electronics

MDPI

Article

Border Gateway Protocol Route Leak Detection Technique Based on Graph Features and Machine Learning

Chen Shen [1], Ruixin Wang [2], Xiang Li [1,*], Peiying Zhang [2], Kai Liu [3,4] and Lizhuang Tan [5,6]

[1] China Academy of Information and Communications Technology (CAICT), Beijing 100037, China; shenchen@caict.ac.cn
[2] Qingdao Institute of Software, College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266580, China; z24070100@s.upc.edu.cn (R.W.); zhangpeiying@upc.edu.cn (P.Z.)
[3] State Key Laboratory of Space Network and Communications, Tsinghua University, Beijing 100084, China; liukaiv@tsinghua.edu.cn
[4] Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China
[5] Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China; tanlzh@sdas.org
[6] Shandong Provincial Key Laboratory of Computing Power Internet and Service Computing, Shandong Fundamental Research Center for Computer Science, Jinan 250014, China
* Correspondence: lixiang3@caict.ac.cn

Abstract: In the Internet, ASs are interconnected using BGP. However, due to a lack of security considerations in the design of BGP, a series of security issues arise during the propagation of routing information, such as prefix hijacking, route leakage, and AS path tampering. Therefore, this paper conducts research on the detection of route leakage. By analyzing BGP routing information, we abstract the routing propagation relationship between ASs into a network topology graph, and extract graph features from the graph abstracted from routing data at certain time intervals. Based on the structural robustness features and centrality measurement features of the graph, we determine whether a route leakage has occurred during the current time period. To this end, we use machine learning methods and propose a weighted voting model. This model trains multiple single models and assigns weights to them, and through the weighted analysis of the results of multiple models, it can determine whether a route leakage has occurred. In addition, to determine the corresponding weights, we use genetic algorithms for identifying route leaks. The experimental results show that the method used in this paper has a high accuracy rate, and compared with a single model, it performs better on multiple datasets.

Keywords: border gateway protocol; route leak detection; genetic algorithms; machine learning

check for updates

Citation: Shen, C.; Wang, R.; Li, X.; Zhang, P.; Liu, K.; Tan, L. Border Gateway Protocol Route Leak Detection Technique Based on Graph Features and Machine Learning. *Electronics* **2024**, *13*, 4072. https://doi.org/10.3390/electronics13204072

Academic Editors: Mehdi Sookhak and Hung-Yu Chien

Received: 28 August 2024
Revised: 30 September 2024
Accepted: 14 October 2024
Published: 16 October 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The entire Internet's top layer interconnects autonomous systems (ASs) through Border Gateway Protocol (BGP) [1]. Each autonomous system has a unique identifier, the autonomous system number. ASs are managed by entities such as Internet service providers, companies, and various universities [2]. Because the internal network needs of each autonomous system are different, it is impossible to standardize the routing rules between autonomous systems. They must be manually configured according to actual situations and experience, which can lead to various problems due to non-standard or incorrect configurations.

Route leakage, a common anomaly in BGP networks, refers to the unauthorized dissemination of routing information across the Internet, which is characterized by its rapid spread and the difficulty in detection. The root cause is that during AS routing forwarding, legitimate route reachability information is leaked to adjacent ASs in violation of routing

policies [1]. To mitigate the impact of route leakage on the global network, the most primary issue is the timely detection of route leakage. Route leakage seems to be a normal BGP exception, but its detection poses many practical difficulties. When a route leakage event occurs, the spread of route messages is legal, but it violates the route leakage forwarding rules, making it impossible to determine whether a route leakage anomaly has occurred solely based on an AS-IP or other mature database mapping information.

Currently, there are relatively few security solutions for route leakage, and existing Border Gateway Protocol security technologies are still unable to effectively address route leakage. However, route leakage attacks account for about 22% of BGP security attacks [3], and there are always large or small route leakage events occurring every day (for example, the Google server interruption incident in 2015, which caused Google's network to be inaccessible in parts of Europe and Asia for nearly 24 h). Therefore, research on how to detect, warn, and prevent route leakage is of great significance for the security of the Border Gateway Protocol and the global Internet security.

We have found that there are two broad approaches to determine if a route leakage has occurred: BGP route leakage research based on commercial relationship inference and route leakage detection based on machine learning.

Early detection methods for BGP route leakage were based on analyzing the commercial relationships needed to make a determination. This method involves analyzing the propagation path of routing messages to determine the relationship between adjacent autonomous systems (ASes) as either customer–provider or peer-to-peer. By examining whether the propagation process of each route violates the valley-free rule, it is possible to identify the occurrence of route leakage.

Guo et al. cited [4] proposed a novel approach to detect BGP route leaks by leveraging the inference of commercial relationships. The core concept of this method is to initially deduce the relationships between autonomous systems through the application of heuristic rules. Subsequently, by integrating these commercial relationships with their model, they formulated four rules that pinpoint route leaks. These rules reveal anomalies compared to the standard operational patterns. The methodology involves tagging autonomous system paths with identified commercial relationships and subsequently assessing potential route leaks by evaluating the adherence of these paths to the established leakage rules.

In recent years, research on BGP anomaly detection based on machine learning models has become increasingly popular. From the perspective of machine learning, the anomaly detection problem can be seen as a binary classification problem [5]. Border Gateway Protocol message structures are relatively complex, and some features extracted from update messages and other messages can be used for the classification of normal and abnormal category data. In this regard, many researchers have proposed various classification methods using different features.

Li et al. [6] proposed an Internet routing forensic framework based on the decision tree algorithm to detect anomalies. This framework can detect route leakages based on the learning rules of past abnormal events. De Urbina Cazenave et al. [7] proposed a similar architecture that can use different data mining algorithms [8], such as Decision Tree (DT), Naive Bayesian (NB), Support Vector Machine (SVM), etc. However, these methods are usually unable to identify the location.

The main idea of BGP route anomaly detection based on machine learning is first, based on the characteristics of the update message volume and "AS PATH" attribute changes during the occurrence of anomalies, then based on the changes in multiple features for feature weighting to obtain a feature evaluation factor [9], or use classifiers such as Support Vector Machine [10] or Decision Tree [11] to train the samples. Finally, based on the feature evaluation factor or the trained classifier, the inter-domain security situation is assessed.

In response to the aforementioned research, this paper proposes a machine learning-based BGP route leak detection method that is based on graph features and a weighted voting mechanism. The work presented in this paper includes the following:

1. The raw data from four route leak incidents are extracted and the AS paths are abstracted into a network topology graph. A series of graph features from the network topology graph are extracted using structural robustness features and centrality measures.
2. A machine learning model based on a weighted voting mechanism is proposed, where multiple base learners are combined to enhance the overall performance of the model and improve its generalization.
3. In this paper, a genetic algorithm will be used to determine the weights of the base learners in the model, thereby further enhancing the model's performance. Additionally, we have implemented real-time monitoring functionality using this model.

2. Methods

2.1. Data Processing

Before the training of a BGP route leak detection model, data collection is the cornerstone of the entire research process. As shown in Figure 1, the diagram illustrates our data processing procedure in detail, including data collection, transformation, cleansing, and the construction of the topological structure.

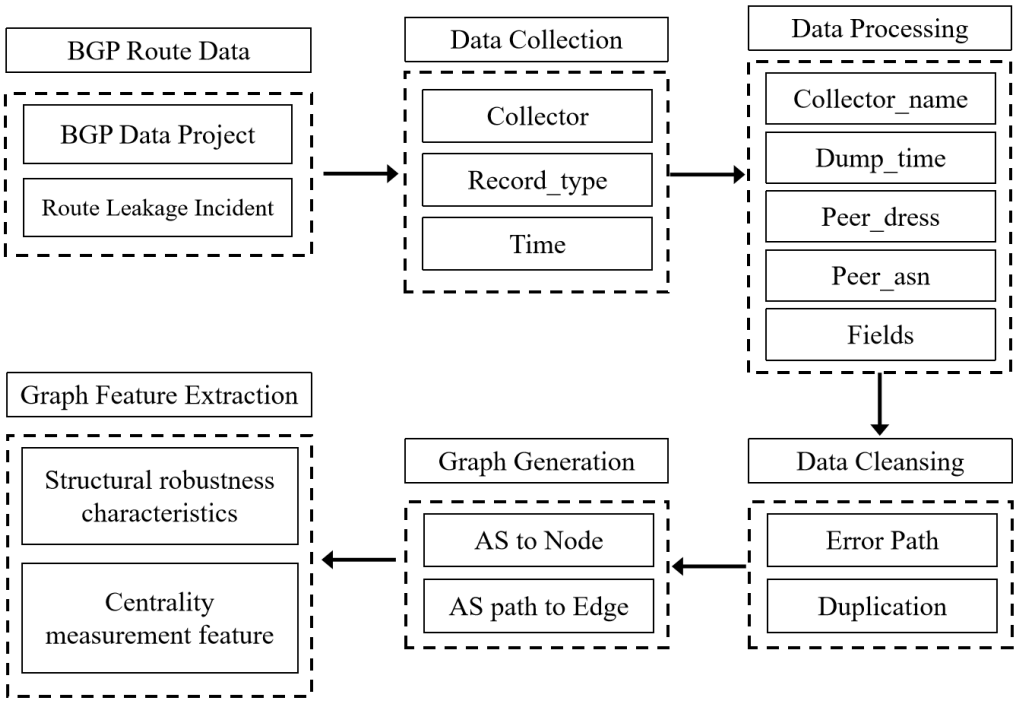


Figure 1. Data collection and processing process.

To ensure that we can accurately and comprehensively acquire the relevant data, we have chosen the RIPE and RouteViews projects as our data sources. These projects are the preferred choice for many researchers due to their rich data resources. The data we obtained originally existed in the MRT (Multi-Threaded Routing Toolkit) format, and subsequently, we utilized the Python tool PyBGPstream (2.0.2) to deeply parse the raw data and convert them into a more analyzable CSV format.

When collecting the corresponding data, we identify the time and region of the route leak events to choose the correct collector. This guarantees that we can conduct further analysis more effectively using the data we have collected.

The route data we obtain contains a vast amount of information, but not all of it is useful for our analysis. Therefore, based on the raw data, we carefully select a series of key information, including the collector, dump time (dump_time), peer address (peer_dress), peer autonomous system number (peer_asn), and fields. These selected pieces of information will play a crucial role in subsequent analysis and model training.

Before delving into the analysis of the converted data, we first perform data cleansing to remove erroneous or redundant information from the BGP raw messages. For instance, loops in the path, message withdrawals, or duplicate transmissions can negatively impact the accuracy and training speed of our model. Thus, data cleansing is an essential step to ensure the quality of the model training.

To conduct network characteristic analysis more effectively, we abstract the process of network transmission into a directed topological graph. In this graph, each AS is regarded as a node in the network topology, and the paths between adjacent ASes are abstracted as edges connecting these nodes. In this way, we can construct the BGP network topology over a period of time, providing a solid foundation for further network characteristic analysis.

2.2. Graph Feature Extraction

When anomalies such as route leaks occur, they result in a significant deviation in the network's routing information from its state prior to the incident. To identify such events, we analyze graph characteristics for signs of route leaks.

Firstly, we capture the evolving state of the network by reading path data at set time intervals—every 2 min. This data collection process is facilitated by the networkx (3.2.1) tool, a powerful utility for constructing and visualizing complex network topologies.

Subsequently, we delve into the meticulously crafted AS network structure topology graph. From this graph, we extract features related to structural robustness and centrality measures [12]. These features provide the analytical foundation for the machine learning model that will be introduced later in the paper.

The extraction of these features is an effective method of transforming network behavior into quantifiable metrics. Each feature serves as an important metric for examining the network's response to anomalies, providing invaluable assessments for the detection of route leaks.

2.2.1. Structural Robustness Characteristics

1. Number of nodes: The total number of all nodes in the graph.
2. Number of edges: The total number of all edges in the graph.
3. Diameter: The eccentricity of a vertex is used to denote the greatest distance between that vertex and all other vertices in the graph. The diameter of a graph represents the maximum value taken over the eccentricities of all vertices.
4. Assortativity [13]: If nodes with a high degree in a graph tend to connect with other high-degree nodes, the network exhibits assortative mixing; conversely, if they tend to connect with low-degree nodes, it indicates disassortative mixing. The assortativity coefficient represents the relationship between connected node pairs and is calculated as follows:

$$r = \frac{\sum_{xy} xy(e_{xy} - a_x b_y)}{\sigma_a \sigma_b} \tag{1}$$

   where r represents the assortativity coefficient of the network. e_{xy} is the number of edges between nodes x and y, a_x and b_y are the degrees of nodes x and y, respectively, and σ_a and σ_b are the standard deviations of the degrees of nodes a and b in the network.
5. Eigenvector centrality: The importance of a node based on both the number of its neighboring nodes and the importance of its neighbors. Assuming x_i represents the importance of a node, c is a proportionality constant, and the calculation of eigenvector centrality is as follows:

$$EC(i) = x_i = c \sum_{j=1}^{n} a_{ij} x_j \tag{2}$$

6. Algebraic connectivity: The second smallest eigenvalue of its Laplacian matrix.

7. Effective graph resistance [14]: A measure used to calculate the sum of effective resistances between all pairs of vertices, assuming the network is treated as an electrical circuit, and the effective resistance between two vertices in the network graph can be calculated through series and parallel operations.
8. Symmetry ratio [15]: The ratio of the number of distinct eigenvalues of the graph to $D + 1$, where D is the diameter.
9. Natural connectivity [16]: Derived from its spectrum as the average eigenvalue of the graph. The calculation method is as follows:

$$\bar{\lambda} = ln\left(\frac{1}{N}\sum_{i=1}^{N} e^{\lambda_i}\right) \tag{3}$$

where N is the number of nodes in the graph, λ_i represents the i-th eigenvalue of the graph.
10. Node connectivity: Equals the minimum number of nodes that must be removed to disconnect the graph.
11. Weighted spectral radius [17]: Defined as the normalized sum of N cycles. The calculation method is as follows:

$$W(G, N) = \sum_{i}(1 - \lambda_i)^N \tag{4}$$

12. Percolation threshold (or percolation limit): The critical proportion of nodes that need to be removed before the graph disintegrates.
13. Number of spanning trees [18]: Can be expressed as a function of the unweighted Laplacian eigenvalues.

$$\xi = \frac{1}{n}\prod_{i=2}^{n} \lambda_i \tag{5}$$

14. Average global efficiency [17]: The average efficiency of all node pairs.
15. Bipartite clustering coefficient [19]: A metric of local connection density:

$$c_u = \frac{\sum_{v \in N(N(u))} c_{uv}}{|N(N(u))|} \tag{6}$$

where $N(N(u))$ is the set of neighbors of the neighbors of node u. c_{uv} is the bipartite clustering coefficient between nodes u and v, where v is a neighbor of u.
16. Number of triangles: The count of triangles that include a node as one of their vertices.

Structural robustness features are concerned with the stability and resilience of the network topology graph in the face of anomalies or attacks. These features usually reflect the overall structure and connectivity strength of the network, such as node degree distribution, clustering coefficient, and path length. This feature is chosen because it reflects the structural changes of the network before and after an anomaly occurs, which is important for detecting anomalies such as route leakage.

2.2.2. Centrality Measurement Feature

1. Harmonic centrality [20]: The sum of the reciprocal of the shortest path distances from all other nodes to node u.
2. Degree of node: The degree of node u, which is the number of edges adjacent to that node.
3. Centrality of node degrees: The proportion of nodes it is connected to.
4. Square clustering coefficient [21]: The square clustering coefficient refers to the score of possible squares that exist on each node, calculated as follows:

$$C_4(v) = \frac{\sum_{u=1}^{k_v} \sum_{w=u+1}^{k_v} q_v(u,w)}{\sum_{u=1}^{k_v} \sum_{w=u+1}^{k_v} [a_v(u,w) + q_v(u,w)]} \tag{7}$$

where $q_v(u,w)$ represents the number of common neighbors between nodes u and w that are also neighbors of node v (i.e., the number of squares formed). $a_v(u,w)$ indicates the presence of an edge between nodes u and w. If there is an edge, $(a_v(u,w) = 1)$; otherwise, if there is no edge, $(a_v(u,w) = 0)$.

5. Mean degree of neighborhood: $N(i)$ is the neighbor of the node and k_j is the degree of the node belonging to $N(i)$. The calculation method is as follows:

$$k_{nn,i} = \frac{1}{|N(i)|} \sum_{j \in N(i)} k_j \tag{8}$$

6. Mediation centrality of a node: The fraction of all pairs of the shortest path by all pairs of nodes v.
7. Proximity of a node: The distance from that node to all other nodes in the graph.
8. Page Rank: Calculated based on the structure of the incoming links.
9. Number of clusters: The maximum number of clusters at each node in the Graph.
10. Eccentricity: The maximum distance from you to all other nodes in the graph from the eccentricity of node u.

Centrality metric features are used to assess the importance of nodes or edges in a network. Common centrality metrics include degree centrality, proximity centrality, median centrality, etc. In a BGP network topology graph, these features can help to identify critical nodes or paths that may play an important role in anomalies such as route leakage.

By conducting feature extraction on the AS network structure topology graph, we save the extracted features into a CSV file. Labels are then assigned to each graph based on the time period it represents: if it is before the start of a route leak event, the label is set to 0; otherwise, the label is set to 1. At the same time, we conduct a time complexity analysis on these features, as shown in Table 1.

Table 1. Complexity analysis of graph features.

Feature	Time Complexity	Spatial Complexity
Number of nodes	O(N)	O(1)
Number of edges	O(N)	O(1)
Diameter	$O(N^3)$	$O(N^3)$
Assortativity	$O(N^3)$	$O(N^3)$
Eigenvector centrality	$O(N^3)$	$O(N^3)$
Algebraic connectivity	$O(N^3)$	$O(N^3)$
Effective graph resistance	$O(N^3)$	$O(N^3)$
Symmetry ratio	$O(N^3)$	$O(N^3)$
Natural connectivity	$O(N^3)$	O(N)
Node connectivity	$O(N^3)$	O(N)
Weighted spectral radius	$O(N^3)$	O(N)
Percolation threshold	$O(N^3)$	O(N)
Number of spanning trees	$O(N^3)$	O(N)
Average global efficiency	$O(N^3)$	O(N)
Bipartite clustering coefficient	$O(N^3)$	O(N)
Number of triangles	$O(N^3)$	O(N)
Harmonic centrality	$O(N^3)$	O(N)
Centrality of node degrees	O(N)	O(1)
Mean degree of neighborhood	$O(N^3)$	O(N)
Proximity of a node	$O(N^3)$	O(N)
Number of clusters	$O(N^3)$	O(N)
Degree of node	O(N)	O(1)
Square clustering coefficient	$O(N^3)$	O(N)
Mediation centrality	$O(N^3)$	O(N)
Page Rank	$O(N^3)$	O(N)
Eccentricity	$O(N^3)$	O(N)

2.3. Weighted Voting Machine Learning Model

Traditional machine learning methods [5] often rely on a single model for predictions, which can lead to overfitting issues, where the model performs well on the training set but poorly on unseen data. The underlying reason is that a single model is too simplistic to capture the underlying features of the data, resulting in a model with low generalization capability. To address this, this paper proposes a weighted voting model based on machine learning.

2.3.1. Model Structure

In previous research on BGP route leak detection, models typically relied on a single base learner. However, not all models maintain good performance across different datasets. It is common for a model to achieve high accuracy in predicting one event but perform poorly in another. To address this issue, this paper employs an ensemble approach, combining multiple base learners into a single model, as depicted in Figure 2. By integrating the results from these base learners and feeding them into our designed Meta Learner, the model can capture a broader spectrum of BGP routing information. This approach leads to more accurate judgments and mitigates the risks of overfitting and underfitting.

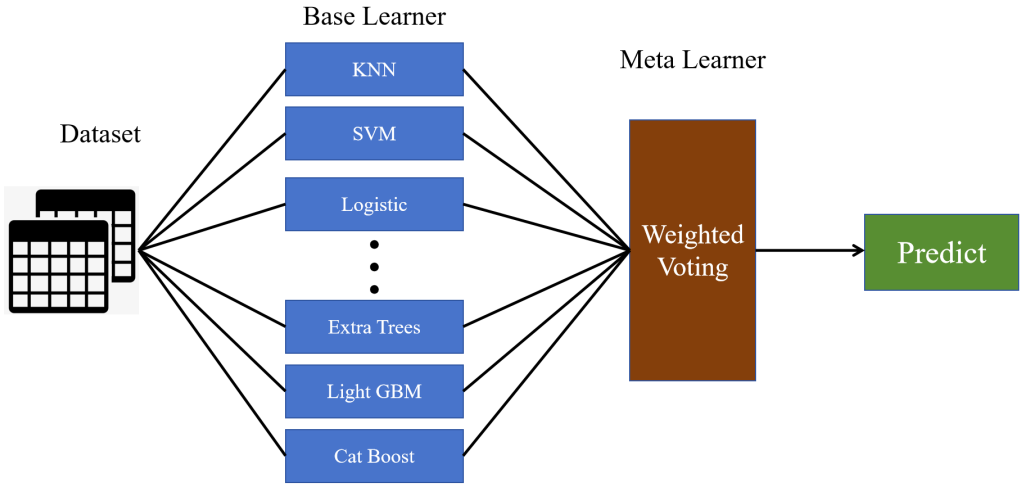


Figure 2. Schematic diagram of weighted voting model.

During the training process of the model, we first train several base learners based on the graph features calculated from the BGP network topology diagram as described in the previous section. These base Learners have different biases and variances. To balance the errors made by individual base learners, we have integrated the results of its training to some extent, and assigned a certain weight to each one, denoted as w_i.

This weight reflects the importance of each model, and models that perform better during training may be assigned higher weights. Based on the weights and the prediction results of the models, we can obtain a final predicted result C, which is calculated as follows:

$$C = argmax(\sum_{i=1}^{N} w_i * p_i(j)) \tag{9}$$

where $p_i(j)(j = 1, 2, 3, \cdots, K)$ represents the confidence level of each model's output for the occurrence of a route leak.

This ensemble method can analyze the weights and confidence levels of each submodel to understand which models contribute more to the final prediction, facilitating the modification and optimization of the model.

2.3.2. Weight Determination

For the weighted voting model proposed in this paper, the determination of the model weights is of paramount importance; appropriate weights can significantly enhance the model's accuracy. To this end, we will employ genetic algorithms [22] to ascertain the w_i weights used within this study. Genetic algorithms, inspired by the process of natural selection, are powerful optimization techniques that can effectively search for optimal or near-optimal solutions within complex spaces. By leveraging genetic algorithms, our aim is to meticulously adjust the weights assigned to each model's vote, thereby maximizing the predictive accuracy of the ensemble model. This methodology enables adaptive and dynamic weight adjustments that consider the performance of each model within the ensemble, ensuring a more robust and responsive predictive system. The process of the genetic algorithm is depicted in Figure 3.

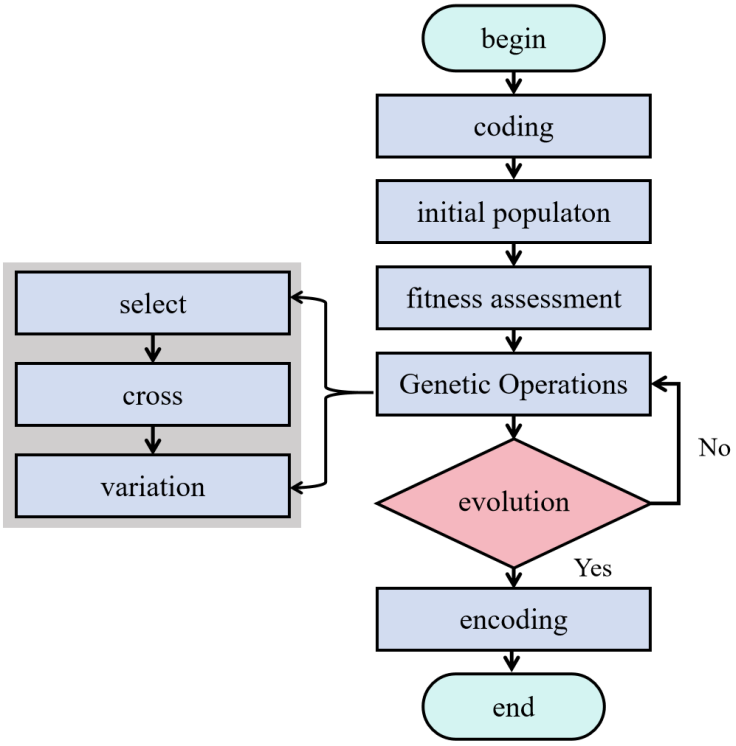


Figure 3. Genetic algorithms.

Firstly, the initialization of the population is carried out. We set the weights of the models to be equal. This step ensures that all models have an equal opportunity to participate at the initial stage. Initializing the weights of each model to the same value means that no model is given an advantage or disadvantage in the early stages of the algorithm, thereby ensuring the fairness and diversity of the algorithm.

$$\mathbf{w} = [w_1, w_2, ..., w_n] \tag{10}$$

where w_i represents the weight of the i-th model.

The fitness function measures the effectiveness of the model fusion strategy, reflecting the consistency between the model predictions and the actual labels through accuracy, providing a standard for the genetic algorithm to evaluate the quality of individuals. The fitness function $F(\mathbf{w})$ is typically defined as the accuracy of the model ensemble after fusion using the weight vector $\mathbf{w}$. The calculation formula for the fitness function is

$$F(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} I(\text{vote}(\mathbf{w}, \mathbf{p}_i) = y_i) \tag{11}$$

where N is the number of test samples, p_i is the prediction result vector of the i-th sample, y_i is the true label, vote($\mathbf{w}$, $\mathbf{p}_i$) is the weighted voting function, and I is the indicator function.

The selection process chooses individuals from the current population to produce offspring based on fitness. The roulette wheel strategy used in this paper is as follows: the selection probability is allocated according to the individual's fitness. The higher the fitness of an individual, the greater the chance of being selected, which helps the spread and accumulation of excellent genes in the population.

$$P(\mathbf{w}) = \frac{F(\mathbf{w})}{\sum_j F(w_j)} \tag{12}$$

Crossover operation is used to generate new individuals by exchanging part of the genes of two parent individuals. Through crossover, new individuals with parental characteristics can be produced, thereby increasing the genetic diversity of the population, which is conducive to exploring the solution space and finding better solutions. To achieve better weight values, this paper employs the method of multi-point crossover [23], which helps to escape from local optima and obtain better weight values.

Assuming the parents are w_1 and w_2, and the offspring are w'_1 and w'_2, we will select n crossover points, dividing the parents into $n+1$ intervals. For each interval k ($k = 1, 2, \ldots, n+1$), the following operations are performed:

- If k is odd, the gene segment from w_1 between crossover points p_{k-1} and p_k is added to w'_1.
- If k is even, the gene segment from w_2 between crossover points p_{k-1} and p_k is added to w'_1.
- Corresponding operations are performed to form w'_2 by selecting the remaining genes from w_1 and w_2 that were not chosen for w'_1.

The solution method is as follows:

$$w'_1 = w_1[0, p_1) + w_2[p_1, p_2) + \cdots + w_1[p_n, L] \tag{13}$$

$$w'_2 = w_2[0, p_1) + w_1[p_1, p_2) + \cdots + w_2[p_n, L] \tag{14}$$

Mutation operation increases the diversity of the population by randomly changing some genes of individuals. Mutation is another key operation in the genetic algorithm, which introduces randomness to prevent the algorithm from converging too early to a local optimal solution. The mutation operation ensures that there is always new genetic information in the population, providing the possibility for the algorithm to jump out of the current solution space and explore new areas. In this paper, we have adopted Gaussian mutation [24], also known as normal distribution mutation. Mutation is carried out by adding a normally distributed random perturbation. Specifically, each gene w_i becomes

$$w'_i = w_i + \mathcal{N}(0, \sigma^2) \tag{15}$$

where w is the weight after mutation, which is a normally distributed random variable with a mean of 0 and a variance of σ^2.

3. Result

3.1. Data Selection

In this paper, we conduct experiments using data from four BGP route leak incidents involving TTNet, Indosat, TM, and AWS [25,26]. The data for these events were obtained from a global BGP open data project, which has multiple monitoring points, each of which can capture routing information for a country or region. In this way, we can capture all routing messages from a particular area as much as possible, thereby constructing a large-scale BGP routing network and conduct further analysis on this basis. Below is an introduction to these routing leak events:

- The TTNet Event, also known as the "AS9121 routing table leak incident", occurred on 24 December 2004. Around 9:20 Greenwich Mean Time, AS9121 began announcing prefixes to its peers and continued past 10:00. AS9121 announced through BGP sessions to other ASes, affecting about 70% of all prefixes (impacting over 106,000 prefixes at the time), resulting in the serious consequence of tens of thousands of network data being either lost or misrouted.
- The Indosat Event, also known as the "AS4761 routing table leak incident", occurred on 2 April 2014, at 18:25 UTC. At the time of the incident, the global routing table contained nearly 500,000 routes, and AS4761 leaked approximately 320,000 routes. The reason why the originating prefixes were not allocated to Indosat was due to incorrect upstream filtering that led to the redistribution of Border Gateway Protocol routing data. This unintentional error significantly impacted various route collectors by increasing the declared IP prefixes.
- The TM Event, also known as the "AS4788 routing table leak incident", involved AS4788 leaking one-third of the global routing table IP prefixes to the backbone network provider Level3 (AS3549). The incident was triggered by a router configuration error at Malaysia Telecom, starting on 12 June 2015, at 8:43 UTC and lasting until 11:45 UTC. Level3 (AS3549) propagated traffic from peers and customers through Malaysia Telecom, which could not handle the traffic, leading to severe packet loss and performance degradation. The performance degradation between the Asia–Pacific region and the rest of Level3's network was particularly noticeable.
- The AWS Event, also known as the "AS200759 routing table leak incident", began on 22 April 2016, at 17:10 UTC. The incident affected a large number of ASes and prefixes, such as Google, Amazon, and Twitter, which have high-traffic prefixes and were severely impacted, causing a significant loss of traffic and connections. The incident occurred due to a maintenance issue with Innofield AG (AS200759) connected to the Swiss Internet Exchange (SwissIX). Innofield AG usually announces an IPv4 and IPv6 prefix to SwissIX. During the maintenance reactivation of the Border Gateway Protocol session, AS200759 assigned Amazon's prefixes to a private AS65021. The prefix announcement was propagated through AS6939 Hurricane Electric (HE), which is a peer with SwissIX. This redirected traffic through Hurricane Electric to the private autonomous system, thereby impairing the reachability of Amazon's autonomous system.

These incidents have been widely utilized by researchers [27]. For each event, we select data from one hour before the incident and one hour after the incident to serve as our training and testing datasets.

3.2. Feasibility Analysis

In order to demonstrate the feasibility of the graph-based method used in this paper, we will take the TTNet event as an example for explanation. To more intuitively show the relationship between graph features and routing leaks, we divide the images into multiple ones based on the order of magnitude of the calculated features.

In Figure 4, it can be observed that after the occurrence of the routing leak, features such as number_of_nodes, number_of_edgestriangles, harmonic_centrality, and average_neighbor_degree have undergone significant changes. Similarly, in Figures 5–7, many features have changed to varying degrees after the event.

From the line charts, it can be seen that the majority of graph features in the TTNet event have experienced noticeable changes in their values after the routing leak occurred. Therefore, using graph features as a basis for classification models for anomaly detection is feasible.

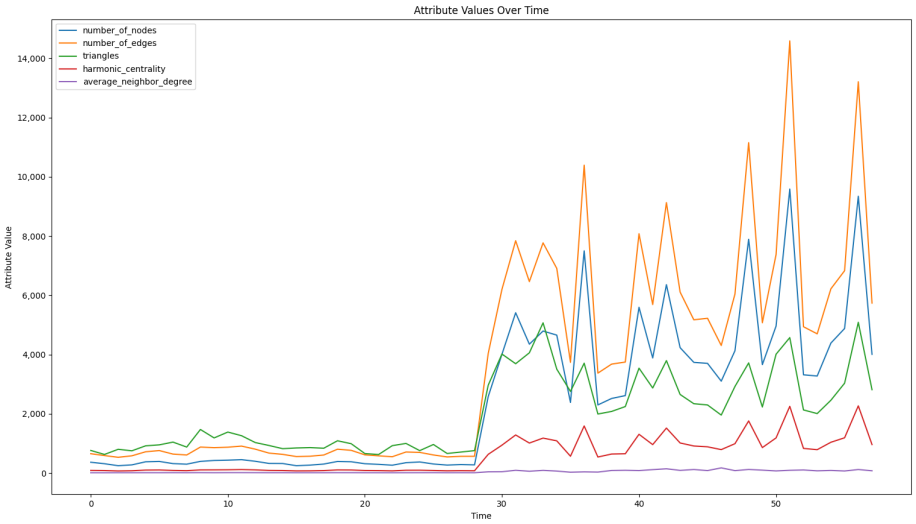


Figure 4. TTNet feature line chart 1.

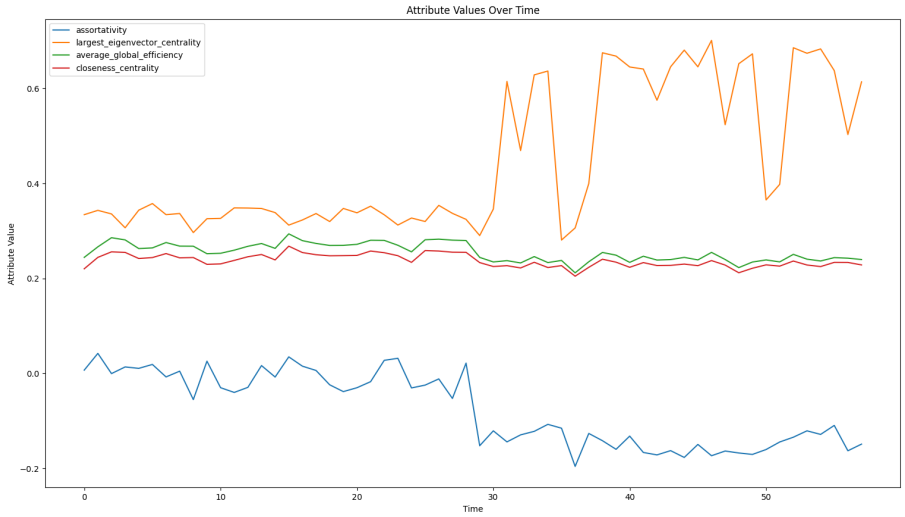


Figure 5. TTNet feature line chart 2.

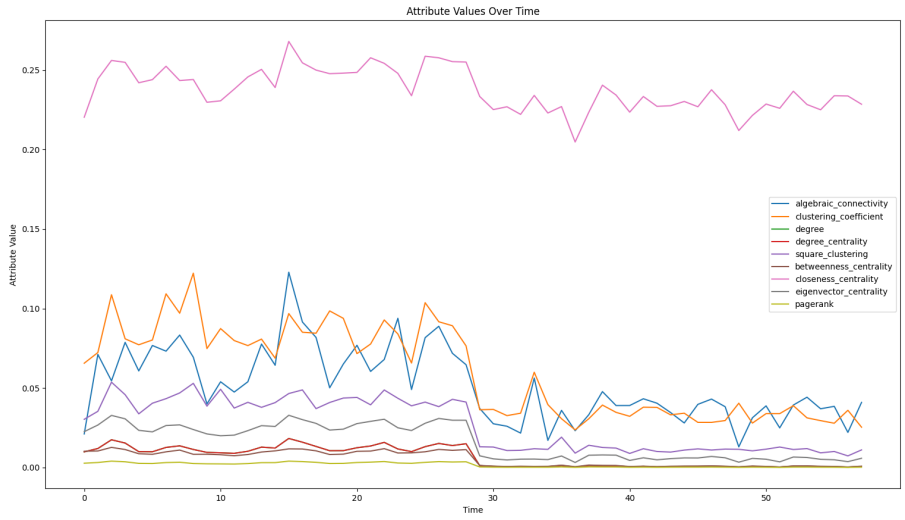


Figure 6. TTNet feature line chart 3.

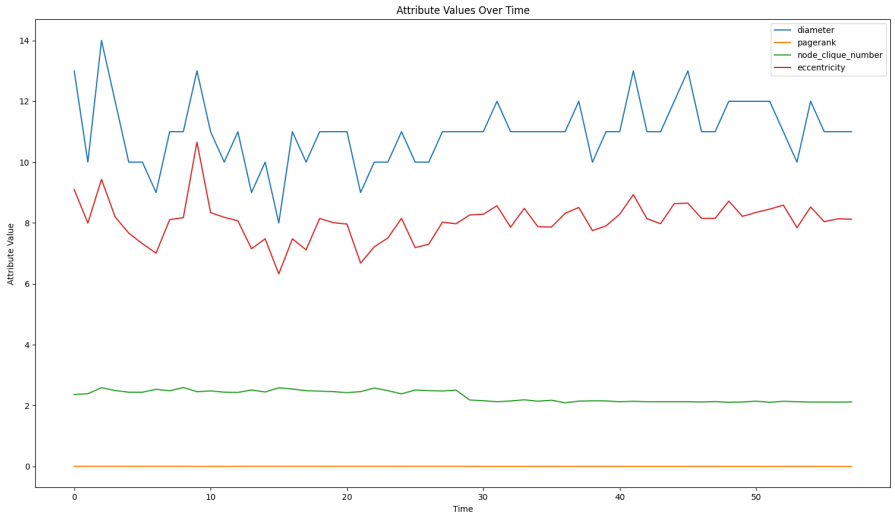


Figure 7. TTNet feature line chart 4.

3.3. Complexity Analysis

The complexity of feature extraction mainly depends on the size of the graph (number of nodes N and number of edges M). The computation of many features involves the global properties of the graph, such as diameter, intermediary centrality, proximity, etc., and the computational complexity of these features is usually $O(N^3)$. However, the computational complexity of most features is $O(N^2)$ and below.

The time complexity of the weighted voting algorithm is $O(K * N)$, where K is the number of models, and N is the number of test samples. The space complexity is O(K), as it is necessary to store the weights. The time complexity of the genetic algorithm for optimizing weights is high because it involves multiple iterations and selection, crossover, and mutation operations. The time complexity of each iteration is $O(P * Q)$, where P is the population size, and Q is the length of the individual (i.e., the number of models). The space complexity is $O(P * Q)$ because it is required to store all individuals in the population.

In general, in addition to the fact that the genetic algorithm involves multiple iterations and complex genetic operations, which increases a certain computational complexity, the overall computational complexity of the algorithm is low and shows good performance.

3.4. Training Set Partition

To more effectively evaluate the performance of the model proposed in this paper, we have employed the Leave-One-Out cross-validation (LOOCV) method [28]. For the four BGP route leak events examined in this paper, we first select one as the validation set, with the remaining samples serving as the training set. For instance, in Dataset1, we let TTNet, Indosat, and TM serve as the training set, with AWS as the test set. The same treatment is applied to other datasets, as shown in Table 2. Through this cycle, each sample is used once as the validation set. This method not only maximizes the use of available data but also ensures that each event is used exactly once for validation, allowing us to accumulate a diverse set of performance metrics. Collectively, these metrics provide a more nuanced and reliable assessment of the model's predictive accuracy.

After completing the cycle through all events, the individual performance scores obtained from each iteration are aggregated to calculate an average score. This final score serves as a comprehensive indicator of the model's overall performance, reflecting its consistency and reliability when faced with different BGP route leak scenarios.

In determining the parameters of the sub-models, we employed the grid search method. We also utilized the grid search method during the training process of the sub-models to train suitable parameters based on the requirements of each model. We then used the models with the optimal parameters obtained to further refine our weighted voting model. The specific parameter settings are shown in Table 3.

Table 2. LOOCV train data division.

Data Name	Train	Test
Data1	TTNet + Indosat + TM	AWS
Data2	TTNet + TM + AWS	Indosat
Data3	TTNet + Indosat + AWS	TM
Data4	Indosat + TM + AWS	TTNet

Table 3. Parameter grid for different models.

Model	Parameter	Value Range
Logistic Regression	penalty	{l1, l2}
	C	10^{-4} to 10^{4} (log scale)
SVM	C	10^{-4} to 10^{4} (log scale)
	gamma	10^{-4} to 10^{4} (log scale)
	kernel	{rbf, poly, sigmoid}
Decision Tree	max_depth	{None, 10, 20, 30, 40, 50}
	min_samples_split	{2, 5, 10}
Random Forest	n_estimators	{50, 100, 200}
	max_depth	{None, 10, 20, 30, 40, 50}
	min_samples_split	{2, 5, 10}
KNN	n_neighbors	1 to 30
	weights	{uniform, distance}
Gradient Boosting Trees	n_estimators	{50, 100, 200}
	learning_rate	{0.01, 0.1, 0.2}
	max_depth	{3, 4, 5, 6, 7, 8}
Extra Trees	n_estimators	{50, 100, 200}
	max_depth	{None, 10, 20, 30, 40, 50}
	min_samples_split	{2, 5, 10}
LightGBM	n_estimators	{50, 100, 200}
	max_depth	{3, 4, 5, 6, 7, 8}
	learning_rate	{0.01, 0.1, 0.2}
CatBoost	n_estimators	{50, 100, 200}
	learning_rate	{0.01, 0.1, 0.2}
	depth	{3, 4, 5, 6, 7, 8}

3.5. Experimental Results

To demonstrate the performance of the model used in this paper, we have selected several common machine learning models for comparison, including Logistic Regression, SVM, Decision Tree, Random Forest, KNN, Gradient Boosting Trees, Extra Trees, LightGBM, CatBoost, and ours. These models are also used as the base learners in our weighted voting model. Table 4 presents the results of the experiment.

The data in the table indicate that the weighted voting model proposed in this paper has outperformed most existing models in terms of accuracy and can consistently maintain a high accuracy rate across different datasets. Specifically, in Data2, although the overall correctness is relatively low, the accuracy of our model is higher than all models, including its base learners. This result not only proves the effectiveness of our model design but also demonstrates the advantage of integrating multiple algorithms when dealing with complex problems. Compared to methods that use a single model, the weighted voting model proposed in this paper has a significantly higher average accuracy across different datasets and exhibits greater generalizability.

In order to demonstrate the performance of our model, we compared it with other BGP route leakage detection methods, as shown in Table 5. It can be observed that our model outperforms the existing models in terms of accuracy. Regarding the model itself, after using grid search for the parameter optimization of sub-models, the model's accuracy is significantly higher than before the optimization. This highlights the superiority of our model approach.

In practical applications, data often come with uncertainty and variability. Our model effectively reduces the risks associated with this uncertainty by integrating predictions from multiple models. This robustness ensures that our model maintains a high level of accuracy in different application scenarios and data distributions, thereby providing a reliable solution to real-world problems.

Table 4. Comparison of model accuracies across different datasets (the bold font indicates that the model's prediction results are optimal).

Model	Data1	Data2	Data3	Data4	Average
Logistic	**98.04**%	72.72%	**96.49**%	72.93%	85.00%
SVM	72.55%	70.91%	49.12%	60.00%	63.14%
Decision Tree	86.27%	89.09%	94.74%	50.91%	80.25%
Random Forest	94.12%	90.91%	85.96%	81.82%	88.20%
KNN	**98.04**%	89.09%	71.92%	47.27%	76.58%
Gradient Boosting Trees	88.24%	90.91%	87.71%	65.45%	62.60%
Extra Trees	98.04%	89.09%	80.70%	72.73%	85.60%
Light GBM	96.08%	69.09%	45.61%	**87.27**%	74.51%
Cat Boost	92.16%	72.73%	40.35%	76.36%	70.37%
Ours	96.08%	**92.72**%	92.98%	**87.27**%	**92.75**%

Table 5. Comparison results with other route leakage detection methods (the bold font indicates that the model's prediction results are optimal).

Methods	Accuracy
SVM-HMM [10]	86.10%
Allahdadi et al. [29]	92.16%
Auto-Encoder [30]	60.00%
SVM-BGPAD [31]	91.36%
Edwards et al. [32]	65.20%
Ding et al. [33]	68.60%
Ours	88.58%
Ours + grid search	**92.75**%

Additionally, during the experimental process, we deployed the monitoring projects built using the aforementioned models to a local environment. By setting up the time and detection points to collect and process data, we achieved the functionality of real-time monitoring. At the same time, we added a detection alarm function. When the monitoring process makes multiple consecutive judgments that a route leak has occurred, we can determine that a route leak has happened during the current time period. To better verify our results, we conducted simulation experiments, the results of which are shown in the Figure 8.

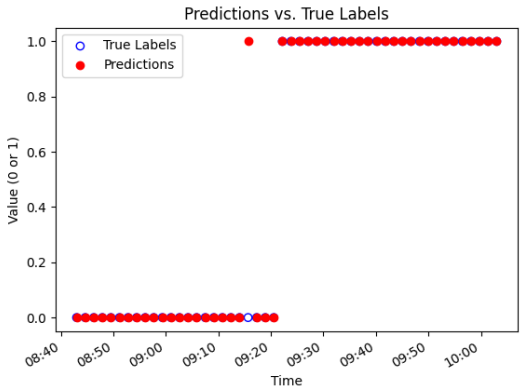


Figure 8. Simulation experiment results.

4. Conclusions

This paper proposes a weighted voting model based on existing machine learning methods to conduct BGP route leak detection. The method combines multiple base learners to enhance the model's accuracy and effectively improve its generalizability. By analyzing the original routing information of the network, this paper abstracts the relationships between ASs into a network topology structure diagram. Based on this, structural robustness features and centrality measurement features are extracted, reflecting the changes in the BGP network topology before and after the occurrence of route leaks through the variation in these graph features. Experimental results indicate that graph features can serve as a basis for judgment in BGP route leak detection. By comparing with other related techniques, the superiority of this method is demonstrated. Furthermore, by integrating the trained model with real-world scenarios, a real-time monitoring function for BGP route leaks has been realized. This function can provide early warnings for the phenomenon of route leaks.

In our work, we have achieved certain improvements in experimental accuracy, but there are still some limitations. In future work, future research will mainly focus on the following two aspects: On the one hand, we will attempt to integrate our model with real-world scenarios, not limited to the real-time monitoring within a certain range in this paper, but aimed at large-scale, multi-regional routing networks. On the other hand, in practical network operation contexts, merely detecting the occurrence of a leak is insufficient. Our future research will delve into methods for rapidly pinpointing the source of the leak and identifying its underlying causes.

Author Contributions: Conceptualization, C.S.; methodology, C.S., X.L. and P.Z.; software, C.S. and R.W.; validation, C.S. and R.W.; investigation, C.S.; resources, C.S. and R.W.; data curation, R.W.; writing—original draft preparation, C.S.; writing—review and editing, X.L. and P.Z.; supervision, P.Z., K.L. and L.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Al-Musawi, B.; Branch, P.; Armitage, G. BGP anomaly detection techniques: A survey. *IEEE Commun. Surv. Tutor.* **2016**, *19*, 377–396. [CrossRef]
2. Roughan, M.; Willinger, W.; Maennel, O.; Perouli, D.; Bush, R. 10 Lessons from 10 years of measuring and modeling the internet's autonomous systems. *IEEE J. Sel. Areas Commun.* **2011**, *29*, 1810–1821. [CrossRef]
3. Jia, J.; Yan, Z.-w.; Geng, G.-g.; Jian, J. Study on bgp route leak. *Chin. J. Netw. Inf. Secur.* **2016**, *2*, 54–61.
4. Guo, X.; Chen, D.; Chen, J.; Xiong, Z. Research on predicting trusted business relationships in autonomous system based on neural networks. In Proceedings of the Third International Conference on Green Communication, Network, and Internet of Things (CNIoT 2023), Chongqing, China, 14–16 July 2023; Volume 12814, pp. 545–550.
5. Zhang, L.; Cui, Y.; Liu, J.; Jiang, Y.; Wu, J. Application of machine learning in cyberspace security research. *Chin. J. Comput.* **2018**, *41*, 1943–1975.
6. Li, J.; Dou, D.; Wu, Z.; Kim, S.; Agarwal, V. An internet routing forensics framework for discovering rules of abnormal bgp events. *ACM SIGCOMM Comput. Commun. Rev.* **2005**, *35*, 55–66. [CrossRef]
7. de Urbina Cazenave, I.O.; Köşlük, E.; Ganiz, M.C. An anomaly detection framework for bgp. In Proceedings of the 2011 International Symposium on Innovations in Intelligent Systems and Applications, Istanbul, Turkey, 15–18 June 2011; pp. 107–111.
8. Witten, I.H.; Frank, E.; Hall, M.A.; Pal, C.J. Data mining practical machine learning tools and techniques. *ACM Sigmod Rec. 2017*, *31*, 76–77. [CrossRef]
9. Zhang, L.; Qiu, H.; Guo, Y.; Duan, H. A threat perception method for inter-domain routing systems based on weighted similarity. *Sci. Sin. Inform.* **2017**, *47*, 878–890. [CrossRef]
10. Al-Rousan, N.M.; Trajković, L. Machine learning models for classification of bgp anomalies. In Proceedings of the 2012 IEEE 13th International Conference on High Performance Switching and Routing, Belgrade, Serbia, 24–27 June 2012; pp. 103–108.
11. Li, Y.; Xing, H.-J.; Hua, Q.; Wang, X.-Z.; Batta, P.; Haeri, S.; Trajković, L. Classification of bgp anomalies using decision trees and fuzzy rough sets. In Proceedings of the 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), San Diego, CA, USA, 5–8 October 2014; pp. 1312–1317.

12. Rueda, D.F.; Calle, E.; Marzo, J.L. Robustness comparison of 15 real telecommunication networks: Structural and centrality measurements. *J. Netw. Syst. Manag.* **2017**, *25*, 269–289. [CrossRef]
13. Newman, M.E. Mixing patterns in networks. *Phys. Rev. E* **2003**, *67*, 026126. [CrossRef]
14. Ellens, W.; Spieksma, F.M.; Van Mieghem, P.; Jamakovic, A.; Kooij, R.E. Effective graph resistance. *Linear Algebra Its Appl.* **2011**, *435*, 2491–2506. [CrossRef]
15. Dekker, A.H.; Colbert, B. The symmetry ratio of a network. In Proceedings of the 2005 Australasian Symposium on Theory of Computing, Newcastle, Australia, 1 January 2005; Volume 41, pp. 13–20.
16. Jun, W.; Barahona, M.; Yue-Jin, T.; Hong-Zhong, D. Natural connectivity of complex networks. *Chin. Phys. Lett.* **2010**, *27*, 078902. [CrossRef]
17. Fay, D.; Haddadi, H.; Uhlig, S.; Moore, A.W.; Mortier, R.; Jamakovic, A. *Weighted Spectral Distribution*; Technical Reports; University of Cambridge, Computer Laboratory: Cambridge, UK, 2008.
18. Ellens, W.; Kooij, R.E. Graph measures and network robustness. *arXiv* **2013**, arXiv:1311.5064
19. Latapy, M.; Magnien, C.; Del Vecchio, N. Basic notions for the analysis of large two-mode networks. *Soc. Netw.* **2008**, *30*, 31–48. [CrossRef]
20. Boldi, P.; Vigna, S. Axioms for centrality. *Internet Math.* **2014**, *10*, 222–262. [CrossRef]
21. Lind, P.G.; Gonzalez, M.C.; Herrmann, H.J. Cycles and clustering in bipartite networks. *Phys. Rev. E—Stat. Nonlinear, Soft Matter Phys.* **2005**, *72*, 056127. [CrossRef]
22. Holland, J.H. Genetic algorithms. *Sci. Am.* **1992**, *267*, 66–73. [CrossRef]
23. De Jong, K.A.; Spears, W.M. A formal analysis of the role of multi-point crossover in genetic algorithms. *Ann. Math. Artif. Intell.* **1992**, *5*, 1–26. [CrossRef]
24. Zhang, X.; Xu, Y.; Yu, C.; Heidari, A.A.; Li, S.; Chen, H.; Li, C. Gaussian mutational chaotic fruit fly-built optimization and feature selection. *Expert Syst. Appl.* **2020**, *141*, 112976. [CrossRef]
25. Cosovic, M.; Obradovic, S.; Junuz, E. Deep learning for detection of bgp anomalies. In *Time Series Analysis and Forecasting: Selected Contributions from ITISE 2017*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 95–113.
26. Yan, H.; Oliveira, R.; Burnett, K.; Matthews, D.; Zhang, L.; Massey, D. Bgpmon: A real-time, scalable, extensible monitoring system. In Proceedings of the 2009 Cybersecurity Applications & Technology Conference for Homeland Security, Washington, DC, USA, 3–4 March 2009; pp. 212–223.
27. Sanchez, O.R.; Ferlin, S.; Pelsser, C.; Bush, R. Comparing machine learning algorithms for bgp anomaly detection using graph features. In Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks, Orlando, FL, USA, 9 December 2019; pp. 35–41.
28. Vehtari, A.; Gelman, A.; Gabry, J. Practical bayesian model evaluation using leave-one-out cross-validation and waic. *Stat. Comput.* **2017**, *27*, 1413–1432. [CrossRef]
29. Allahdadi, A.; Morla, R.; Prior, R. A Framework for BGP Abnormal Events Detection. *arXiv* **2017**, arXiv:1708.03453.
30. McGlynn, K.; Acharya, H.B.; Kwon, M. Detecting BGP route anomalies with deep learning. In Proceedings of the IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Paris, France, 29 April–2 May 2019; pp. 1039–1040.
31. Dai, X.; Wang, N.; Wang, W. Application of machine learning in BGP anomaly detection. *J. Phys. Conf. Ser.* **2019**, *1176*, 032015. [CrossRef]
32. Edwards, P.; Cheng, L.; Kadam, G. Border gateway protocol anomaly detection using machine learning techniques. *SMU Data Sci. Rev.* **2019**, *2*, 5.
33. Ding, Q.; Li, Z.; Batta, P.; Trajković, L. Detecting BGP anomalies using machine learning techniques. In Proceedings of the 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Budapest, Hungary, 9–12 October 2016; pp. 3352–3355.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.