 from

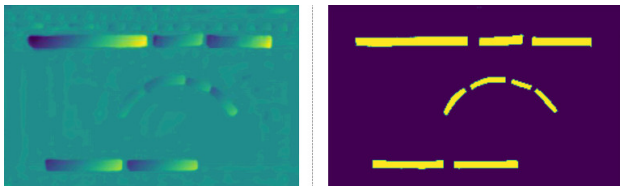


Figure 9. Appling the contextual attention model

the contextual attention will be further used to predict high level features (knowledge) of text regions to enhance the accuracy of the text segmentation algorithm. In this approach, four main types of high-level features are of particular interest: the centerline of text regions, boundary points, corner points of the text polygon, and the offset of points on the centerline from the boundary points.

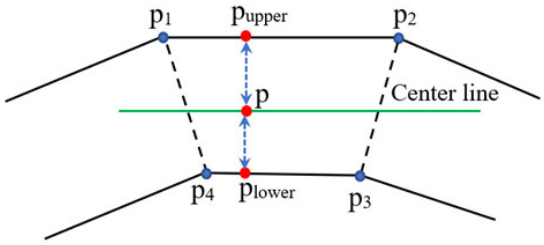


Figure 10. Predict centerline feature map

The extraction of these high level features is handled by four parallel blocks within a Fully Convolutional Network (FCN), as described at the end of Figure 6. Each block consists of four 1×1 convolution layers.The output of each block corresponds to a specific type of high-level feature map that needs to be determined. Specifically, the first convolution block allows for the prediction of the feature map representing the center line of text regions (C) - Figure 10. The second convolution block allows predicting the feature map TBO_ Map (TBO Figure 6). This map defines the upper and lower boundary points (p_upper, p_lower) for each point p in the centerline feature map C and estimates the offset between them (Figure 11). The third convolution

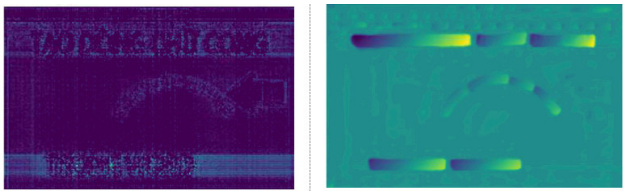


Figure 11. The offset of p and the pair of points (p_upper, p_lower)

block allows predicting the FBO_Map (FBO Figure 6). This map identifies the four corner points of each text region (e.g., four vertices p1, p2, p3, p4 in Figure 11) and estimates the offset of these four points to the pixels in the boundary map C. The fourth convolution block allows predicting the TCO_Map (TCO), determining the offset between the pixels in the centerline map C and the center of the text region's bounding box.

4. Text instance segmentation

Text instance segmentation is the final processing step in the text detection pipeline. The input to this step consists of 128 feature maps (of size $H\times W\times C$, where H=192, W=256, C=128) from the F^2 set obtained from the contextual attention model in the previous stage.
This involves determining the positions of text regions, represented by their enclosing polygons, in the input image. Building upon ideas from [16], [28], [34], [35], the text instance segmentation algorithm is executed as follows:

- B1. Identify the center line of the text region.
- B2. Sample center points along the center line.
- B3. Extract the boundary points of the text region.
- B4. Group the boundary points in a clockwise direction to form text regions.

a) Input image b) Text centerline

Figure 12. Determining the text centerline

Where the text's centerline is regarded as a scaled-down representation of the text region. Figure 12(b) shows the centerline of the input text line in Figure 12(a). Determining the text's centerline is done based on the centerline feature map C (constructed from the contextual attention model mentioned above).
The method of sampling center points is performed from left to right along the centerline, with center points taken at equal intervals (Figure 13). Based on the sampled center

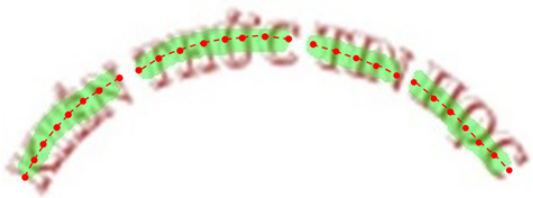


Figure 13. Sampling the center points

points, the following processing step will proceed to determine the pairs of boundary points (upi and loi) Figure 14.

Finally, by connecting the boundary points clockwise, we

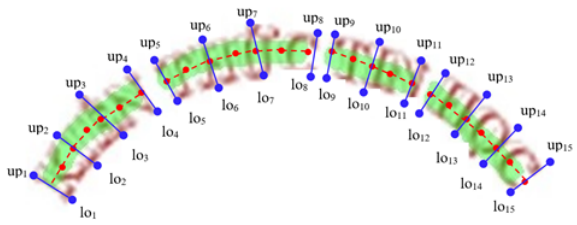


Figure 14. Determine the boundary point pairs

will obtain the complete bounding box (position and shape) of the text regions (Figure 15).

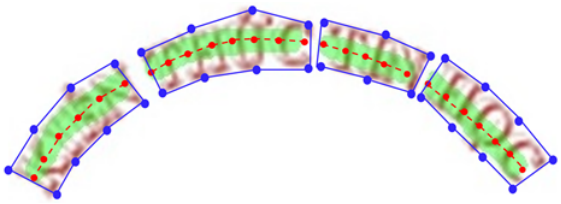


Figure 15. Determine the boundary point pairs

IV. Experimentation

1. Experimental environment

We use the Python 3 environment to implement the algorithm and experimental model. The experimental computer is configured with an Intel Core i7/9700 4.7 GHz (Max Turbo Frequency) processor, 32 GB RAM. The computer is also equipped with Nvidia Tesla K80 12 GB $\times$ 2 GDDR5 graphics card.

2. Experimental Data

Both English and Vietnamese belong to the Latin alphabet system. The Vietnamese alphabet almost encompasses the entire English alphabet. In reality, Vietnamese foreign context documents often contain many English terminologies (even half English and half Vietnamese, for example, on signs, directions, advertisements, etc.). Therefore, to evaluate the method's effectiveness, the authors conducted experiments on both Vietnamese and English datasets. Specifically, four experimental datasets were used, including:

- ICDAR 2015. This dataset was collected for the ICDAR 2015 text recognition competition [36], including 1000 scene text images for training and 500 for testing. The text images were collected using Google Glass, and the text in the images is very diverse and random (Figure 16). This image set has been annotated at the word level.

Figure 16. ICDAR 2015 dataset

- Total-Text. The Total-Text dataset [37] is an outdoor scene image dataset containing curved text with various orientations (multi-oriented). This dataset comprises a total of 1255 training images and 300 testing images (Figure 17). All images in this dataset have been labeled at the word level.

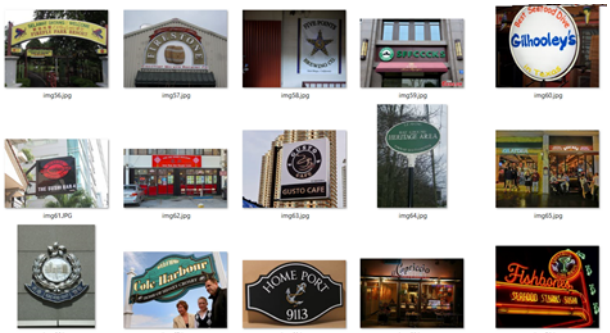


Figure 17. Total-Text dataset

- VinText. The VinText Vietnamese scene text dataset was collected by the VinAI Institute [31], comprising a total of 1200 training images and 300 testing images (Figure 18).

Figure 18. VinText dataset

The images in this dataset were collected in the Hanoi region and its surrounding areas, featuring diverse content,

including images of advertisement signs, street names, shop names, offices, text on various modes of transportation, and more. All images in this dataset have been labeled at the word level.

- VNSceneText. This is a dataset of scene text images collected directly by the authors in the streets of Ba Ria / Vung Tau and Ho Chi Minh City under completely natural conditions using smart phone devices (Iphone 7, Iphone 12, Oppo Reno 8). The dataset includes 3000 images, with 2400 images for training and 600 images for testing. The text in this dataset is very diverse in terms of types (images taken from advertising signs, traffic signs, street names, signs on buildings, vehicles, and images of various other types of documents).

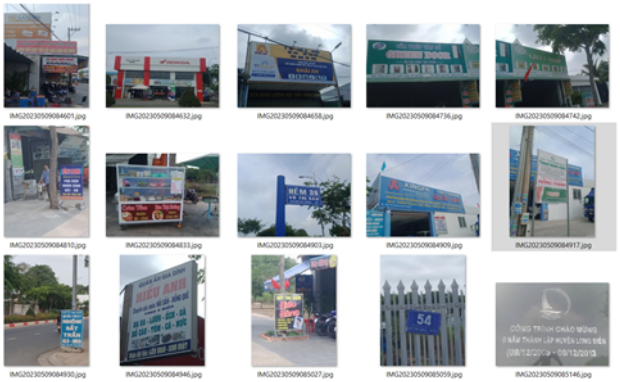
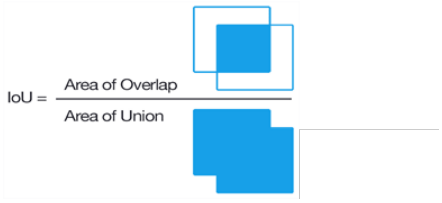


Figure 19. VNSceneText dataset

3. Evaluation Metrics

In the context of object detection in general and text detection in scene images, IoU (Intersection over Union) is commonly used to determine the accuracy of an object or text region detection. IoU is defined as the area of intersec-



tion between the detected text region and the corresponding ground truth text region (stored in the ground truth file) over the area of their union. A text region is considered detected correctly if its corresponding IoU (Intersection over Union) score is not less than a predefined threshold Γ. To assess the effectiveness of the method, we use the Precision, Recall, and H_mean metrics:

- Precision is the ratio of the number of correctly detected text regions to the total number of detected text regions: $Precsion = \frac{T_P}{T_P+F_P}$

- Recall is the ratio of the number of correctly detected text regions to the total number of actual text regions: $Recall = \frac{T_P}{T_P+F_N}$

- H_mean is the harmonic mean between the precision and recall of the detection results: Where T_P (True Positive) is the number of correctly detected text regions in the image, F_P (False Positive) is the number of text regions falsely detected (not actual text but detected as text), and F_N is the text regions that were not detected (missed).

$$H_mean = \frac{2xPrecisionxRecall}{Precision+Recall}$$

To evaluate the runtime performance of the algorithms, we use the FPS (Frames Per Second) metric, defined as the number of image frames processed in one second

4. Experimental Results

The experimental process was conducted sequentially on each dataset. For each dataset, we first trained the network using the training data. To ensure the stability of the model, from the training data of each model, we applied data augmentation techniques such as changing the image scale (random-scale), rotating the image (random-rotate), blurring the image (random-blur), adjusting image contrast (random-contrast), and flipping the image. For each image in the training set, we typically generated an additional 10-15 images for training purposes.

During the training process, the four components of the model (backbone network, aggregation model, contextual attention model, and text instance segmentation) are trained sequentially. In particular, the backbone network is trained using transfer learning with pre /trained weights from ImageNet [38]. The fundamental parameters for training the network models are set as follows: Epoch: 5000; Batch size (per GPU): 16; Num-workers: 4; Optimizer: Adam, beta1: 0.9, beta2: 0.999; Learning-rate: 0.001; Regularizer: L2 normalization.

To obtain visually informative evaluation results, we compared the performance of the proposed method with the EAST [16], SAST [28], FCENet [29], and DB++ [30] methods, as mentioned above. Specifically, for the ICDAR 2015 and Total-Text datasets, we compared the experimental results of the proposed method with the results published in [16], [28], [29], [30]. For the VinText and VNSceneText datasets, to facilitate a fair comparison, we first trained EAST, SAST, FCENet, and DB++ models directly on the VinText training and VNSceneText training datasets using transfer learning with pre-trained weights from [37]. Subsequently, we evaluated and compared the performance of these methods on the VinText and VNSceneText testing datasets.

Additionally, to assess the effectiveness of the aggregation model, we conducted experiments and compared the model's performance with the case where the fusion model is not used (denoted as 'Not Fusion'). Specifically, in 'Not Fusion,' the input images are passed through the backbone network for feature extraction, the contextual attention mechanism is applied directly to the feature maps obtained from the backbone, and text instance segmentation is performed. The experimental results of the methods on the ICDAR 2015 dataset are presented in detail in Table I1.

TABLE I
THE EXPERIMENTAL RESULTS ON THE ICDAR 2015 DATASET

Method	Precision	Recall	H_mean	FPS
EAST [16]	78.33	83.27	80.72	13.2
SAT [28]	87.09	86.72	86.91	–
FECNet[29]	84.02	85.01	84.60	-
DB++ [30]	90.90	83.90	87.30	10
Not Fusion	84.07	85.23	85.12	6.3
VNSTD	87.53	86.94	87.23	8.4

From the experimental results, it is evident that the fusion model not only significantly enhances the accuracy of text detection, especially for arbitrarily shaped and sized text, but also optimizes the processing time for the contextual attention model.

Figure 20. Some results on the ICDAR 2015 dataset

Figure 20 illustrates some results of the proposed method on the ICDAR 2015 dataset. This dataset focuses on evaluating the adaptability of algorithms to real-world images captured in urban environments with natural lighting and diversity: indoor and outdoor images, close-up and distant shots, images with single or multiple text lines, and text in arbitrary shapes. The experimental results of the methods on the Total-Text dataset are presented in detail in Table II.

TABLE II
THE EXPERIMENTAL RESULTS ON THE TOTAL-TEXT DATASET

Method	Precision	Recall	H_mean	FPS
EAST [16]	–	–	-	–
SAT [28]	76.86	83.77	80.17	–
FECNet[29]	79.80	87.40	83.40	–
DB++ [30]	88.90	83.20	86.00	28.0
Not Fusion	77.26	84.02	80.34	20.5
VNSTD	84.32	88.17	86.20	26.6

Figure 21 shows some results of the algorithm on the Total-Text dataset. This dataset is focused on evaluate the algorithms' adaptability to diverse text images, including short, long, curved, vertical, horizontal, and various other text types.

Figure 21. Some results on the Total-Text dataset

The experimental results show that the DB++ algorithm and the proposed VNSTD algorithm achieve higher accuracy than the others. In the case of not using the fusion

model, the model's accuracy significantly decreases. This indicates that fusion and representing features at various scales can effectively handle text with different shapes and sizes.

The experimental results of the methods on the VinText dataset are presented in detail in Table III.

TABLE III
THE EXPERIMENTAL RESULTS ON THE VINTEXT DATASET

Method	Precision	Recall	H_mean	FPS
EAST [16]	71.24	73.38	72.30	11
SAT [28]	83.12	85.03	84.06	8.5
FECNet[29]	83.23	84.57	83.89	9.2
DB++ [30]	84.05	83.90	83.97	8.3
Not Fusion	83.26	84.04	83.32	6.2
VNSTD	85.63	87.94	86.77	7.9

Some results of the proposed method on the VinText dataset are presented in Figure 22.

Figure 22. Some results on the VinText dataset

TABLE IV
THE EXPERIMENTAL RESULTS ON THE VNSCENETEXT DATASET

Method	Precision	Recall	H_mean	FPS
EAST [16]	70.73	72.16	71.44	10.03
SAT [28]	83.25	84.17	83.70	10.60
FECNet[29]	81.23	82.68	81.95	11.50
DB++ [30]	84.05	81.02	82.50	9.30
Not Fusion	82.23	83.40	82.27	7.60
VNSTD	85.14	87.23	86.17	8.80

Figure 23 displays some results of the proposed method on the VNSceneText dataset.

Figure 23. Some results on the VNSceneText dataset

V. CONCLUSION

In this paper, we propose an efficient approach to address the problem of Vietnamese text detection in scene images using deep learning. The proposed method is based on the idea of utilizing deep neural network architectures to learn various geometric attributes to reconstruct the polygonal representation of text regions.

The proposed network architecture consists of four main components: (1) The backbone network (resnet-50) for feature extraction from the input image; (2) The fusion model, constructed as a Feature Pyramid Network (FPN) to integrate and represent features at multiple scales, aiding in the detection of text regions of various sizes; (3) The contextual attention model, structured as a deep convolutional network, trained to capture contextual information (both short-range and long-range dependencies) at each pixel to obtain more representative features; (4) The text instance segmentation model, implemented using semantic segmentation, is responsible for determining the positions

and bounding boxes of all text regions in the input image based on the detection of center lines, center points, and boundary points for grouping them into corresponding text regions.

Notably, during the clustering process in step 4, the algorithm integrates low-level features (pixel-level) with high-level object knowledge to adapt and effectively address complex text shapes (curved, slanted, irregular font sizes). Experimental results show that this is a feasible approach to tackle the problem of Vietnamese text detection in outdoor images.

ACKNOWLEDGEMENT

We would like to sincerely thank the Basic Research and Technology Development Task 'Research on Improving Image Retrieval Efficiency with Relevance Feedback Using Graph Convolutional Neural Network', project code: CSCL02.05/23-24;3, for their support during this research.

REFERENCES

[1] Zobeir Raisi, Mohamed A. Naiel, Paul Fieguth, Steven Wardell, John Zelek, "Text Detection and Recognition in the Wild: A Review", https://doi.org/10.48550/arXiv.2006.04305.

[2] S. Long, X. He, and C. Yao. Scene text detection and recognition: The deep learning era. *Int. J. Comput. Vision*, pp.1–24, 2020.

[3] S. M. Hanif and L. Prevost, "Text detection and localization in complex scene images using constrained adaboost algorithm," *in Proc. Int. Conf. on Doc. Anal. and Recognit*, pp 1-5, 2009.

[4] K. Wang, B. Babenko, and S. Belongie, "End-to-end scene text recognition," *in Proc. Int. Conf. on Comp. Vision*, pp.1457–1464, 2001.

[5] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven,"PhotoOCR: Reading text in uncontrolled conditions," *in Proc. IEEE Int. Conf. on Comp. Vision*, pp. 785–792, 2013.

[6] S. Tian, Y. Pan, C. Huang, S. Lu, K. Yu, and C. Lim Tan, "Text flow: A unified text detection system in natural scene images," *in Proc. IEEE Int. Conf. on Comp. Vision*, pp.4651–4659, 2015.

[7] H. Cho, M. Sung, and B. Jun, "Canny text detector: Fast and robust scene text localization algorithm," *in Proc. IEEE Conf. on Comp. Vision and Pattern Recognit*, pp.3566–3573, 2016

[8] B. Shi, X. Bai, and S. Belongie, "Detecting oriented text in natural images by linking segments," *in Proc. IEEE Conf. on Comp. Vision and Pattern Recognit*, pp.2550–2558, 2017.

[9] Y. Zhu, C. Yao, and X. Bai, "Scene text detection and recognition: Recent advances and future trends," *Frontiers of Comp. Sci*, vol. 10, no. 1, pp.19–36, 2016.

[10] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu. "Textboxes: A fast text detector with a single deep neural network", *In AAAI Conf. on Artificial Intelligence*, 2017

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. E. Reed. "SSD: single shot multibox detector", *In European Conf. Comput. Vision*, 2016.

[12] M. Liao, B. Shi, and X. Bai. "Textboxes++: A single-shot oriented scene text detector", *IEEE Trans. Image Processing*, 27(8), pp.3676–3690, 2018.

[13] Y. Liu and L. Jin. "Deep matching prior network: Toward tighter multi-oriented text detection", *In Proc. Conf. Comput. Vision Pattern Recognition*, 2017.

[14] P. He, W. Huang, T. He, Q. Zhu, Y. Qiao, and X. Li " Single shot text detector with regional attention", *In Proc. Int. Conf. Comput. Vision*, pp.3047–3055, 2017.

[15] M. Liao, Z. Zhu, B. Shi, G. Xia, and X. Bai. "Rotation-sensitive regression for oriented scene text detection", *In Proc. Conf. Comput. Vision Pattern Recognition*, pp.5909–5918, 2018.

[16] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. "EAST: an efficient and accurate scene text detector", *In Proc. Conf. Comput. Vision Pattern Recognition*, 2017.

[17] W. He, X. Zhang, F. Yin, and C. Liu." Deep direct regression for multi-oriented scene text detection", *In Proc. Int. Conf. Comput. Vision*, 2017.

[18] L. Xie, Y. Liu, L. Jin, and Z. Xie. "Derpn: Taking a further step toward more general object detection", *In AAAI Conf. on Artificial Intelligence*, volume 33, pp.9046–9053, 2019.

[19] B. Shi, X. Bai, and S. J. Belongie." Detecting oriented text in natural images by linking segments", *In Proc. Conf. Comput. Vision Pattern Recognition*, 2017.

[20] J. Tang, Z. Yang, Y. Wang, Q. Zheng, Y. Xu, and X. Bai. "Seglink++: Detecting dense and arbitrary-shaped scene text by instanceaware component grouping", *Pattern recognition*, 2019.

[21] Z. Zhang, C. Zhang, W. Shen, C. Yao, W. Liu, and X. Bai. "Multioriented text detection with fully convolutional networks", *In Proc. Conf. Comput. Vision Pattern Recognition*, 2016.

[22] C. Xue, S. Lu, and F. Zhan. "Accurate scene text detection through border semantics awareness and bootstrapping", *In European Conf. Comput. Vision*, pp. 355–372, 2018.

[23] M. Liao, P. Lyu, M. He, C. Yao, W. Wu, and X. Bai. "Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes", *IEEE Trans. Pattern Anal. Mach. Intell*, 2019.

[24] P. Lyu, M. Liao, C. Yao, W. Wu, and X. Bai. "Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes", *In European Conf. Comput. Vision*, pp. 67–83, 2018.

[25] K. He, G. Gkioxari, P. Dollar, and R. Girshick." Mask R-CNN", *In Proc. Int. Conf. Comput. Vision*, pp. 2961–2969, 2017.

[26] W. Wang, E. Xie, X. Li, W. Hou, T. Lu, G. Yu, and S. Shao. "Shape robust text detection with progressive scale expansion network", *In Proc. Conf. Comput. Vision Pattern Recognition*, pp. 9336–9345, 2019.

[27] Z. Tian, M. Shu, P. Lyu, R. Li, C. Zhou, X. Shen, and J. Jia. "Learning shape-aware embedding for scene text detection", *In Proc. Conf. Comput. Vision Pattern Recognition*, pp. 4234–4243, 2019.

[28] Pengfei Wang, Chengquan Zhang, Fei Qi, Zuming Huang, Mengyi En, Junyu Han, Jingtuo Liu, Errui Ding, and Guangming Shi " A Single-Shot Arbitrarily-Shaped Text Detector based on Context Attended Multi-Task Learning", *In Proceedings of the 27th ACM International Conference on Multimedia (MM '19)*, 2019.

[29] Yiqin Zhu, Jianyong Chen, Lingyu Liang, Zhanghui Kuang, Lianwen Jin, Wayne Zhang, "Fourier Contour Embedding for Arbitrary-Shaped Text Detection", *CVPR*, 2021.

[30] M. Liao, Z. Zou, Z. Wan, C. Yao and X. Bai, "Real-Time Scene Text Detection With Differentiable Binarization and Adaptive Scale Fusion", *in IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 45, no. 01, pp. 919-931, 2023.

[31] N. Nguyen et al., "Dictionary-guided Scene Text Recogni-

tion," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA*, pp. 7379-7388, 2021.

[32] N. T. Pham, V. D. Pham, Q. Nguyen-Van, B. H. Nguyen, D. N. Minh Dang and S. D. Nguyen, "Vietnamese Scene Text Detection and Recognition using Deep Learning: An Empirical Study," *6th International Conference on Green Technology and Sustainable Development (GTSD), Nha Trang City, Vietnam*, pp. 213-218, 2022.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need", *In Adv. Neural Inf. Process. Syst. (NIPS)*, pp. 5998–6008, 2017.

[34] Shangbang Long, Jiaqiang Ruan, Wenjie Zhang, Xin He, Wenhao Wu, and Cong Yao, "TextSnake: A flexible representation for detecting text of arbitrary shapes", *In Eur. Conf. Comp. Vis. (ECCV)*, pp. 20–36, 2018.

[35] Wenhai Wang, Enze Xie, Xiang Li, Wenbo Hou, Tong Lu, Gang Yu, and Shuai Shao, "Shape Robust Text Detection With Progressive Scale Expansion Network", *In IEEE Conf. Comp. Vis. Patt. Recognit. (CVPR)*, pp. 9336–9345, 2019.

[36] Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, et al, "ICDAR 2015 competition on robust reading", *In Int. Conf. Doc. Anal. Recognit. (ICDAR). IEEE*, pp. 1156–1160, 2015.

[37] Chee Kheng Ch'ng and Chee Seng Chan, "Total-Text: A comprehensive dataset for scene text detection and recognition", *In Int. Conf. Doc. Anal. Recognit. (ICDAR)*, Vol. 1, pp. 935–942, 2015.

[38] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database", *In IEEE Conf. Comp. Vis. Patt. Recognit. (CVPR). IEEE*, pp. 248–255, 2009.

Huynh Van Huy graduated from Hue University with a Bachelor's degree in Computer Science in 2005 and obtained a Master's degree in Information Technology from Lac Hong University in 2012. Currently, the author is a Ph.D. student in Computer Science at Lac Hong University.
Email: huynhvanhuy@gmail.com

Nguyen Thi Thanh Tan obtained her Bachelor's and Master's degrees in Computer Science and Information Technology from the Hanoi University of Science and Technology in 1999 and 2001, respectively. She earned her Ph.D. in Computer Science from the Vietnam Academy of Science and Technology. Currently, the author is working at the Electric Power University. Her primary research interests include Artificial Intelligence, Computer Vision, and Big Data Analysis. The author has published over 40 articles in reputable national and international journals.
Email: tanntt@epu.edu.vn

Ngo Quoc Tao obtained his Bachelor's degree in "Mathematical Assurance for Computers and Computing Systems" in 1997, an Associate Professor in Computer Science in 2002, and a Senior Researcher in 2018. Now, he is working at the Institute of Information Technology, Vietnam Academy of Science and Technology. He is conducting research in the fields of Pattern Recognition, Image Processing, Knowledge Technology, and Artificial Intelligence.
Email: nqtao@ioit.ac.vn

A Method for Change Impact Analysis of C# Projects

Hoang-Viet Tran, Nguyen Thi Mai Loan, Doan Duc Kien, Nguyen Ha Trang, Le Van Huy, Pham Ngoc Hung
VNU University of Engineering and Technology, 144 Xuan Thuy str., Cau Giay dist., Hanoi, Vietnam

Correspondence: Hoang-Viet Tran, thv@vnu.edu.vn
Communication: received 21 Jun 2023, revised 15 Oct 2023, accepted 26 Nov 2023
DOI: 10.32913/mic-ict-research.v2024.n1.1211

***Abstract*: Change Impact Analysis (CIA) is a common technique for identifying the potential effects of a change in software on other components. Despite the popularity of C# in the software industry, there have not been any CIA methods for C# projects. This paper proposes a new method for performing static CIA in C# projects based on existing methods for object-oriented languages. The main idea behind the method is constructing a dependency graph by doing static source code analysis. This enables the use of WaveCIA, a CIA algorithm leveraging dependency graphs. The implementation of this method in a CIA tool has proven its effectiveness in determining impacted components in C# projects, which can be further used for regression testing.**

Change Impact Analysis (CIA) is a common technique for identifying the potential effects of a change in software on other components. This solution has been made to analyze the impact of code changes in different languages like Java, C/C++, etc. Despite the popularity of C# in the software industry, there have not been any CIA methods for C# projects. This paper proposes a new method for performing static CIA in C# projects based on existing methods for object-oriented languages, to generate dependency graphs of that project's source code. The key idea behind the method is to construct a dependency graph by doing static source code analysis. This enables the use of Wave-CIA, a CIA method leveraging dependency graphs. The results of analyzing the source code to build the dependency graph between the components in the project have been shown in the CIA4CS tool. We did some tool testing with several C# projects and the results showed high coverage for components and dependencies, that has proven its effectiveness in determining impacted components in C# projects. Finally, we provide some methodological discussion at the end of the paper.

Keywords: *Internet of Things (IoT), deep learning, IoT device identification, attention features fusion method (AFF), IoT device detection, feature fusion; image processing.*

I. Introduction

In the software life cycle, changing requirements often happens because of various reasons including adding new features, changing customer requirements, improving performance, etc. However, a small change in the source code can have some unpredictable effects and hidden bugs deep inside the application that even programmers and testers are not aware of. As a result, change requests introduce several challenges for software maintainers. The developers have many difficulties in identifying which modules need to be modified to accomplish a change. Testers must minimize the number of test cases for regression testing. The project manager and higher-level managers are faced with reducing costs and time for change management. As a result, determining how changing source code components affects other components is one of the main problems in the software development project and the software life cycle. The optimal and effective solution to solve the above problem is the analysis of the effects of changes (Change Impact Analysis - CIA [1], [2]). CIA helps identify the affected parts of the software application when making changes to its source code in a new version. For this reason, the CIA is an important solution in software development and maintenance, especially in reducing the maintenance effort in many senses.

Many researchers published many papers addressing the CIA problem [3–5]. In 2007, Huang and Song proposed a dynamic impact analysis based on program executions [4]. They considered the characteristics of object-oriented programs and performed dependency analysis by removing elements that do not have a dependency on the changed elements. In 2010, Sun et al. presented a static CIA method that considers different impact mechanisms and rules of different change types, to calculate the impact sets [3]. They present an intermediate representation for object-oriented Java programs, then perform CIA based on the impact rules of different change types. In 2020, Mondal et al. [5] proposed a method to find impact sets using rule association and code clone. The rule association relies on the previous change history of program entities. Code

clones are detected using a clone detector. Mondal et al. performed the investigation on some projects written in Java, C++, and C#. Although many methods have been proposed representing many different viewpoints addressing the CIA problems, there is no detailed method that applies Wave-CIA for C# projects.

In practice, the CIA solution has been implemented to analyze projects in many different languages on a number of tools. First, JRipples[1] is a tool for program comprehension during incremental change. It manages the organization of the steps that comprise the impact analysis and subsequent change propagation [6]. However, this tool only supports Java programming language, not C#. Second, JArchitect[2] is a tool that supports a large number of code metrics and allows visualization of dependencies using directed graphs and a dependency matrix. The tool also performs code base snapshots comparison and validation of architectural and quality rules. JArchitect also supports Java, not C#. Third, JCIA is a tool for change impact analysis of Java EE applications. This tool analyzes the source code of Java EE applications for building the dependency graphs (called JDG) [7]. Fourth, CodeSurfer [8] is a popular code visualization tool used to explain and display source code clearly and understandably. It can be used to create charts and graphs to help users visualize and track the flow of data and the execution process of the source code. It supports programming languages like C, C++, Java, and Ada. However, CodeSurfer does not support C#. Finally, Tochal [9] is a tool that implements a DOM-sensitive hybrid change impact analysis technique for JavaScript through a combination of static and dynamic analysis. Tochal also does not support C#. Consequently, the above tools support many programming languages, not C#, which is one of the most popular programming languages.

This paper introduces a Wave-CIA-based method, named CIA4CS, for analysis of the impact of changes in C# projects. It is based on WAVE-CIA, a novel CIA approach based on call graph mining, which until now, no one has applied and implemented for C#. In CIA4CS, two versions of the project source code are analyzed and converted into two dependency graphs, respectively. Then, CIA4CS compares the two dependency graphs to find the components that are changed between the two versions of the source code. Those components will be put into the execution of a CIA method named WAVE-CIA to find the affected components. There are two main contributions of CIA4CS. First, CIA4CS provides detailed algorithms for the process from code analysis to change computation. Second, CIA4CS applies a well-known method named WAVE-CIA in C# projects that there has been no published research before. A tool supporting the CIA4CS has been built and applied for testing a number of C# projects. The test results show that CIA4CS gives high accuracy in analyzing the effect of changing one component on other components.

The rest of this paper is organized as follows. Section II presents some background concepts being used in this paper. Next, Section III shows the details of CIA4CS methods. Then, Section IV describes the CIA4CS Tool which includes the implementation of CIA4CS. Initial experiment results and discussions are presented in Section V. After that, related works to CIA4CS are discussed in Section VI. Finally, we conclude the paper in Section VII.

II. BACKGROUNDS

In this section, we present some background concepts which will be used in this paper.

1. Change Impact Analysis

Change Impact Analysis (CIA) [1, 10] is an important solution in the process of software development and maintenance. The basis of this approach is to evaluate whether a component is affected or not based on its dependency on the changed component when analyzing the source code. From there, we can predict the scope and impact of the change on the code and recommend components that need to be retested. This is an effective solution for all stakeholders in software projects. For testers and developers, CIA is used to identify test cases that involve changed components when the source code changes. As a result, it is possible to reduce the number of test cases that need to be performed, reducing the time, effort, and cost required for testing while ensuring high-quality software. For project managers, based on the results of the analyzed impact, they can evaluate the level of impact to estimate time, costs, etc. In addition, it allows investors and customers to determine the total cost to invest in the change.

2. An Overview of C#

C# (or C sharp) is a simple, modern, object-oriented, powerful, and versatile programming language developed by Microsoft based on C++ and Java [11]. C# has a syntax quite similar to those languages. However, it has been developed and improved to be simpler and more accessible. C# comes along with and is an important part of the .NET framework. Currently, C# is an open-source programming language, is fast, and can be compiled on different computer platforms (Windows, Linux, and Mac OS). Although it is an object-oriented programming language, the latest versions of C# also support some features of functional languages

[1] https://en.wikipedia.org/wiki/JRipples
[2] https://www.jarchitect.com/

such as *immutable data types*, *delegate*, *local function*, etc. For this reason, analyzing C# source code becomes more difficult than analyzing the source code of other object-oriented languages, which only have *class*, *method*, and *field*. In fact, C# source code is managed with two levels: *project* and *solution*.

- *Project* is the basic level of management. Each project consists of one or more C# source code files. After the compilation process, each project is corresponding to a generated executable file or library. Project information is contained in a file with the extension *.csproj*. Normally, all project components are located in one folder.
- *Solution* is a higher level of source code management. A solution consists of one or more projects. Projects in the same solution can reference each other, allowing one project to reuse source code from another project.

3. Abstract Syntax Tree

An abstract syntax tree (AST) is a data structure that represents the structure of the source code. AST focuses on representing the logical components of the source code. Each node in the tree represents a structural object that appears in the source code, such as a name, function, etc. Nodes can contain child nodes which are more specific components of the parent node. For example, an expression node can include child nodes such as name, type, and value. AST is commonly used in compilers and is often the result of the source code parser's parsing phase. AST can be seen as an intermediate structure to represent various forms of source code.

4. Project Component Structure Tree

The structure tree is the result of analyzing the project source code. Each element in the structure tree represents an actual component of the project. On the branch level, the root represents the root of the C# project followed by sub-components at each level, and so on until the most granular level. Component types are defined in accordance with the components defined in the C# project.

Table I lists all the component types in the C# project tree and their meaning in a software project.

TABLE I
COMPONENT TYPES IN THE C# PROJECT COMPONENT TREE STRUCTURE

ID	Component	Define
1	Unknown	Unknown components
2	Class	Classes in C# project
3	Interface	Interface in a C# project
4	Struct	Representing the struct data type in C# project
5	Enum	Representing the enum data type in C# project
6	Field	Field in a C# project
7	Property	Property in a C# project
8	Method	Method in a C# project
9	Delegate	Representing delegate in C# project
10	LocalFunction	Representing Local Function in C# project
11	File	Files in C# project
12	Folder	Folders in C# project
13	Project	C# Project Root Component

TABLE II
C# PROJECT DEPENDENCIES

ID	Dependency	Description
1	*member*	Expressed when the wrapper component contains another component in the project.
2	*inheritance*	A class inherits another class, a class implements an interface, an interface inherits another interface
3	*use*	A method uses another component in the project.
4	*invocation*	A method calls another method
5	*override*	Represents when a method overrides another

5. Dependency Graph

The project component structure tree contains only the components. For this reason, it only describes the architecture of components of a project. Thus, it is not enough information to use the project component structure tree for CIA purpose. The reason is that it does not show the dependency relationship between two arbitrary components. To visualize the components in the project and the relationships between those components, we represent them with a dependency graph.

Definition: *The dependency graph G is defined as a pair $\{V, E\}$ where $V = \{v_0, v_1, \ldots, v_n\}$ is a list of nodes that represent components like class, method, property, etc. and $E = \{(v_i, v_j) \mid v_i, v_j \in V\} \subset V \times V$ is a list of directed edges. Each edge (v_i, v_j) represents a dependency between v_i and v_j which means v_i depends on v_j.*

All dependency types are presented in Table II. These types of dependencies represent relationships between components on the dependency graph. If component A has one of those five dependencies with component B, then A depends on B. For example, if A uses B, there will be an arrow from node A to node B on the dependency graph.

6. Libraries Used In Source Code Analysis

a) Microsoft.CodeAnalysis.CSharp

Microsoft.CodeAnalysis.CSharp is a library published by Microsoft. This library provides API to generate abstract syntax trees from C# source code, interacts with the nodes in the tree, and provides API for analyzing static bindings.

We can use this library to find the definition of an identifier in the source code. For example, we can find where the definition of a function that is being called in the function call, or we can find the definition of a data type when we declare a variable. These features are necessary for building the dependency graph for analyzing the impact of change. The dependency graph is generated by traversing the vertices of the abstract syntax tree. The edges of the graph are generated by defining the associations in the statements.

b) Microsoft.CodeAnalysis.CSharp.Workspaces

Microsoft.CodeAnalysis.CSharp.Workspaces is a library developed based on Microsoft.CodeAnalysis.CSharp. It has extensive features to help process the source code of C# solutions and projects. Using this library is necessary since C# projects are often organized into solutions and projects whilst Microsoft.CodeAnalysis.CSharp mainly works with individual pieces of C# code. In our tool implementing CIA4CS, this library is used to determine and obtain the AST of C# source code files in a solution when the path to that solution file is known.

7. Change type

When comparing two versions of source code, we need to find the difference between them, specifically the set of changed components. The set of changed nodes is called the change set (CS). Changes can be classified into three categories: addition, deletion, and syntax change. In particular, the type of "syntax change" covers all types of component changes (except additions and deletions). This includes body changes, return types, declarations, etc. These types of components are shown in Table III

TABLE III
NODE CHANGE TYPES

No.	Change type	Description	Example
1	Deletion	Delete a component of the old version	Remove a property in class C
2	Addition	Add a component that the old version does not have	Add a method in class C
3	Syntax change	Change a component that is already in an old version	Change the return type of method M

In addition to the change nodes, we also have impact nodes. These nodes represent the components affected by the change from the previous version.

8. WAVE-CIA

WAVE-CIA is a CIA method proposed by Li et al. [10] in 2013. The method is to compute a set of impacted components from a change set and dependency graph. The original idea of the WAVE-CIA method is "the propagation of water waves". By throwing some stones into a quiet water surface to create a center of propagation, waves are formed on the surface of the water. These water waves interfere with each other and the intersection points become new centers of propagation. Let's consider the dependency graph as a water surface and the change set as the set of stones thrown into the water. The key idea of WAVE-CIA method is that a component is more likely to be impacted if it has dependency relationships with more changing components. These changed components will propagate the impact to other components in the graph. The intersection component mentioned above is called the *core* set defined as follow:

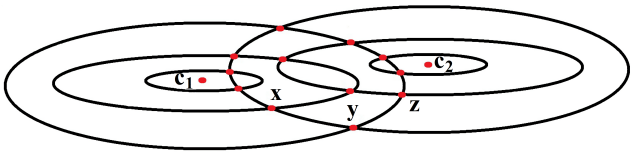


Figure 1. An example of the *core* set.

Definition 1: (Core set) Core set is a set of components that are cohesive and more likely impacted by multiple changed components in the change set.

In Figure 1, let components c_1 and c_2 be two components in the change set. Then, they propagate their impact waves to other components in the given graph. Components $\{x, y, z\}$ are put in the core set because both c_1 and c_2 can reach them.

To compute the core set from a changed set and a dependency graph, we use the neighbor set (NS) of vertices that do not belong to the changed set in the dependency graph. If the neighbor set of a vertex has multiple components that belong to the changed set, then the vertex belongs to the core. The neighbor set of a component is defined as follows:

Definition 2: (Neighbor set) Given a vertex in a dependency graph, its neighbor set is a set of vertices that are reachable from it within a given distance. The distance between two vertices in a dependency graph is defined to be the least number of edges needed to move from one vertex to the other.

III. CIA4CS: A CIA METHOD FOR C# PROJECTS

This section gives a detailed description of the proposed method for performing change impact analysis of C# projects. For a given C# project source code, the method consists of three main phases. First, an AST is obtained using Microsoft.CodeAnalysis.CSharp library. Next, a dependency graph is generated by traversing the AST in the

depth-first search order. For the comparison of two source code versions of the same project, two corresponding dependency graphs are generated and compared to identify the change set. Finally, the execution of a CIA algorithm with the change set as input determines the core set and impact set. In this paper, we employ the Wave-CIA [10] method to perform the change impact analysis. The overview of CIA4CS is shown in Figure 2. The following sections present each phase in more detail.

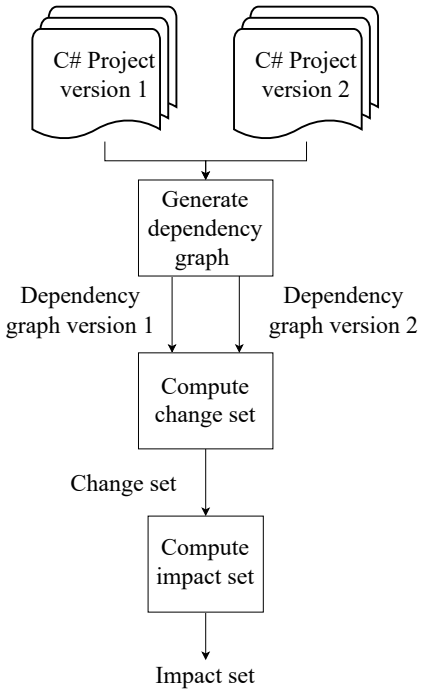


Figure 2. An overview of CIA4CS.

1. Dependency Graph Generation

In this phase, a dependency graph is constructed based on the AST of C# project source code. The construction process contains three main steps which are shown in Figure 3.

- Step 1: Starting from the project directory, CIA4CS filters out the C# source files (*.cs) and builds a tree of directories and files.
- Step 2: For each source code file, the Microsoft.CodeAnalysis.CSharp library is used to generate the corresponding AST.
- Step 3: Traverse the generated AST to obtain the necessary information about the components such as classes, methods, properties, etc. For each appropriate node in the AST, a new node is added to the dependency graph. Then, its dependencies with other nodes are created.

In Step 1, the path to a C# project directory, which consists of C# source files needed for analysis, is used to

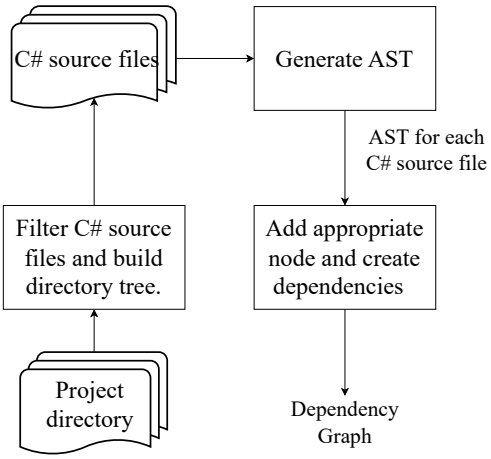


Figure 3. Dependency graph generation.

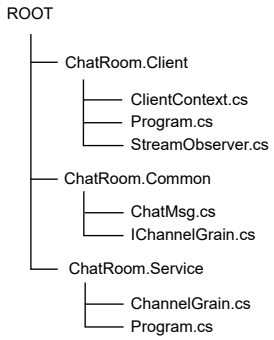


Figure 4. Directory tree for the ChatRoom project

construct a directory tree. A C# project directory is usually composed of a .sln file and one or more .csproj files, which contain information about the project's dependencies and runtime. The directory tree is built with the directory that contains the .sln file as root and other nodes corresponding to the files and directories hierarchy. Figure 4 illustrates a directory tree of ChatRoom[3] project, a sample C# online chat application.

In Step 2, the AST for each C# source file is constructed using Microsoft.CodeAnalysis.CSharp. This process contains two following smaller steps. First, a concrete syntax tree is produced by calling the method CSharpSyntaxTree.ParseText, which takes C# source code as an argument. A concrete syntax tree is a complete representation of a program's source code, including nodes corresponding to keywords, parentheses, operators, etc., referred to as syntax tokens in the library. Second, the concrete syntax tree is pruned to obtain a more compact version which is the corresponding abstract syntax tree (AST). This can be done by removing syntax tokens from the concrete syntax tree.

[3]https://github.com/dotnet/samples/tree/main/orleans/ChatRoom

```
1  int gcd(int a, int b)
2  {
3    if (b == 0)
4      return a;
5    return gcd(b, a % b);
6  }
```

Listing 1: Example program

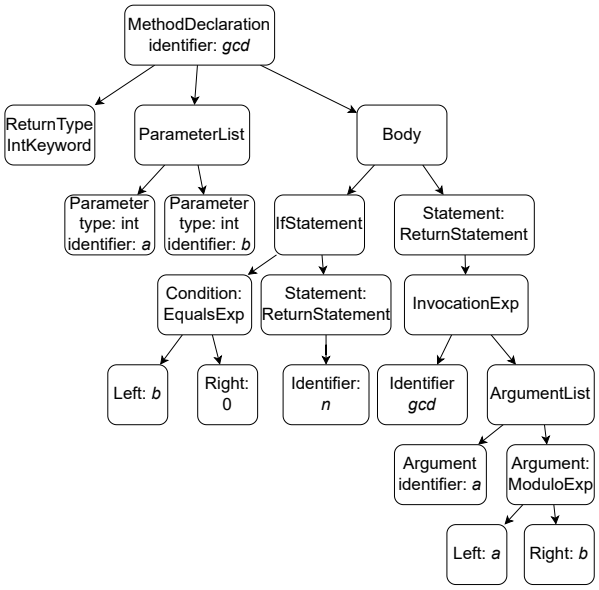


Figure 5. AST for the example program.

To illustrate this process, a simple C# method that finds the greatest common divisor (GCD) of two integers is given in Listing 1. The corresponding AST is shown in figure 5.

In Step 3, for each node in the AST, a corresponding node is added to the dependency graph. The detailed process is described in Algorithm 1. The main idea of the algorithm is to traverse the AST tree in the depth-first search order. For each node in the tree, CIA4CS creates an edge connecting that node and its parent, then performs static binding to locate dependent nodes, and recursively processes all child nodes.

The algorithm starts by checking if there are nodes in the dependency graph corresponding to the node we are traversing in the AST tree (lines 3-4). If not, the algorithm creates a new node with the identifier (ID) being the fully qualified name of the node in the AST tree (lines 5 - 6). The ID of a node is generated by Microsoft.CodeAnalysis.CSharp library via binding resolution. For instance, if field B is inside class A, which is wrapped in namespace N, its ID will be N.A.B. Since there cannot be two fields with the same name inside the same class, this hierarchical naming system results in uniquely defined IDs. For methods, in addition to

Algorithm 1: Generate dependency graph

```
1  Input: ast_node: root of the AST; parent_node:
     parent node in dependency graph; dep_graph: the
     dependency graph to be generated
2  begin
3      node_id ← fully qualified name of ast_node;
4      graph_node ← find node with ID node_id in
         dep_graph;
5      if graph_node = NULL then
6          graph_node ← create new node in dep_graph
             with ID node_id;
7      end
8      if parent_node! = NULL then
9          Create new edge from parent_node to
             graph_node with MEMBER dependency type;
10     end
11     Process inner statements, return types, etc. to handle
         other dependency types;
12     foreach child_node in ast_node.children do
13         call Algorithm 1
             (child_node, graph_node, dep_graph);
14     end
15 end
```

their name, their parameter types are also present in their ID. This can prevent two overloads of a method from being assigned identical IDs. For example, suppose two methods named C are members of class A mentioned above. The first one accepts a string as a parameter, making its ID N.A.C(string), while the ID of the second one might be N.A.C(int, int) if it requires two integer inputs. Back to the algorithm, Then, we proceed to create an edge between the parent node and the child node to represent the MEMBER type dependency between the parent and the child (lines 8 - 9). Thus, a project component structure tree is a tree that includes all vertices and edges whose dependency type is MEMBER. This is followed by processing inner statements for methods, or return types for variables, properties, and so on to create edges in the graph with other types of dependencies (line 11). This step is primarily concerned with a technique called semantic analysis. The semantic analysis attempts to obtain extra information about an AST expression or statement node, particularly what declaration node a variable or a method call refers to. This is useful to the algorithm as the ability to determine which node a method call or a type refers to enables the addition of appropriate edges. In Microsoft.CodeAnalysis.CSharp, semantic analysis is provided via the SemanticModel class, which contains the semantic information of a C# source file. Specifically, each node in the AST has an associated “symbol”, which contains information regarding its declarations, type, references, etc. The symbol can be accessed by calling the method SemanticModel.GetSymbolInfo, which takes a node in the syntax tree as input. The result is a generic

symbol object that can be further cast to a specific symbol type such as method, class, or interface to obtain more concrete information. For instance, a symbol associated with a method call provides pieces of information about the corresponding method such as its return type, parameters, and parent class. Finally, the algorithm recursively calls itself with each children node of *graph_node* (lines 12 - 13). Each child node, referred to as *child_node* in the algorithm, is a child of the AST node used as input. To generate the dependency graph, Algorithm 1 is executed with three inputs: the root of AST (*ast_node*), *NULL* (*parent_node*), the required dependency graph (*dep_graph*).

2. Change Set Computation

After generating dependency graphs for the two versions of the source code, CIA4CS finds a set of nodes that are changed in the new version (hereafter called version 2) in comparison with the old version (hereafter called version 1). Algorithm 2 describes the process in detail. The algorithm takes two dependency graphs A and B generated from version 1 and version 2 of the source code as inputs, respectively. Some examples of each type of change are given in Table III.

Algorithm 2 begins by declaring sets of nodes (*change_set*, *deletion_nodes*, *changed_nodes*, *added_nodes*) corresponding to the change types and initializing them as an empty set (∅) (lines 4 - 7). Then, for each node in graph A, the algorithm finds a node with the same ID in graph B (lines 8 - 9). It is possible that graph B contains nodes with the same IDs as class A as long as their signatures do not change. In other words, if a node and its parents keep their name in the new version, which means its ID is unchanged, it can be found using its ID. If this is correct, the corresponding syntax tree of the two nodes are compared to see if there is a change in syntax (lines 10 - 11). If there is a change in syntax, the node is changed in version 2 in comparison with version 1. The algorithm adds it to *changed_nodes*. If the node cannot be found, it is added to the set of deleted nodes (*deleted_nodes*) (lines 14 - 15). Similarly, the algorithm iterates through all nodes in graph B to find the added nodes and adds them to *added_nodes* (lines 18 - 23). Finally, the algorithm returns the union of *added_nodes*, *deleted_nodes*, and *changed_nodes* as the required change set.

3. Compute The Impact Set

In this paper, we use the WAVE-CIA which is a well-known CIA algorithm proposed by Li et al. [10] in 2013. The algorithm is to compute a set of impacted components from a change set and a dependency graph. Given

Algorithm 2: Change set calculation

```
1  Input: A: dependency graph of version 1; B: dependency graph of version 2.
2  Output: change_set: the set of nodes which was changed/deleted/added in the second version.
3  begin
4      change_set ← ∅ ;
5      deleted_nodes ← ∅;
6      changed_nodes ← ∅;
7      added_nodes ← ∅ ;
8      foreach nodeA in A.nodes do
9          nodeB ← find node in B with the same ID as nodeA ;
10         if nodeB != NULL then
11             if the syntax tree of nodeA and nodeB do not match then
12                 changed_nodes ← changed_nodes ∪ {nodeA};
13             end
14         else
15             deleted_nodes ← deleted_nodes ∪ nodeA;
16         end
17     end
18     foreach nodeB in B.nodes do
19         nodeA ← find node in A with the same ID as nodeB;
20         if nodeA = NULL then
21             added_nodes ← added_nodes ∪ {nodeB};
22         end
23     end
24     change_set ← added_nodes ∪ deleted_nodes ∪ changed_nodes;
25     return change_set;
26 end
```

a change set (CS) and a dependency graph, WAVE-CIA algorithm consists of two steps to compute the impact set as follows: Step 1: Core set computation; Step 2: Impact set computation. Sections below present more details about the computation process.

a) Compute Core Set

As mentioned above, the core set consists of components that are likely to be impacted by multiple changes. The computation of the core set is described in Algorithm 3. The algorithm accepts the dependency graph (*dep_graph*), the change set (*change_set*), and the given limited distance (*depth*) as inputs. After processing, the algorithm returns the core set (*core_set*) as its output. For each vertex in the dependency graph (*dep_graph*), the algorithm retrieves its set of neighbors with a limited distance of depth (*depth*) (line 5 - 7). If the number of neighbors in the change set of a vertex is greater than one, the vertex will be pushed into the temporary core set (*temp_set*) (lines 9 - 10). The final core set is the union of the temporary core set (*temp_set*) and the change set (*change_set*) (line 14).

Core Set Computation Example

Algorithm 3: Core computation

```
1  Input: dep_graph: dependency graph; change_set: set
     of changed components; depth: limited distance.
2  Output: core_set: core set.
3  begin
4      temp_set ← ∅;
5      foreach vertex in dep_graph.V do
6          if vertex ∉ change_set then
7              neighbor_set ← get_neighbor_set(vertex, depth) ;
8              neighbor_change_set ← neighbor_set ∩ change_set;
9              if |neighbor_change_set| > 1 then
10                 temp_set ← temp_set ∪ {vertex};
11             end
12         end
13     end
14     core_set = change_set ∪ temp_set;
15     return core_set;
16 end
```

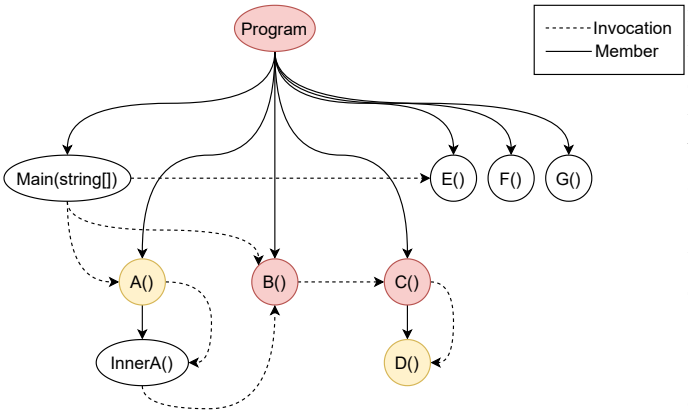


Figure 6. An example of core set.

Let's consider an example shown in Figure 6. In this case, we set the limit distance $depth$ to 2 and the change set to $A(), D()$ (in yellow). We compute the core set for the dependency graph in this example using the following steps:

1) We identify the set of neighbors of vertices that are not in the change set (i.e., $E()$, $F()$, $G()$, $B()$, $C()$, $Main(string[])$, $Program$, $InnerA()$). For example, $E()$ has a neighbor set of $\{C(); A(); Main(string[]); Program; E(); F(); G()\}$. The reason is that these vertices can be reached from $E()$ within 2 edges. Similarly, the neighbor set of $B()$ is $\{C();\ D();\ A();\ InnerA();\ Main(string[]);\ Program;\ E();\ F();\ G()\}$, and so on.
2) We identify all vertices belonging to the neighbor set which are in the change set. For instance, the neighbor set of $E()$ contains $A()$ which is in the change set. In the meantime, the neighbor set of $B()$ has $A()$ and $D()$ which are in the change set, etc.
3) Since the number of neighbors in the change set of $E()$ is 1 (not greater than 1), $E()$ is not in the core set. On the other hand, $B()$ has 2 neighbor vertices in the change set (greater than 1), so $B()$ is in the core set. Similar computation for the remaining vertices, we obtain a core set as follows $\{B(), C(), Program\}$ (in red).

Identifying the core set helps increase the accuracy of the impact set. The core set obtained from vertices not in the changed component set will be combined with the change set to compute the impact set. In the above example, the core set will be $\{B(), C(), Program, A(), D()\}$.

b) Compute Impact Set

Although the core set can give the impact set with few false positives, there are other components that are really impacted but not included in the core. To reduce false negatives, we need to expand from the core set to find the impact set. To obtain a complete set of impacted components, the key idea of the process is as follows. In the beginning, the impact set is initialized as the core set. With each vertex that is not in the impact set, we can obtain the set of components that depend on it and the set of components it depends on. If both of these sets exist vertices belonging to the impact set, the components under consideration will be added to the impact set.

Details of the impact set computation process are described in Algorithm 4. The algorithm accepts the dependency graph (dep_graph) and the core set ($core\_set$) as its inputs. When the algorithm stops, it returns the impact set ($impact\_set$) as its output. When the algorithm starts, it initializes the impact set ($impact\_set$) as core set ($core\_set$) (line 4). For each vertex ($vertex$) that does not belong to the impact set, the algorithm finds the set of components that depend on it ($depender\_set$) and the set of components it depends on ($dependee\_set$) (lines 8 - 9). If both sets contain some impacted components then that vertex is added to the impact set (lines 10 - 11). These steps are repeated until the impact set remains unchanged and the final impact set ($impact\_set$) is returned.

Impact Set Calculation Example

Let's consider an example shown in Figure 7 with the given core set $\{B(), C(), Program, A(), D()\}$ (in red). To calculate the impacted component set, we consider the core set as the initial impact set. Then, we iterate over the vertices that are not in the impact set.

In the first iteration, the vertex $Main(String[])$ depends on vertex $A()$ and is depended upon by vertex $Program$. Since both vertices are already in the impact

Algorithm 4: Impact Set Computation

```
1  Input: dep_graph: dependency graph; core_set: core
     set.
2  Output: impact_set: set of impact components.
3  begin
4      impact_set ← core_set;
5      while impact_set is unstable do
6          foreach vertex in dep_graph do
7              if vertex ∉ impact_set then
8                  depender_set ←
                     get_depender_set(vertex);
9                  dependee_set ←
                     get_dependee_set(vertex);
10                 if (impact_set ∩ depender_set ≠ ∅) ∧
                     (impact_set ∩ dependee_set ≠ ∅) then
11                     impact_set ← impact_set ∪
                         {vertex};
12                 end
13             end
14         end
15     end
16     return impact_set;
17 end
```

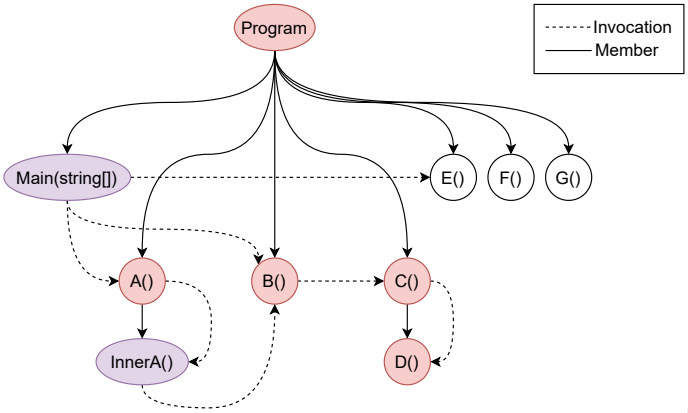


Figure 7. Impact set of dependency graph.

set, $Main(String[])$ belongs to the impact set. Similarly, the vertex $InnerA()$ also belongs to the impact set. For this reason, the two vertices $\{Main(String[]), InnerA()\}$ (in purple) are added to the impact set.

In the second iteration, since there is no more vertex that can be added to the impact set, the algorithm ends and gets the final impact set as $\{B(), C(), Program, A(), D(), Main(String[]), InnerA\}$. The impact set includes $Main(String[])$ and $InnerA()$. This result shows that a more accurate impact set has been constructed.

IV. Implementation

We have implemented the CIA4CS method in a tool named CIA4CS Tool by using C# programming language. CIA4CS Tool can compute the impact set of a certain change set in a given C# project. For a given project, when the user uploads the project folder to CIA4CS Tool, it analyzes and displays the project structure and the dependency graph of components in the project.

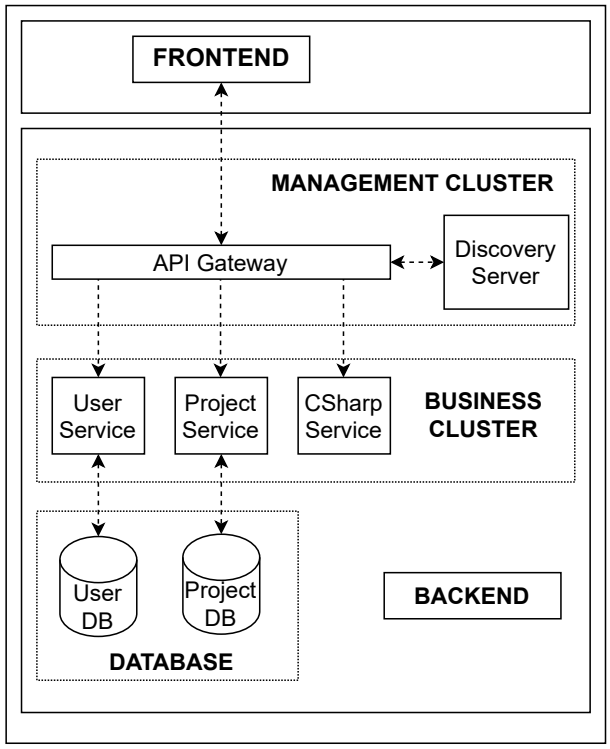


Figure 8. System architecture.

CIA4CS Tool is implemented using the microservices architecture which is based on the JCIA architecture [7]. The architecture of CIA4CS Tool is shown in Figure 8. The tool is built as a website, where the front-end and back-end communicate with each other using API via an API gateway. The back-end of CIA4CS Tool includes Management Cluster and Business Cluster. In which, Management Cluster contains services related to the network navigation, coordination, and management. In the meantime, Business Cluster contains services that directly affect the user interface:

- **User Service:** This service provides user-related functions, including account registration, login, and logout.
- **Project Service:** This service is responsible for managing projects uploaded to CIA4CS Tool. The provided functions include saving and getting the list of projects and versions.
- **CSharp Service:** This service is used to analyze C# source code. The provided functions include comparing versions and computing the impact set.

In CIA4CS Tool, CIA4CS is implemented in CSharp Service. Other services are inherited from the JCIA tool [7].

CIA4CS Tool allows the user to upload the source code of a C# project as a .zip file or a link of a Git project.

Figure 9 shows a screenshot of a CIA4CS session. The project is represented as a dependency graph. We can see that in the Dependency graph section in figure 9. The rectangle nodes represent the components of the C# project. In the normal state, these nodes are black. With the CIA4CS tool, users can select any node and set it as a changed node. After being set, these nodes will be displayed in the Change set section. In addition, in the Dependency graph section, selected changed components are represented by yellow nodes. Then, we can determine the impact set. In the Dependency graph section, nodes in purple are the components impacted by the selected change element. These components are displayed as a list in the Impact set section.

In addition, the CIA4CS Tool has a function to compare two given source code versions of a project. A screenshot of this function is shown in Figure 10. The project and its components are shown in the Version Compare section. Normally, project nodes are shown in black. When a node has a different color, this means that the node is either changed or impacted by some changes. In Figure 10, nodes in green are added whilst nodes in red are deleted and nodes in yellow are changed nodes. Finally, nodes in purple represent the impact set of the new version in comparison with the old version. In addition, the impact set is also displayed in the Impact nodes section.

V. Experiments

We have performed experiments to evaluate the method with C# projects. The experiment was conducted on one open-source library and one project which are publicly available on GitHub[4], one of the world's most famous source code-sharing websites. The tested source code is as follows:

- **eShopOnWeb**[5]: This is a Microsoft's reference ASP.NET Core project. The project is close to industrial projects and large in size.
- **CoreSearch**[6]: These are search library projects that are small and easy to control.
- **LiteNetLib**[7]: This is a UDP library for Mono and .NET.

For each of the above GitHub repositories, two versions are compared at the *project* level (which corresponds to .csproj files) to obtain the impact set. For eShopOnWeb, the two versions tagged as netcore2.1 and netcore2.2 are chosen as the old and new versions, respectively. For LiteNetLib, the candidate versions are 0.9.5.2 and 1.0.1.1. For CoreSearch, since there is no published release, we have tested the two commits: 64dd4858cfb37016f954096110a340d0382df212 (for the old version) and cab1aa05ce9c62b571fe075f8c3f0b18868805ed (for the new version). In these experiments, we use $depth = 2$ when computing the core set. The effectiveness of the proposed method is evaluated through some key indicators: the number of detected impacted components, the percentage of impacted nodes, the time required, and the memory consumed. Performance metrics are measured using the BenchmarkDotNet[8] library. The number of C# source code files and the number of lines of code denote the size of the project.

Experimental results are shown in Table IV and V. In these Tables, the columns are as follows:

- **Project**: the project under test.
- **F1/F2**: the number of source code files in the old and new versions of the project, respectively.
- **LOC1/LOC2**: the number of lines of code in the old and new versions of the project, respectively.
- **Num**: the number of impacted components detected by CIA4CS Tool.
- **Total**: the number of components in the dependency graph.
- **%**: the percentage of impacted components in the dependency graph.
- **Mem**: average memory (in MB) allocated to complete one analysis phase.
- **Time**: the average time (in ms) required to complete an analysis phase.

From the experimental results shown in Table IV and V, we have the following observations.

For eShopOnWeb project:

- The eShopOnWeb project contains three children projects which are Infrastructure, ApplicationCore, and Web.
- The number of impacted components are 144, 155, and 371 for projects Infrastructure, ApplicationCore, and Web, respectively.
- The percentage of impacted components are 84.21%, 77.11%, and 70.15% for projects Infrastructure, ApplicationCore, and Web, respectively.
- CIA4CS Tool can detect all expected impacted component according to *depth* parameter.

For Coresearch project:

[4]https://github.com/
[5]https://github.com/dotnet-architecture/eShopOnWeb
[6]https://github.com/pilotpirxie/coresearch
[7]https://github.com/RevenantX/LiteNetLib
[8]https://github.com/dotnet/BenchmarkDotNet

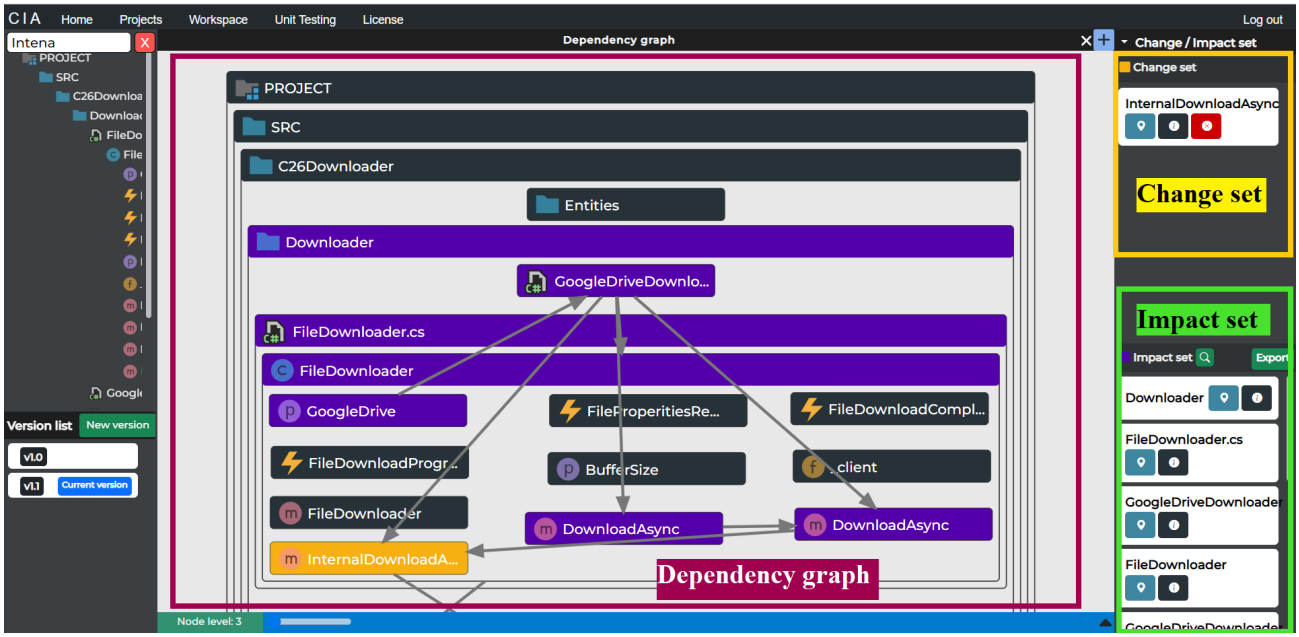


Figure 9. Impact set in CI44CS Tool.

TABLE IV
EXPERIMENTAL RESULTS OF COMPARING TWO VERSIONS OF SOURCE CODE ON ESHOPONWEB

Component	F1	LOC1	F2	LOC2	Num	Total	%	Mem	Time
Infrastructure	17	1664	35	3268	144	171	84.21	81.51	1027.2
ApplicationCore	35	543	35	571	155	201	77.11	82.85	1203.5
Web	42	5985	52	6964	371	529	70.13	156.51	1881.7

TABLE V
EXPERIMENTAL RESULTS OF COMPARING TWO VERSIONS OF SOURCE CODE ON CORESEARCH

Component	F1	LOC1	F2	LOC2	Num	Total	%	Mem	Time
Webserver	3	152	3	167	10	111	9	40.49	542.0
coresearch	6	630	6	645	2	100	2	34.92	468.8

TABLE VI
EXPERIMENTAL RESULTS OF COMPARING TWO VERSIONS OF SOURCE CODE ON LITENETLIB

Component	F1	LOC1	F2	LOC2	Num	Total	%	Mem	Time
LibSample	16	1456	18	1468	150	505	29.70	64.92	975.8
LiteNetLib	34	8368	41	9061	1030	2881	35.75	240.49	2542.0

- The Coresearch project contains two children projects which are Webserver and coresearch.
- The number of impacted components are 10 and 2 for projects Webserver and coresearch, respectively.
- The percentage of impacted components are 9% and 2% for projects Webserver and coresearch, respectively.
- In Webserver, we see the result is not as expected, the actual number of changes is more. The reason is that the Webserver class only changes the namespace. However, the tool considers that one Webserver class is deleted and another Webserver class is added.

For LiteNetLib project:

- The LiteNetLib project contains two children projects which are LibSample and LiteNetLib.
- The number of impacted components are 150 and 1030 for projects LibSample and LiteNetLib, respectively.
- The percentage of impacted components are 29.70% and 35.75% for projects LibSample and LiteNetLib, respectively.
- CIA4CS Tool can detect all expected impacted component according to *depth* parameter.

Regarding the memory and time usage of CIA4CS, we can see that the memory and time used are proportional to

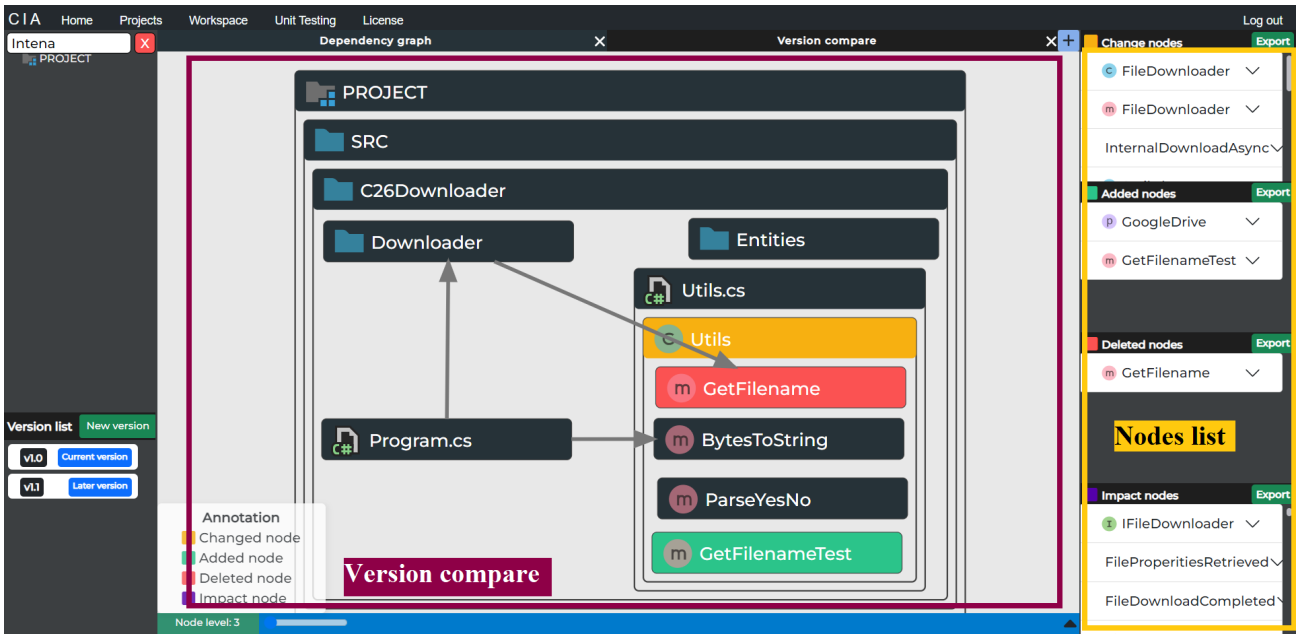


Figure 10. Compare two versions of the source code in the CIA4CS Tool.

the size of the project.

Regarding the acceptance validation of CIA4CS and the experimental results, there are three points as follows.

- CIA4CS is based on Wave-CIA [10] method, which has its own philosophy to consider which components are affected. This should not be compared with other CIA methods. The contribution of CIA4CS is to provide details for C# project source code analysis and dependency graph construction. In turn, the dependency graph will be used as the input for the Wave-CIA method for analysis.
- We do not discuss whether the found impacted components are *true positive* or *false positive* in this paper. The reason is that this depends on the philosophy of the Wave-CIA method. Discussing the correctness of this can be found in the research of Li et al. [10].
- The validation of CIA4CS lies in the source code analysis and dependency graph construction. This is shown through Section III.1.

VI. Related Works

There are many works that have been recently proposed for the CIA method by several authors. Those are CIA methods for many different programming languages such as C++ [12, 13] and other software artifacts [14], etc. In addition, many tools are implemented to support those methods [8], [15], [9], [16], [17], [18].

Chaumun et al. [12] propose a change impact model which is defined at the conceptual level, and map it to the C++ programming language. The conceptual model is comprised of components, relationships, and change types. The impact of a change depends on two main factors the type of change and the type of relationship between both code entities. They are interested in measuring the influence of high-level design on maintainability, and how inter-class dependencies influence changeability. We share the interest in the CIA problem as Chaumun but we focus on solving the problem for C# projects.

Zalewski et al. [13] propose a conceptual change impact analysis (CCIA) which determines the impact of changes in the conceptual specification of generic libraries. The analysis is organized in a pipe-and-filter manner which contains two algorithms. The first one is to detect the impact of changes on the compatibility of different versions of concept specifications. The second one is to detect the impact of changes to the degree of genericity of an algorithm. We share an interest in the CIA problem but we focus on C# projects.

Pirklbauer et al. [14] present a CIA process and a tool that supports many types of visualization methods. The tool is featured with a quality assurance component. In this process, software artifacts such as source code, configuration files, and database scripts are used as the inputs. Then, those artifacts are used to generate a dependency database and to do the CIA process. The process presented by Pirklbauer is a general method and it is not detailed enough for C#. We share the interest of Pirklbauer about the CIA process but we focus on C#.

CodeSurfer [8] is a tool that is available at no cost for C/C++. This tool provides many capabilities to understand a program by providing the results of a static-semantic analysis to the user. It performs a lot of program analyses such as pointer analysis, data flow analysis, and system dependency graphs. However, CodeSurfer does not provide CIA functionalities. We share an interest in program analysis, however, we focus on the CIA functionalities and the C# projects.

Rigi[9] is a visual software understanding tool. Rigi supports reverse engineering, visualization, exploration, and redocumentation [15]. The tool is provided with parsers to retrieve information from the source code. In addition, the tool contains an exchange format to store retrieved, transform, and abstract the information, etc. However, the tool only supports C/C++ and Cobol, not for C#. We share an interest in program analysis but we focus on the CIA and C# projects.

Petrenko et al. [19] are concerned with enhancing impact analysis to cope with a variable granularity of analyzed software artifacts. They argue that a single granularity is insufficient and leads to imprecise analysis. Therefore, they studied how a variable granularity improves the precision of impact analysis, saving the programmers the extra work that is needed to inspect the false positives. They discuss different granularities, including the granularity of classes, the granularity of all program components that include class members, and the granularity of code fragments. They extended the JRipples [20] tool to cope with this variable approach. The algorithms for finding inspected sets at the different granularities were implemented as a plug-in for JRipples. This plug-in substitutes the programmer and simulates all actions from the programmers. We share the interest of CIA for software projects but we focus on units of C# projects by applying the WAVE-CIA method for the dependency graph of C# project.

S.Black et al. [17] proposes a method for the reformulation of Yau and Collofello's ripple-effect algorithm [21]. This method is implemented in the Ripple Effect and Stability Tool (REST) to compute the ripple effect for C programs. REST builds a dependency matrix of the program and applies four different McCabe complexity measures to compute the ripple effect and stability measures for the source code. They performed an evaluation to gauge the correlation between the original algorithm and the REST tool. Then, they concluded that both match, whereas REST achieves more accurate results. We share an interest in improving the quality of maintenance projects but we focus on the CIA of C# projects.

Alimadadi et al.[9] propose a hybrid analysis method for change impact analysis of JavaScript-based web applications that combines the advantages of both static and dynamic analysis techniques to obtain a more complete impact set. This analysis is DOM-sensitive and aware of the event-driven, dynamic and asynchronous entities, and their relationship in JavaScript. It creates a novel graph-based representation capturing these relations, which is used for detecting the impact set of a given code change. Their approach is implemented in a tool called Tochal (TOol for CHange impact AnaLysis)[10]. We share an interest in the CIA problem but we focus on the CIA of C# projects.

Mohamad et al. [22] approach Change Impact Analysis (CIA) method to support the developer to find potential affections or dependency information in source code. The prototype called CIA-V which is a combination of CIA and visualization method based on C++ Object-Oriented language to enhance program understanding ability, will assist the developers to understand the program through diagrams. It also helps to save time to find the affected code. There are three models which are involved to implement the visualization: CATIA [16], Relationship Analysis, and Dependencies Analysis. We share an interest in the CIA problem but we focus on applying the WAVE-CIA [10] for C# projects.

Gwizdala et al. [18] propose the JTracker tool to guide programmers while changing software. This is an extension to the Java environment JBuilder5 and assists a programmer during change propagation. JTracker marks the neighbors which can be impacted when the programmer changes a class and informs the programmers about them. If these changes are necessary, the programmers will make modifications using a standard editor. After that, JTracker automatically marks additional classes. The change propagation continues until no marked classes exist in the program. JTracker scans the source code and creates a database of classes and their dependencies. JTracker was written in Java as an extension of the Borland JBuilder5 environment. We share an interest in the CIA problem. However, we focus on applying the WAVE-CIA [10] for C# projects.

In 2007, Huang and Song introduced a dynamic impact analysis approach that relies on program executions [4]. They took into account the peculiarities of object-oriented programs and conducted a dependency analysis by eliminating components that are not dependent on the modified components. We share an interest in CIA analysis. However, this paper focuses on the Java programming language rather than C#.

In 2010, Sun et al. introduced a static CIA method that

[9]https://rigi.io/

[10]https://github.com/saltlab/tochal

accounts for various impact mechanisms and rules associated with different change types to compute the impact sets [3]. They proposed an intermediate representation for object-oriented Java programs and subsequently performed CIA based on the impact rules of different change types. Sun focused on the Java programming language rather than C#.

In 2020, Mondal et al. proposed a method to find impact sets using rule association and code clone detection [5]. The rule association is based on the previous co-change history of program entities. Code clones are detected using a clone detector. Mondal et al. conducted an investigation on several projects written in Java, C++, and C#. While we share an interest in the CIA problem, our focus lies in applying WAVE-CIA [10] to C# projects.

VII. Conclusion

This paper proposes a Wave-CIA-based method, named CIA4CS, to analyze the impact of changed components of C# projects. The method has been implemented in CIA4CS Tool and used to perform some experiments with some common C# projects in the community. Experimental results show that the tool effectively detects the change set and impact set of the given source code. The tool is a solution to support both the development team and the software company in their daily work in software quality assurance.

In the future, several features of CIA4CS Tool will be implemented for the user to have a more complete solution with a better user experience. First, we will add the ability to analyze the dependencies between *.cshtml files in ASP.NET projects or *.xaml files in WPF projects. In addition, other optimizations can also be implemented such as improving the speed of the dependency graph-building process, comparing versions, improving the algorithm to determine the impact set, or defining existing dependencies. Moreover, the representation of components on the graph for large-scale projects needs to be improved and optimized. Last but not least, some features like the report export and CI/CD integration support need to be added to the tool. By implementing these improvements, we aim to help users have a better user experience, and make the tool capable of being used practically in an enterprise environment.

References

[1] M. Kretsou, A. A. Elvira-Maria Arvanitou, I. Deligiannis, and V. C. Gerogiannis, "Change impact analysis: A systematic mapping study," *Journal of Systems and Software*, vol. 174, no. 110892, (2021).

[2] B. Li, X. Sun, H. K. N. Leung, and S. Zhang, "A survey of code-based change impact analysis techniques," *Software Testing*, vol. 23, (2013).

[3] X. Sun, B. Li, C. Tao, W. Wen, and S. Zhang, "Change Impact Analysis Based on a Taxonomy of Change Types," in *2010 IEEE 34th Annual Computer Software and Applications Conference*, (2010): 373–382.

[4] L. Huang and Y.-T. Song, "Precise dynamic impact analysis with dependency analysis for object-oriented programs," in *5th ACIS International Conference on Software Engineering Research, Management & Applications (SERA 2007)*, (2007): 374–384.

[5] M. Mondal, B. Roy, C. K. Roy, and K. A. Schneider, "Associating Code Clones with Association Rules for Change Impact Analysis," in *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, (2020): 93–103.

[6] J. Buckner, J. Buchta, M. Petrenko, and V. Rajlich, "Jripples: A tool for program comprehension during incremental change," 06 (2005): 149 – 152.

[7] L. B. Cuong, V. S. Nguyen, D. A. Nguyen, P. N. Hung, and D. H. Vo, "Jcia: A tool for change impact analysis of java ee applications," in *Information Systems Design and Intelligent Applications*, V. Bhateja, B. L. Nguyen, N. G. Nguyen, S. C. Satapathy, and D.-N. Le, Eds. Singapore: Springer Singapore, (2018): 105–114.

[8] P. Anderson and M. Zarins, "The CodeSurfer software understanding platform," in *13th International Workshop on Program Comprehension (IWPC'05)*, (2005): 147–148.

[9] S.Ãlimadadi, A. Mesbah, and K.P̃attabiraman, "Hybrid DOM-Sensitive Change Impact Analysis for JavaScript," in *29th European Conference on Object-Oriented Programming (ECOOP 2015)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), J. T. Boyland, Ed., vol. 37. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, (2015): 321–345. [Online]. Available:

[10] B. Li, Q. Zhang, X. Sun, and H. Leung, "Wave-cia: A novel cia approach based on call graph mining," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, ser. SAC '13. New York, NY, USA: Association for Computing Machinery, (2013): 1000–1005. [Online]. Available:

[11] ECMA-334, *C# Language Specification*. Ecma Internationa, 4th Edition, 06 (2006).

[12] M. Chaumun, H. Kabaili, R. K. Keller, and F. Lustman, "A change impact model for changeability assessment in object-oriented software systems," *Science of Computer Programming*, vol. 45, no. 2, (2002): 155–174, special Issue on Software Maintenance and Reengineering (CSMR 99). [Online]. Available:

[13] M. Zalewski and S. Schupp, "Change Impact Analysis for Generic Libraries," in *2006 22nd IEEE International Conference on Software Maintenance*, (2006): 35–44.

[14] G. Pirklbauer, C. Fasching, and W. Kurschl, "Improving change impact analysis with a tight integrated process and tool," in *2010 Seventh International Conference on Information Technology: New Generations*, (2010): 956–961.

[15] H. Kienle and H. Müller, "Rigi—an environment for software reverse engineering, exploration, visualization, and redocumentation," *Science of Computer Programming*, vol. 75, 04 (2010): 247–263.

[16] S. Ibrahim, M. Munro, and A. Deraman, "A requirements traceability to support change impact analysis," *Asian J Inform Technol*, vol. 4, 01 (2005).

[17] S. Black, "Computing ripple effect for software maintenance," *Journal of Software Maintenance*, vol. 13, 07 (2001): 263–279.

[18] S. Gwizdala, Y. Jiang, and V. Rajlich, "JTracker - A Tool for Change Propagation in Java." 04 (2003): 223 – 229.

[19] M. Petrenko and V. Rajlich, "Variable granularity for improving precision of impact analysis," in *2009 IEEE 17th In-*

ternational Conference on Program Comprehension, (2009): 10–19.
[20] J. Buckner, J. Buchta, M. Petrenko, and V. Rajlich, "JRipples: a tool for program comprehension during incremental change," in *13th International Workshop on Program Comprehension (IWPC'05)*, (2005): 149–152.
[21] S. Yau, J. Collofello, and T. MacGregor, "Ripple effect analysis of software maintenance," in *The IEEE Computer Society's Second International Computer Software and Applications Conference, 1978. COMPSAC '78.*, (1978): 60–65.
[22] R. Mohamad, "A change impact analysis approach using visualization method," 01 (2010).

Hoang-Viet Tran received his B.E. from Hanoi University of Science and Technology in 2005, MSc., and Ph.D. from VNU University of Engineering and Technology (2014, 2021). He is now a lecturer at VNU University of Engineering and Technology. His research interests include model checking, assume-guarantee verification, software testing, software evolution, and artificial intelligence for software engineering.

Email: thv@vnu.edu.vn

Nguyen Thi Mai Loan received a Bachelor's degree in Information Technology from VNU University of Engineering and Technology, Hanoi, Vietnam, in 01/2024. She is now a developer at Rikkeisoft Corporation. Her research interests include software testing, mobile applications, and web development.

Email: 20020114@vnu.edu.vn

Doan Duc Kien is currently a third-year Computer Science student at VNU University Of Engineering and Technology, Hanoi, Vietnam. His research interests include automated software testing and applying machine learning methods in software quality assurance.

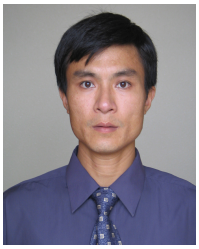
Email: 21020207@vnu.edu.vn

Nguyen Ha Trang received a Bachelor's degree in Information Technology from VNU University of Engineering and Technology, Hanoi, Vietnam, in 2024. She is now a software developer at FPT Software, Hanoi. Her research interests include web development, software quality assurance, and automated testing.

Email: 20020482@vnu.edu.vn

Le Van Huy is currently a 4^{th} year Information Technology student at VNU University of Engineering and Technology, Ha Noi, Vietnam. His research interests include web, cloud computing, and automated testing.

Email: 20020197@vnu.edu.vn



Pham Ngoc Hung received his B.S. degree from VNU University of Engineering and Technology in 2002, M.S. and PhD. degrees from Japan Advanced Institute of Science and Technology (2006, 2009). He is now an assoc. prof. at VNU University of Engineering and Technology. His research interests include model checking, assume-guarantee verification, model-based testing, and software evolution.

Email: hungpn@vnu.edu.vn

A Recommendation Method for an Online Programming Portal

Nguyen Duy Anh[1], Nguyen Duy Phuong[2], Nguyen Van Tien[2]

[1] Department of Cognitive Science and Artificial Intelligence Tilburg University Tilburg, Netherlands
[2] Faculty of Information Technology 1 Posts and Telecommunications Institute of Technology Hanoi, Vietnam

Correspondence: Nguyen Van Tien, tiennv@ptit.edu.vn
Communication: received 28 Aug 2023, revised 6 Oct 2023, accepted 16 Oct 2023
DOI: 10.32913/mic-ict-research.v2024.n1.1227

***Abstract*: An online programming portal is a valuable resource for Information Technology students to enhance their programming skills. It offers an environment where educators and learners can create problems, generate test data, program, and automatically evaluate tests. Numerous universities, both domestic and international, have achieved remarkable success by developing digital content for these portals. However, a common challenge for learners is locating problems that align with their expertise in the vast database covering various topics. In this paper, we propose a collaborative filtering recommendation approach integrated into the Online Programming Portal. This method aims to suggest a suitable set of problems based on each user's programming proficiency. Experiments conducted on the Online Programming Portal at the Post & Telecommunications Institute of Technology (PTIT) demonstrate a significant enhancement in students' online problem-solving results.**

Keywords: *Online Programming Portal Collaborative Filtering Recommendation, Content-based Filtering Recommendation, Item-based Collaborative Filtering Recommendation, User-based Collaborative Filtering Recommendation.*

I. INTRODUCTION TO THE ONLINE PROGRAMMING PORTAL

The computer community is witnessing an unprecedented development of online programming technologies. Online programming technologies provide learners with a multi-language programming environment, allowing them to code and automatically grade their programming results. [1]. Accompanying learners on online programming portals are universities and major economic corporations, all aiming to train and recruit a high-quality IT workforce. Leading IT corporations such as Microsoft, Google, Facebook, Apple, and Samsung, not only establish their online programming portals but also provide financial support and digital content for online programming portals of other universities. [1–3]. Major universities such as MIT, Stanford, and Baylor consider online programming portals as essential tools for teaching, training, and assessing the programming skills of IT engineers. Recognizing the effectiveness of online programming technologies, several universities in Vietnam, including Hanoi National University, Ho Chi Minh City National University, Hanoi University of Science and Technology, and Posts and Telecommunications Institute of Technology, have developed their online programming portals and successfully implemented them in teaching and honing the programming skills of students.

For online programming portals, the most crucial resource is the digital content repository integrated into the platform [1]. This digital content repository consists of a collection of problems, each accompanied by corresponding test datasets. Each problem requires multiple test datasets, with each dataset specifying a correct property that the programming solution must achieve. A programming solution is considered good if it passes all the test datasets with defined time and memory constraints. The digital content repository is constructed and maintained by experts from the organization owning the online programming portal. The larger the digital content repository, the more attractive the online programming platform becomes, leading to a greater number of users' participation [1–3]

The next important resource for an online programming portal is its user community. A large user community indicates the high value of the portal's digital content repository [1]. Most users participate in learning programming, while some join to enhance their programming skills. Others engage in showcasing their programming proficiency and may even compete in national and international programming competitions to validate their expertise and potentially enter the high-quality job market or achieve awards in programming contests at the national and international levels.

Some online programming portals, such as https://codeforces.com/, https://topcoder.com/, and

https://www.hackerrank.com/, attract hundreds of thousands of programmers worldwide to participate, as their digital content is continuously updated. Therefore, building an advisory system to provide tailored digital content and learning progress based on each user's programming abilities is of utmost importance for these online programming portals.

There have been several studies on recommendation systems in online learning environments. Recommendation systems in education have also been explored in recent research trends. In 2019, Carlo De Medio and colleagues conducted research on content recommendation systems for instructors on online learning platforms using the Moodle platform [4]. Asra Khalid and Karsten Lundqvist proposed a new recommendation method for Massive Open Online Courses (MOOCs) in 2021 [5].

Some studies by authors Huynh Ly Thanh Nhan and Nguyen Thai Nghe have focused on predicting students' academic performance using open-source systems [6] or course selection advisory systems [7]. However, these studies may not align well with online programming learning platforms.

In this paper, we propose a collaborative filtering advisory method to enhance the effectiveness of online programming portals. The method is approached by considering the collaborative advisory problem on the online programming portal as a graph-based search problem. Leveraging graph representation, we propose a method to predict the suitability of programming content for each user based on their programming abilities. The method is experimentally evaluated on real-world data collected from the online programming portal of the Posts and Telecommunications Institute of Technology, and it shows promising results compared to traditional collaborative filtering methods. For ease of tracking, in Section 2, we present the collaborative filtering advisory problem for online programming portals. Section 3 introduces the method for collaborative filtering advisory on online programming portals, and the final section presents the experimentation and evaluation of the results.

II. COLLABORATIVE FILTERING ADVISORY PROBLEM FOR ONLINE PROGRAMMING PORTALS

Collaborative filtering recommendation is a popular method in building advisory systems widely applied in e-commerce [8, 9]. The method is built from a set of users $U = \{u_1, u_2, \ldots, u_n\}$ and a set of items $P = \{p_1, p_2, \ldots, p_m\}$. Each user $u_i \in U$ provides their ratings for certain items $p_x \in$ P with a numerical value $r_{ix} \in \Omega$ (where Ω can be a set of integers or a set of real numbers). The rating matrix R is the sole input for collaborative filtering recommendation methods [9]. For ease of presentation, we can use the shorthand notation $i \in U$ and $x \in P$ instead of $u_i \in U$ and $p_x \in P$ respectively. Based on the rating matrix $R = \{r_{ix} : i = 1, 2, \ldots, n; x = 1, 2, \ldots, m\}$, collaborative filtering recommendation methods exploit aspects related to user communities with similar interests to provide the most suitable items for each user. The core philosophy of collaborative filtering is that users with similar preferences in the past may have shared interests in the future. Each user in the collaborative filtering advisory system operates independently from other users.

The problem of content recommendation for each user on the online programming portal can be viewed as a typical collaborative filtering recommendation problem. In this context, the user set $U = \{u_1, u_2, .., u_n\}$ consists of users participating in the online programming portal. The set U is automatically collected as users register to join the online programming platform. The item set $P = \{p_1, p_2, \ldots, p_m\}$ includes problems along with their corresponding test datasets already loaded into the online programming portal. The programming results of each user $i \in U$ solving problem $x \in P$ is automatically recorded by the online programming portal with a rating value r_{ix}. Specifically, r_{ix} is recorded as 1 if the programming solution of user $i \in U$ satisfies all the test datasets for problem $x \in P, r_{ix}$ is recorded as -1 if the programming solution of user $i \in U$ does not fully meet all the test datasets for problem $x \in P$, and r_{ix} is recorded as 0 if the user $i \in U$ has not solved problem $x \in P$. The objective of the collaborative filtering recommendation approach is to offer a collection of problems that match the programming skills of individual users on the online programming portal.

There are various proposals to address the collaborative filtering recommendation problem, and they can be categorized into two main approaches: Memory-Based Collaborative Filtering and Model-Based Collaborative Filtering [8, 9]. In this paper, we focus on the Memory-Based collaborative filtering method. Memory-Based Collaborative Filtering is approached through two primary methods: User-Based Collaborative Filtering [10, 11] and Item-Based Collaborative Filtering [11, 12]. Each method has its own advantages, exploiting aspects related to users or items. A common feature of both methods is that they utilize the entire rating dataset to predict the preferences of users for items they have not encountered before. Essentially, these are lazy learning or example-based learning methods used in machine learning [13]. Each method is carried out through four steps, as described below [10–12].

Step 1. Calculate the similarity between user pairs or item pairs. At this step, we can use correlation measures

or similarity metrics to compute the similarity between user pairs or item pairs [10–12]. Let u_{ij} be the similarity between user $i \in U$ and user $j \in U$, and p_{xy} be the similarity between item $x \in P$ and item $y \in P$. Then, the Pearson correlation between user $i \in U$ and user $j \in U$ is determined by formula (1), and the similarity between item $x \in P$ and item $y \in P$ is determined by formula (2) [10, 12].

$$u_{ij} = \frac{\sum_{x \in P_i \cap P_j} (r_{ix} - \overline{r_l})(r_{jx} - \overline{r_j})}{\sqrt{\sum_{x \in P_i \cap P_j} (r_{ix} - \bar{r}_l)^2} \sqrt{\sum_{x \in P_i \cap P_j} (r_{jx} - \overline{r_j})^2}} \quad (1)$$

$$u_{ij} = \frac{\sum_{x \in P_i \cap P_j} (r_{ix} - \bar{r_i})(r_{jx} - \bar{r_j})}{\sqrt{\sum_{x \in P_i \cap P_j} (r_{ix} - \bar{r_i})^2} \sqrt{\sum_{x \in P_i \cap P_j} (r_{jx} - \bar{r_j})^2}} \quad (2)$$

Where,

$$P_i = \{x \in P \mid r_{ix} \neq 0\} \quad (3)$$

$$\bar{r_i} = \frac{1}{|P_i \cap P_j|} \sum_{x \in P_i \cap P_j} r_{ix} \quad (4)$$

$$\bar{r_j} = \frac{1}{|P_i \cap P_j|} \sum_{x \in P_i \cap P_j} r_{jx} \quad (5)$$

$$U_x = \{i \in U \mid r_{ix} \neq 0\} \quad (6)$$

$$\bar{r_x} = \frac{1}{|U_x \cap U_y|} \sum_{i \in U_x \cap U_y} r_{ix} \quad (7)$$

$$\bar{r_y} = \frac{1}{|U_x \cap U_y|} \sum_{i \in U_x \cap U_y} r_{iy} \quad (8)$$

Step 2. Determine the neighbor set for the user in need of recommendations. At this step, we simply sort the u_{ij} or p_{xy} values in descending order, where $i \in U$ represents the user requiring item recommendations from the set of items $x \in P$. Then, select the first K_i users as the neighbor set for user i, or select the first K_x items as the neighbor set for item x [10, 12].

Step 3. Generate predictions for the user in need of recommendations. The most common method to generate predictions for user $i \in U$ on a new item $x \in P$ is through formula (9), and for items, it is through formula (10) [10–12].

$$r_{ix} = \bar{r_i} + \frac{\sum_{j \in K_i} (r_{jx} - \bar{r_j}) u_{ij}}{\sum_{j \in K_i} |u_{ij}|} \quad (9)$$

$$r_{ix} = \frac{\sum_{y \in K_x} p_{xy} r_{iy}}{\sum_{y \in K_x} |p_{xy}|} \quad (10)$$

Step 4. Formulate the recommendations. Select the top k new items with the largest rix values to recommend to user i, or choose the top k users i with the largest pix values to recommend to these users [10–12].

Despite many successes in collaborative filtering recommendation systems, the User-Based and Item-Based collaborative filtering methods still face some challenges, such as sparse data, new users, and new items [9]. Particularly, collaborative filtering methods heavily depend on the real-world semantics of data values in different application domains. In this paper, we propose a graph-based collaborative filtering method that overcomes these issues. The method proves to be effective even in cases of sparse data.

III. COLLABORATIVE FILTERING RECOMMENDATION ALGORITHM FOR ONLINE PROGRAMMING PORTALS

In order to enhance the quality of recommendations, this section presents a proposed collaborative filtering advisory algorithm for online programming portals. The method is implemented by representing the relationships between users and content items on the online programming portal as a bipartite graph. Leveraging this property, we treat the problem at hand as a graph search problem. The method is carried out as follows.

1. The method estimates the level of suitability of users for items

Let's consider a system with n users U = $U = \{u_1, u_2, .., u_n\}$ and m items $P = \{p_1, p_2, \ldots, p_m\}$. In this context, the user set U is automatically collected as users register to join the online programming portal. The item set P includes problems along with their corresponding test datasets already loaded into the online programming portal. The rating matrix $R = \{r_{ix} : i = 1, 2, \ldots, n; x = 1, 2, \ldots, m\}$ is automatically recorded in the online programming portal using formula (11).

$$r_{ix} = \begin{cases} 1 \text{ if user } i \in U \text{ submits the correct problem } x \in P \\ 0 \text{ if user } i \in U \text{ has not submitted the problem } x \in P \\ -1 \text{ if user } i \in U \text{ submits the incorrect problem } x \in P \end{cases} \quad (11)$$

It is evident that the rating matrix R, defined by formula (11), can be represented as a bipartite graph. One side of the graph comprises the user set U, while the other side includes the problem set P within the online programming portal. The edges of the graph consist of pairs $e = (i, x)$, where $i \in U$ and $x \in P$. There are no edges connecting vertices within the same side. In other words, there are no edges connecting user vertices to other user vertices, and there are no edges connecting item vertices to other item vertices. For instance, in a system with a rating matrix comprising 5 users and 7 items, as shown in Table 1, the corresponding bipartite graph representation is depicted in Figure 1. Here, the value $r_{ix} = 1$ indicates that user i has solved problem x correctly, $r_{ix} = -1$ indicates that user i

TABLE I
THE RATING MATRIX R

Users	Items						
	p1	p2	p3	p4	p5	p6	p7
u1	1	0	-1	1	0	-1	0
u2	0	1	-1	1	-1	0	-1
u3	-1	1	1	-1	0	0	1
u4	-1	-1	0	0	1	-1	1
u5	0	1	0	-1	1	1	0

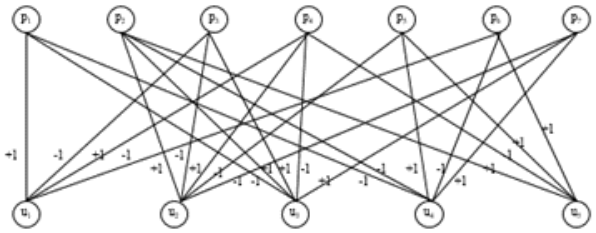


Figure 1. Bipartite graph corresponding to R

has not solved problem x correctly, and $r_{\text{ix}} = 0$ indicates that user i has not attempted problem x yet.

According to the property of the bipartite graph, the paths from the vertex of user $i \in U$ to the vertex of item $x \in P$ always have odd lengths ($L = 1, 3, 5, 7 \cdots$). For example, the paths $u1-p4-u3-p2, u1-p3-u2-p7, u1-$ $p1-u2-p2$ are paths of length 3 , and the paths $u1-p4-$ u2-p2-u3-p7, u2-p3-u1-p6-u4-p1, u1-p1-u3-p7-u4-p2 are paths of length 5 . All paths from the vertex of user $i \in U$ to the vertex of item $x \in P$ can be classified into three types: type 1 includes paths that only pass through edges with positive weight (+1), type 2 includes paths that only pass through edges with negative weight (-1), and type 3 includes paths that pass through edges with either positive (+1) or negative (-1) weights. For example, $u1-p4-u3-p2, u1-$ $p4-u2-p2-u3-p7$ are type 1 paths; $u1-p3-u2-p7, u2-p3-u1-p6-u4-p1$ are type 2 paths; $u1-p1-u2-p2, u1-p1-u3-$ $p7-u4-p2$ are type 3 paths.

By visual observation, if the majority of the paths of length L from vertex $i \in U$ to vertex $x \in P$ belong to type 1 among all three types of paths, then the prediction that item x is suitable for user i is most likely correct. In this case, we predict the opinion of user i about the new item x to be $r_{\text{ix}} = 1$. If the majority of the paths of length L from vertex $i \in U$ to vertex $x \in P$ belong to type 2 among all three types of paths, then the prediction that item x is not suitable for user i is most likely correct. In this case, we predict the opinion of user i about the new item x to be $r_{\text{ix}} = -1$. In the final case, if the majority of the paths of length L from vertex $i \in U$ to vertex $x \in P$ belong to type 3 among all three types of paths, then we cannot predict whether the item x is suitable or not for user i. In this case, we predict the opinion of user i about the new item x to be $r_{\text{ix}} = 0$. Based on this observation, we propose the method to calculate the suitability of users for new items as follows.

Let $W = \{w_{ix} : i = 1, 2, .., n; x = 1, 2, \ldots, m\}$ be the adjacency matrix representing the bipartite graph of the rating matrix R, defined by formula (12). $W^1 = \left\{w^1_{ix} : i = 1, 2, \ldots, n; x = 1, 2, \ldots, m\right\}$ is the adjacency matrix representing the bipartite graph for positive rating values, determined by the formula (13). $W^2 = \left\{w^2_{ix} : i = 1, 2, \ldots, n; x = 1, 2, \ldots, m\right\}$ is the adjacency matrix representing the bipartite graph for negative rating values, determined by the formula (14). In other words, we divide the bipartite graph representation of the rating matrix R into two subgraphs, where subgraph W^l only includes edges with positive weights (+1), and subgraph W^2 only includes edges with negative weights (−1).

$$W = \begin{cases} 1 & \text{if } r_{ix} \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

$$W^1 = \begin{cases} 1 & \text{if } r_{ix} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

$$W^2 = \begin{cases} 1 & \text{if } r_{ix} < 0 \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

In that case, the total number of all paths with length L from vertex $i \in U$ to vertex $x \in P$ in the entire rating matrix R is determined by the formula (15) [14]. This is also the count of all three types of paths from vertex $i \in U$ to vertex $x \in P$. The number of type 1 paths (paths that pass through edges with weight 1) from vertex $i \in U$ to vertex $x \in P$ is determined by the formula (16) [14]. The number of type 2 paths (paths that pass through edges with weight -1) from vertex $i \in U$ to vertex $x \in P$ is determined by the formula (17) [14]. The number of type 3 paths (paths that pass through edges with weight 1 or -1) from vertex $i \in U$ to vertex $x \in P$ is determined by formula (18) [14]. In these formulas, W^T is the transpose matrix of W, $\left(W^1\right)^T$ is the transpose matrix of W^1, and $\left(W^2\right)^T$ is the transpose matrix of W^2.

$$W^L = \begin{cases} W & \text{if } L = 1 \\ W \cdot W^T \cdot W^{L-2} & \text{if } L = 3, 5, \ldots \end{cases} \tag{15}$$

$$\left(W^1\right)^L = \begin{cases} W^1 & \text{if } L = 1 \\ W^1 \cdot \left(W^1\right)^T \cdot \left(W^1\right)^{L-2} & \text{if } L = 3, 5, \ldots \end{cases} \tag{16}$$

$$\left(W^2\right)^L = \begin{cases} W^2 & \text{if } L = 1 \\ W^1 \cdot \left(W^2\right)^T \cdot \left(W^1\right)^{L-2} & \text{if } L = 3, 5, \ldots \end{cases} \tag{17}$$

$$\left(W^3\right)^L = W^L - \left(W^1\right)^L - \left(W^2\right)^L \quad (18)$$

So, we have determined the number of each type of path with length L from vertex $i \in U$ to vertex $x \in P$. Let's call MAX the number of paths with length L that is the largest from vertex $i \in U$ to vertex $x \in P$, which is determined by the formula (19). Then, the method to predict the level of suitability for user $i \in U$ with new items $x \in P$ is determined by the formula (20).

$$MAX = max\left\{w_{ix}^L : i = 1,\ 2,\ ..,n; x = 1,\ 2,\ ..,m\right\} \quad (19)$$

$$r_{ix} = \left\{ \begin{array}{ll} 1 & \text{if } \frac{(w^1)_{ix}^L}{MAX} > 0.5 \\ 0 & \text{if } \frac{(w^3)_{ix}^L}{MAX} > 0.5 \\ -1 & \text{if } \frac{(w^2)_{ix}^L}{MAX} > 0.5 \end{array} \right\} \quad (20)$$

In the prediction formula (20), to limit the pairs (i, x) that have a small number of paths with length L but have a larger number of paths with length L belonging to each type compared to w_{ix}^L, we compare it with the maximum value in the matrix W^L for comparison. The predicted value of user i 's opinion for the new item x is $r_{ix} = 1$ when the ratio $\frac{(w^1)_{ix}^L}{MAX} > 0.5$. This means that the number of paths with length L from vertex i to vertex x must be sufficiently large, and the majority of these paths must pass through edges with positive weights.

Similarly, the predicted value $r_{ix} = 1$ when the number of paths with length L from vertex i to vertex x must be sufficiently large, and the majority of these paths must pass through edges with negative weights. The predicted value $r_{ix} = 0$ when the number of paths with length L from vertex i to vertex x must be sufficiently large, and the majority of these paths must pass through edges with both positive and negative weights. Based on the prediction method developed according to (20), we propose a collaborative filtering recommendation algorithm for the online programming portal as in Section III.2.

2. The Collaborative Filtering Recommendation Algorithm for Online Programming Portal

The Collaborative Filtering Recommendation Algorithm for Online Programming Portal (GraphBased) is performed through four steps as shown in Algorithm 1. In Step 1 of the algorithm, we construct bipartite graphs. The graph W is used to determine the number of paths with length L from vertex $i \in U$ to vertex $x \in P$. The graph W^1 is used to determine the number of paths with length L from vertex $i \in U$ to vertex $x \in P$ that only traverse edges with positive weights. The graph W^2 is used to determine the number of paths with length L from vertex $i \in U$ to vertex $x \in P$ that only traverse edges with negative weights.

In Step 2 of the algorithm, we find the number of paths with length L from vertex $i \in U$ to vertex $x \in P$ in the graph W. This allows us to observe indirect relationships between

Algorithm 1 The GraphBased algorithm.

Inputs: The rating matrix R is determined according to the formula (11).
Outputs: The list of k most suitable new items $p_x \in P$ for the user $p_i \in U$.
The steps are as follows:
Step 1. Constructing bipartite graphs from the rating matrix R:
1.1. Build the adjacency matrix representing the bipartite graph over the entire R according to the formula (12): $W = \begin{cases} 1 & \text{if } r_{ix} \neq 0 \\ 0 & \text{otherwise} \end{cases}$
1.2. Build the adjacency matrix representing the bipartite graph for ratings with a value of 1 using the formula (13):

$$W^1 = \left\{ \begin{array}{ll} 1 & \text{if } r_{ix} > 0 \\ 0 & \text{otherwise} \end{array} \right.$$

1.3. Build the adjacency matrix representing the bipartite graph for ratings with a value of -1 using the formula (14): $W^2 = \begin{cases} 1 & \text{if } r_{ix} < 0 \\ 0 & \text{otherwise} \end{cases}$
Step 2. Finding the number of paths of length L for each type:
2.1. Find the number of paths of length L from vertex $i \in U$ to vertex $x \in P$ over the entire R using formula (15): $W^L = \begin{cases} W & \text{if } L = 1 \\ W \cdot W^T \cdot W^{L-2} & \text{if } L = 3, 5, \ldots \end{cases}$
2.1. Find the number of type 1 paths of length L from vertex $i \in U$ to vertex $x \in P$ using formula (16):

$$\left(W^1\right)^L = \left\{ \begin{array}{l} W^1 \\ W^1 \cdot \left(W^1\right)^T \cdot \left(W^1\right)^{L-2} \quad \text{if } L = 3, 5, \ldots \end{array} \right.$$

2.2. Find the number of type 2 paths of length L from vertex $i \in U$ to vertex $x \in P$ using formula (17):

$$\left(W^2\right)^L = \left\{ \begin{array}{ll} W^2 & \text{if } L = 1 \\ W^1 \cdot \left(W^2\right)^T \cdot \left(W^1\right)^{L-2} & \text{if } L = 3, 5, \ldots \end{array} \right.$$

2.3. Find the number of type 3 paths of length L from vertex $i \in U$ to vertex x ∈ P using formula (18):

$$\left(W^3\right)^L = W^L - \left(W^1\right)^L - \left(W^2\right)^L$$

Step 3. Predicting the opinions of $i \in U$ regarding the new items $x \in P$:
3.1. Find the number of paths of length L from vertex $i \in U$ to vertex $x \in P$ over the entire R using formula (15): $MAX = \max\left\{w_{ix}^L : i = 1, 2, \ldots, n; x = 1, 2, \ldots, m\right\}$
3.2. Generate predictions for the opinions of $i \in U$ regarding the new items $x \in P$:

$$r_{ix} = \left\{ \begin{array}{ll} 1 & \text{if } \frac{(w^1)_{ix}^L}{MAX} > 0.5 \\ 0 & \text{if } \frac{(w^3)_{ix}^L}{MAX} > 0.5 \\ -1 & \text{if } \frac{(w^2)_{ix}^L}{MAX} > 0.5 \end{array} \right.$$

Step 4. Providing recommendations for user $i \in U$ for the new items $x \in P$:
4.1. Sort r_{ix} in ascending order of weights.
4.2. Select the first k new items with $r_{ix} = 1$ to recommend to user i.

user pairs in determining similarity. The result obtained is matrix W^{L}, which records the number of all types of paths with length L from vertex $i \in U$ to vertex $x \in P$. Next, we find the number of paths with length L that traverse edges with positive weights in the matrix $\left(W^{1}\right)^{L}$, and the number of paths with length L that traverse edges with negative weights in the matrix $\left(W^{1}\right)^{L}$. Finally, we determine matrix $\left(W^{1}\right)^{L}$ which records the number of paths with length L that traverse edges with both positive and negative weights.

In Step 3 of the algorithm, we proceed to fill in the predicted values rix according to the formula (20). In this step, some labels initially having $r_{\mathrm{ix}} = 0$ are replaced with the value 1 , while others are replaced with the value -1 . The remaining parts have a value of 0 , corresponding to the situation where we cannot make a prediction. The detailed algorithm is illustrated in Algorithm 1.

IV. EXPERIMENTATION AND EVALUATION

The GraphBased algorithm proposal was experimentally conducted on a dataset collected from the online programming portal of the Posts and Telecommunications Institute of Technology. The process of constructing the dataset and the experimental results were evaluated and compared with other methods according to the procedure described below.

1. Experimentation Methodology

Firstly, the entire test data is divided into two parts, one part U_{tr} is used as the training data, and the remaining part U_{te} is used for testing. The U_{tr} set contains 80% of the ratings, and the U_{te} set contains 20% of the ratings. The training data is used to build the model using the algorithm described above. For each user i in the test dataset, their ratings (already available) are split into two parts: O_{i} and P_{i}.O_{i} is considered as the known data, while P_{i} is the ratings to be predicted using the training data and O_{i} [9, 10, 12].

The prediction error MAE_{u} for each customer u in the test dataset is calculated as the mean absolute error between the predicted values and the actual values for all items in the set P_{u}.

$$MAE_u = \frac{1}{|P_u|} \sum_{y \in P_u} \left| \hat{r}_{uy} - r_{uy} \right| \tag{21}$$

$$MAE = \frac{\sum_{u \in U_{te}} MAE_u}{|U_{te}|} \tag{22}$$

2. Test Dataset

The collaborative filtering method was directly collected by the research team from the online programming portal of the Posts and Telecommunications Institute of Technology. The dataset was collected from August 2020 to June 2023, with 6,435 users registered on the online programming portal. The repository of programming tasks was built within one year, consisting of 1,245 problems for learners to program and be automatically graded. The total number of problem submissions recorded in the online programming portal until June 2023 was 1,151,000, submitted by 1,435,000 users. For each submission, if user i correctly solved problem x, the portal recorded a value of 1. If user i attempted but did not solve problem x, the portal recorded a value of -1. If user i did not attempt problem x at all, the portal recorded a value of 0. Each user could submit a problem multiple times, but the system only recorded the final result with values 1 or -1. Out of the 6,435 users, we filtered down to 6,120 users who participated in programming at least 20 problems correctly or incorrectly for testing purposes.

We randomly selected 2,000, 3,000, and 4,000 users from the 6,120 users as the training data and randomly selected 400, 600, and 800 users from the remaining users as the test data. To evaluate the performance of the proposed method compared to other methods in cases with limited data, we adjusted the number of problems programmed by each user in the test data to be 5, 10, and 20, with the remaining submissions for prediction.

3. Comparison and Evaluation

The proposed GraphBased method in Section III was tested and compared with the following methods:

- UserBased method using Pearson correlation, which is a user-based collaborative filtering method presented in Section II [10, 11].

- ItemBased method using Pearson correlation, which is an item-based collaborative filtering method presented in Section II [11, 12].

The MAE values in Table 2 were estimated from the average of 10 random experiments. The test results showed that the proposed method consistently yielded lower MAE values than the UserBased and ItemBased methods in all cases where the known data was 5, 10, or 20 evaluations. Specifically, in the case of very sparse data with 5 known evaluations in the training dataset of 2000 users, the MAE values for the UserBased, ItemBased, and GraphBased methods were 0.2158, 0.2172, and 0.208, respectively. The MAE values of the UserBased, ItemBased, and GraphBased methods tended to decrease as the size of the training dataset increased to 3000 and 4000 users. However, the GraphBased method still outperformed the other methods, indicating its effectiveness even in sparse collaborative filtering scenarios.

TABLE II
PERFORMANCE COMPARISON OF DIFFERENT RECOMMENDER SYSTEMS.

Training dataset size	Method	Number of known ratings/reviews		
		5	10	20
2000 users	UserBased	0.2158	0.2117	0.2042
	ItemBased	0.2172	0.2146	0.1992
	GraphBased	**0.2068**	**0.1984**	**0.1872**
3000 users	UserBased	0.2147	0.2012	0.1924
	ItemBased	0.2145	0.2116	0.1967
	GraphBased	**0.2043**	**0.1877**	**0.1792**
4000 users	UserBased	0.2037	0.2117	0.1942
	ItemBased	0.2012	0.2146	0.1820
	GraphBased	**0.1987**	**0.1784**	**0.1712**

In the case of relatively complete data, specifically with 20 known evaluations, the MAE value of the GraphBased method is significantly lower than the other methods. The MAE values of the GraphBased method are 0.1812, 0.1792, and 0.1712, while both the UserBased and ItemBased methods yield MAE values > 1.820. This can be explained by the fact that the user-based and item-based collaborative filtering methods directly estimate the degree of similarity between user-item pairs based on the intersection of item or user evaluations. However, with the graph-based method, we can infer the degree of similarity over sets of paths with positive weights and sets of paths with negative weights while not making predictions with paths containing both positive and negative weights. This allows us to leverage indirect relationships in the prediction results.

The number of students participating in programming learning and skill development has doubled within 4 months after implementing the collaborative filtering recommendation method into the online programming portal. The number of student submissions has increased from 2000 per week to 8000 per week. The average online practice time for each student is 98 hours, compared to 8 hours of offline practice according to the teaching program. These results further affirm the important role of the online programming portal in enhancing the programming skills of learners. The recommendation method established on the online programming portal is an effective tool that supports users in finding suitable problems based on their programming abilities.

V. CONCLUSION

The paper has proposed a collaborative filtering recommendation method on the online programming portal of the Posts and Telecommunications Institute of Technology to provide each user with a set of problems suitable for their programming abilities. The proposed model has proven effective even in cases of sparse data, which is a common challenge for other collaborative filtering recommendation models. The research team has also constructed a relatively comprehensive collaborative filtering dataset to share with the research community interested in using it. The proposed collaborative filtering method has brought practical benefits to online programming learners at the Posts and Telecommunications Institute of Technology.

In addition to the collaborative filtering dataset, we are currently working on building datasets for content-based recommendations based on user profiles, problem topic information, user-submitted programs, and user comments on solutions to problems on the online programming portal. This will serve as valuable data for implementing content-based recommendation methods, hybrid recommendation methods and experimenting with other recommendation approaches.

REFERENCES

[1] S. Wasik, M. Antczak, J. Badura, A. Laskowski, and T. Sternal, "A survey on online judge systems and their applications," *ACM Comput. Surv. (CSUR)*, vol. 51, pp. 1–34, 2018.

[2] J. Liu, S. Zhang, Z. Yang, Z. Zhang, J. Wang, and X. Xing, "Online judge system topic classification," 2018, pp. 993–1000.

[3] P. Antonucci, C. Estler, D. Nikolić, M. Piccioni, and B. Meyer, "An incremental hint system for automated programming assignments," 06 2015, pp. 320–325.

[4] C. Medio, C. Limongelli, F. Sciarrone, and M. Temperini, "Moodlerec: A recommendation system for creating courses using the moodle e-learning platform," *Computers in Human Behavior*, vol. 104, 03 2020.

[5] A. Khalid, K. Lundqvist, A. Yates, and M. a. Ghazanfar, "Novel online recommendation algorithm for massive open online courses (nor-moocs)," *PLOS ONE*, vol. 16, p. e0245485, 01 2021.

[6] H.-L. Thanh-Nhan and N. Thai-Nghe, "A system for predicting students's course result using a free recommender system library - mymedialite," 11 2013.

[7] D. Tran Thanh, L. Sang, N. Thanh Hai, and N. Thai-Nghe, "Course recommendation with deep learning approach," *Communications in Computer and Information Science*, vol. 1306, pp. 63–77, 11 2020.

[8] F. Tawfiq, A. M. Rahma, and H. Abdul wahab, "Recommendation systems for e-commerce systems an overview," *Journal of Physics: Conference Series*, vol. 1897, p. 012024, 05 2021.

[9] N. P. Raval and V. Khedkar, "A review paper on collaborative filtering based moive recommedation system," 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:220844886

[10] Z. Zhang, T. Peng, and K. Shen, "Overview of collaborative filtering recommendation algorithms," *IOP Conference Series: Earth and Environmental Science*, vol. 440, p. 022063, 03 2020.

[11] P. Thakkar, K. Varma (Thakkar), V. Ukani, S. Mankad, and S. Tanwar, *Combining User-Based and Item-Based Collaborative Filtering Using Machine Learning: Proceedings of ICTIS 2018, Volume 2*, 01 2019, pp. 173–180.

[12] M. Kharita, A. Kumar, and P. Singh, "Item-based collaborative filtering in movie recommendation in real time," 12 2018, pp. 340–342.
[13] B.-B. Cui, "Design and implementation of movie recommendation system based on knn collaborative filtering algorithm," *ITM Web of Conferences*, vol. 12, p. 04008, 01 2017.
[14] T. Phuong, D. T. Lien, and N. D. Phuong, "Graph-based context-aware collaborative filtering," *Expert Systems with Applications*, vol. 126, 02 2019.

Nguyen Duy Anh born on September 26, 2001, is currently a fourth-year student majoring in Artificial Intelligence at Tilburg University in the Netherlands. His research interests include machine learning, artificial intelligence, and information filtering.
Email: ndanh@gmail.com

Nguyen Duy Phuong born on February 20, 1965 in Hanoi. Graduated from Hanoi University of Science in 1998, defended a doctoral thesis at the University of Engineering and Technology, Hanoi National University in 2010. Currently working at the Post and Telecommunications Institute of Technology. Research focus: Machine Learning Applications in Information Filtering, Expert Systems, and Artificial Intelligence.
Email: ndaphuong@gmail.com

Nguyen Van Tien received a Bachelor's degree in Information Technology in 2018 and a Master's degree in Information Systems in 2020 from the Posts and Telecommunications Institute of Technology. From June 2020 to the present, Mr. Nguyen Van Tien has been a lecturer at the Faculty of Information Technology 1, Posts and Telecommunications Institute of Technology. His main research directions include high-performance computing systems and deep learning-based recommendation systems
Email: nvtien@gmail.com

A Meta-Heuristic Approach for Enhancing Performance of Associative Classification

Nguyen Quoc Huy[1], Tran Anh Tuan[1], Nguyen Thi Ngoc Thanh[2], Do Nhu Tai[3]

[1]Saigon University, [2]Ho Chi Minh City Open University, [3]University of Economic HCMC

Correspondence: Do Nhu Tai, taidn@ueh.edu.vn
Communication: received 30 Nov 2023, revised 12 Jan 2024, accepted 22 Jan 2024
DOI: 10.32913/mic-ict-research.v2024.n1.1246

***Abstract*: Associative Classification is an interesting approach in data mining to create more accurate and easily interpretable predictive systems. This approach is often built on both association rule mining and classification techniques, to find a set of rules called association rules for classification (CAR) of label attributes. There are many kinds of associative classification such as CPAR, CBA, CMAR but the accuracy is still low on large datasets, and the running time is not reasonable as well. This paper proposes a heuristic approach to significantly enhance the performance of Associative Classification algorithms in running time, reducing the rule set, and accuracy on large data. Experimental results show that heuristic searching the optimal data set makes the associative classification more useful on big data, and reasonable in practice.**

Keywords: *Associative Classification, Heuristic search, CAR, Feature Selection*

I. INTRODUCTION

The association classifier is a supervised learning model that uses association rules to assign labels. The model uses association rules to label new data. Thus, the model can be seen as a list of "if-then" clauses: if a new data meets the left-hand attributes of the rule, then the data will be classified according to the right-hand value of the rule [1, 7, 8, 9]. Association classifiers without requiring prior knowledge of feature significance is very effective in heterogeneous high-dimensional data [12]. Most classifiers combine sequential scanning of the set of rules and labeling new data by matching the attributes on the left-hand side of rules. Association classifier rules inherit several metrics from association rules, such as Support or Confidence, that can be used to rank or filter the rules in the model and evaluate their quality. There are many different associative classification methods such as CBA, CMAR, and CPAR.

CBA uses techniques in association rules to classify data, this method has higher performance than traditional classification techniques. The limitation of this method is that the number of rules generated is too large in case the support threshold is low. CMAR applies FP-tree efficiently, consuming less memory and space than the CBA method. However, FP-tree will meet limitations if the data has a large number of attributes, and the memory capacity is not enough to fit. CPAR is a type of association classifier based on predicted association rules. This method combines the advantages of association classification and traditional rule-based classification by generating rules directly during the training phase to avoid missing important rules.

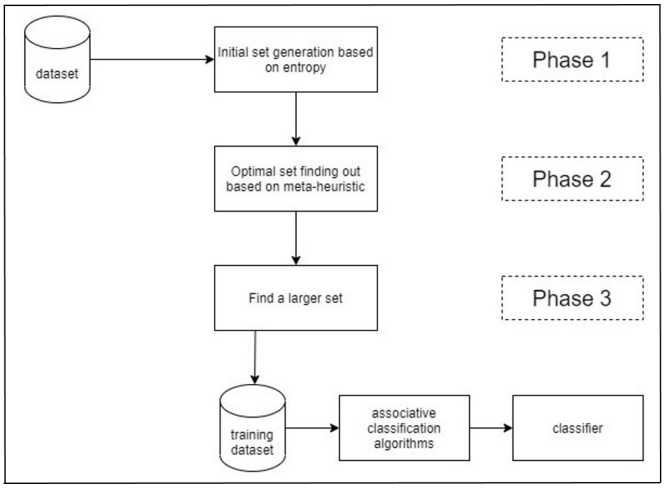


Figure 1. Three phases of heuristic approach

However, association classifier still faces many limitations in terms of time and accuracy rate of classification in case of the attributes is large [5, 6, 9, 13]. Table 1 shows the accuracy of classification, running time, as well as the number of rules of CAR algorithms in SPMF [13] are not reasonable at all. SPMF is an open-source software and data mining library written in Java, the latest version is 2.56.

This paper proposes a heuristic approach to remove unimportant attributes to enhance the associative clasification method. Our approach works in three stages as

TABLE I
LIMITATIONS OF CAR IN SPMF[13]

	CPAR	CBA	CMAR
Accuracy	0.5429	0.9828	0.8012
Rules	55	22	453
Times(ms)	1120	945343	1500

shown in Figure 1. This improvement helps descrease running time while significantly increasing accuracy rate of classification.

Phase 1 selects the initial attribute set according to the Entropy. At this stage, attributes are eliminated based on the Entropy relationship of that attribute with the class attribute. The Entropy represents a correlation to check whether the attributes play an important role with the class attribute in classification and decide whether or not to remove those attributes. At this phase, we temporarily select a candidate set that will be the initial set in phase 2.

Phase 2 finds the optimal set of attributes derived from the initial set in phase 1. The original attribute set is divided into subsets and a associative classification (AC) algorithm is run based on these subsets of data. Based on the accuracy of the model, we can add or remove randomly some attributes on the current set to obtain a new subset, and run AC repeatedly until getting the best subset of attributes. This method create subsets by using a greedy approach and evaluates the goodness of all subsets derived from the initial set in phase 1 instead of testing all combinations of possible attributes.

Phase 3 expands the optimal attribute set to increase classification accuracy. Expand the optimal set of attributes in phase 2 by adding each remaining attribute to find a new optimal set with higher accuracy.

The main contributions focus on part II, part III. The rest content is arranged as follows: part II describes our method in three phases of feature selection, and three algorithms correspondingly. The experimental results and evidences of effectiveness are described in part III, and the conclusion is presented in part IV.

II. OUR PROPOSED METHOD

With large data sets and many attributes, the cost of selecting an optimal attribute set is huge. The proposed method combines the Entropy measure and the heuristic approach to select features with three phases. Phase 1 is mainly filtering unimportant attributes according to the Entropy measure (the Entropy value between an attribute and the class attribute) and feature selection method in the form of Hill Climbing. The accuracy of the model after removing attributes was tested again using the cross-validation method. The selected attribute set will be the optimal attribute set for classification.

Algorithm 1 Phase1(FSAC)

Input: minSup, minConf, minAllConf; D: dataset (D_{train}, D_{test});

Output: D_e(all best attributes)

1: e := entropy threshold ;
2: D_e := attrribute a_i from D_{train} where entropy(a_i, label) ≤ e;
3: neighbor=[e,e ± △];
4: e' := random(neighbor);
5: $D_{e'}$:= attrribute a_i from D_{train} where entropy(a_i, label) ≤ e;
6: **repeat**
7: $model_e$:= AC-RG(D_e);
8: $accuracy_e$:= $model_e$.predict(D_{test});
9: neighbor=[e,e ± △];
10: e' := random(neighbor);
11: $model_{e'}$:= AC-RG($D_{e'}$);
12: $accuracy_{e'}$:= $model_{e'}$.predict(D_{test});
13: **if** ($accuracy_e \leq accuracy_{e'}$) **then**
14: e:=e'
15: **end if**
16: **until** $accuracy_e \leq accuracy_{e'}$
17: **return** D_e

The Entropy value is in the range [0,1], attributes with large Entropy values that will not contribute much information to the classification should be eliminated. In data sets with a few attributes, we can calculate all the entropies of the attributes and arrange them from highest to lowest. Then, each attribute is checked repeatedly for removing it from high to low entropy and evaluate again the accuracy of the remaining set of attributes. If the accuracy is equal to or greater than the old value (without removing the attribute), then the removed attribute should really be removed. Because this is a bottom-up approach, the cost of eliminating the initial large data set is not feasible. We proposes an initial Entropy threshold, instead of testing Entropy in order from high to low as in the Phase 1 algorithm.

The FSAC [11] is the algorithm for phase 1. With the initial entropy threshold, the FSAC algorithm finds the optimal entropy value that eliminates all redundant attributes. The remaining attribute set is a relatively good attribute set. Line 7 is AC-RG(D_e) algorithm that generates a large set of rules, Associative classification is a special case of association rule discovery in which only the class attribute is considered on the rule's right-hand side. Line 8 calculates the accuracy of the model obtained from line 7.

The code from lines 6 to 16 repeats the improvement until the model's accuracy reaches its peak. In each iteration, based on the current entropy neighbor (lines 11, 12), and the entropy level e', the algorithm determines the attribute set De including the attributes whose entropy are less than e'. In other words, in this iteration the algorithm tried to remove attributes which are redundant from the training set. Line 13 checks whether the removal is correct or not, if so, updates the better value. After exiting the loop, the algorithm finds a relatively good set of attributes like line number 16. The dataset D is divided into 2 groups Dtrain and Dtest according to the ratio 80/20. Based on Dtrain to find a set of classification rules, and Dtest is used to test the accuracy of the set of classification rules.

Algorithm 2 Phase2.1(Hill Climbing)

Input: The initAttr set, the subAttr set, initValid, delta
Output: bestAttr(all best attributes)

```
1:  for (; ;) do
2:      newAttr := GenAttributes(initAttr, subAttr, delta);
3:      validation := AC(newAttr);
4:      if validation > initValid  then
5:          initValid := validation;
6:          bestAttr := newAttr
7:          subAttr := allAttr - bestAttr
8:      end if
9:  end for
10: return bestAttr
```

After implementing Phase 1, we obtain an initial set of attributes that are relatively good but not really optimal because of the limitations of the Filter method for each attribute. Phase 2 bases on the heuristic approach, the starting solution is the initial set of attributes obtained from phase 1. At this phase, attributes are divided into two attribute groups (bestAttr and subAttr). The bestAttr attribute set contains the attributes obtained from phase 1, temporarily considered a good solution.

This task iterates until finding the best set with the same number of attributes compared to the attribute set obtained from phase 1. In each iteration, the new attribute set generated from the GenAttributes function in line 3 is tried, based on the delta value (neighborhood parameter) to give a delta number of any random attribute in the bestAttr set, and add the delta number of random attributes from the subAttr set to the bestAttr set. If the new attribute set has better results than the current bestAttr attribute set, update it again. This process stops when after a finite number of iterations a better set of attributes is still not found. The subAttr set contains the remaining attributes. The heuristic algorithms in Phase 2 (Hill-Climbing or Simulated Annealing) find out whether there is any set that is better than the current bestAttr set. If so, the bestAttr set will be updated. This task is from lines 2 to 9 in the algorithm 2 (Hill-Climbing), and is from lines 2 to 16 in the algorithm 3 (Simulated Annealing).

After completing phase 2, we have the optimal attribute set with the same number of attributes as in phase 1. Phase 3 is the expanding the number of attribute sets to find a better value than the set of attributes obtained in phase 2. Lines 2 to 7 in the Phase 3 algorithm is a task that tries to find a more optimal extended set by combining the set obtained in phase 3. with each remaining attribute in the subAttr set.

Algorithm 3 Phase2.2 (Simulated Annealing)

Input: The initAttr set, the subAttr set, initValid, delta, T, k
Output: bestAttr

```
1:  for (; ;) do
2:      newAttr := GenAttributes(initAttr, subAttr, delta);
3:      validation := AC(newAttr);;
4:      if validation > initValid  then
5:          initValid := validation;
6:          bestAttr := newAttr
7:          subAttr := allAttr - bestAttr
8:      else
9:          △f=|validation - initValid|
10:         r := random r(0,1)
11:         if r > exp(-△f/kT)  then
12:             initValid := validation;
13:             bestAttr := newAttr
14:             subAttr := allAttr - bestAttr
15:         end if
16:     end if
17: end for
18: return bestAttr
```

III. EXPERIMENTS

The experimental environment is processed centrally on a computer configured with Intel(R), Core(TM) i7-6820HQ CPU @ 2.70GHz (8 CPUs), 2.7GHz and Windows 10 operating system. In this paper, the experiment has 3 phases as follows:

- Phase 1: Choose the entropy threshold and initial attribute set
- Phase 2: Find the optimal set of attributes by using HC and SA algorithms
- Phase 3: Find a larger attribute set with better results than the original attribute set

Algorithm 4 Phase 3

Input: The initAttr set, the subAttr set, initValid
Output: bestAttr

```
1: bestAttr := initAttr;
2: for i ← 0; i ≤ subAttr.length; i++ do
3:     newAttr := initAttr + subAttr [i];
4:     validation := AC(newAttr);
5:     if validation > initValid then
6:         initValid := validation;
7:         bestAttr := newAttrs;
8:     end if
9: end for
10: return bestAttrs
```

1. Simple Dataset

The first experiment is implemented on Mushroom data [2]. Because this data is used by the original ACAC [1, 10] algorithm, it has 8125 rows and 23 columns. In the data, the first attribute is a target attribute named 'class' to classify mushrooms as poisonous or non-poisonous (edible=e, poisonous=p). The thresholds are minSupp = 0.1, minConf = 0.8, minAllConf = 0.5.

In Figure 2, vertical axis describes the accuracy of model, the horizontal axis describes the entropy of each removed attribute. The graph peaks at accuracy of 99.51% correspond to the entropy score of 0.757, with 8 attributes. However, the attributes are removed continuously with entropy values (0.745, 0.727, 0.714), and the accuracy still keeps at 99.51%. The classification performance will go down if one more attribute is removed. So the the entropy value of 0.714 is the best value, and gains the highest accuracy of 99.51%.

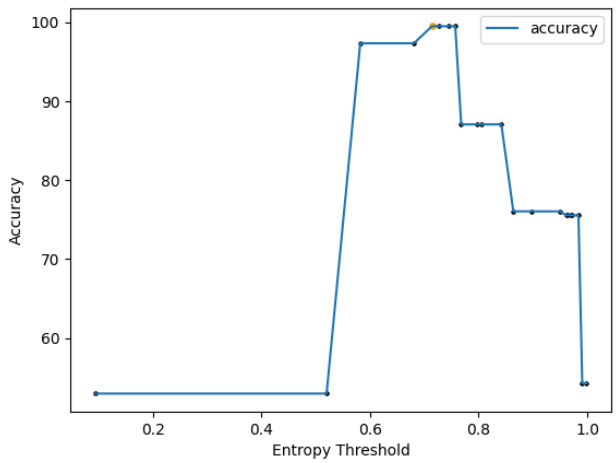


Figure 2. The accuracy of classification

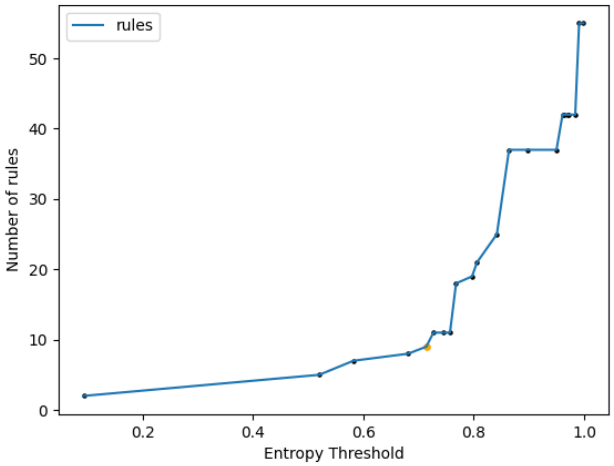


Figure 3. The optimal CAR set

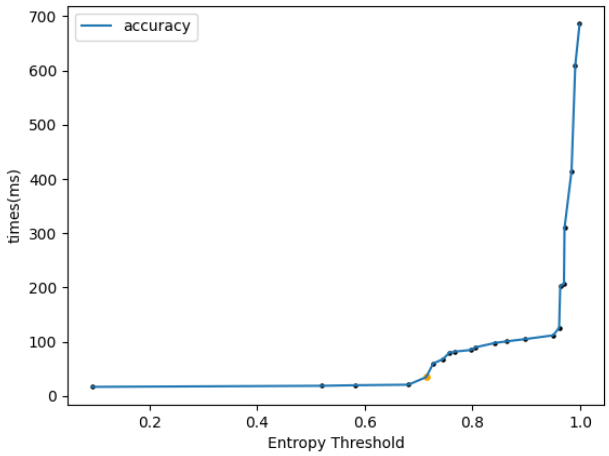


Figure 4. Running time of processing

After phase 1, the prediction model achieves very high accuracy in small datatsets (we can see the performance of CPAR, CBA, CMAR on 5 attributes of Mushroom in Table 2), so the phase 2 and phase 3 is not necessary to implement more. Our research focuses on the challenge of associative classification problem, the large dataset with many attributes, so we do experiments on more challenge dataset, SpamEmail.

TABLE II
REASONABLE VALUES ON MUSHROOM

	CPAR	CBA	CMAR
Accuracy	0.9951	0.9397	0.8876
Rules	9	8	14
Times(ms)	169	146	200

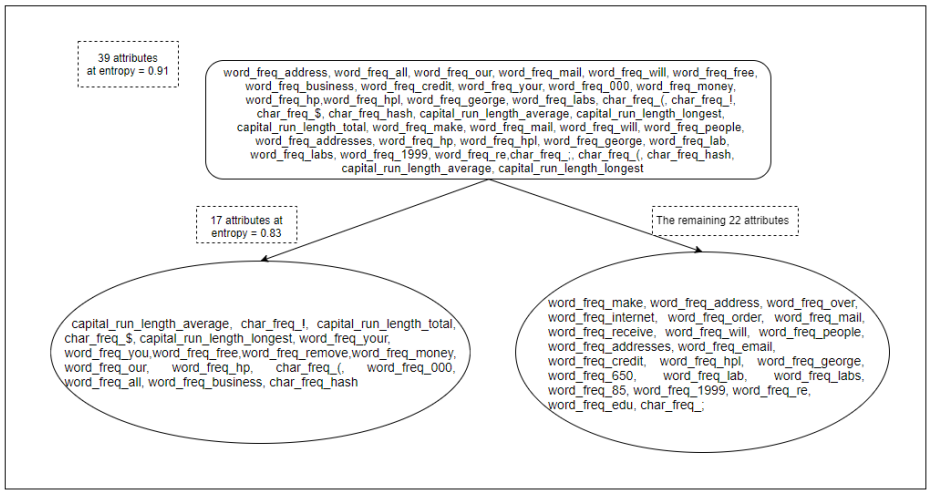


Figure 5. Divide 2 subsets from output of phase 1

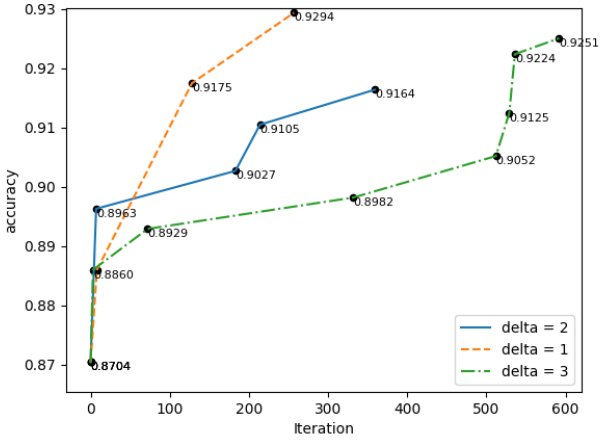


Figure 6. Wrapper with delta neighbor by SA

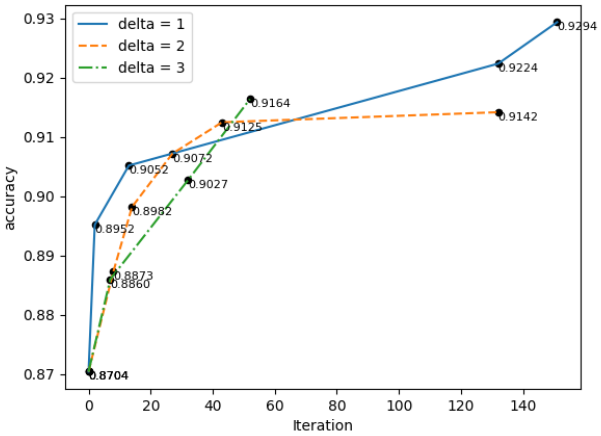


Figure 7. Wrapper with delta neighbor by HC

2. Large Dataset

The Spam Emails Dataset [3] is selected for experiment, it is a rather complex data set, with 4602 rows and 58 columns. In particular, the last attribute is a label attribute called 'spam' to classify emails as spam or spam (spam=1, spam=0). This data is also used in paper[11]. The thresholds are minSupp = 0.1, minConf = 0.9, minAllConf = 0.5.

a) Phase 1: Selecting the Entropy threshold, the initial set

Among the remaining 57 attributes, there are many attributes that have no value in classification and need to be eliminated. To eliminate, we uses entropy to determine the ability of each attribute to contribute to the classification. Any attribute with a high entropy value will be removed from the data set, the new data set will be tried again using the ACAC method for classification. This model will be tested using cross-validation method (80% - 5000 lines of data used for training, 20% - 1001 lines of data used for testing). If the precision increases then the attribute removal is correct, and this is done until the precision reaches its highest value. In other words, work stops when accuracy is degraded.

TABLE III
OUTPUT OF PHASE 1 ON SPAMEMAIL

Entropy	Attributes	rules	times(ms)	Accuracy(%)
0.91	39	?	?	?
0.885	33	3108	7.783E+13	94.2
0.88	31	2239	3.5E+11	94.4
0.875	29	1612	11E+9	94.9
0.87	28	1254	261000000	94.06
0.865	27	989	51000000	93.04
0.86	26	692	10000000	92.4
0.85	24	338	2200000	90.99
0.84	22	109	35170	88.93
0.83	17	24	343	87.04
0.81	14	17	151	81.11
0.79	13	13	145	80.61

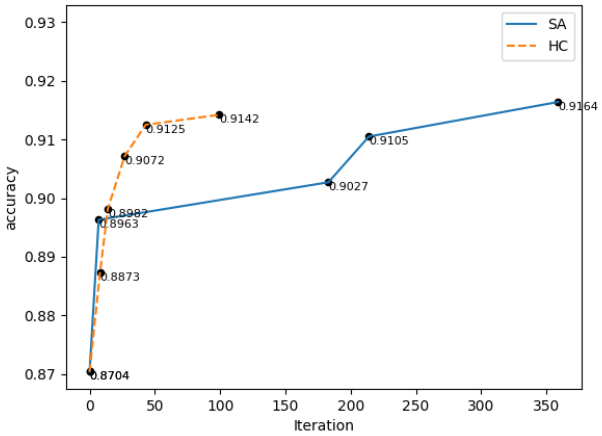


Figure 8. Comparison between HC and SA

TABLE IV
OUTPUT OF PHASE 3

Entropy	Attributes	times(ms)	rules	Accur(%)
0.308	capital_run_length_a	7862	170	9294
0.589	capital_run_length_t	7165	170	0.9294
0.658	capital_run_length_l	6891	170	0.9294
0.786	word_freq_hp	18390	170	0.9294
0.789	char_freq_(	7029	169	0.9294
0.815	word_freq_all	17124	219	0.9294
0.830	word_freq_george	19007	170	0.9294
0.835	word_freq_mail	19316	332	0.9305
0.836	word_freq_address	18231	225	0.9294
0.836	word_freq_hpl	18023	170	0.9294
0.838	word_freq_will	12262	267	0.9294
0.846	word_freq_email	13710	244	0.9294
0.866	word_freq_credit	**13292**	**278**	**0.9305**
0.871	word_freq_re	12716	194	0.9294
0.877	word_freq_people	17010	185	0.9294
0.878	word_freq_1999	18207	170	0.9294
0.884	char_freq_;	17159	327	0.9305
0.887	word_freq_addresses	17589	204	0.9305
0.892	word_freq_edu	18668	211	0.9294
0.898	word_freq_labs	17062	170	0.9294
0.898	word_freq_85	17108	170	0.9294
0.901	word_freq_650	17400	171	0.9294

In this case chosing word_freq_credit because of its performnance (Table 4, column Atrributes, accuracy = 0.9305). After go through 3 phase, We have the best attributes set so we can use Associative classfication algorithms to build prediction model.

TABLE V
HEURISTIC OUTPUT OF SPAMEMAIL

	CPAR	CBA	CMAR
Accuracy	0.9305	0.848	0.6916
Rules	278	184	158
Times(ms)	13292	2257889	3496

Based on Table 3, the entropy threshold = 0.91 is chosen to reduce the number of attributes from 57 to 39, thereby reducing space, processing time and improving evaluation indicators. Although the entropy threshold = 0.91 helps improve the limitation score compared to the original data set, the number of 39 attributes is still too large for one processing. Therefore, we propose an approach that is to divide the set of 39 attributes into 2 subsets. The first subset (original data set) includes 17 attributes at entropy threshold = 0.81, and the second subset includes the remaining 22 attributes. The reason for choosing the entropy threshold = 0.81 as the initial attribute set for experimentation is because at this threshold, although the perfomance is not the best, it is not bad to be improved and expanded more.

b) Phase 2: Find out the optimal set

After selecting the initial attribute set, there are 17 attributes at the entropy threshold = 0.81. The next step is to find a set of 17 attributes that are more optimal than the original 17 attributes. To do that, the experimental part chooses delta values of 1, 2 and 3 respectively, each different delta value will give different results. On this experimental, we select HC and SA to implement in heuristic search.

In Figure 6, it shows the peak changes (accuracy) of the data set over iterations of the program. At delta = 1, the number of peak changes when finding a set with higher accuracy than the current set is 7 times, this number at delta = 3 is 3 times and delta = 2 is 4 times. The following data sets tend to have higher accuracy than the original (0.8704) with each new peak being found, and the highest peak achievable at delta=3 across iterations is 0.9294. For delta = 3, finding the peak with accuracy = 0.9294 takes less time and also has higher accuracy than delta = 2 or delta = 3.

SA (Figure 7) is considered an improved approach to improve the disadvantages of Hill Climbing, we can refer charts comparing the indicators of each approach on the same data set (17 attributes) at different delta values.

c) Phase 3: Find a larger attribute set that has better result than the original attribute set

From phase 2, we have selected a set of rules including 17 attributes with accuracy = 0.9294 (highest peak) and the remaining 22 attributes. In phase 3, we will find a set of 18 attributes with accuracy higher than 0.9294.

To concretize the above goal, this experimental part will combine the set of 17 attributes and each of the remaining 22 attributes. Each set of 18 attributes will give different possible results.

IV. CONCLUSION

The main advantage of AC is that the set of "If-Then" rules has the ability to infer new knowledge that other classification methods can not do. Another important advantage

of AC is the ease of interpretation of labeling for new data. AC algorithms have limitations that often occur when minsupp is set to a very small value.

Experimental results show that AC algorithms on small and simple data such as Mushroom (23 attributes) are very effective. On large and complex data like SpamEmail (58 attributes), the algorithms show limitations (as shown in Table 3).

The heuristic approach still maintains almost the same classification accuracy, but running time and rule pruning are more efficient (as shown in Table 5 compared to Table 3). The comparison on the same classification accuracy (0.9305 vs 0.9304), processing time (13292 vs 51000000) is 3750 times faster, the number of rules is much reduced (278 vs 989). The running time is more important than the number of rules, that why we select 278 rules instead of 204 rules.

REFERENCES

[1] Z. Huang, Z. Zhou, T. He, and X. Wang, "ACAC: Associative Classification Based on All-Confidence," *IEEE International Conference on Granular Computing*, pp. 289–293, 2011.

[2] Mushroom Classification Dataset, [Online]. Available: *https://www.kaggle.com/datasets/uciml/mushroom-classification.*

[3] Spam Emails Dataset, [Online]. Available: *https://www.kaggle.com/datasets/yasserh/spamemailsdataset*

[4] H. F. Ong, C. Y. M. Neoh, V. K. Vijayaraj, Y. X. Low, "Information-Based Rule Ranking for Associative Classification," *ISPACS*, 2022.

[5] M. Abrar, A. Tze and S. Abbas, "Associative Classification using Automata with Structure based Merging," *IJACSAA*, vol. 10, 2019.

[6] D. L. Olson and G. Lauhoff, "Market Basket Analysis" in Descriptive Data Mining," *Springer Singapore*, 2019.

[7] K. D. Rajab, "New Associative Classification Method Based on Rule Pruning for Classification of Datasets," *IEEE Access*, vol. 7, pp. 157783-157795, 2019.

[8] H. F. Ong, N. Mustapha, H. Hamdan, R. Rosli and A. Mustapha, "Informative top-k class associative rule for cancer biomarker discovery on microarray data," *Expert Systems with Applications*, vol. 146, 2020.

[9] Majid Seyf, Yue Xu, Richi Nayak, "DAC: Discriminative Associative Classifcation", *SN Computer Science*, 2023.

[10] E.R.Omiecinski, "Alternative interest measures for mining associations in databases", *IEEE Transactions on Knowledge and Data Engineering*, vol 15, pp. 57-69, 2003.

[11] N. Q. Huy, T. A. Tuan, N. T. N. Thanh, "An efficient algorithm that optimizes the classification association rule set", *VNICT 26*, pp. 13-19, 2023.

[12] K.H.T Dam,T. G Wilson,A Legay,R Veroneze, "Packer classification based on association rule mining", *Applied Soft Computing*, Vol 127, Issue C, 2022.

[13] Philippe Fournier-Viger, "SPMF - An Open-Source Data Mining Library", *https://www.philippe-fournier-viger.com/spmf/index.php*, Version 2.59, 2022.

Quoc-Huy Nguyen received the B.S. in Information System major from HCM City University of Science, Vietnam in 2002. In 2008, he got Master in Computer Science from HCM City University of Science. In 2014, he got Doctoral Degree in Computer Science from Japan Advanced Institute of Science and Technology (JAIST), Japan. From 2008 to now, he is a lecturer in Faculty of Information Technology, Saigon University, Vietnam. His research interests are Data Mining, Computer Game, Computer Vision, Knowledge Based Systems. Currently, he is a senior lecturer, and being Head of Software Engineering Department, Saigon University.

Thi Ngoc-Thanh Nguyen received the B.S. in Information System major from HCM City University of Science, Vietnam in 2006. In 2014, she got Master in Computer Science from HCM City University of Information Technology. At present, she is a lecturer in Faculty of Information Technology, Ho Chi Minh City Open University, Vietnam. Her research interests are Data Mining, Computer Vision.

Anh-Tuan Tran is currently a 4th year student majoring in Information Technology at Saigon University (SGU), Ho Chi Minh City. His research interests are Data Mining, Heuristic algorithms.

Nhu-Tai Do received the B.S. in Information System major from HCM City University of Foreign Language – Information Technology, Vietnam in 2005, the M.S. in Information System Management from International University, Vietnam National University at HCMC, Viet Nam in 2017, and his Ph.D. degree in Artificial Intelligence Convergence, Chonnam National University, South Korea, in 2021. From 2005 to 2017. From 2021 to 2023. From 2023 to now, he is the lecturer in University of Economics Ho Chi Minh City-UEH Vietnam. His interests are pattern recognition, deep learning, computer vision, video understanding, and medical analysis.

Exploring the Feasibility of Meeting the MDS Criterion in Random Matrices and Searching for Efficient Random Involutory Hadamard MDS Matrices

Tran Thi Luong
Academy of Cryptography Techniques, Tan Trieu, Thanh Tri, Hanoi, Vietnam
Correspondence: Tran Thi Luong, luongtranhong@gmail.com
Communication: received 20 sep 2023, revised 11 Jan 2024, accepted 21 Jan 2024
DOI: 10.32913/mic-ict-research.v2024.n1.1231

***Abstract*: Maximum distance separable codes (MDS codes) have been studied for a long time in error correcting code theory and have important applications in cryptography. There are many different methods have been studied to build MDS matrices such as: from MDS codes, Cauchy matrices, Hadamard matrices, Vandermonde matrices, circulant and circulant-like matrices etc. In this paper, the construction of MDS matrix from random matrices is studied, and the possibility of satisfying the MDS condition of any random matrix is presented. Then, the method of random search of MDS matrices will be given. In addition, we propose algorithms to randomly search for efficient involutory Hadamard MDS matrices of size 4, and 8. These results will be meaningful to determine the possibility that MDS matrices can be obtained from random matrices and diversify the methods of searching for MDS matrices. The involutory Hadamard MDS matrices make the encryption and decryption processes identical, which makes them very efficient for both hardware and software implementation.**

Keywords: *MDS matrix, random matrices, Hadamard MDS matrix, Circulant-like MDS matrix*

I. INTRODUCTION

MDS matrices [5, 11, 16, 17] play an important role in cryptographic applications. They are often used for the diffusion layer of block ciphers and hash functions to provide high diffusion. Therefore, many well-known ciphers have used MDS matrices in their diffusion layer such as AES, Twofish, KHAZAD, Anubis, SHARK, Square Hierocrypt, Camellia, Manta, MUGI, WHIRLPOOL, etc.

Because of the benefits of MDS matrices, many different methods have been studied to generate them, including: from MDS codes [5, 13, 14, 15], from suitable matrices such as Cauchy matrices [1, 3], Hadamard matrices [2, 7, 10, 12], Vandermonde matrices [10], Companion matrix [6, 9], recursive MDS matrices [18-22], etc.

The method of constructing MDS matrices from MDS codes [5, 13, 14, 15] has the advantage of being able to generate a large number of MDS matrices, but the disadvantage is that we cannot control the elements within the matrix. On the other hand, constructing MDS matrices from the Vandermonde matrix [10] and the Cauchy matrix [1, 3] allows for the advantage of building large-sized MDS matrices. However, the elements in these matrices often have high Hamming weights, leading to significant execution costs.

The method of constructing MDS matrices from recursive matrices [18-22] and Companion matrices [6, 9] has the advantage of being convenient for hardware implementation. However, recursive matrices are not necessarily MDS matrices, so they always need to be checked for the MDS condition. On the other hand, checking this condition for large-sized recursive matrices can be challenging. As for Hadamard matrices, they are a special type of matrix with many advantages in execution. Works on constructing MDS matrices from Hadamard matrices [2, 7, 10, 12] also provide various methods to generate matrices of this form. However, these works do not approach the construction randomly, as well as the ability to control the elements in this matrix.

Recently, in [24], the authors shared findings from the all-encompassing exploration of (iterative) MDS matrices over $GL(4, F2)$. The investigation specifically delves into circulant MDS matrices ranging from orders 4 to 8, Hadamard MDS matrices of orders 4 and 8, as well as recursive MDS matrices derived from companion matrices of order 4, and recursive MDS matrices stemming from

sparse DSI matrices spanning orders 4 to 8. In this paper, the authors sought involutory matrices but over the field $GL(4, F2)$; furthermore, they have not considered the fixed points of these matrices. In [25], the authors introduced novel construction frameworks, specifically, transposition-permutation path configurations for 3×3 involutory and MDS permutation-equivalent matrices across F_{2^3} and F_{2^4}. They created 3×3 involutory and MDS matrices across F_{2^3} and F_{2^4}, subjecting all these matrices to analysis by identifying their permutation-equivalent matrices. Subsequently, an examination was conducted to identify any distinctive permutation patterns, particularly within matrices of this size. Consequently, they identified a total of 28,088 unique transposition-permutation path configurations, offering a direct means to construct 3×3 involutory and MDS matrices from any representative 3×3 involutory and MDS matrix over F_{2^3} and F_{2^4}. However, this work only focuses on matrices of size 3×3. In [26], a fresh hybrid approach is introduced, amalgamating search-based techniques and direct construction methods, aiming to produce the complete set of 4×4 involutory maximum distance separable (MDS) matrices over F_{2^m}. The innovative method significantly reduces the complexity of the search space to the square root of n, with n denoting the count of all 4×4 invertible matrices over F_{2^m} subject to exploration. Consequently, this approach facilitates the generation of all 4×4 involutory MDS matrices over and . In this work, the authors solely concentrated on matrices of size 4×4 over F_{2^3} and F_{2^4}.

One of the other methods to search for MDS matrices is to search from random matrices. A random matrix is a matrix whose elements are chosen randomly, independently, with uniform probability in some finite field $GF(p^r)$, where p is a prime number. In [23], the authors presened an algorithm of generating random MDS matrices, but the article is quite sketchy, and the authors cannot control the coefficients of the generated matrix according to this algorithm.

Our contributions: In this paper, a lower bound for the probability of satisfying the MDS condition of any random matrix is presented. Then, a method of random search of MDS matrices is given and based on specific examples to present an estimate for this possibility. In addition, we propose efficient random search algorithms for involutory Hadamard MDS matrices of size 4, 8. These results will be meaningful to determine the possibility that MDS matrices can be obtained from random matrices and diversify the methods of searching for MDS matrices. With our search approach, involutory Hadamard matrices can be found efficiently for implementation due to the presence of elements with small Hamming weights. In addition, the involutory Hadamard MDS matrices make the encryption and decryption processes identical, which makes them very efficient for both hardware and software implementation.

The paper is organized as follows. In Section II, preliminaries are introduced. Section III presents a lower bound for the possibility of satisfying the MDS condition of any random matrix and a method for random search of MDS matrices along with specific examples to present an estimate for this probability. Then is to search for efficient random Hadamard MDS matrices for executions of size 4, and 8. Section IV is the conclusion.

II. PRELIMINARIES AND RELATED WORKS

2.1. MDS matrix. The MDS matrices come from the theory of error correction codes with maximal distance separable codes or MDS codes. To better understand the MDS matrix, we can see the important theorems about the MDS matrix in the code theory as follows:

Theorem 1: ([5, p. 33]). If C is a $[n, k, d]$ code, $n-k \geq d-1$.

The code C is called an MDS code if $n - k = d - 1$ (or $d = n - k + 1$).

Theorem 2: ([5, page 321]). A code $[n, k, d]$ with a generator matrix $G[I|A]$ where A is a $k\times(n-k)$ matrix is MDS if and only if every square submatrix (generated from any i rows and any i columns, for any $i = 1, 2, ..., min\{k, n-k\}$ of A is nonsingular.

Thus, it can be seen that the MDS matrix is a very special matrix whose every square sub-matrix is nonsingular.

2.2. Hadamard MDS matrix. The authors in [2, 7, 10, 12] use Hadamard matrices to construct MDS matrices for their block cipher design.

Elumalai and Reddy in [12] defined the Hadamard matrix as follows:

Definition 1: With k elements $\alpha_0, \alpha_1, ..., \alpha_{k-1}$, a matrix $A = [a_{i,j}]$ is called a Hadamard matrix if each element in this matrix is constructed as: $a_{ij} = \alpha_{i \oplus j}$

The authors in [8] also defined a Hadamard matrix of size 2^n in a different way as follows:

A matrix H of size $2^n\times2^n$ is a Hadamard matrix over the finite field $GF(2^q)$ (FFHadamard) if it can be represented as follows: $H = \begin{bmatrix} U & V \\ V & U \end{bmatrix}$, and two submatrices U, V are matrices over $GF(2^p)$.

An example of a 4×4 FFHadamard matrix is as follow:

$$\mathbf{H} = had(a_0, a_1, a_2, a_3) = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_3 & a_0 & a_1 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} (*)$$

where $h_{ij} = a_{i \oplus j}$

Comment 1. *Each Hadamard matrix A over a finite field will have the following property:* $A^2 = \gamma \cdot I$ *where* γ *is a constant. When* $\gamma = 1$ *, then A is an involutory matrix.*

III. SOME RESULTS

In this section, some results will be presented on the method of finding MDS matrices from random matrices, in which a lower bound for the possibility of satisfying the MDS condition of any random matrix is given. Then, a method of random search of MDS matrices is presented and an estimate for this probability is given on specific examples.

3.1. MDS matrices of size 2

Let $A = [a_{i,j}]_{m\times m}$ be an any random matrix over $GF(p^r)$, where p is a prime number.

Then, it can be easily seen that the number of *sub-matrices* of size $i(1 \le i \le m)$ created by any i rows and i columns of the matrix A will be equal to $(C_m^i)^2$.

In the case of $p = 2$, $r = 1$ we see that there is only one MDS matrix of size 1, and there are no MDS matrices of size 2 or higher.

For the case $GF(p^r)$ and $r > 1$, we have the following lemma:

Lemma 1: The number of MDS matrices of size 2 over $GF(p^r)$ for $r > 1$, is equal to $P^3(P-1)$, where $P = p^r - 1$.

Proof: Indeed, the number of singular matrices of size 2 over $GF(p^r)$ (with all four elements being nonzero) is equal to P^3. Also, the number of possible matrices of size 2 with all four elements being non-zero over $GF(p^r)$ is equal to P^4. Therefore, the number of MDS matrices of size 2 is equal to $P^4 - P^3 = P^3(P-1)$. ■

3.2. The ability to satisfy the MDS condition of random matrices

In this section, an any random matrix is partitioned into disjoint square matrices of size 2, producing a series of Bernoulli's tests on these matrices. On this basis, the probability that an any random matrix not being MDS is estimated.

Let $A = [a_{i,j}]_{m\times n}$, $(m \ge 2)$ be an any random matrix over $GF(p^r)$.

The partition of the disjoint submatrices of size 2 of the matrix A **is as follows:**

- Consider pairs of consecutively disjointed rows and columns of the matrix A .

- Denote N is the number of disjoint sub-matrices of size 2 created by the above mentioned pairs of rows and columns of the matrix A. Then, $N = [\frac{m}{2}]^2$. These matrices are N completely random and independent matrices of size 2.

- According to the proof of the Lemma 1, the probability that each of these N matrices being MDS is equal to $\frac{P^3(P-1)}{P^4} = 1 - \frac{1}{P}$; the probability that each matrix not being MDS is equal to $\frac{1}{P}$. Note that, the larger P is, the higher the probability to choose the MDS matrix is.

Series of Bernoulli's tests for the N above matrices of size 2

Consider the following event: Ω =There are at least $i(1 \le i \le N)$ matrices in N above matrices not being MDS. Obviously, if Ω occurs, then the matrix A is not MDS.

Then, the probability of the event Ω is determined as follows:

$$P(\Omega) = \sum_{i=1}^{N} C_N^i \left(\frac{1}{p}\right)^i \left(1 - \frac{1}{p}\right)^{N-i} \tag{1}$$

On the other hand, it is to have:

$$\sum_{i=1}^{N} C_N^i \left(\frac{1}{p}\right)^i \left(1 - \frac{1}{p}\right)^{N-i} = 1 \tag{2}$$

Then:

$$P(\Omega) = 1 - \left(1 - \frac{1}{p}\right)^N \tag{3}$$

Comment 2. *The larger N is, the smaller* $(1 - \frac{1}{P})$ *is, that is, the larger* $P(\Omega)$ *is.*

Probability that the random matrix $A = [a_{i,j}]_{m\times m}$ **over** $GF(p^r)$ **is not MDS**

Denote the event $\overline{\Omega}$ = The matrix A is not MDS.

It is to have:

$$P(\overline{\Omega}) \ge P(\Omega) \tag{4}$$

Because, as suppose matrix A has N sub-matrices of order 2, but in addition to sub-matrices of order 2, A can also have sub-matrices of level 3, level 4, ..., of order m. So in addition to the submatrices of order 2 which may not be MDS, the higher order submatrices may not be MDS either. Therefore, we have formula (4).

Obviously, it is difficult to directly compute the value of $P(\overline{\Omega})$, so we can estimate the lower bound of this value by computing $P(\Omega)$.

Comment 3. *From the Comment 1 and the formulas (3), (4), it can be seen that, when N is larger or when the size of the matrix m is larger, the probability for the matrix A being not MDS is larger.*

These results are also significant for estimating the number of MDS matrices of any size m.

Example 1.

- For $m = 4$ and $GF(2^8)$ it is to have: $P = 255$ and $N = \frac{m^2}{4} = 4$. Then, $P(\Omega) = 1 - (1 - \frac{1}{P})^4 = 0,0156$, that is, the probability that the matrix A is not MDS $\geq 0,0156$.

- For $m = 8$ and $GF(2^8)$ it is to have: $P = 255$ and $N = \frac{m^2}{4} = 16$. Then, $P(\Omega) = 1 - (1 - \frac{1}{P})^4 = 0,0609$, that is, the probability that the matrix A is not MDS $\geq 0,0609$.

- For $m = 64$ and $GF(2^8)$ it is to have: $P = 255$ and $N = \frac{m^2}{4} = 1024$. Then, $P(\Omega) = \cong 0.984$.

Thus, the larger the matrix size is, the smaller the probability that the matrix being MDS is. In the case of $m \geq 64$, it is almost certain that a random matrix obtained will not be an MDS matrix.

3.3. Searching for MDS matrix from random matrices and probability

Consider $A = [a_{i,j}]_{m\times m}$ is a matrix with elements taken independently, randomly with uniform probability over $GF(p^r)$. The method of random search of MDS matrices is implemented through Algorithm 1 as follows:

Algorithm 1 $m \times m$ base

Input: An empty matrix A of size $m \times m$ over the finite field $GF(p^r)$.

Output: An MDS matrix A of size $m \times m$ over $GF(p^r)$.

Step 1: Construct a matrix **A** by randomly taking the elements of $GF(p^r)$ to fill in the cells of the matrix A.

Step 2: Check if this matrix is MDS:

+ If true, then save this matrix to a file.

+ If false, then go back to Step 1 to create another random matrix A

Experiment:

We experimented this method to find the MDS matrices over the field $GF(2^8)$, and obtained the following results:

- *Searching for MDS matrices of size 4 from random matrices of size 4: with 500 random matrices, 56 MDS matrices of size 4 are obtained.*

Assume the ratio that a random matrix of size 4 is not MDS is P_0.

With a sample size of $n = 500$, we have the ratio that a random matrix of size 4 on a particular sample is not an MDS is: $f_0 = \frac{444}{500} = 0.888$. For accuracy is $1 - \alpha = 95\% \rightarrow u_{1-\frac{\alpha}{2}} = u_{0,975} = 1.96$. Then, the precision of the confidence interval is:

$\varepsilon = u_{1-\frac{\alpha}{2}}\sqrt{\frac{f_0(1-f_0)}{n}} = 1.96\sqrt{\frac{0.888(1-0.888)}{500}} \cong 0.0276$. And, the confidence interval of p_0 is: $(f_0 - \varepsilon; f_0 + \varepsilon) = (0.8604; 0.9156)$

We see that the interval estimate for the true p_0 is larger than the lower bound value 0.0156 (in the example) theoretically calculated.

-Searching for MDS matrices of size 8 from random matrices of size 8: with 500 random matrices, 34 MDS matrices of size 8 are obtained.

Assume the ratio that a random matrix of size 8 is not MDS is p_1 .

With a sample size of , we have the ratio that a random matrix of size 8 on a particular sample is not an MDS is: $f_1 = \frac{466}{500} = 0.932$. For accuracy is $1 - \alpha = 95\%$. Then, $u_{1-\frac{\alpha}{2}} = u_{0.975} = 1.96$. Then, the precision of the confidence interval is:

$\varepsilon = u_{1-\frac{\alpha}{2}}\sqrt{\frac{f_0(1-f_0)}{n}} = 1.96\sqrt{\frac{0,932(1-0.932)}{500}} \cong 0.0221$

And, the confidence interval of p_1 is: $(f_1 - \varepsilon; f_1 + \varepsilon) = (0.9099; 0.9541)$.

It can be seen that, almost certainly, the resulting random matrices will not be MDS matrices.

Comment 4. These results help us to estimate the efficiency of the method of choosing MDS matrices from random matrices for cryptographic applications, and can also help determine the quantity of MDS matrices of any size m. In addition, these results show that the probability that a random matrix is MDS is very small, from which it can be seen that other methods of constructing MDS matrices are more efficient than this one. *However, when we apply the method of random search of MDS matrices with specific matrix types, this method becomes very useful. For example, with Hadamard matrices, the nature of these matrices is not necessarily satisfied by themselves that they are MDS matrices, so our random search for elements in the these matrices, then checking whether the obtained matrices satisfy the MDS condition can help us to find a lot of efficient Hadamard MDS matrices for execution, and have good use in cryptography.*

In the next section, we will present in detail the random search method for involutory Hadamard MDS matrices for execution.

3.4. Finding random 4×4 and 8×8 Hadamard MDS matrices efficient for execution

From Algorithm 1, we improve by not only finding a random MDS matrix A of any form, but we also randomly searching for MDS matrices of a particular form. For example, here we can fix the initial matrix A to be of Hadamard form. Hadamard matrices are capable of being self-inverting matrices, so it is very convenient to implement when the encryption and decryption processes at the linear layer using MDS matrices are exactly the same. Therefore, in random search, we will find elements such that the sum of the elements in a row of the Hadamard matrix is 1 so that the matrix is involutory.

If A is a Hadamard matrix of size m, then from the Definition 2, it can be seen that there are only m distinct

elements in the matrix A, and these m elements will be filled into matrix A in the Hadamard form in Definition 2. For example, these matrices have the form (*) for $m = 4$, and these matrices have the following form for $m = 8$.

$$\mathbf{H} = had(a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7) = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \\ a_1 & a_0 & a_3 & a_2 & a_5 & a_4 & a_7 & a_6 \\ a_2 & a_3 & a_0 & a_1 & a_6 & a_7 & a_4 & a_5 \\ a_3 & a_2 & a_1 & a_0 & a_7 & a_6 & a_5 & a_4 \\ a_4 & a_5 & a_6 & a_7 & a_0 & a_1 & a_2 & a_3 \\ a_5 & a_4 & a_7 & a_6 & a_1 & a_0 & a_3 & a_2 \\ a_6 & a_7 & a_4 & a_5 & a_3 & a_2 & a_0 & a_1 \\ a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \end{bmatrix} (**)$$

where the elements $a_i (0 \geq i \geq 7)$ are distinct.

To find the efficient Hadamard matrices, we restrict the elements $a_i \in [01_x, ..., 08_x]$.

We propose **Algorithm 2** to find efficient 4×4 involutory Hadamard MDS matrices, **Algorithm 3** to find efficient 8×8 involutory Hadamard MDS matrices for execution.

Algorithm 2 4×4 base

Input: An empty matrix A of size 4×4 over the finite field $GF(2^8)$; The set T includes a list of tuples of the form $(a_0, a_1, a_2, a_3), T = \emptyset$.

Output: An MDS matrix A of size 4×4 over $GF(2^8)$.

Step 1: Set up the matrix A in the form of a Hadamard matrix $(*)$ with 4 different symbolic elements $a_i \in GF(2^8), 0 \geq i \geq 3$.

Step 2: Select three distinct elements $a_i \in \{01_x, ..., 07_x\} (0 \geq i \geq 2)$. Calculate $a_3 = 1 - \sum_{i=0}^{2} a_i$.

Step 3: If a_3 matches any $a_i (0 \geq i \geq 2)$ element, return to Step 2. Otherwise:

+ If the tuple (a_0, a_1, a_2, a_3) does not belong to T, then fill these elements in the Hadamard matrix A; and add this tuple (a_0, a_1, a_2, a_3) to T.

+ Otherwise, go back to Step 2.

Step 4: Check if the Hadamard matrix obtained after Step 3 is MDS.

+ If true, then save this matrix to a file.

+ If false, then go back to Step 2.

Note that, with the construction as in Algorithm 2, Algorithm 3, the resulting Hadamard matrices from these algorithms will be involutory Hadamard MDS matrices.

Note that, as the Hadamard matrix is of size 4×4 in the form $(*)$, on the other hand, to ensure the sum of elements in a row of the Hadamard matrix A equals 1, in Step 2 of Algorithm 2, we only select 3 distinct elements, and the remaining element in the row of the Hadamard matrix is computed as: $a_3 = 1 - \sum_{i=0}^{2} a_i$. Similarly, we only select 7 distinct elements in Step 2 of **Algorithm 3**.

Algorithm 3 8×8 base

Input: An empty matrix A of size 8×8 over the finite field $GF(2^8)$; The set T includes a list of tuples of the form $(a_0, a_1, a_2, a_3, ..., a_6), T = \emptyset$.

Output: An MDS matrix A of size 8×8 over $GF(2^8)$.

Step 1: Set up the matrix A in the form of a Hadamard matrix $(**)$ with 8 different symbolic elements $a_i \in GF(2^8), 0 \geq i \geq 7$.

Step 2: Select three distinct elements $a_i \in \{01_x, ..., 08_x\} (0 \geq i \geq 6)$. Calculate $a_7 = 1 - \sum_{i=0}^{6} a_i$.

Step 3: If a_7 matches any $a_i (0 \geq i \geq 6)$ element, return to Step 2. Otherwise:

+ If the tuple $(a_0, a_1, a_2, a_3, ..., a_6)$ does not belong to T, then fill these elements in the Hadamard matrix A; and add this tuple $(a_0, a_1, a_2, a_3, ..., a_6)$ to T.

+ Otherwise, go back to Step 2.

Step 4: Check if the Hadamard matrix obtained after Step 3 is MDS.

+ If true, then save this matrix to a file.

+ If false, then go back to Step 2.

We choose the primitive polynomial for $GF(2^8)$ as: $x^8 + x^6 + x^5 + x^2 + 1$.

Since the elements $a_i (0 \leq i \leq m - 1)$ are distinct, for matrices of size 4, we try $7 \times 6 \times 5 = 210$ involutory Hadamard matrices and then obtain 16 MDS matrices. Thus, the ratio of 4×4 involutory Hadamard MDS matrices obtained from randomly generated involutory Hadamard matrices according to Algorithm 2 is $16/210 \approx 0.076$.

The proposed algorithms, including Algorithms 2 and 3, can generate involutory Hadamard MDS matrices. Therefore, the encryption and decryption processes using these matrices are performed identically, resulting in significant execution cost savings. In software, if the encryption process is represented as $y = A.x$, then the decryption process is $x = A.y$. Thus, the encryption and decryption processes are identical, allowing the use of the same lookup table for both encryption and decryption. In hardware, it is possible to design encryption and decryption circuits to be identical, resulting in cost savings in implementation.

In addition, we proceed to calculate the number of fixed points and the number of XORs of the obtained matrices. The results are shown in Table I.

Table I is a list of 16 involutory Hadamard MDS matrices of size 4 obtained. The resulting matrices have a small number of XORs that range from 192 to 200. The number of fixed points equal to 4 is quite small. Compared with AES's Mixcolumn matrix, this matrix has the number of fixed points equal to 2^{16} which is very large. Therefore, the proposed MDS matrices have a much smaller number

TABLE I
LIST OF 16 INVOLUTORY HADAMARD MDS MATRICES OF SIZE 4 WITH PRIMITIVE POLYNOMIAL $x^8 + x^6 + x^5 + x^2 + 1$

No	Involutory Hadamard MDS Matrix of size 4×4	Number of XORs	Number of fixed points
1	$\begin{bmatrix}2&4&6&1\\4&2&1&6\\6&1&2&4\\1&6&4&2\end{bmatrix}$	192	4
2	$\begin{bmatrix}2&5&7&1\\5&2&1&7\\7&1&2&5\\1&7&5&2\end{bmatrix}$	224	4
3	$\begin{bmatrix}3&4&7&1\\4&3&1&7\\7&1&3&4\\1&7&4&3\end{bmatrix}$	240	4
4	$\begin{bmatrix}1&2&4&6\\2&1&6&4\\4&6&1&2\\6&4&2&1\end{bmatrix}$	192	4
5	$\begin{bmatrix}1&2&5&7\\2&1&7&5\\5&7&1&2\\7&5&2&1\end{bmatrix}$	224	4
6	$\begin{bmatrix}1&2&6&4\\2&1&4&6\\6&4&1&2\\4&6&2&1\end{bmatrix}$	192	4
7	$\begin{bmatrix}1&2&7&5\\2&1&5&7\\7&5&1&2\\5&7&2&1\end{bmatrix}$	224	4
8	$\begin{bmatrix}1&3&4&7\\3&1&7&4\\4&7&1&3\\7&4&3&1\end{bmatrix}$	240	4
9	$\begin{bmatrix}1&3&5&6\\3&1&6&5\\5&6&1&3\\6&5&3&1\end{bmatrix}$	240	4
10	$\begin{bmatrix}1&3&6&5\\3&1&5&6\\6&5&1&3\\5&6&3&1\end{bmatrix}$	240	4
11	$\begin{bmatrix}1&3&7&4\\3&1&4&7\\7&4&1&3\\4&7&3&1\end{bmatrix}$	240	4
12	$\begin{bmatrix}1&4&6&2\\4&1&2&6\\6&2&1&4\\2&6&4&1\end{bmatrix}$	240	4
13	$\begin{bmatrix}1&4&7&3\\4&1&3&7\\7&3&1&4\\3&7&4&1\end{bmatrix}$	240	4
14	$\begin{bmatrix}1&5&6&3\\5&1&3&6\\6&3&1&5\\3&6&5&1\end{bmatrix}$	240	4
15	$\begin{bmatrix}1&5&7&2\\5&1&2&7\\7&2&1&5\\2&7&5&1\end{bmatrix}$	224	4
16	$\begin{bmatrix}3&5&6&1\\5&3&1&6\\6&1&3&5\\1&6&5&3\end{bmatrix}$	240	4

of fixed points, in favor of safety.

If we bring back the same method of evaluating the number of XORs that we perform, the AES matrix has a number of XORs of 152, Hierocrypt's MDS matrix is 448. Thus, the 4×4 involutory Hadamard MDS matrices we propose not only have a small number of fixed points, but also a smaller number of XORs than the AES and Hierocrypt MDS matrices.

With matrices of size 8, we try $8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 40.320$ involutory Hadamard MDS matrices and then we obtain only one MDS matrix. That's the matrix $Had(0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x13)$. This matrix has the number of fixed points is 1, the number of XORs is 1.272. This matrix has the smallest possible number of fixed points, so it is very good one, but the number of XORs is quite large, larger than the MDS matrix of the Kalyna block cipher when we perform the same evaluation method. Kalyna's is 952.

In general, the Hadamard MDS matrices we propose have a small number of fixed points, and the number of XOR operations is quite small. Their biggest advantage, on the other hand, is that they are involutory MDS matrices, so they're really useful for execution.

Compare with other MDS matrix generation methods.

In Table II, we present a comparison of our methods for generating MDS matrices with other approaches. With the proposed algorithms, we fixed the matrix format as Hadamard, and the coefficients in these matrices can be controlled. Specifically, in Algorithm 2 and Algorithm 3, we select the elements in the set , so we can create MDS matrices with low execution cost. Meanwhile, with matrices generated from MDS codes such as Reed-Solomon codes, and other methods we cannot control the coefficients of the matrices. The methods for generating MDS matrices from Cauchy [1, 3] and Vandermonde matrices [10] often produce matrices with elements having high Hamming weight and do not control the number of fixed points.

In comparison with the work [24], the authors of the current paper focus solely on the execution cost of involutory matrices, without considering their fixed points. In this study, we prioritize matrices with a small number of fixed points, and then select matrices with low execution costs. Meanwhile, works [25, 26] primarily concentrate on matrices of sizes 3×3 and 4×4 over F_{2^3} and F_{2^4}. The authors in [25] and [26] also did not calculate the fixed points of the obtained matrices.

On the other hand, in Algorithm 2 and Algorithm 3, we select elements from the set $\{01_x, ..., 08_x\}$ rather than exhaustively searching through all possible elements, making the algorithms run quite quickly. The time taken to search

TABLE II
COMPARISON OF OUR MDS MATRIX GENERATION METHODS WITH OTHER APPROACHES

No.	Our algorithms (Algorithm 2, Algorithm 3)	Method for generating MDS matrices from MDS code	Methods for generating MDS matrices from Cauchy matrices [1, 3], Vandermonde matrices [10]	Method for generating MDS matrices in [24]	Method for generating MDS matrices in [25]	Method for generating MDS matrices in [26]
Hamming weight of elements in the matrix	Small	Large	Large	Small	Small	Small
Matrix size	4x4 and 8x8	4x4, 8x8, 16x16	4x4, 8x8, 16x16	4x4 and 8x8	3x3	4x4
Checking the number of fixed points	Yes	No (number of fixed points may be large)	No (number of fixed points may be large)	No	No	No
Involutory property of the resulting matrix	Yes	No	No	Yes	Yes	Yes
Ability to control elements in the matrix	Yes	No	No	Yes	Yes	Yes

for 210 4×4 matrices and verify the MDS condition was 87.5 minutes. The time taken to search for 40,320 8×8 matrices and verify the MDS condition was 896 hours.

Furthermore, the most crucial objective is to find MDS matrices with a small number of fixed points and a low number of XOR operations, ensuring both security and minimizing execution costs. In this paper, we are not only concerned with matrices that have low execution cost, but we also need to select matrices with the small number of fixed point to ensure the security of the block cipher in the future.

IV. CONCLUSION

In this paper, the possibility of satisfying the MDS condition of any random matrices is presented, and the method of finding MDS matrices from random matrices is given. In addition, we propose algorithms to randomly search for involutory Hadarmard MDS matrices. These results will be meaningful to determine the possibility that MDS matrices can be obtained from random matrices and diversify the methods of searching for MDS matrices. The obtained involutory Hadamard MDS matrices are those in which both the encryption and decryption matrices are efficient for execution, so they are very significant in the implementation of today's cryptographic algorithms. On the other hand, they also have small number of fixed points, which is highly favorable for the security of the block cipher and cryptographic algorithms that use them in the future.

REFERENCES

[1] M. Elhosary, N. Hamdy, I.A. Farag, A.E Rohiem, "Optimum Dynamic Diffusion of Block Cipher Based on Maximum Distance Separable Matrices", *International Journal of Information & Network Security (IJINS)*, Vol.2, No.4, August 2013, pp. 327–332, ISSN: 2089–3299.

[2] A. M. Youssef, S. E. Tavares and H. M. Heys, "A New Class of Substitution Permutation Networks", *Workshop on Selected Areas in Cryptography*, SAC '96, Workshop Record, pp. 132–147, 1996.

[3] A. M. Youssef, S.Mister and S.E. Tavares, "On the Design of Linear Transformations for Substitution Permutation Encryption Networks", *in Workshop on Selected Areas in Cryptography* (SAC'97), pp. 40–48, 1997.

[4] A. R. Rao, P. Bhimasankaram, "*Linear Algebra*", Second Edition, Hindustan Book Agency.

[5] F. J. MacWilliams, N. J. A. Sloane, Bell Laboratories, Murray Hill, NJ 07974, U. S. A, "*The Theory of Error-Correcting Codes*", North-holland publishing company amsterdam, New York, Oxford, 1977.

[6] J. Daemen, L. Knudsen, and V. Rijmen, "The block cipher Square", *Fast Software Encryption* (FSE'97), LNCS 1267, pp. 149–165, Springer-Verlag, 1997.

[7] K. C. Gupta and I. G. Ray, "On Constructions of Involutory MDS Matrices", In *AFRICACRYPT* 2013, pp. 43–60, Springer 2013.

[8] K. C. Gupta, I. G. Ray, "On Constructions of MDS Matrices From Circulant-Like Matrices For Lightweight Cryptography", *Technical Report* No. ASU/2014/1, Dated : 14th February, 2014.

[9] K. C. Gupta, I. G. Ray, "On constructions of mds matrices from companion matrices for lightweight cryptography". In A. Cuzzocrea, C. Kittl, D. E. Simos, E. Weippl, L. Xu, A. Cuzzocrea, C. Kittl, D. E. Simos, E. Weippl, and L. Xu, editors, CD-ARES Workshops, volume 8128 of *Lecture Notes in Computer Science*, pp. 29–43. Springer, 2013.

[10] M. Sajadieh, M. Dakhilalian, H. Mala, B. Omoomi, "On construction of involutory MDS matrices from Vandermonde Matrices in GF(2q)", *Designs, Codes and Cryptography*, 64, 287–308.

[11] P. Junod and S. Vaudenay, "Perfect diffusion primitives for block cipher – Building efficient MDS matrices", *in Selected Areas in Cryptology (SAC 2004)*, LNCS 3357, pp. 84–99, Springer-Verlag, 2004.

[12] R. Elumalai, Dr. A. R. Reddy, "Improving Diffusion Power of AES Rijndael with 8×8 MDS Matrix", *International Journal of Scientific & Engineering Research*, Vol. 2, Issue 3, March-2011.

[13] R. Klima, N. Sigmon, E. Stitzinger, "Applications of abstract algebra with maple", *CRC Press*, Boca Raton London New York Washington, D.C, 1999.

[14] S. Lin and D. Costello, "*Error Control Coding: Fundamentals and Applications*", Prentice Hall, 2004.

[15] T. P. Berger, A.V. Ourivski, "Construction of new MDS codes from Gabidulin codes", *LACO, University of Limoges, France*, 2013.

[16] A. Venkateswarlu, A. Kesarwani and S. Sarkar, "On the Lower Bound of Cost of MDS Matrices", *IACR Transactions on Symmetric Cryptology*, pp. 266–290, 2022.

[17] X. Zhou and T. Cong, "Construction of generalized-involutory MDS matrices", *Cryptology ePrint Archive*, 2022.
[18] K. C. Gupta, S. K. Pandey and Venkateswarlu, "Almost involutory recursive MDS diffusion layers", *Design, Codes and Cryptography*, 87 (2018), 609–626.
[19] S. Kolay and D. Mukhopadhyay, "Lightweight diffusion layer from the kth root of the mds matrix", *IACR Cryptology ePrint Archive*, vol. 498, 2014.
[20] T. T Luong, "Constructing effectively mds and recursive mds matrices by reed-solomon codes", *Journal of Science and Technology on Information Security of Viet Nam Government Information Security Commission*, vol.3, no. 2, pp. 10–16, 2016.
[21] T. T. Luong, N. N. Cuong and B. D. Trinh, "4x4 Recursive MDS Matrices Effective for Implementation from Reed-Solomon Code over GF(q) Field", *International Conference on Modelling, Computation and Optimization in Information Systems and Management Sciences – MCO 2021*, pp 386–391, 2021.
[22] K. C. Gupta, S. K. Pandey and S. Samanta, "Construction of Recursive MDS Matrices Using DLS Matrices", *In Progress in Cryptology-AFRICACRYPT 2022: 13th International Conference on Cryptology in Africa*, AFRICACRYPT 2022, Springer Nature Switzerland, pp. 18–22, 2022.
[23] P. Freyre, N. Díaz, R. Díaz and C. Pérez, "Random generation of MDS matrices", *Proceedings of Current Trends in Cryptology CTCrypt2014*, Russia, 2014. DOI:10.13140/RG.2.1.5111.2160
[24] A. Kesarwani, S. Sarkar and A. Venkateswarlu, "Exhaustive search for various types of MDS matrices", *IACR Transactions on Symmetric Cryptology*, 2019, 231–256.
[25] M. Kurt Pehlivanoğlu, M. Ali Demir, F. Büyüksaraçoğlu Sakallı, S. Akleylek and M. Tolga Sakallı, "On the Construction Structures of Involutory MDS Matrices over ", *In Nonlinear Dynamics and Applications: Proceedings of the ICNDA*, Cham: Springer International Publishing, pp. 587–595, 2022.
[26] G. Tuncay, F. B. Sakallı, M. K. Pehlivanoğlu, G. G. Yılmazgüç, S. Akleylek and M. T. Sakallı, "A new hybrid method combining search and direct based construction ideas to generate all involutory maximum distance separable (MDS) matrices over binary field extensions", *PeerJ Computer Science*, 9, 2023, e1577.

Tran Thi Luong received a Bachelor's degree from Hanoi University of Science, Hanoi, Vietnam, in 2006; a Master's degree in cryptographic technique at the Academy of Cryptography Techniques, Hanoi, Vietnam, in 2012; Ph.D. degree in cryptographic technique at the Academy of Cryptography Techniques, Hanoi, Vietnam in 2019. Now, she is Vice Dean of Information Security Department of the Academy of Cryptography Techniques. Her research interests include Cryptography, Coding theory, and Information Security.
Email: luongtranhong@gmail.com

Data Power Optimization Algorithm for Downlink Multi-ARS Small Cell Communication Systems

Bui Anh Duc[1], Tran Manh Hoang[2], Nguyen Thu Phuong[1], Xuan Nam Tran[1], Pham Thanh Hiep[1]

[1] Advanced Wireless Communications Group, Le Quy Don Technical University, Hanoi, Vietnam.
[2] Telecommunication University, Nha Trang, Khanh Hoa, Viet Nam.

Correspondence: Pham Thanh Hiep, thanhhiep@lqdtu.edu.vn
Communication: received 6 Dec 2023, revised 16 Dec 2023, accepted 21 Jan 2024
DOI: 10.32913/mic-ict-research.v2024.n1.1248

***Abstract*: In this paper, we investigate a novel downlink Small Cell (SC) system based on the Cell Free (CF) model, with the deployment of Unmanned Aerial Vehicles (UAVs) as Aerial Relay Stations (ARSs) simultaneously serving users in a specific area. The new SC model concept is expressed in the user-centric idea that the user will choose only one ARS with the best conditions for communication, and the concept of "cell" or "cell boundary" will no longer exist. The Time Division Duplex (TDD) mechanism is used for system-wide communication protocol, and the Minimum Mean Squared Error (MMSE) technique is applied for downlink channel estimation. We establish the downlink user throughput expression and then propose a Bisection optimization algorithm to control power for downlink data transmission to improve system quality and save energy. We evaluate the downlink multi-ARS SC communication system with and without power optimization in different aspects, such as altering the number of users, the number of ARSs, the number of antennas, and the transmission channel estimation interval. The results also reveal that the system quality is significantly improved when applying the downlink power factor optimization problem.**

Keywords: *Small Cell, Multi-ARS, power optimization, bisection algorithm.*

I. Introduction

In recent years, the rapid proliferation of wireless devices has posed numerous challenges for the telecommunications industry. Therefore, the next generation must focus on solving more practical problems such as high speed in gigabits per second, low latency, high throughput, and high spectrum efficiency within a specific coverage area or application scenario. The infrastructure of the present day is easily capable of supporting advanced techniques and scenarios. Potential candidates include Small Cell (SC) systems that feature well-connected nodes and high-capacity backhaul links, as well as Unmanned Aerial Vehicles (UAVs) communications that function as aerial relay stations (ARSs) with flexible deployment [1, 2]. The challenge of integrating ARS communication and SC systems is inherently innovative and holds significant potential for reciprocal advancements in the communications technology industry.

To begin with, it is evident that ARSs communication is an exceptional field that has been the subject of numerous studies examining various aspects. Due to its advantageous characteristics, including rapid response time, high mobility, cost-effectiveness, flexible deployment, favorable channel conditions, and precise controllability, ARS-enabled wireless mobile communication finds extensive utility not only in everyday situations but also in military and rescue operations [3]. The authors also investigated various application scenarios for ARS communication, including single UAV relaying networks, multi-user UAV relaying networks, multi-hop UAV relaying networks, and the Internet of UAVs. In addition, the Amplify-and-Forward (AF) or Decode and-Forward (DF) relaying scheme, Time Division Multiple Access (TDMA), Orthogonal Frequency Division Multiple Access (OFDMA) protocols, and the problem of optimizing capacity and ARS trajectory in the ARS communication network are also introduced in detail [4].

Moreover, a conventional SC system is predicated on an extremely dense deployment of Base Stations (BSs), which are considerably smaller in size than a typical microcell. SC models are, in essence, BSs intended to serve a restricted coverage area that is approximately one hundred times smaller in size than conventional macro-BSs. By focusing their transmission at low power and a coverage radius of 50–150 meters, SC models enhance energy efficiency and minimize transmission loss through the reduction of the distance between users and BSs [2]. As a result, the

deployment of SC models has attracted the interest of mobile operators seeking to improve system quality in various homes or geographical areas.

The combination of ARS communication and SC models for user cluster coverage through beamforming and mm-wave techniques has also been extensively investigated. The results show that the system quality depends on the height of the ARS, the number of ARS, and the size of the user cluster [5]. To decrease the power consumption of SC systems, researchers developed a power optimization strategy. This framework utilizes a novel control algorithm that can autonomously deactivate the BSs of the redundant SC model in response to network traffic requirements [6, 7].

1. Motivation

With the above-mentioned basic survey of SC and ARS communication systems, as well as the current state and high demands of the telecommunications industry and future directions. As a result, there are some compelling reasons to combine the SC system and ARS communication.

Firstly, conventional SC systems provide benefits such as high-speed processing, monitoring, and flexible deployment with picocells, femtocells, and microcells. However, the nature of the conventional SC model still exists in the concept of cells with a small scope. In practice, more cells lead to more complex signal processing, increased inter-cell interference, and affect the radio connection. Therefore, a novel approach is required to enhance and develop the conventional SC model to guarantee the concept of "Cell" is removed.

Secondly, the survey reveals that limited studies have been conducted on the issue of integrating ARS communication and SC models. In particular, a completely new SC model is developed when considering no cells. Furthermore, when SC is combined with ARS communication, a combined model with outstanding characteristics such as high LoS probability, flexible deployment, and simple signal processing is created. These issues demonstrate that this combined model is entirely feasible for real-world scenarios such as rescue missions, military operations, and connecting IoT devices at common civilian events.

Finally, the simple power optimization problem is always prioritized to be applied in accordance with the models of the cellular networks to reduce the influence of interference and improve system quality. In particular, in the context of the emergence of ARS communication, it will save energy and prolong the life of the wireless network.

2. Contribution

Based on the aforementioned research motivations, we develop a new downlink SC structure-assisted ARS based on the Cell Free (CF) multi-ARS system model. Simultaneously, a simple Bisection optimization algorithm is proposed to address the problem of controlling data transmission power in order to improve system quality, deploy a flexible system, and save energy. The detailed structure of our research is shown in the following important contributions:

- We develop the downlink SC communication model on the CF system, so there is no concept of cell or cell boundary with all ARS deployed and serving the same users in a specific area [8]. The SC model is established according to the user-centric idea, in which each user chooses only one ARS with the best channel conditions for communication.
- We design the channel model according to advanced LTE standards provided by the International Telecommunication Union (ITU) and the 3rd Generation Partnership Project (3GPP) to support airborne vehicles. The Time Division Duplex (TDD) protocol is applied to the network-wide communication protocol. In addition, the Minimum Mean Squared Error (MMSE) channel estimation is applied to downlink channel estimation.
- We establish the expression for the downlink throughput per user. Next, we deploy the bisection algorithm to solve the linear problem under the power control factor constraint, thereby optimizing the power for the downlink communication system from the ARSs to the users.
- We evaluate the system quality by comparing the user throughput without and with data power coefficient optimization in different aspects, such as changing the number of users, number of ARSs, and channel estimated interval in the downlink.

The subsequent sections are organized as follows: Section II discusses the downlink multi-ARS SC system. Section III addresses signal processing, including downlink channel estimation, downlink data transmission, and power optimization. Section IV displays numerical findings, and Section V outlines our concluding observations. Table I is structured for convenient reference to the mathematical symbol notations.

II. Downlink Multi-ARS Small Cell System

1. Channel model

We investigate the model of a novel multi-ARS SC structure applied in various real-world scenarios as de-

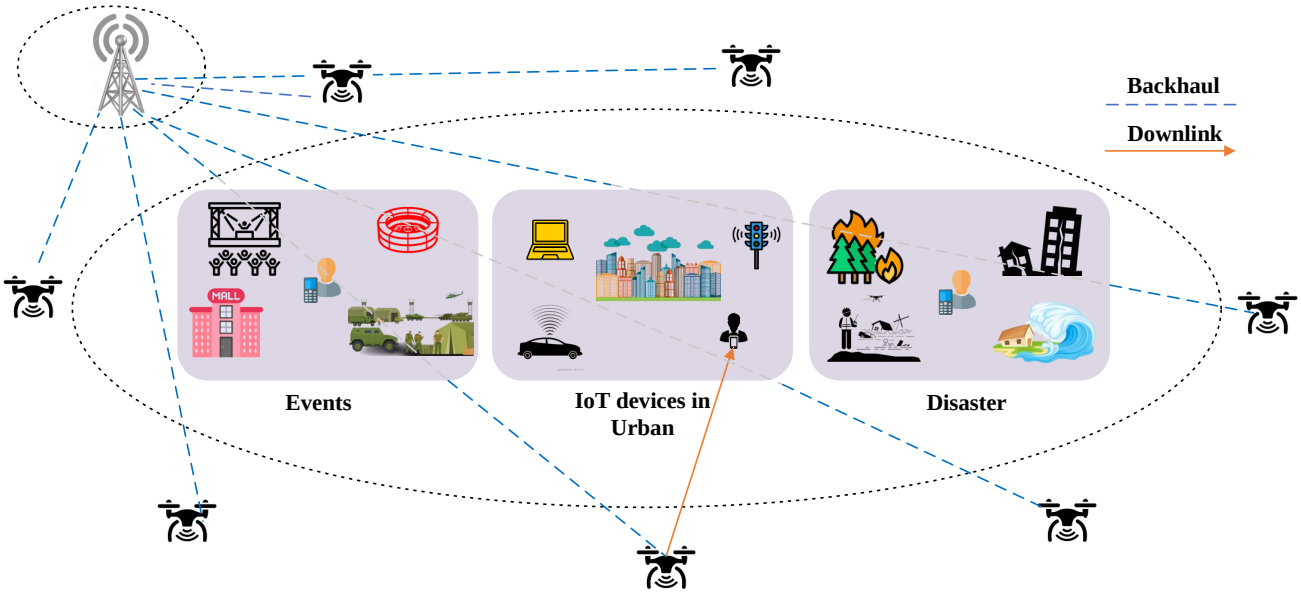


Figure 1. The downlink multi-ARS Small Cell system.

TABLE I
MATHEMATICAL SYMBOLS EMPLOYED IN THIS ARTICLE

Notation	Description
$(.)^*$	Conjugate
$(.)^H$	Conjugate-Transpose
$\mathbb{E}\{\cdot\}$	Expectation operator
$(.)^{-1}$	Inverse
$(.)^T$	Transpose
$\|.\|$	The Euclidean norm
$\mathbf{I}_N$	The $N \times N$ identity matrix
$\mathbb{C}^{m\times n}$	A vector or matrix with size of $m \times n$
τ_k^{d}	Downlink data power control coefficient
$\alpha_{\mathrm{p}}^{\mathrm{d}}$	Downlink channel estimation interval
K	Number of Users
N	Number of antennas at ARS
S	Number of ARSs

scribed in Figure 1. We first design a multi-ARS CF communication model with S ARS and K ground users [8]. ARSs are configured with N antennas, while users are equipped with a single antenna. Moreover, ARSs and users are randomly distributed in a specific area. We examine under the conditions of the multi-ARS CF model, meaning that all ARSs serve users at the same frequency and time resource. The ARSs connect to the ground base station (GBS) through a perfect backhaul channel.

In this paper, we illustrate the channel from sth ARS to kth user as follows:

$$\mathbf{g}_{sk} = \sqrt{\beta_{sk}}\mathbf{h}_{sk}, \tag{1}$$

where the small-scale fading is denoted by $\mathbf{h}_{sk}$ and is considered under the assumption that it follows an independent and identically distributed distribution (i.i.d.) $\mathcal{CN}(0, \mathbf{I}_N)$ random variable (RV) vectors. Moreover, the large-scale fading is illustrated below.

$$\beta_{sk} = 10^{\frac{\mathrm{PL}_{sk}}{10}}, \tag{2}$$

$\mathrm{PL}_{\mathrm{lk}}$ is the path loss provided by the advanced LTE standard of the ITU for the Urban Micro (UMi) [9, Table B-2] communication scenarios.

$$\mathrm{PL}_{sk} = \tag{3}$$
$$\begin{cases} \mathrm{PL}_{\mathrm{LoS}} = \max\{\mathrm{PL}', 30.9 + \\ \left(22.25 - 0.5\log_{10}(h_{ARS})\right)\log_{10}(d_{3\mathrm{D}}) + 20\log_{10}(f_c)\} \\ \mathrm{PL}_{\mathrm{NLoS}} = \max\{\mathrm{PL}_{\mathrm{LoS}}, 32.4 + \\ \left(43.2 - 7.6\log_{10}(h_{ARS})\right)\log_{10}(d_{3\mathrm{D}}) + 20\log_{10}(f_c)\}, \end{cases}$$

where PL', $\mathrm{PL}_{\mathrm{NLoS}}$, and $\mathrm{PL}_{\mathrm{LoS}}$ represent free space, Non-Line-of-Sight (NLoS), and Line-of-Sight (LoS) path loss, respectively.

2. Multi-ARS Small Cell model

The deployment of the SC model brings numerous significant benefits and has become one of the crucial solutions for the next generation of networks, as mentioned in Section I.1. We propose a multi-ARS SC downlink model to be established under the assumption that each user is served by a single ARS. It means that for each user, the ARS with the best signal contribution will be selected, and the ARS selection problem is based on the best large-scale

fading coefficient. The mathematical model is specified as follows:

$$s_k \triangleq \underset{s\in\{\in ARSs\}}{\arg\max}\ \beta_{sk}. \tag{4}$$

We take into consideration the requirement that the time interval be sufficiently short to avoid handover between ARSs [10]. The problem of handovers can be addressed in future research. Furthermore, unlike the CF model, the SC model does not have the constraint of the harden channel condition. This means that in the CF system, the channel gain is nearly equal to its mean value [11]. However, in the SC model, the channels are single fading Rayleigh channels. Therefore, both users and ARS must perform channel estimation. In this study, we investigate a communication system employing the TDD protocol [12] as shown in Figure 2, and focus on examining downlink channel estimation and downlink data transmission.

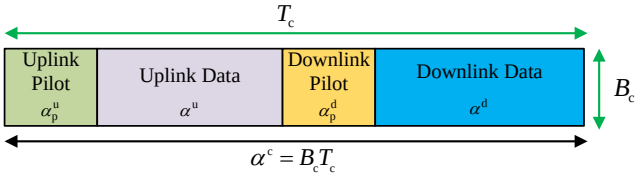


Figure 2. Time-division duplex (TDD) protocol.

III. Signal Processing

1. Downlink channel estimation

We define α_p^d as the period of time for downlink channel training and $\sqrt{\alpha_p^d}\boldsymbol{\phi}_k \in \mathbb{C}^{\alpha_p^d \times 1}$, $\|\boldsymbol{\phi}_k\|^2 = 1$ is the pilot sequence sent from the s_kth ARS, with ρ_p^d denotes the transmit power per the pilot symbol.

Subsequently, the received pilot signal at kth user show as follows:

$$\mathbf{y}_p = \sqrt{\alpha_p^d \rho_p^d}\mathbf{g}_{s_k k}\boldsymbol{\phi}_k^H + w_p, \tag{5}$$

where w_p is additive noise at user. Next, the projecting of $\mathbf{y}_p$ to $\boldsymbol{\phi}_k$, the MMSE estimation for $\mathbf{g}_{s_k k}$ is

$$\begin{aligned}\hat{\mathbf{g}}_{s_k k} &= \mathbb{E}\left\{\mathbf{g}_{s_k k}\breve{\mathbf{y}}_p^H\right\}\left(\mathbb{E}\left\{\breve{\mathbf{y}}_p\breve{\mathbf{y}}_p^H\right\}\right)^{-1}\breve{\mathbf{y}}_p \\ &= c_{s_k k}\breve{\mathbf{y}}_p\end{aligned} \tag{6}$$

where $\breve{\mathbf{y}}_p \triangleq \mathbf{y}_p\boldsymbol{\phi}_k$, and

$$c_{s_k k} \triangleq \frac{N\sqrt{\alpha_p^d \rho_p^d}\beta_{s_k k}}{N^2\alpha_p^d\rho_p^d\sum\limits_{k'=1}^{K}\beta_{s_{k'}k}\left|\boldsymbol{\phi}_k^H\boldsymbol{\phi}_{k'}\right|^2 + 1}. \tag{7}$$

According to [13], the channel estimation from s_kth ARS to kth user is represented as follows:

$$\hat{\mathbf{g}}_{s_k k} = \mathbf{g}_{s_k k} - \boldsymbol{\varepsilon}_{s_k k}, \tag{8}$$

where $\varepsilon_{s_k k}$ is error of channel estimation and is independent of $\hat{\mathbf{g}}_{s_k k}$. Moreover, we have $\hat{\mathbf{g}}_{s_k k} \sim \mathcal{CN}\left(0, \mu_{s_k k}\right)$, and $\varepsilon_{s_k k} \sim \mathcal{CN}\left(0, \beta_{s_k k} - \mu_{s_k k}\right)$ [13] with $\mu_{s_k k}$ is mean-square of the estimated channel vector $\hat{\mathbf{g}}_{s_k k}$, can be computed as

$$\begin{aligned}\mu_{s_k k} &\triangleq \mathbb{E}\left\{\left|\hat{\mathbf{g}}_{s_k k}\right|^2\right\} = N\sqrt{\alpha_p^d\rho_p^d}\beta_{s_k k}c_{s_k k} \\ &= \frac{N^2\alpha_p^d\rho_p^d\beta_{s_k k}^2}{N^2\alpha_p^d\rho_p^d\sum\limits_{k'=1}^{K}\beta_{s_{k'}k}\left|\boldsymbol{\phi}_k^H\boldsymbol{\phi}_{k'}\right|^2 + 1}.\end{aligned} \tag{9}$$

2. Downlink data transmission

K users rely on the pilot sequence received from ARSs to select the best ARS for data transmission. Denote $\sqrt{\tau_k^d}q_k$, where $\mathbb{E}\left\{|q_k|^2\right\} = 1$ is the symbol sent from the s_kth ARS to the kth user with the power control factor of τ_k^d. The signal at kth user is described below:

$$\begin{aligned}\mathbf{y}_k &= \sum_{n=1}^{N}\sqrt{\rho^d}\sum_{k'=1}^{K}\mathbf{g}_{s_{k'}k}\sqrt{\tau_{k'}^d}q_{k'} + w_k \\ &= \sum_{n=1}^{N}\sqrt{\rho^d}\hat{\mathbf{g}}_{s_k k}\sqrt{\tau_k^d}q_k + \sum_{n=1}^{N}\sqrt{\rho^d}\boldsymbol{\varepsilon}_{s_k k}\sqrt{\tau_k^d}q_k \\ &+ \sum_{N=1}^{N}\sqrt{\rho^d}\sum_{k'\neq k}^{K}\mathbf{g}_{s_{k'}k}\sqrt{\tau_{k'}^d}q_{k'} + w_k,\end{aligned} \tag{10}$$

where ρ^d represent the normalized downlink transmit SNR, and $w_k \sim \mathcal{CN}(0,1)$ is additive Gaussian noise. By considering the last three components of Eq. (10) as noise components, we obtain the Eq. (11) for the downlink data transmission rate of kth user, as shown on the next page.

It can be observed that in Eq. (11), $\left|\hat{\mathbf{g}}_{s_k k}\right|^2$ follows an exponential distribution with mean $\mu_{s_k k}$. Therefore, the expression for the achievable rate of kth user in Eq. (11) can be transformed by using the exponential integral function Ei(.)[14, Eq. (8.211.1)] as follow:

$$R_k^d = -\left(\log_2 e\right)e^{1/\bar{\mu}_{s_k k}}\ \mathrm{Ei}\left(-\frac{1}{\bar{\mu}_{s_k k}}\right), \tag{12}$$

where

$$\bar{\mu}_{s_k k} \triangleq \frac{\rho^d\tau_k^d\mu_{s_k k}}{\rho^d\tau_k^d\left(\beta_{s_k k} - \mu_{s_k k}\right) + N\rho^d\sum\limits_{k'\neq k}^{K}\tau_{k'}^d\beta_{s_{k'}k} + 1}. \tag{13}$$

3. Power optimization

The power control problem brings numerous benefits aimed at enhancing system quality, managing interference,

$$R_k^{\mathrm{d}} = \mathbb{E}\left\{\log_2\left(1 + \frac{\rho^{\mathrm{d}}\tau_k^{\mathrm{d}}\left|\hat{\mathbf{g}}_{s_k k}\right|^2}{\rho^{\mathrm{d}}\tau_k^{\mathrm{d}}\left(\beta_{s_k k} - \mu_{s_k k}\right) + N\rho^{\mathrm{d}}\sum\limits_{k'\neq k}^{K}\tau_{k'}^{\mathrm{d}}\beta_{s_{k'}k} + 1}\right)\right\}. \quad (11)$$

ensuring service quality synchronization, and conserving energy for the entire system. Some power optimization methods contribute to the very high overall throughput of the system. However, within them, there are some users with very good throughput, but there are also some users with very poor throughput. Therefore, this issue leads to unfairness, or, in other words, uneven service quality for users.

The max-min power optimization problem is intended to achieve a certain fairness among users. Fairness can be attained by increasing the data transmission power of users with low throughput. In our proposed model, power optimization is addressed through a max-min problem using the Bisection algorithm, where both the objective functions and constraints are semi-linear and linear. The mathematical model is specifically described as follows:

$$(\mathbf{P1}) \quad \max_{\tau_k^{\mathrm{d}}} \ \min_{k=1,\ldots,K} R_k^{\mathrm{d}} \quad (14a)$$

$$\text{subject to } 0 \le \tau_k^{\mathrm{d}} \le 1, \ k = 1, \ldots, K. \quad (14b)$$

Because R_k^{d} is the monotonically increasing function of $\bar{\mu}_{s_k k}$. Hence, (P1) problem is reformulated as follows:

$$(\mathbf{P2}) \quad \max_{\tau_k^{\mathrm{d}}} \ \min_{k=1,\ldots,K} \bar{\mu}_{s_k k} \quad (15a)$$

$$\text{subject to } 0 \le \tau_k^{\mathrm{d}} \le 1, \ k = 1, \ldots, K. \quad (15b)$$

To address the optimization problem, the objective functions and constraints must be linear or quasi-linear functions. (P2) is a maximization problem of $\min_{k=1,\ldots,K}\bar{\mu}_{s_k k}$ with regard to τ_k^{d}. It is observed that in P2, (15b) is a linear function, whereas (15a) does not guarantee linear. By introducing the slack variable ξ, we transform (15a) expression with ξ as the upper bound of $\min_{k=1,\ldots,K}\bar{\mu}_{s_k k}$ [15, Chapter 5, 6]. Thus, (P2) has an additional constraint is $\xi \le \bar{\mu}_{s_k k}$.

Specifically, P2 is reformulated as follows:

$$(\mathbf{P3}) \quad \max_{\tau_k^{\mathrm{d}}, \xi} \xi \quad (16a)$$

$$\text{subject to } \xi \le \bar{\mu}_{s_k k}, \ k = 1, \ldots, K \quad (16b)$$

$$0 \le \tau_k^{\mathrm{d}} \le 1, \ k = 1, \ldots, K. \quad (16c)$$

The (P3) problem is a linear problem with respect to ξ that can be successfully solved via the Bisection method as follows:

Algorithm 1 Bisection Algorithm

1: **Initialization**: Select the initial values of ξ_{min} and ξ_{max} define a range of relevant values of the objective function in (P3). Denote a tolerance value $\epsilon = 10^{-2}$
2: **Set:** $\xi := \frac{\xi_{min}+\xi_{max}}{2}$. Solve the(P3) quasi-linear problem
3: **If** (P3) problem is feasible, set $\xi_{min} := \xi$, else set $\xi_{max} := \xi$
4: **Stop** if $\xi_{max} - \xi_{min} \le \epsilon$ Otherwise, return to Step 2.

Remark 1: In optimization problems, it is often challenging to find an exact absolute solution, as in problems with explicit formulas. Specifically, in the bisection algorithm, the exact solution is a point within the search interval. Therefore, as the search intervals become smaller, the search solution gets closer to reaching the optimal point. The accuracy of the solution in the optimization problem depends on the tolerance value. The smaller the tolerance value, the approximate solution gets closer to the optimal solution. Typically, the tolerance value is set between 10^{-2} to 10^{-5}.

IV. Numerical results and discussions

We evaluate the system performance of multi-ARS based on the SC model using the Cumulative Distribution Function (CDF) of user throughput both without and with optimized downlink data power. Additionally, we also experiment with the system quality when varying parameters such as the number of ARS, the number of antennas, the time allocated for downlink channel estimation, and the large number of users.

1. Parameters and Setup

We exame with throughput per user, taking into account estimated channel interval. The throughput expression is expressed as follows:

$$T_k^{\mathrm{d}} = B\frac{1 - \alpha_{\mathrm{p}}^{\mathrm{d}}/\alpha^{\mathrm{c}}}{2}R_k^{\mathrm{d}}. \quad (17)$$

Without data power optimization, the ARSs transmit at full power, i.e., $\tau_k^{\mathrm{d}} = 1$

With data power optimization, the ARSs implement transmission with the optimized power control coefficients (τ_k^{d}) as described in Section III.3.

TABLE II
PARAMETERS OF THE SYSTEM USED FOR SIMULATION.

Parameter	Value
Carrier frequency	1.9 GHz
Bandwidth (B)	20MHz
Coherence bandwidth (B_c)	200 KHz
Coherence time (T_c)	1 ms
ARS height	$22.5m \leq h_{ARS} \leq 300m$
ρ_p^d, ρ^d	100, 200 mW
Coherence interval α^c	200 samples
The distributed area (S)	1 km^2

2. Results and Discussions

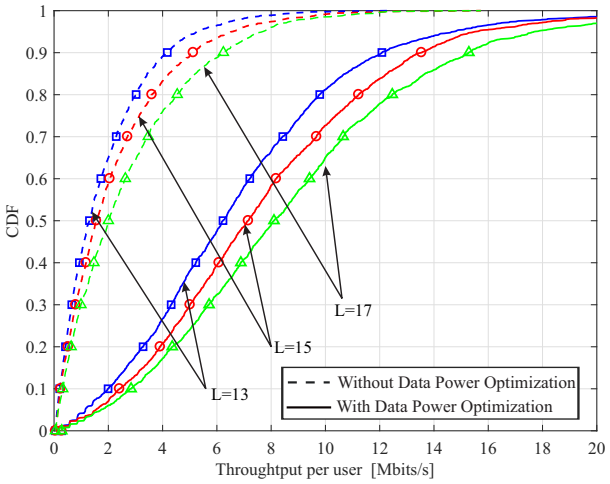


Figure 3. CDFs of the per-user throughput without and with data transmission power optimization, $L = 13, 15, 17$, $K = 10$, $N = 1$, and $\alpha_p^d = 10$.

Figure 3 illustrates CDF of user throughput as the number of deployed ARSs varies. On the one hand, it can be observed that the power optimization problem significantly improves throughput efficiency. For example, with a total of deployed ARSs ($L = 13$), the 90%-likely user throughput without power optimization is only around $0.1Mbits/s$, whereas with power optimization, it is $2Mbits/s$. The reason for this is that when the transmit power is optimized, the ARSs transmit with just enough power to manage interference or noise at the receiver, thereby improving user throughput.

On the other hand, as the total number of deployed ARSs increases sequentially with $L = 13, 15, 17$, user throughput also improves. As the L value increases, the probability of selecting an ARS with the best channel condition, corresponding to the SC model, also increases. This ensures an improvement in user service quality.

In Figure 4, we examine user throughput as the number

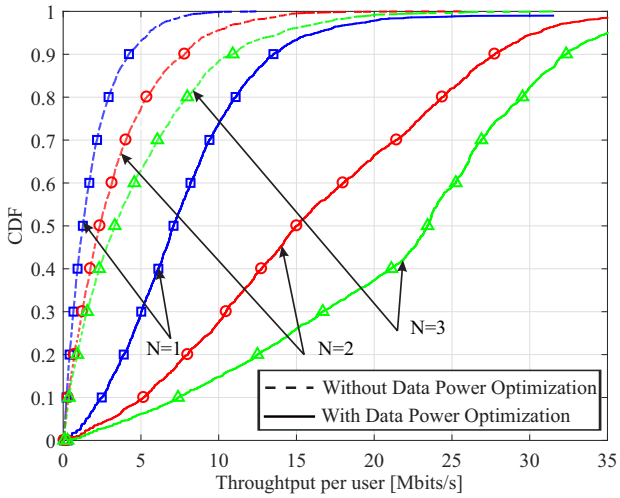


Figure 4. CDFs of the per-user throughput without and with data transmission power optimization, $L = 15$, $K = 10$, $N = 1, 2, 3$, $\alpha_p^d = 10$.

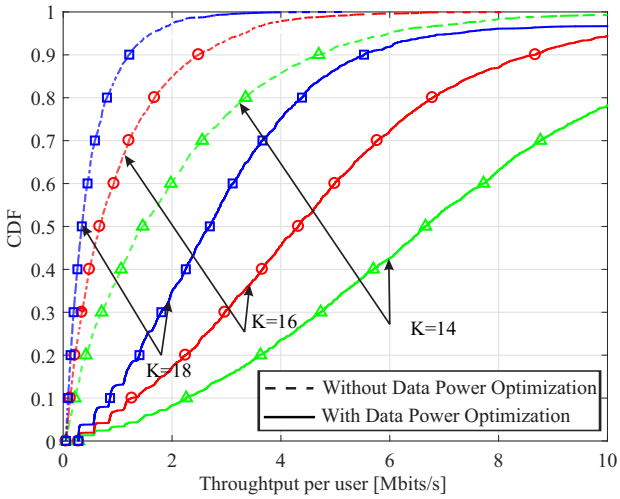


Figure 5. CDFs of the per-user throughput without and with data transmission power optimization, $L = 15$, N-$ARS = 15$, $K = 14, 16, 18$, $N = 1$, and $\alpha_p^d = 10$.

of antennas on ARS varies. User throughput increases significantly as the number of antennas per ARS increases ($N = 1, 2, 3$). In practice, it can be observed that as the number of antennas increases, the diversity gain also increases, leading to improved user service capabilities.

Figure 5 shows the CDF of user throughput as the number of users varies. In this simulation result, we investigate the system performance under real-world conditions with a varying number of users both larger and smaller than the deployed number of ARSs. The results indicate that with a high number of users, the user connections to ARSs require more competition, especially for users at a distance,

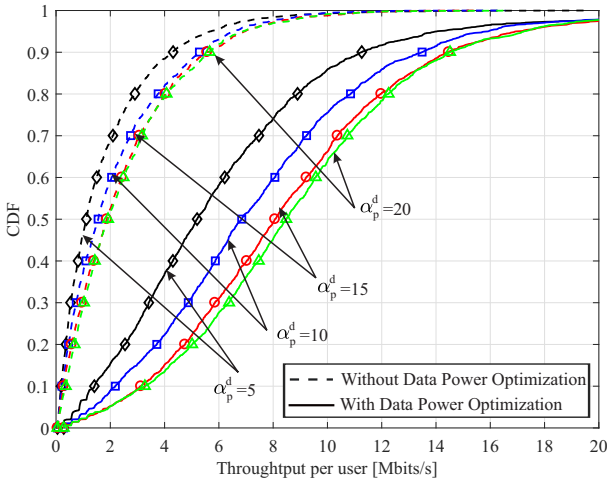


Figure 6. CDFs of the per-user throughput without and with data transmission power optimization, $L = 15$, $K = 10$, $N = 1$, $\alpha_p^d = 5, 10, 15, 20$.

leading to reduced user throughput. However, even with a high number of users, the power optimization problem still yields significant effectiveness.

In Figure 6, we investigate user throughput as the length of the downlink channel training pilot sequence varies, or, in other words, the time allocated for downlink channel training. It can be observed that as the time allocated for channel estimation increases ($\alpha_p^d = 5, 10, 15, 20$), user throughput improves.

However, with $\alpha_p^d = 20$, user throughput slightly increases and tends to saturate. The reason for this is that as the α_p^d value increases, indicating an increase in the time allocated for channel estimation, the quality of channel estimation improves. However, if the α_p^d value continues to increase, it will impact the data transmission time. Therefore, as the α_p^d value increases, user throughput tends to saturate and decrease.

V. Conclusion

This study has investigated a multi-ARS downlink system based on the SC model. By deploying ARSs to jointly serve ground users in a specific area, similar to the CF model. Then, the selection of an ARS with the best channel condition to serve users is referred to as the SC model. In addition, a power control algorithm is proposed to enhance the system quality. The system quality is evaluated based on various parameters such as the total number of deployed ARSs, the number of users, the number of antennas, and the length of the pilot sequence. The results also reveal that user throughput improves when applying the power control problem. However, at present, we are examining the connection between ARS and GBS as an ideal backhaul channel. In future work, we will study other optimization algorithms and apply deep reinforcement learning to the multi-ARS SC model to improve system performance, reduce complexity and signal processing time.

Acknowledgement

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.02-2021.56.

References

[1] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, D. Niyato, O. Dobre, and H. V. Poor, "6G Internet of Things: A comprehensive survey," *IEEE Int. J.*, vol. 9, no. 1, pp. 359–383, Aug. 2022.

[2] J. Tanveer, A. Haider, R. Ali, and A. Kim, "An overview of reinforcement learning algorithms for handover management in 5G ultra-dense small cell networks," *Applied Sciences*, vol. 12, no. 1, p. 426, 2022.

[3] B. Alzahrani, O. S. Oubbati, A. Barnawi, M. Atiquzzaman, and D. Alghazzawi, "UAV assistance paradigm: State-of-the-art in applications and challenges," *Journal of Network and Computer Applications*, vol. 166, p. 102706, 2020.

[4] B. Li, S. Zhao, R. Miao, and R. Zhang, "A survey on unmanned aerial vehicle relaying networks," *IET Communications*, vol. 15, no. 10, pp. 1262–1272, 2021.

[5] M. A. Ouamri, D. Singh, M. A. Muthanna, A. Bounceur, and X. Li, "Performance analysis of UAV multiple antenna-assisted small cell network with clustered users," *Wireless Networks*, vol. 29, no. 4, pp. 1859–1872, 2023.

[6] Q. Alsafasfeh, O. A. Saraereh, A. Ali, L. Al-Tarawneh, I. Khan, and A. Silva, "Efficient power control framework for small-cell heterogeneous networks," *Sensors*, vol. 20, no. 5, p. 1467, 2020.

[7] N. Parvaresh and B. Kantarci, "A continuous actor–critic deep Q-learning-enabled deployment of UAV base stations: Toward 6G small cells in the skies of smart cities," *IEEE Open Journal of the Communications Society*, vol. 4, pp. 700–712, 2023.

[8] B. A. Duc, T. M. Hoang, N. T. Phuong, X. N. Tran, and P. T. Hiep, "Optimizing power for data transmissions in uplink cell-free multi-ABSs communication systems," in *2022 Int. Conf. Advanced. Technol. Commun. (ATC)*. IEEE, Nov. 2022, pp. 23–28.

[9] 3GPP, "Technical specification group radio access network; study on enhanced LTE support for aerial vehicles," *TR 36.777, Dec 2017*.

[10] Z. Liu and L. Dai, "A comparative study of downlink MIMO cellular networks with co-located and distributed base-station antennas," *IEEE Transactions on Wireless Communications*, vol. 13, no. 11, pp. 6259–6274, 2014.

[11] S. Elhoushy, M. Ibrahim, and W. Hamouda, "Cell-free massive MIMO: A survey," *IEEE Commun. Surveys & Tutorials.*, vol. 24, no. 1, pp. 492–523, Oct. 2022.

[12] T. L. Marzetta and H. Yang, *Fundamentals of massive MIMO*. Cambridge University Press, 2016.

[13] S. M. Kay, *Fundamentals of statistical signal processing: estimation theory*. Prentice-Hall, Inc., 1993.

[14] D. Zwillinger and A. Jeffrey, *Table of integrals, series, and products*. Elsevier, 2007.

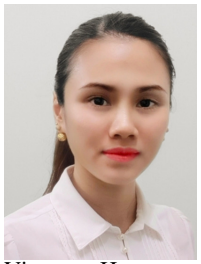
[15] E. K. Chong, W.-S. Lu, and S. H. Żak, *An introduction to optimization*. John Wiley & Sons, 2023.

Bui Anh Duc received a B.S. degree in Communication Command at Telecommunications University, Ministry of Defense, Vietnam, in 2011, and an MS degree from Le Quy Don Technical University, Vietnam in 2017. He is currently pursuing a Ph.D degree at Le Quy Don Technical University, Hanoi, Vietnam. His research interests include unmanned aerial vehicle communications, MIMO and Cell Free Massive MIMO systems, and signal processing for wireless cooperative communications.

E-mail: buiducsqtt@lqdtu.edu.vn

Tran Manh Hoang received the B.S. degree in communication command from Telecommunications University, Ministry of Defense, Nha Trang, Vietnam, in 2002, the B.Eng. degree in electrical engineering from Le Quy Don Technical University, Ha Noi, Vietnam, in 2006 the M.Eng. degree in electronics engineering from Posts and Telecommunications Institute of Technology, Ho Chi Minh City, Vietnam, in 2013 and the Ph.D degree from Le Quy Don Technical University, Hanoi, Vietnam, in 2018. He is currently working as a Lecturer with Telecommunications University Khanh Hoa, Vietnam. His research interests include energy harvesting, UAV, short packet communication, non-orthogonal multiple access, and MIMO, RIS, signal processing for wireless cooperative communications.

E-mail: tranmanhhoang@tcu.edu.vn

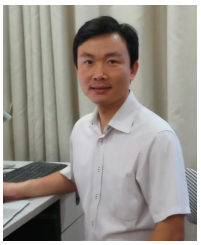


Nguyen Thu Phuong received the B.S, M.S and PhD degrees from Le Quy Don Technical University, Vietnam in 2008, 2012 and 2016, respectively. She is now a lecturer at Faculty of Radio-Electronics Engineering, and a significant member of the Advanced Wireless Communication Group, Le Quy Don Technical University, Hanoi, Vietnam. Her research interests are in the area of emerging technologies for future wireless communications: Energy harvesting, Non-orthogonal multiple access (NOMA), space-time processing, space-time coding, spatial modulation and index modulation, and massive MIMO systems.

E-mail: phuong.nt@mta.edu.vn

Xuan Nam Tran received his Master of Engineering (ME) in Telecommunications Engineering from University of Technology Sydney, Australia in 1998, and Doctor of Engineering in Electronic Engineering from The University of Electro-Communications, Japan in 2003. He is currently a full professor and head of a strong research group on advanced wireless communications, Le Quy Don Technical University. Prof. Tran is the founding chair and currently the chapter chair of the Vietnam Chapter of IEEE Communications Society. He is a member of IEEE, IEICE and the Radio-Electronics Association of Vietnam (REV). His research interests are in the area of emerging technologies for future wireless communications: Energy harvesting, Non-orthogonal multiple access (NOMA), space-time processing, space-time coding, spatial modulation and index modulation, and massive MIMO systems, signal processing for wireless cooperative communications.

E-mail: namtx@mta.edu.vn



Pham Thanh Hiep received a B.E. degree in Communications Engineering from National Defence Academy, Japan, in 2005; received the M.E. and Ph.D. degrees in Physics, Electrical, and Computer Engineering from Yokohama National University, Japan, in 2009 and 2012, respectively. Now, he is a lecturer at Le Quy Don Technical University, Ha Noi, Viet Nam. His research interests lie in the area of wireless communication technologies and signal processing.

E-mail: thanhhiep@lqdtu.edu.vn