Anomaly Detection Algorithm for Network Device Configuration based on Configuration Statement Tree

Yuan-Cheng Shen[1], Rui Ban[2], Xin Chen[1], Run-Duo Hua[2] and Yun-Hai Wang[1,*]

[1]*School of Computer Science and Technology, Shandong University, Qingdao 266200, China*

[2]*China information technology designing & consulting institute, Beijing 100000, China*

E-mail: remoteshen@gmail.com; banrui@dimpt.com; huard@dimpt.cpm; 1341040663@qq.com; CloudSeaWang@gmail.com

Received Feb 19, 2023; accepted Mar 7, 2023.

Abstract The problem of device configuration anomalies is becoming increasingly significant with the development of network communication equipment. Traditional detection tools usually only detect spelling, formatting, and other issues and cannot identify logic problems. Consequently, engineers' experience plays a critical role in detecting such anomalies. To improve network service quality, reduce repetitive work, and address issues like slow detection speed, weak detection capabilities, and poor versatility of traditional tools, this paper draws on the design concept of abstract syntax trees and proposes an innovative unsupervised anomaly detection algorithm based on "configuration statement trees." The algorithm can identify seven types of detectable anomalies and provides recommendations for anomaly localization and modification plans. The paper evaluates and compares the algorithm based on indicators such as detectable types, runtime, accuracy, and recall using configurations from the operator's current network operation. The results demonstrate that the algorithm has good robustness and can effectively address network communication issues resulting from configuration anomalies in network communication equipment.

Keywords anomaly detection, cluster analysis, automatic inspection of equipment, abstract syntax tree, co-occurrence corpus analysis, unsupervised learning, association analysis

Regular Paper

The work is supported by the Human Computer Interaction Theory and Methods for Visual Analysis of Big Data in Ubiquitous Computing Environment No. 62141217.

[*]Corresponding Author

1 Introduction

In recent years, the growth of 5G technologies and the surge in end-user numbers have expanded communication networks and the operational scope of network operators, attracting numerous manufacturers to the field, providing network equipments and softwares[1]. Operators often partner with different manufacturers to balance service quality and costs. These Original Equipment Manufacturers (OEMs) offer specialized hardware and softwares, each with distinct protocols and specifications, resulting in complex network structures. Concurrently, real-time processing technologies like auto driving are reshaping communication network efficiency[2, 3]. As user demands grow and network structures evolve, maintaining network infrastructure faces unprecedented challenges[4].

Configuration anomalies are on the rise, causing network communication issues. Operators team up with OEMs for comprehensive solutions, including hardware, software, and tools. This approach heavily relies on skilled personnel to manage anomalies. Commonly, providers offer rule-based tools like "PTN Automated Network Inspection Tools" [5]. While effective for simpler issues, these tools struggle with complex anomalies like logic errors. This limits their utility in precise configuration anomaly detection, especially as network equipment evolves.

To address limitations in rule-based methods and tool inflexibility, the concept of "intelligent operation and maintenance" has emerged [6, 7]. Led by experts, this idea seeks a unified detection model adaptable to various configurations. By leveraging big data and AI, it aims to screen network device configurations comprehensively [8, 9]. The goal is to relieve engineers from anomaly tasks, focusing on quality configurations. Liu et al.'s proposal uses AI association analysis for anomaly detection [10]. However, real-world anomalies, often structural, challenge manual detection. AI models trained using association rules might miss these anomalies.

To address the outlined challenges, this study proposes a comprehensive strategy. Firstly, by leveraging insights from operators' profiles within the existing network, an innovative method is introduced that combines big data analysis, frequency analysis, and co-occurring corpus analysis [11]. This harmonious integration extracts operational profiles from the existing network [12], enabling the identification of potential anomalies in prevalent configuration files. Importantly, this approach mitigates subjectivity and minimizes the impact of varying technical expertise.

Simultaneously, the paper employs the abstract syntax tree design principle to introduce an innovative approach [13]. Specifically, a unique methodology transforms each configuration file into a structured statement tree, named the "configuration statement tree". Using features from these trees and the unsupervised K-Means clustering algorithm, anomalous configuration files are efficiently detected. Additionally, the study ingeniously uses the abstract syntax tree design concept again [14], establishing a novel mechanism for converting configuration files. This approach culminates in an effective anomaly detection framework, identifying anomaly locations, classifying types, and providing corrective actions. Notably, this configuration anomaly detection strategy adeptly identifies anomalies across different manufacturers and versions, with an automated process reducing engineers' workload and enhancing network service quality.

2 Related work

Configuration anomaly detection is a prominent research area in anomaly detection. Proposed methods fall into rule-based, statistics-based, and artificial intelligence-based categories. Rule-based methods use predefined rules for detection, statistics-based methods rely on frequency analysis, and AI-based methods mine anomalies through association rules between configuration statements.

2.1 Anomaly detection based on rule-based judgment

Rule-based anomaly detection methods offer precision and potential for perfect detection, achieving 100% accuracy for predefined anomalies. However, they require expert-crafted rules, limiting adaptability. Existing tools like ZTE's ZXTIM, NetNumenTM U31, and Huawei's SmartKit NSE rely on rule-based approaches, but they struggle with complex anomalies, focusing on spelling errors and formatting issues. Even efforts to enhance flexibility, like Liu Xin et al.'s rule categorization [15], don't address the fundamental limitations of rule-based methods. Moreover, rule-based detection can be time-consuming and prone to operational disruptions, as seen in ZTE's ZXTIM case.

2.2 Anomaly detection based on statistical methods

Statistical anomaly detection is a conventional method that assumes high-frequency events are normal, while low-frequency events are anomalies [16]. This approach relies on statistical modeling and defines deviations from the model as anomalies.

Yu Yijiao et al.[17] utilized a corpus-based method to analyze mutual information distribution in high-frequency Chinese character strings. Their study examined two-word, three-word, and four-word strings, revealing challenges in text construction and anomaly analysis when using corpus analysis and statistical methods to study word relationships [18]. The ARIMA algorithm, introduced by PINCOMBEB, employs prediction-based anomaly detection through threshold comparison, requiring substantial data training and human analysis to enhance reliability. Recently, efforts by experts have explored co-occurring word frequency analysis for anomaly detection [19]. This method identifies textual anomalies by categorizing low-frequency corpora as anomalies through statistical analysis of word, sentence, or paragraph occurrences before and after one another. However, when applied to configuration co-occurring corpora, the method often neglects the structural information of the entire configuration context. Consequently, the anomaly detection outcomes may lack comprehensiveness and accuracy. Scholars have attempted to build AI models for natural language tasks, like expression or anomaly detection, using techniques such as neural networks and knowledge graphs. These models learn from abundant spoken or written expressions to extract network structures or knowledge repositories, enabling language recommendations or anomaly detection. However, this process requires extensive sample training, consuming time and resources. Moreover, these methods are not perfectly suited for the complexities of flexible natural language.

Statistical methods are valuable for detecting configuration anomalies due to the structured format of configuration files. Unlike the complexities of natural language, these files have standardized components like device details, IP addresses, port numbers, and passwords. Statistical analysis efficiently reveals probable errors, allowing researchers to categorize potential error types. Our study employs co-occurrence frequency analysis on configuration file corpora to identify anomaly types.

2.3 Anomaly detection based on artificial intelligence

Artificial intelligence algorithms are a prominent focus in anomaly detection due to their smart algorithmic analysis and decision-making capabilities. Anomaly detection algorithms, mainly based on machine learning, are currently widely used. These machine learning-based methods can be categorized as supervised, semi-supervised, and unsupervised approaches.

Liu et al.'s Opprentice is a notable anomaly detection framework [20]. It employs KPI anomaly cycle labeling and extracts various anomalous features for training a random forest classifier. This allows for automatic threshold adjustments, enhancing detection efficiency. However, Opprentice relies on supervised learning, necessitating extensive manual data labeling. Yang et al. introduced a Gaussian Mixture Model (GMM)-based approach [21], calculating data probabilities to detect anomalies after labeling data as "normal." Rashidi et al. proposed a Bayesian Networks-based method [22] that captures relationships in a corpus, particularly effective for textual anomaly detection.

As 5G network construction expands, major operators confront the challenges of network scale and heightened demands. In configuration anomaly detection for network equipment, there's a growing emphasis on AI-driven correlation analysis methods. These advanced techniques, such as the Fp-Growth algorithm[23], explore dependencies among statements in vast configuration files. A notable example is Liu Shiguo et al.'s work at China Unicom, proposing an AI correlation analysis-based method[10]. Their approach constructs a detection model using weak correlation rules from AI correlation analysis. This model, trained on labeled anomalous data, swiftly identifies configuration anomalies within extensive configuration datasets.

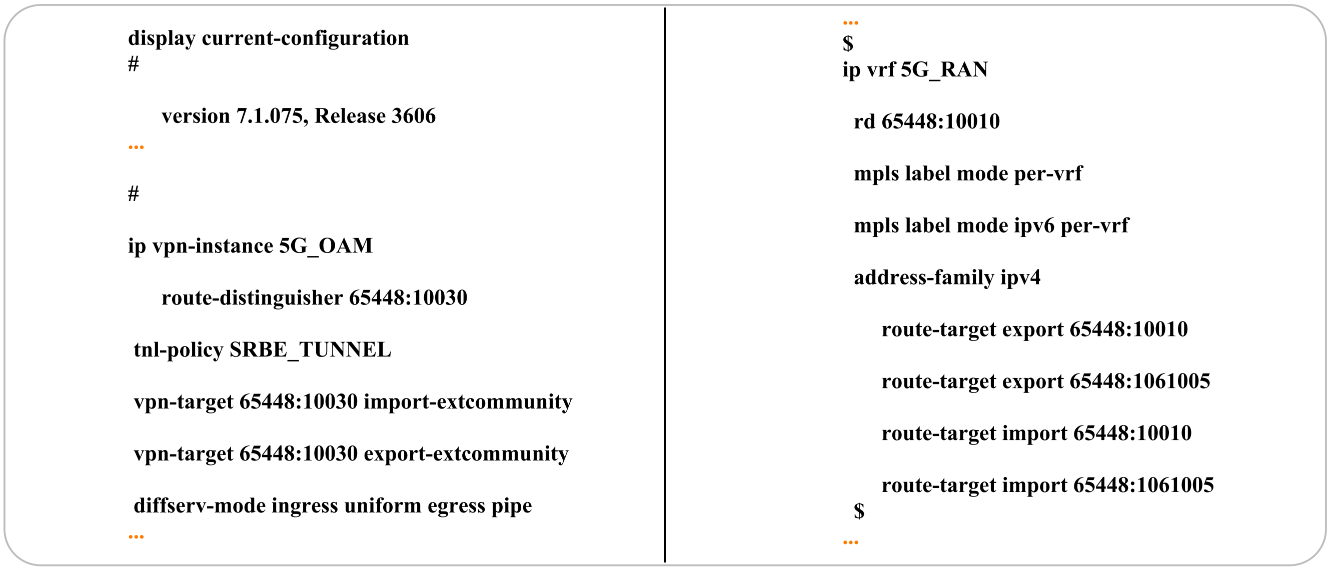
Recent prominent work in machine learning for anomaly detection largely relies on supervised or semi-supervised approaches. These methods necessitate extensive manual labeling of data and features. However, in the context of configuration anomaly detection, this entails pre-labeling a substantial volume of abnormal samples. Yet, the inherent rarity of anomalous configurations makes it challenging to amass sufficient labeled data. Furthermore, accurately and comprehensively labeling such data relies on the expertise of operations and maintenance personnel. The heavy human involvement in supervised or semi-supervised configuration anomaly detection methods often leads to less thorough and precise detection.

Given the repetitive and structured nature of configuration files, this paper introduces a solution. It suggests an unsupervised anomaly detection algorithm utilizing a configuration statement tree (see Section 3.2). This algorithm transforms the configuration file into a "configuration statement tree," extracting features based on the tree's structure. An unsupervised clustering algorithm then categorizes the configuration files, pinpointing anomalous configurations.

3 Anomaly Detection Based on Configuration Statement Trees

To tackle these challenges, this chapter proposes a method that integrates big data analysis, frequency analysis, and co-occurring corpus analysis to identify potential anomalies within current network profiles. It also introduces a technique that structures profiles into trees, extracts features using these trees, and applies unsupervised K-Means clustering to detect anomalies. Finally, an efficient anomaly detection method is presented, offering precise localization, categorization, and recommended corrective actions.

In this study, we adopt a “traditional statistical method” to determine potential anomaly types in configuration files. We utilize clustering to identify irregular configuration file structures and employ rule matching to suggest anomaly correction strategies. Instead of approaching the anomaly detection problem through supervised machine learning and model training, we optimize each module for specific solutions. Given the relative regularity of configuration files, statistical methods prove faster and more accurate for anomaly extraction compared to supervised machine learning. For detailed insights, please refer to Section 3.1. While configuration files exhibit regularity, their syntax rules are more flexible than those of programming languages, allowing for relaxed ordering. To address this, we propose a tree-based method for anomaly localization, leveraging configuration statement trees to cluster and identify anomalies. This method efficiently identifies anomalies by considering the structural information of the configuration file. For specifics, see Section 3.2, which elaborates on the structure of configuration statements. In recommending anomaly modification schemes, we apply traditional rule matching for the seven well-defined anomaly types. For rarer undefined anomalies, we provide direct location information. This approach outperforms neural networks and other machine learning methods in terms of speed. Details can be found in Section 3.3.



```
display current-configuration
#

     version 7.1.075, Release 3606
...

#

ip vpn-instance 5G_OAM

     route-distinguisher 65448:10030

 tnl-policy SRBE_TUNNEL

 vpn-target 65448:10030 import-extcommunity

 vpn-target 65448:10030 export-extcommunity

 diffserv-mode ingress uniform egress pipe
...
```

```
...
$
ip vrf 5G_RAN

  rd 65448:10010

  mpls label mode per-vrf

  mpls label mode ipv6 per-vrf

  address-family ipv4

      route-target export 65448:10010

      route-target export 65448:1061005

      route-target import 65448:10010

      route-target import 65448:1061005
  $
...
```

Fig. 1. The left is part of the statement of a configuration file of "H3C" manufacturer, and the right is part of the statement of a configuration file of "ZTE" manufacturer.

3.1 Determination of the type of exception

Network equipment configurations vary across manufacturers and versions, encompassing flexible syntax rules and diverse anomalies. Existing inspection tools primarily detect basic errors, like spelling and parameter format inconsistencies, often overlooking complex logic anomalies that impact device operation. Figure. 1 displays configuration statements from "H3C" and ZTE, revealing structured components and crucial parameters. Unfortunately, due to limitations in technology and experience, engineers often miss anomalies during detection. Scholars explore AI methods like neural networks for anomaly detection, but these approaches demand substantial time and human effort. In contrast, network equipment configuration files are regularized text with distinct information types. Considering these factors, we opt for traditional statistical methods to identify anomaly types within the configuration files.

Based on empirical observations, this paper logically infers that equipment configuration anomalies are infrequent and rare events. Since a large number of anomalies would disrupt network services, it's crucial to address these issues. Building upon the inference that configuration anomalies are typically infrequent, this section proposes utilizing big data analysis, co-occurrence corpus analysis, and frequency analysis to identify potential anomaly types within existing network configurations. Experimental results demonstrate that the statistical methods employed in this study can identify over 90% of configuration anomalies, with the remaining less than 10% being exceptionally rare and irregular. In comparison to neural networks and knowledge graphs, traditional statistical methods offer a more accurate and expedient means of determining anomaly types within configuration files.

3.1.1 Configuration file preprocessing

In this section, the configuration file undergoes a preprocessing step. Parameters like ip addresses, port numbers, passwords, and user information are matched and replaced with standardized wildcards using regular expressions. The remaining words are then categorized into "keywords" and "non-keywords" through frequency analysis. "Keywords" represent high-frequency, generic, and regular terms, while "non-keywords" encompass low-frequency and less generic terms. These "non-keywords" usually include supplementary information like equipment serial numbers, location names, and equipment names. These terms lack the necessity for detection. The algorithm for categorizing "keywords" and "non-keywords" is detailed below:

(1) Obtaining high-frequency words: In order to obtain the "keywords" in the configurations and find the configuration words that are most worthy of excavation and research, this section firstly calculates the word frequencies of all the remaining words except those that have been replaced in the above section. The configuration files of different manufacturers are evenly sampled, totaling 3,000 copies and more than 2 million lines of configuration statements. All the words are counted as a set S, $S = \{A_1, A_2, \ldots, A_n\}$, where the total number of occurrences of each word Ai is $N(A_i)$, and the word frequency of word A_i, $f(A_i)$:

$$f(A_i) = \frac{N(A_i)}{\sum_{i \in S} N(A_i)} \tag{1}$$

Calculate the word frequency f (Ai) for each word Ai in the set S. Then sort all the words according to the word frequency in ascending order, and take the top 1% words from them to get $F = \{B_1, B_2, ..., B_p\}$ is the set of high frequency words.

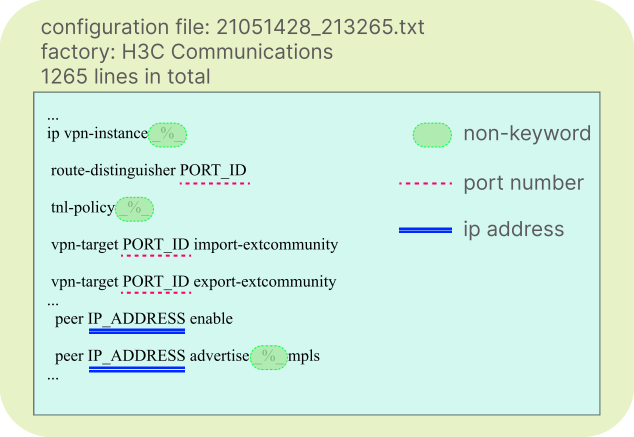
(2) get "active" words: some words, such as place names, in a configuration of a large number of repeated, but in other configuration files appear very rarely, these words in the previous step may be categorized into the collection of high-frequency words, but they are obviously not the final section to look for the " key words", so in this step, we need to calculate the "document frequency" of all words, that is, to find those words "active" in each configuration file. "Keywords", so in this step, we need to calculate the "document frequency" of all the words, that is, to find out those "active" in the various profiles of the word. For the words in the set S, first calculate the number of documents containing each word $D(A_i)$, then the document frequency $w(A_i)$ of the word A_i is:

$$w(A_i) = \frac{D(A_i)}{\sum_{i \in S} D(A_i)} \tag{2}$$

Calculate the document frequency $w(A_i)$ of each word A_i in the set S. Then sort all the words according to the document frequency in ascending order, and take the top 1% of the words to get $G = \{C_1, C_2, ..., C_p\}$ as the set of "active" words. C_p is the set of "active" words.

(3) Division of "keywords" and "non-keywords": the above two steps in the calculation of the set F and the set G to do intersection operations, you can get the "keyword set", written as The remaining words which are not in the set T are "non-keywords".

Finally, to the set T obtained in the above steps, the keywords in the configuration file are saved, and the non-keywords are replaced with uniform wildcards to obtain the preprocessed configuration file, as shown in Figure. 2. The parts circled with color blocks in the figure are the words that are replaced with wildcards according to the above rules.



configuration file: 21051428_213265.txt
factory: H3C Communications
1265 lines in total

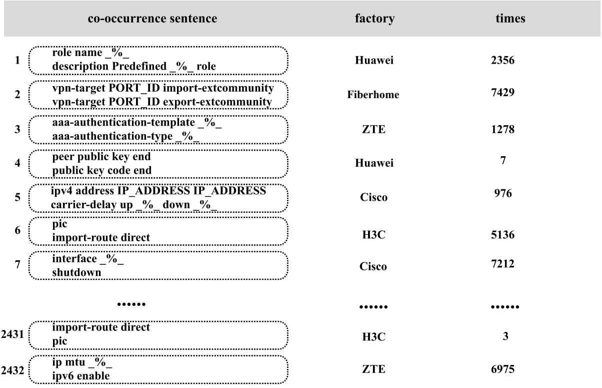
```
...
ip vpn-instance_%_
 route-distinguisher PORT_ID
 tnl-policy_%_
 vpn-target PORT_ID import-extcommunity
 vpn-target PORT_ID export-extcommunity
...
  peer IP_ADDRESS enable
  peer IP_ADDRESS advertise_%_mpls
...
```

Fig.2. Example of a pre-processed profile.

3.1.2 co-occurring corpus analysis

Due to significant correlations among configuration statements, this section employs "co-occurrence corpus analysis" to explore potential anomalies in the network configuration. Figure. 3 is a part of the "co-occurrence sentences" corpus. The approach involves tallying adjacent sentence and word occurrences in the configuration file to create a co-occurring corpus. Analyzing this corpus reveals patterns, like the instance where 'pic import-route direct' appears 5136 times, while 'import-route direct pic' occurs 3 times. As a result, it's concluded that No. 12431 exhibits a sentence order anomaly. Consequently, three types of abnormalities are identified: configuration statement order issues, redundancy, and omissions. Additionally, a “co-occurring words” corpus is constructed and analyzed for potential word-related anomalies, including the possibility of omitted spaces between words.

In addition, in this section, high-frequency words and sentences are used as templates for frequency analysis of configurations. Sentences and words are sorted according to their frequency, and the top 1% of words and phrases are taken. Then the edit distance between the remaining words and sentences and these high-frequency words and sentences is calculated[24]. By analyzing those low-frequency words and phrases with small editing distance between them and the high-frequency words and phrases, we found that there are spelling mistakes in the configuration statements, and these words with spelling mistakes are all in the keyword set T, so they are identified as keyword spelling anomalies. After the above analysis and exploration, this section specifies five types of configuration anomalies: keyword spelling errors, omission of spaces between keywords, configuration statement order anomalies, missing configurations, and configuration redundancy. Together with the ip address and port number, which are the “high incidence area” of configuration anomalies, a total of seven clear and detectable types of anomalies are given.



	co-occurrence sentence	factory	times
1	role name _%_ description Predefined _%_ role	Huawei	2356
2	vpn-target PORT_ID import-extcommunity vpn-target PORT_ID export-extcommunity	Fiberhome	7429
3	aaa-authentication-template _%_ aaa-authentication-type _%_	ZTE	1278
4	peer public key end public key code end	Huawei	7
5	ipv4 address IP_ADDRESS IP_ADDRESS carrier-delay up _%_ down _%_	Cisco	976
6	pic import-route direct	H3C	5136
7	interface _%_ shutdown	Cisco	7212
	······	······	······
12431	import-route direct pic	H3C	3
12432	ip mtu _%_ ipv6 enable	ZTE	6975

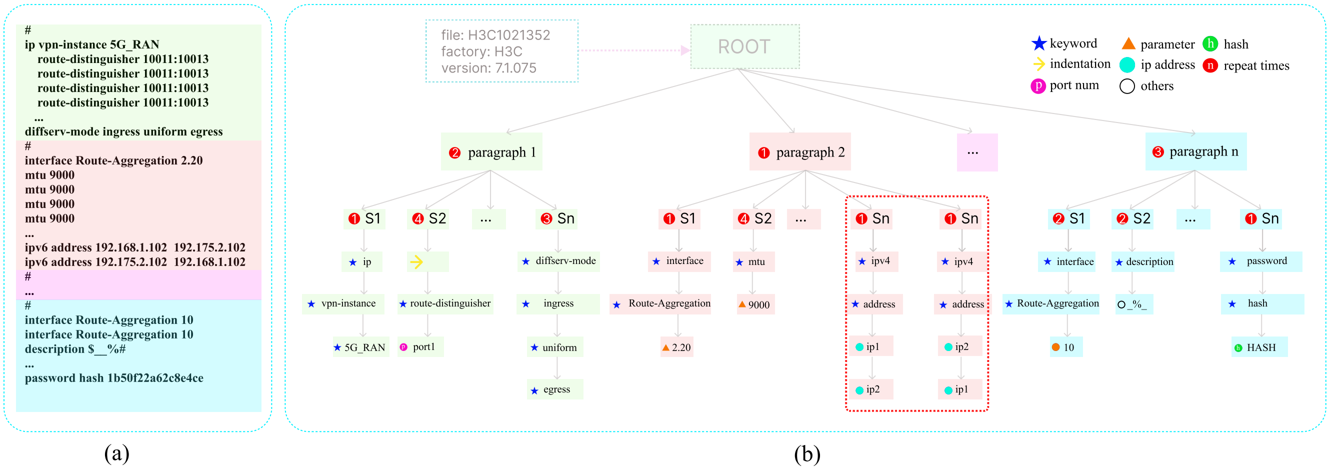
Fig.3. Part of the “co-occurrence” corpus.

While this paper has successfully employed statistical methods to uncover certain anomalies within the network configuration, there are limitations. For instance, this section overlooks the analysis of anomalies related to ip addresses and port numbers, hindering the identification of specific anomalies. Additionally, the analysis of paragraphs, sentences, and words is solely based on statistical metrics, lacking consideration for sentence and paragraph structures, limiting the scope of anomaly detection. To address these gaps, the following section draws inspiration from the abstract syntax tree concept. It introduces a novel approach, constructing a “configuration statement tree” to capture structural information within the configuration file. This tree-based approach enables feature extraction and leverages an unsupervised clustering algorithm to detect abnormal configuration files.

3.2 Configuration statement tree construction

Traditional rule-based and statistically-based configuration anomaly detection methods often lack flexibility due to fixed detection rules. Similarly, statistical methods may not fully consider the configuration file’s structure. Drawing inspiration from programming language design and compilers, which utilize abstract syntax trees (ASTs) for comprehensive program analysis[25], this section adopts the concept of ASTs. It proposes constructing a “configuration statement tree” from the configuration file, enabling feature extraction. This tree-based approach, combined with an unsupervised clustering algorithm, aids in detecting abnormal configuration files.

The “configuration statement tree” constructed in this paper is a multinomial tree structure with more than three layers, the first layer of the statement tree is the configuration file information node, which contains the file name, manufacturer, version of the device and other information; the second layer is the paragraph node, which contains the start and end position of the paragraph (line information) and the maximum number of times that the paragraph structure is continuously repeated; The third level for the sentence node, contains the sentence where the line location information and the sentence structure of the maximum number of consecutive repetitions; the fourth level from the word



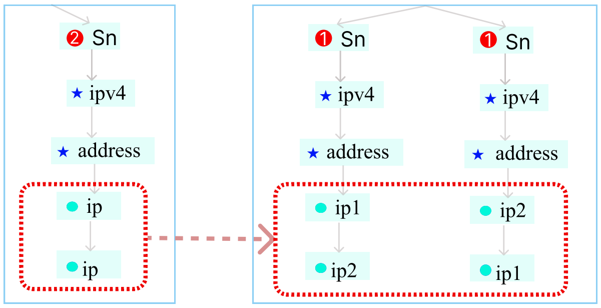
```
#
ip vpn-instance 5G_RAN
   route-distinguisher 10011:10013
   route-distinguisher 10011:10013
   route-distinguisher 10011:10013
   route-distinguisher 10011:10013
  ...
diffserv-mode ingress uniform egress
#
interface Route-Aggregation 2.20
mtu 9000
mtu 9000
mtu 9000
mtu 9000
...
ipv6 address 192.168.1.102  192.175.2.102
ipv6 address 192.175.2.102  192.168.1.102
#
...
#
interface Route-Aggregation 10
interface Route-Aggregation 10
description $__%#
...
password hash 1b50f22a62c8e4ce
```

Fig.4. (a) is a statement fragment in the configuration file, (b) is the "configuration statement tree" of the configuration file.

node for the configuration of the statement tree, each word node contains the content of the word, the type of words, keywords or non-keywords or ip address or port number, and so on. Such as Figure. 4 in this tree configuration statement shows that the configuration file named "H3C 1021352", the manufacturer of H3C Communications, the configuration of the version of 7.1.075. paragraph by 'paragraph 1', 'paragraph 2',... , 'paragraph n' composition, 'paragraph n'. The paragraph nodes in the second level record the starting and ending positions of the paragraphs and the number of consecutive repetitions, as shown in the figure, 'paragraph 1' is repeated twice, 'paragraph 2' is repeated once, the ellipses are for omitting the unshown paragraphs, and 'paragraph n' is repeated three times. The S node in the third layer represents the sentence, which also records the position information of the line where the sentence is located and the number of consecutive repetitions of the sentence. Taking paragraph1 in the figure as an example, paragraph1 consists of 'sentence1(S_1)', 'sentence2(S_2)', the omitted part is the paragraph omitting the undisplayed sentence, and 'sentence n(Sn)' in turn. where, where, 'S_1' is repeated 1 time, 'S_2' is repeated 4 times,... , 'S_n' is repeated 3 times. Below the S node are the sentence nodes.Sentence 1(S_1) consists of three keywords, so S_1 is: 'ip vpn-instance 5G_RAN', and sentence 2(S_2) consists of a hierarchical character, a keyword, and a port number in that order, so S_2 is: 'quad route-distinguisher port'.

For ip address, port number, hash password, parameters and other information in the construction of the configuration statement tree will take into account some of their own characteristics: ip address and port number within the same paragraph will usually have a strong correlation, for example, Figure. 4 in the red dashed box circled two configuration statements, 'ipv4 address 192.168.1.102 192.75. 2.10' and 'ipv4 address 192.75.2.102 192.168.1.102', which appear in sequence, in accordance with the above construction rules, they will be abstracted into two identical sentence templates stored in the same sentence node. As shown in Figure. 5a, however, because the ip addresses of these two sentences have strong correlation, they should be treated as two independent sentences when constructing the tree. Configuration syntax tree not only need to minimize the cost of the most concise way as much as possible to record the structure of the configuration file and the content of the information, but also need to be more reasonable as much as possible to record the logical relationship between the parameters of the configuration statement. Therefore, in the construction of the configuration statement tree, for the same paragraph ip address, port number, hash password, registration number and other parameters, you need to distinguish between different contents of the same type. For example, the above two statements with ip address appear in sequence, because they are in the same paragraph, so ‘192.168.1.102’ will be recorded as IP_1, and ‘192.75.2.102’ will be recorded as IP_2. in the generation of the configuration statement tree, these two sentences are abstracted into ‘ipv4 address IP_1 IP_2’ and ‘ipv4 address $IP_2 IP_1$’, as shown in Figure. 5b. Since the logical relationship of ip addresses is taken into account, these two sentences cannot be merged into a single sentence. When faced with other parameters to build the configuration statement tree, it is treated in the same way.

According to the above steps each configuration file is constructed into a configuration statement tree, for each configuration statement tree i, the number of its paragraph nodes are counted separately, denoted as the number of sentence nodes, as the number of keywords, as the tree Next. We generates a set of feature vectors, the set of these feature vectors T is used as the feature input to the K-Means algorithm, and adopts the Euclidean distance[26] as the distance measure. The value of K is chosen according to the version number of the configuration file recorded in the node of the configuration statement tree, because there is no difference in the writing logic of the same version of the configuration file. 20,000 configuration files are selected as the training set, which are divided into 5 categories according to the version number, i.e. the value of K is 5. Therefore, the clustering algorithm runs as follows: 1) randomly select the initialized 5 samples as the initial clustering centers; 2) for each sample in the dataset, compute the Euclidean distance from the 5 clustering centers and classify it into the category corresponding to the cluster center with the smallest distance; 3) recalculate the distance for each category. For each class recalculate its clustering center. After clustering all the profiles, the outliers in each cluster are the abnormal profiles, while the documents in the center of the cluster can be regarded as the correct “templates”.



(a)

(b)

Fig. 5. (a) is the branch constructed without considering the association between the parameter contents in the paragraph, (b) is the branch constructed after considering the association between the parameter contents.

In this section, we create a configuration statement tree for each configuration file to maximize the preservation of the structural content characteristics of the configuration file. In the process of constructing the tree, this section also proposes to individually process parameters such as ip address, port number, hash password, and so on, in order to ensure that the configuration statement tree can maximize the recording of the logical relationship between the parameters. This section then uses the unsupervised K-Means algorithm to cluster these statement trees to obtain a model for abnormal configuration file detection, extracting those outliers as abnormal documents, while those cluster centers are used as ”templates” for correct configurations. In the next section, this paper will localize the anomalies in the sieved anomalous profiles and give the recommended modifications.

3.3 Recommendations for locating and modifying anomalies

The preceding clustering model effectively identifies abnormal profiles, leaving the current section’s focus on precisely locating anomalies within these profiles, as outlined in Section 3.1. Should these profiles meet any of the seven anomalies outlined in the same section, corresponding correction schemes are recommended. This recommendation employs an enhanced rule-based approach. While contemporary efforts explore machine learning and neural networks for more generalized recommendation algorithms, this text’s statistical method to detect seven primary anomalies, covering over 90% of common anomalies, offers well-defined definitions. Furthermore, leveraging the clustering results from the configuration statement tree, an algo-

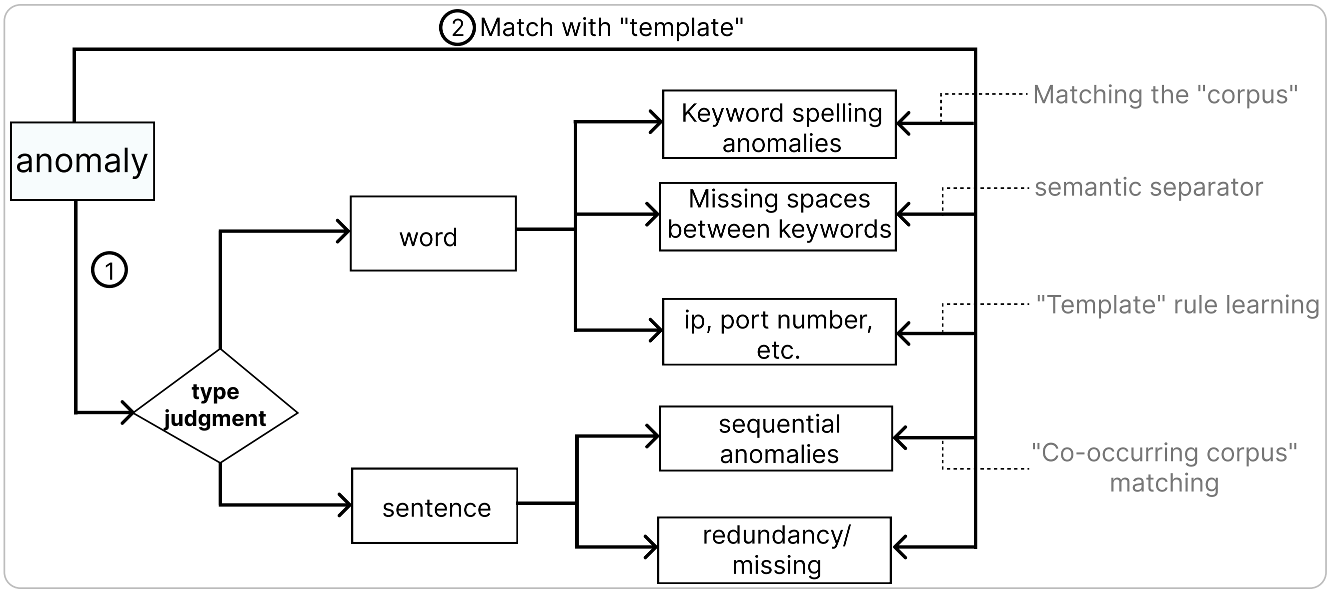


Fig.6. Recommended method steps for locating and modifying exceptions.

rithm is devised to rapidly pinpoint the anomalous sentence or word. Thus, compared to machine learning and neural network-based methods, the traditional rule-based recommendation approach boasts superior accuracy and operational efficiency.

Figure. 6 shows the process of anomaly localization and modification recommendation. First, the anomaly profile and the "template" profile extracted from the clustering in the previous section are compared one by one in the order of paragraph nodes, sentence nodes and word nodes. The anomalies can be quickly localized to the paragraph, sentence or word. If the abnormal node is a single word node, it means that it is a word abnormality. The word anomalies presented in Section 3.1 include: misspelled keywords, missing spaces between keywords, and, specifically, ip address and port number anomalies. The following is a discussion of how to identify each of these anomalies and their recommended modifications in turn:

(1) *Keyword spelling anomaly.* For non-keyword anomalies, a comparison with the corresponding "template" word is performed. If the template word is a keyword and the editing distance between the two words is less than 2, it's identified as a keyword spelling error. Similarly, if the template word is a keyword and the edit distance is less than 2, it's considered a misspelling, and the correct word is taken from the template. For edit distances greater than 2, a check in the "keyword corpus" is done to match words with edit distances less than 2. If found, it's confirmed as a keyword misspelling, with the correct spelling derived from the corpus. If not, it's identified as a non-keyword misspelling.

(2) *Lack of space between keywords.* If the type of the exception node is "other words", and it can be determined that it is not a spelling error of the keyword, then use the words in the "keyword corpus" as a reference to perform semantic disambiguation, if the disambiguation results can be matched in the corpus, then it can be clear that the exception is "lack of space between keywords". If all the results can be matched in the corpus, then it can be clear that the exception is "lack of space between keywords". Finally, it gives the location of the exception, the type of the exception and the recommended modification scheme.

(3) *The abnormal node is "other words".* It is confirmed that it is not one of the above two types of abnormality after matching, the reason for the abnormality may be an abnormality in the parameters of the ip address, port number, and so on. If the type of the cor-

responding normal node in the template is “ip address, port number, etc.”, the reason for the exception is the wrong format of these parameters. Since the statement tree records the logical pattern between each ip and port parameter in a paragraph, you can find the correct content of the abnormal ip address or port number parameter by following the pattern in the “template”.

(4) *The abnormal of ip address and port number.* If the abnormal node type successfully matches “ip address” or “port number”, which means that there is no error in the format of ip and port number, then it can be inferred that the logical configuration of these parameters is abnormal. If there is no error in the format of ip and port number, it can be inferred that the logical configuration of these parameters is abnormal. Therefore, in this paper, we refer to the format of the template to learn the correct writing logic of the parameters, and adjust the logical order of the ip address and port number anomalies.

For sentence-level anomalies identified in Section 3.1, these anomalies might include sequential anomalies, missing configuration statements, or redundancy. Identifying anomalies in statements is simpler than in words. The following outlines how to identify each of these anomalies and provide recommended modifications:

(1) There are two ways to determine if a sentence is out of order. First, we try to match with the corresponding branch of the “template” tree to see if there is a corresponding order relationship between the matched sentence and the abnormal sentence. And if so, we will give the location of the abnormality, the type of abnormality and the recommended modification plan. However, due to the merging of statements, the “template” may not be able to match directly. If the exception type cannot be derived by matching the “template tree” directly, this section bundles the exception statement and its next configuration statement into a set of corpus R. We bundle the anomalous statements and their next configuration statements into a corpus R, and query whether R is a “low-frequency corpus” in the “co-occurring statement corpus” constructed in Section 3.1, and if it is indeed a “low-frequency corpus”, we give the anomaly localization, anomaly type, and the recommended modification scheme. If it is indeed a “low-frequency corpus”, it will give the anomaly localization, anomaly type and recommended modifications.

(2) For missing or redundant statements, this paper matches with the “template” tree to find out whether there are omissions or over-allocations, and if so, it gives the location of the anomaly, the type of the anomaly, and the recommended modification plan.

4 Evaluation

This chapter evaluates the proposed anomaly detection algorithm in two main aspects. Firstly, it compares the algorithm with traditional methods like ZXSEM/TIM400 and Guangdong Unicom’s AI correlation analysis-based method[10], as well as Bayesian Networks[11, 22] and FpGrowth-based anomaly detection[23]. This highlights the algorithm’s comprehensiveness. Secondly, the algorithm’s accuracy, recall, and efficiency are quantitatively assessed to demonstrate its superiority over the other two methods. The experiments are conducted on a Windows 10 system with Intel(R) Core i7-9700 CPU and 16GB RAM, using Python 3.6, a single thread, and one CPU core.

4.1 Experimental data sources and security issues

The datasets used for algorithm experimentation and evaluation are derived from network equipment configuration files provided by a domestic operator. These files come from five manufacturers: ZTE, H3C Communications, Huawei, Fiberhome Communications, and Cisco. Due to security and privacy concerns, the data is encrypted before experimentation. Ip addresses and port numbers are encrypted following the method by Zhao Xiaohang and Yi Jun[27, 28]. Each bit of the ip address and port number is assigned a random value, and its ASCII code is added for encryption, with the randomly generated values serving as the encryption key. This approach retains the sequential and mu-

Table1. Comparison of detection capabilities of different algorithms

Exception type	Ours	Traditional algorithm	AI correlation analysis	Bayesian net	FpGrowth
Keyword spelling	✓	✓	X	✓	✓
Missing space	✓	✓	X	✓	✓
ip anomaly	✓	✓	X	X	X
port anomaly	✓	✓	X	X	X
command Missing	✓	X	✓	✓	✓
port anomaly	✓	X	✓	✓	✓
sentence order anomaly	✓	X	✓	✓	✓

tual relationship information of ip addresses and port numbers while supporting third-party intervention and confidentiality requirements. For sensitive router information like usernames and passwords, this paper employs a hash function and ternary group-based cipher table encryption algorithm[29] to ensure data security and user privacy protection.

4.2 Analysis and Comparison of Detectable Anomaly Types

In this section, configurations from five manufacturers, H3C, Huawei, Cisco, ZTE, and Fiberhome, are used, and 200 cities are randomly taken from each manufacturer, and 100 profiles are taken from each city, totaling 100,000 profiles as a dataset, and the algorithms of this paper and the traditional rule-based as well as AI-based correlation analysis methods are respectively used to conduct multiple experiments on this dataset.

Experiments are conducted to detect the seven types of configuration anomalies proposed in this paper. As shown in Table. 1, the traditional rule-based ptn network patrol tool can only recognize three types of anomalies, as keyword spelling and space missing, format anomalies of ip address and port number, which are simple anomalies that don't involve the configuration logic, and although the current AI contextual correlation analysis algorithms based on the AI can solve the problems on the logical correlation, it performs poorly in the traditional error detection, and the The types of anomaly detection supported by Bayesian Networks and FpGrowth are also not as rich as the methods in this paper. Obviously, compared with several other methods, this paper's algorithm is more comprehensive in its ability to detect anomalies, and this paper's algorithm is more generalizable.

In addition, the configuration statement tree used in this paper's algorithm is unsupervised, so that even when the type of anomaly is unknown, i.e., in addition to the seven types of anomalies defined in this paper's statistical methodology, the clustering algorithm of the configuration statement tree can still locate the positions of other anomalies and return these structural anomalies in real time to the operation and maintenance personnel for analysis. Typically, traditional rule-based or correlation-based analysis methods are supervised or semi-supervised. To achieve anomaly detection, you need to artificially generalize the types of anomalies and annotate the dataset, which is obviously affected by the limitations of people's thinking, and is both time-consuming and incomplete. For example, in a configuration file, there are configuration statements 'aaa session-limit telnet' and 'aaa session-limit ssh', which do not seem to belong to any category of anomalies, and they are both high-frequency statements, and the frequency of co-occurrence of the two statements is very high, and engineers are hard to find any anomalies of these two statements together. However, when analyzing the statement tree clustering, these two sentences were thrown as exceptions. After confirmation by the operation and maintenance engineers and analysis of the configuration passages, it was found that the two configurations could not be used due to insufficient privileges of the device, rather than a problem with the two configurations statements themselves. The algorithm in this paper is an unsupervised one that does not require manual labeling of anomalous data and can automatically detect various anomalies. After comparative analysis, the algorithm in this paper achieves more comprehensive and intelligent anomaly detection.

4.3 Quantitative assessment

4.3.1 accuracy

Accuracy is a key measure of algorithm effectiveness. Since our unsupervised clustering method can't directly assess overall accuracy, we focus on the seven most probable anomaly types identified through statistical analysis. Using 10,000 data samples from five vendors, we conduct two steps. First, the algorithm scans and corrects the dataset to remove anomalies. Then, for each of the seven types, we introduce 20 artificial errors per configuration. The algorithm is run on this adjusted dataset to calculate accuracy.

As traditional ptn network inspection tools rely on clear matching rules for anomaly detection, their accuracy assessment lacks significance. Therefore, our paper's algorithms are mainly compared with AI correlation analysis, Bayesian Networks (BAN) based anomaly detection, and FpGrowth based anomaly detection methods.

In Table. 2, we statistically analyze the accuracy of anomaly detection for various types such as keyword spelling, missing spaces, ip address, port number, missing commands, command redundancy, and sentence order. Our paper's algorithms achieve an average accuracy rate of 86%, exceeding 90% in the cases of keyword spelling and missing spaces. In comparison, correlation analysis-based algorithms achieve an average accuracy rate of 85%, with none surpassing 90%. This analysis demonstrates the significant accuracy improvement of our algorithm.

4.3.2 recall rate

Recall is a reliability measure indicating the proportion of actual anomalies detected by an algorithm out of the total anomalies that should have been identified. As our approach employs an unsupervised clustering algorithm, directly evaluating its recall for all anomalies is not feasible. Consequently, the recall assessment here centers solely on the seven most probable anomaly types identified through our co-occurrence frequency analysis. Using the same dataset of 10,000 entries mentioned earlier, we apply our algorithm after rectifying the dataset by adding human-generated anomalies. This assessment compares our algorithm with AI correlation analysis, Bayesian Networks (BN), and FpGrowth anomaly detection algorithms. As traditional rule-based ptn inspection tools don't utilize statistical or machine learning techniques and don't involve recall considerations, we exclude them from this recall comparison.

The experimental results are shown in Table. 3, the recall of keyword spelling, keyword missing space, ip address anomaly, and command missing can reach more than 90%, and the average recall reaches 85.1%. In anomaly detection, the gap between this paper's algorithm and AI-based association analysis algorithms as well as Bayesian Networks or FpGrowth-based methods in terms of misspellings and missing spaces is not that big. For example, 'route' is misspelled as 'rote' in the correct example 'port link-mode route', 'address - family ipv4 unicast' misses a space and is written as 'ipv4unicast', the recall of such anomalies is usually high, but if there is a problem of statement ordering like A+B vs. B+A, the algorithm in this paper greatly outperforms the other algorithms, e.g., statement A is 'address-family ipv4', while statement B is ' route-distinguisher 65448:20001', the reversal of the front-to-back order of A and B does not result in a syntax error, and the writing of A before B or A after B is syntactically correct. However, for traditional ptn inspection tools, correlation analysis algorithms, or configuration anomaly detection algorithms based on Bayesian Networks or FpGrowth, because they use human-made rules or they need to manually label a part of the data during training, once the order of AB statements is switched, due to the fact that there is no corresponding rule in the algorithms or the algorithms themselves do not learn the corresponding rule to match it, the order of AB statements will be changed. The algorithm itself does not learn the corresponding rules

Table2. Comparative analysis of accuracy

Exception type	Ours	AI correlation analysis	Bayesian net	FpGrowth
Keyword spelling	91%	X	87%	85%
Missing space	94%	X	91%	87%
ip anomaly	89%	X	X	X
port anomaly	82%	X	X	X
command Missing	88%	86%	71%	44%
port anomaly	84%	83%	68%	79%
sentence order anomaly	82%	82%	75%	77%

Table3. Comparative analysis of recall rates

Exception type	Ours	AI correlation analysis	Bayesian net	FpGrowth
Keyword spelling	93%	X	90%	88%
Missing space	97%	X	93%	89%
ip anomaly	86%	X	X	X
port anomaly	84%	X	X	X
command Missing	83%	82%	78%	74%
port anomaly	85%	86%	51%	62%
sentence order anomaly	87%	81%	24%	38%

to match with them, it cannot find the corresponding anomalies, resulting in the trained anomaly detection model does not learn the features of such anomalies, which leads to its low recall rate. The configuration syntax tree used in this paper adopts an unsupervised clustering algorithm to match the abnormal configuration profiles by comparing the abnormal configuration statement tree with the normal “template” statement tree, and adopts a statistical method to determine the types of possible anomalies, which abandons the way of manually inputting the rules or manually annotating some of the anomalies, and has a stronger robustness and higher reliability. The algorithm in this paper has a higher recall rate because it is more robust and reliable than the manual rule input or manual labeling of some anomalies.

In addition, in the course of the experiments in this section, in addition to verifying the recall of the seven defined anomalies identified using statistical methods in 10,000 sample files, 27 additional anomalies that are not within the scope of these seven anomalies are thrown out, which are distributed in 16 configuration files in the same province of the same backbone network. Engineers refer to this information to quickly launch a targeted screening of the backbone network node configurations in this province, and finally successfully resolve the configuration anomalies in this province based on the information provided by the algorithm in this paper. The algorithm in this paper makes full use of the “strong regularity” of the configuration statement and the “flexibility” of writing, and adopts a combination of unsupervised machine learning models and statistical theory, combining the advantages of unsupervised methods that do not require manual pre-labeling of data and the advantages of statistical methods in dealing with large amounts of data. It combines the advantages of unsupervised methods, which do not require manual pre-labeling of data, and statistical methods, which are efficient and accurate in dealing with a small number of anomalies in large-scale data. Compared with pure statistical methods and pure machine learning methods, the algorithm in this paper has advantages in terms of comprehensiveness, accuracy and efficiency.

4.3.3 efficiency

In this section, the algorithm of this paper is tested with the rule-based ptn inspection tool, respectively, by executing the two algorithms and counting the time they take to accomplish the same task. In this section, 2000 randomly selected configurations from each of the 5 different manufacturers each, totaling 10000 configuration files and 17521900 lines of configuration statements.

When employing the ptn autopatrol tool, the first experiment was forcibly halted after 86 minutes due to its lack of robustness. The subsequent attempt took 8 hours and 32 minutes, averaging 3.07 seconds per file for detection. Conversely, utilizing our algorithm completed the task in 1 hour and 18 minutes, averaging about 0.47 seconds per configuration file. Comparatively, AI correlation analysis, Bayesian Networks, and FpGrowth-based methods took approximately 1 hour and 35 minutes for the same task, slightly slower than our algorithm.

In summary, it can be clearly seen that the detection time of the rule-based ptn inspection tool on the dataset is about 6 times that of this paper's algorithm, and this paper's algorithm is far ahead of the traditional algorithm in terms of efficiency. For the configuration anomaly detection algorithm based on Bayesian Networks and the configuration anomaly detection algorithm based on FpGrowth, this paper's algorithm has a slight advantage in efficiency, but the accuracy, recall, and generalization ability of this paper's algorithm are much higher than these two algorithms.

4.3.4 Algorithm Stability

To assess the algorithm's stability, we focus on the clustering model's stability. We randomly selected 2,000 configuration files from each of the five manufacturers, creating 100 training groups for 100 distinct clustering models. Subsequently, 10,000 configuration files were randomly chosen, generating 20 sets of random anomaly configurations for each file, totaling 200,000 anomalies. These configuration files were then processed through each of the 100 models to determine their detection rates. Three features were chosen for experimentation: paragraph nodes, sentence nodes, and keywords (PSK features), which were ultimately adopted in this paper; bottom-level nodes and paragraph nodes (PD features); and sentence nodes and keywords (SK features). The experiments were conducted using these three feature inputs.

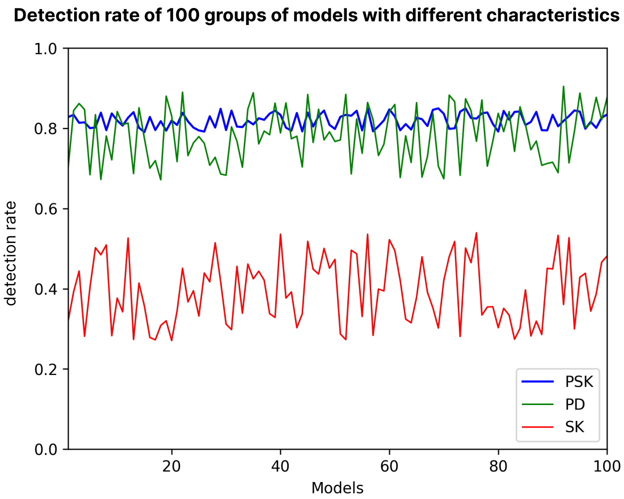


Fig. 7. Detection rate of 100 groups of models with different characteristics.

Figure. 7 illustrates the stability assessment of the models. The x-axis shows the serial numbers of the 100 models, and the y-axis indicates the proportion of detected anomalies compared to preset anomalies. The blue line corresponds to the model used in this paper, showing a smooth trend and indicating high stability. In contrast, the red and green lines represent results from the use of other feature selections, displaying lower algorithm stability. This is attributed to limited feature coverage across different manufacturer versions, resulting in significant detection variations for various configuration file versions. In this study, the inclusion of paragraph nodes, sentence nodes, and keywords ensures comprehensive representation for all manufacturers and versions. The experimental outcomes emphasize the robust stability, generalization, and high reliability of the proposed algorithm, model, and feature selection.

5 Conclusion

In this paper, we propose an unsupervised network device configuration anomaly detection method based on configuration statement tree, which draws on the design concept of abstract syntax tree, and firstly proposes to construct the configuration file of the network device into a "configuration statement tree" to retain the information of the configuration file, and then extracts the features of the configuration file through the configuration statement tree, and subsequently The unsupervised K-Means clustering algorithm is used to find out the abnormal configuration files. In this paper, frequency analysis and co-occurrence corpus analysis of profiles are performed to determine seven detectable types of anomalies, and anomaly localization and recommended modifications are given. From the results of quantitative evaluation, the average accuracy and average recall of the algorithm are greater than 85%, and some scenarios reach more than 90%. For a scale of ten million row configurations, the speed of the algorithm in this paper is about 6 times faster compared to the traditional ptn inspection tool. In addition, besides the seven types of anomalies identified, for other possible anomalies, the algorithm can give their localization, which is then handed over to the operation and maintenance engineers for manual judgment. However, the algorithm in this paper still has the following shortcomings: the configuration statement tree does not give a link between ip addresses, port numbers and other parameters across segments, and it cannot detect segment-to-segment related anomalies; the algorithm in this paper has a certain degree of versatility, but it puts too many configuration files from different manufacturers together to perform the anomaly detection accuracy and recall will be reduced; this paper performs better in anomaly localization, and anomaly screening, but it does not have the same performance when it comes to The recommended modification scheme for anomalies is still based on the traditional rule matching method.

References

[1] P. J. Willis. "The challenges in building a carrier-scale IP network". In: *BT Technology* 18.3 (2000), pp. 11–14.

[2] Gozde Bakioglu and Ali Osman Atahan. "AHP integrated TOPSIS and VIKOR methods with Pythagorean fuzzy sets to prioritize risks in self-driving vehicles". In: *Applied Soft Computing* (2021), p. 106948.

[3] Yushan Siriwardhana et al. "A survey on mobile augmented reality with 5G mobile edge computing: architectures, applications, and technical aspects". In: *IEEE Communications Surveys & Tutorials* (2021), pp. 1160–1192.

[4] G. H. LIU, X. C. Meng, X. R. Zhou, et al. "Exploring the optimization of China Unicom packet domain IP bearer network architecture for 5G". In: *Telecommunications Technology* 12 (2019), pp. 95–98.

[5] W. Q. WANG. "PTN network inspection solution for LTE". In: *Science and Technology Innovation* 27 (2020), pp. 62–63.

[6] H. M. LIU and G. CHEN. "Innovative research and practice of network operation and maintenance system based on centralization and intelligence". In: *China New Communication* 17.02 (2015), pp. 68–71.

[7] J. CUI. "Introduction to the construction of intelligent operation and maintenance mode of 5G network". In: *Technology and Market* 28.05 (2021), pp. 126–127.

[8] Theo Araujo et al. "In AI we trust? Perceptions about automated decision-making by artificial intelligence". In: *AI & SOCIETY* (2020), pp. 611–623.

[9] Shivam Gupta et al. "Artificial intelligence for decision support systems in the field of operations research: review and future scope of research". In: *Annals of Operations Research* (2022), pp. 1–60.

[10] X. W. LIU, D. D. MA, X. B. YE, et al. "Application of AI based Configuration Audit System in 5G Backhaul Network". In: *Designing Techniques of Posts and Telecommunications* 8 (2021), pp. 15–19.

[11] T. L. LIN, J. G. CHEN, W. J. GUO, et al. "Application of big data analysis methods in 5G precision construction". In: *Changjiang Information and Communication* 35.06 (2022), pp. 230–232.

[12] Michael J. Hofmann et al. "Simple Co-Occurrence Statistics Reproducibly Predict Association Ratings". In: *Cognitive Science* 42.7 (2018), pp. 2287–2312.

[13] J. ZHANG, X. WANG, ZHANG H., et al. "A novel neural source code representation based on abstract syntax tree". In: *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE. 2019, pp. 783–794.

[14] Sinaga Kristian P. and Ming-Sung Yang. “Unsupervised K-Means clustering algorithm”. In: *IEEE Access* 8 (2020), pp. 80716–80727.

[15] Liu X. et al. “Rule-based anomaly detection for inter-domain routing systems”. In: *Journal of the National University of Defense Technology* 03 (2006), pp. 71–76.

[16] GS, Smrithy, and Ramadoss Balakrishnan. “A Statistical-Based Light-Weight Anomaly Detection Framework for Wireless Body Area Networks”. In: *The Computer Journal* (2022), pp. 1752–1759.

[17] Y. J. YU, Y. F. YIN, and Q. LIU. “Analysis of the distribution pattern of high-frequency Chinese character string mutual information based on large-scale corpus”. In: *Computer Science* 41.10 (2014), pp. 276–282.

[18] B. PINCOMBE. “Anomaly Detection in Time Series of Graphs Using ARMA Processes”. In: *Asor Bulletin* 24.1 (2005), pp. 67–75.

[19] Akram Sadat Jafari Roodbandi, Alireza Choobineh, et al. “Research outputs in ergonomics and human factors engineering: a bibliometric and co-word analysis of content and contributions”. In: *International Journal of Occupational Safety and Ergonomics* 28.4 (2022), pp. 2010–2021.

[20] D. P. LIU, Y. J. ZHAO, H. W. XU, et al. “Opprentice: Towards Practical and Automatic Anomaly Detection through Machine Learning”. In: *ACM. 15th Internet Measurement Conference.* Tokyo, Japan: ACM, 2015, pp. 211–224.

[21] X. W. YANG, L. J. LATECKI, and D. POKRAJAC. “Outlier Detection with Globally Optimal Exemplar-based GMM”. In: *International Conference on Data Mining.* Sparks, Nevada, USA: SDM, Apr. 2009, pp. 145–154.

[22] L. Rashidi, S. Hashemi, and A. Hamzeh. “Anomaly detection in categorical datasets using bayesian networks”. In: *International Conference on Artificial Intelligence and Computational Intelligence.* 2011, pp. 610–619.

[23] Lior Shabtay et al. “A guided FP-Growth algorithm for mining multitude-targeted item-sets and class association rules in imbalanced data”. In: *Information Sciences* (2021), pp. 353–375.

[24] Mahdi Boroujeni et al. “Approximating edit distance in truly subquadratic time: Quantum and mapreduce”. In: *Journal of the ACM* 68.3 (2021), pp. 1–41.

[25] Denis Merigoux, Raphael Monat, and Jonathan Protzenko. “A modern compiler for the french tax code”. In: *Proceedings of the 30th ACM SIGPLAN International Conference on Compiler Construction.* 2021.

[26] Zibo Dong. “Analytical research on 3D point cloud segmentation algorithm based on improved Euclidean distance”. PhD thesis. North China Electric Power University, 2021, pp. 4–38.

[27] X. H. ZHAO. *Research on encryption method based on DNA computing.* Tech. rep. Zhengzhou Institute of Light Industry, 2013.

[28] J. Y and M. Q. “Design of user password authentication scheme based on ACSII code and random numbers”. In: *Computer and Digital Engineering* 39.03 (2011), pp. 102–104.

[29] J. D. CAO. “Research on cryptographic table encryption algorithm based on Hash function and triplet”. In: *Software Guide* 11.11 (2012), pp. 54–56.