Learning with Semantics: Towards a Semantics-Aware Routing Anomaly Detection System

Yihao Chen[†], Qilei Yin[‡], Qi Li[*‡], Zhuotao Liu[*‡], Ke Xu[♮‡],
Yi Xu[*‡], Mingwei Xu[*‡], Ziqian Liu[§], Jianping Wu[♮‡]

[†]*Department of Computer Science and Technology & BNRist, Tsinghua University*
[‡]*Zhongguancun Laboratory*
[*]*Institute for Network Sciences and Cyberspace, Tsinghua University*
[♮]*Department of Computer Science and Technology, Tsinghua University*
[§]*China Telecom*

Abstract

BGP is the de facto inter-domain routing protocol to ensure global connectivity of the Internet. However, various reasons, such as deliberate attacks or misconfigurations, could cause BGP routing anomalies. Traditional methods for BGP routing anomaly detection require significant manual investigation of routes by network operators. Although machine learning has been applied to automate the process, prior arts typically impose significant training overhead (such as large-scale data labeling and feature crafting), and only produce uninterpretable results. To address these limitations, this paper presents a routing anomaly detection system centering around a novel network representation learning model named BEAM. The core design of BEAM is to accurately learn the unique properties (defined as *routing role*) of each Autonomous System (AS) in the Internet by incorporating BGP semantics. As a result, routing anomaly detection, given BEAM, is reduced to a matter of discovering unexpected routing role churns upon observing new route announcements. We implement a prototype of our routing anomaly detection system and extensively evaluate its performance. The experimental results, based on 18 real-world RouteViews datasets containing over 11 billion route announcement records, demonstrate that our system can detect all previously-confirmed routing anomalies, while only introducing at most five false alarms every 180 million route announcements. We also deploy our system at a large ISP to perform real-world detection for one month. During the course of deployment, our system detects 497 true anomalies in the wild with an average of only 1.65 false alarms per day.

1 Introduction

The Border Gateway Protocol (BGP) is the de facto inter-domain routing protocol to achieve global connectivity in the Internet. BGP establishes Internet-wide routing paths by exchanging route announcements among the networks operated by different organizations, referred to as Autonomous Systems (ASes). Each route announcement carries AS-level path information for reaching certain prefixes (*i.e.,* a block of IP addresses). At its steady state, every AS learns an AS-path to reach every globally routable Internet prefix. Despite its global adoption, BGP itself has no built-in authentication mechanism. As a result, a misbehaved AS can announce arbitrary routes in the Internet, either due to deliberate attacks or misconfigurations. These bogus routes form serious threats to routing security, namely BGP hijacking (*i.e.,* forcing certain traffic to go through a malicious AS) and BGP route leaks (*i.e.,* redirecting traffic over unintended links). Over the past decade, the Internet has witnessed several severe BGP incidents. For example, a Swiss company leaked over 70,000 routes in 2019 [1] and a British company hijacked more than 31,000 prefixes in 2021 [2]. A cryptocurrency platform recently confirmed the loss of $1.9 million after a BGP hijacking attack [3]. Although several security extensions have been proposed to counter these threats, *e.g.,* BGPsec [4], psBGP [5] and S-BGP [6], they are not widely deployed, possibly due to incompatibility with the current Internet architecture. Besides, while RPKI [7] has gained traction in providing authoritative information about IP prefix ownership, its effectiveness is largely limited by the incomplete deployment of ROV [8]. More importantly, RPKI is not designed to mitigate route manipulation attacks or route leaks.

Detecting routing anomalies in the global Internet is the first step towards secure Internet routing. The community has proposed significant research in this regard [9–18]. However, they typically rely on extensive analysis of routing data from multiple sources. More crucially, these methods require non-trivial human supervision to produce reasonable results. The advance in Machine Learning (ML) motivates the community to apply ML techniques to automate and simplify anomaly detection by recognizing different patterns of route announcements [19–29]. However, existing methods require large datasets with manual labels and/or handcrafted features, imposing significant overheads on data collection and model update. Moreover, many of these methods learn deep latent features for classification, producing largely uninterpretable results. As a consequence, these methods provide limited prac-

tical guidance for network operators to fix routing anomalies.

To address these challenges, we present a routing anomaly detection system centering around a novel network representation learning model, BEAM (**B**GP s**E**m**A**ntics aware network e**M**bedding). Instead of learning any latent or opaque features, BEAM enables interpretable and accurate routing anomaly detection based on the intrinsic routing characteristics of ASes that are derived from the *domain specific knowledge of BGP semantics*. Specifically, we propose the concept of *AS routing role* to meaningfully characterize ASes in BGP route announcements. The design of routing role is derived from the AS business relationship graph (rather than any handcrafted features), because an AS's business relationship with its neighboring ASes determines how the AS chooses to update the route announcements received from neighbors, and how the newly generated route announcements are further propagated [30]. Given accurate modeling of ASes' routing roles, anomaly detection is reduced to a matter of detecting unexpected AS routing role churns from the original route to the new route announcement.

The key design challenges in obtaining routing roles are two-fold. First, the available dataset of route announcements could contain non-trivial noises due to the unrevealed routing anomalies in the Internet [31]. For instance, AS 4134 leaked over 70,000 routes in 2019 [1] and AS 55410 hijacked more than 31,000 prefixes in 2021 [2]. As a result, computing routing roles directly from the noisy route announcement dataset (using either raw announcements or statistical features) could lead to high false positive rates [20, 24, 25]. Second, due to the dynamism and scale of Internet routing, AS routing roles are evolving over time.

To address the above challenges, BEAM employs a novel embedding mechanism to learn an embedding vector for each AS based on the AS graph constructed from AS relationships. The key of BEAM's embedding is to preserve an AS's proximity and hierarchy properties that are essential to its routing role. The exact definitions of proximity and hierarchy are given in §3.2. The embedding vectors are further employed to uniquely represent and interpret the routing roles of ASes, based on which our routing anomaly detection system reports routing anomalies upon observing abnormal routing role churns. Further, we design our learning mechanism to ensure that the embedding vectors can properly capture routing roles despite the ever-changing Internet routing and topology.

We validate our system on 18 real-world route announcement datasets collected from global vantage points[1]. The entire datasets include over 11 billion route announcement records spanning from 2008 to 2021. The experimental results show that BEAM produces interpretable results regarding AS routing role changes, based on which our system correctly identifies all previously-confirmed routing anomalies, while only incurring at most five false alarms every 180 million route announcements (about 1.61 false alarms per day). We also deploy our system at a large ISP to detect routing anomalies over a one-month period. We further visualize AS routing roles to achieve interpretable routing behavior analysis. Our work can serve as a complement to existing BGP security extensions, such as RPKI, to protect against both BGP hijacking and BGP route leaks.

[1] A vantage point is a BGP participant (*e.g.,* a router) that provides public access to its routing table and/or its received route announcements.

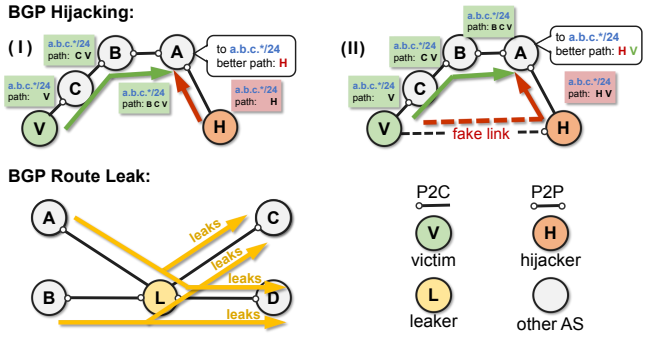

Figure 1: **Illustrations of BGP anomalies.** In BGP hijacking, the adversary can either (I) falsely claim the ownership of a prefix, or (II) announce a fake yet more preferable route. In BGP route leak, routes are propagated to unintended ASes.

To summarize, our contributions are four-fold:

- We design the first BGP semantics aware network representation learning model, BEAM, that accurately captures the routing roles of ASes.
- We develop an unsupervised routing anomaly detection system based on BEAM, which achieves real-time detection in an interpretable manner without requiring labeled routing data or feature engineering.
- We validate our system by conducting experiments on 18 RouteViews datasets with more than 11 billion route announcement records. Our system can detect all confirmed anomalies with minor false alarms.
- We deploy our system at a large ISP to collect real-world detection results for a month. The system detects 497 true anomalies in the wild from over 150 million live route announcements, with only 1.65 daily false alarms on average.

2 Background

Inter-domain Routing Protocol. The Internet contains over 73,000 advertised Autonomous Systems (ASes) as of Jun 2023. Each AS consists of several networks under the management of the same organization and is identified by a uniquely allocated non-negative integer called AS Number (ASN). BGP is the de facto inter-AS routing protocol to achieve global connectivity of the Internet. BGP is a path-vector routing protocol that maintains AS-level path information, which gets updated as BGP announcements propagate in the network. Upon receiving a BGP announcement, an AS, following

its *routing policy*, may stop further propagating the announcement, or append its ASN to the AS-path and send the updated announcement to a *selective* set of neighbors.

Business relationship largely determines one AS's routing policy [30, 32]. Two neighboring ASes typically have three types of business relationships[2]: provider-to-customer (P2C), peer-to-peer (P2P) and customer-to-provider (C2P), where a customer AS pays its provider for connectivity while two peering ASes forward traffic to each other free of charge. Thus, the inter-domain routing system of the Internet can be reconstructed as an AS graph based on AS relationships. This AS-level topology exhibits hierarchy [30], with several well recognized Tier-1 (large-scale) ASes. However, the topology is not strictly hierarchical and is flattening over time [34].

BGP Anomalies. Although widely deployed, BGP lacks built-in authentication, *i.e.,* one AS can broadcast virtually arbitrary BGP announcements to disrupt the security and reliability of Internet routing. BGP anomalies can be classified into two categories: hijacking and route leak, as illustrated in Fig. 1. BGP hijacking itself has two subcategories: (i) falsely claiming the ownership of a prefix or (ii) announcing fake paths (usually more preferable than real paths) to prefixes. The first type of hijacking is solvable by Route Origin Validation (ROV) [7], which is experiencing gradual deployment. Yet, the second type of hijacking usually needs per-hop path validation protocols such as BGPsec [4] that has very limited deployment. Also, BGP hijacking can be classified as *prefix hijacking* (targeting a prefix of others) or *subprefix hijacking* (targeting a subset of others' prefix, *i.e.,* subprefix).

The other category of BGP anomalies is route leak: a misbehaved AS propagates BGP announcements to another AS in violation of the intended policies, resulting in traffic forwarded through unintended links. The Gao-Rexford model [30] describes the restrictions on BGP route propagation and can be used to identify BGP route leak (*i.e.,* the valley-free criterion). For example, in 2019, AS 21217 (*Safe Host*) broke the valley-free criterion by propagating announcements received from its providers (*e.g.,* AS 13237 (*euNetworks GmbH*)) to another provider AS 4134 (*China Telecom*), redirecting large amounts of Internet traffic destined for European mobile networks through *China Telecom* [1].

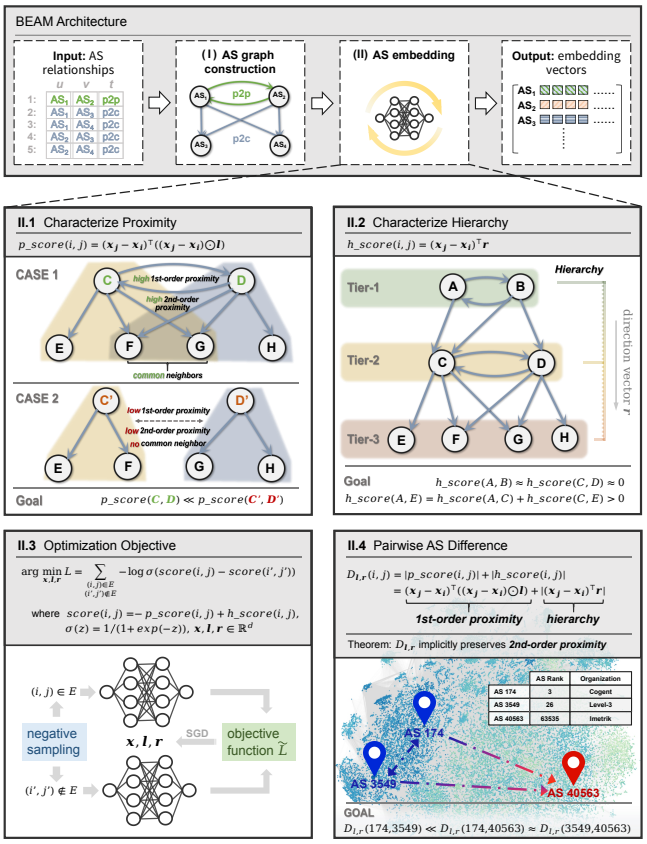

Figure 2: **Learning AS routing roles via BEAM.** BEAM takes AS relationships as the input and outputs the embedding vectors that represent AS routing roles. (**II.1**) BEAM characterizes the proximity between ASes by *p_score*. If two ASes are directly connected and have the same business relationships with many common neighbors, their proximity tends to be high, *i.e.,* a lower *p_score*. (**II.2**) BEAM characterizes the hierarchy among ASes by *h_score*. If an AS must traverse multiple consecutive P2C links to reach another AS, their hierarchy difference should be large, *i.e.,* a higher *h_score*. (**II.3**) BEAM utilizes one joint objective to optimize both *p_score* and *h_score*; negative sampling is also applied. (**II.4**) The function $D_{\boldsymbol{l},\boldsymbol{r}}$ measures the routing-role difference between two ASes. A higher $D_{\boldsymbol{l},\boldsymbol{r}}$ value means higher difference.

3 Semantics Aware Analysis

3.1 BEAM Overview

We propose a novel network representation learning model, BEAM, to learn the routing roles of ASes. The routing roles meaningfully characterize the ASes in BGP route announcements and are utilized to detect Internet routing anomalies. As shown in Fig. 2, BEAM takes the AS business relationships as the input and generates the embedding vector for each AS by (i) constructing an AS graph, and (ii) performing AS embedding. To compute AS embedding vectors, BEAM designs distance functions to measure two routing characteristics of the ASes: the proximity (see Fig. 2(II.1)) and the hierarchy (see Fig. 2(II.2)). The exact definitions of the two characteristics are given in §3.2. We design a dedicated optimization objective such that BEAM preserves both the proximity and the hierarchy through the embedding (see Fig. 2(II.3)). After obtaining the optimized embedding vectors, we can further measure the pairwise AS difference in terms of their routing roles (see Fig. 2(II.4)).

[2]We ignore complex AS relationships [33]. They are much more unusual and play a minor role in defining the overall routing behavior of an AS.

To our best knowledge, BEAM is the first dedicated network representation learning model that fully integrates BGP semantics into the training process and enables meaningful and accurate representation of ASes' routing roles. Applying network representation learning, rather than using "raw" AS business relationships, is essential to characterize ASes' routing roles. In particular, our network representation learning model can capture the global routing characteristics of each AS and translate them into embedding vectors, while the original AS business relationships can only indicate the local routing policy between two directly connected ASes. Further, with the embedding vectors, we can quantify the difference in routing roles between *any* pair of ASes, regardless of whether they are connected or not. This enables us to detect the unexpected AS routing role churns in the global Internet that capture routing anomalies.

3.2 Model Formulation

Definition 1 (AS Graph)**.** An AS graph is a directed graph $G = (V, E)$ where each vertex $v \in V$ represents an AS and each directed edge $e = (u, v) \in E$ represents a P2C relationship from u to v.

We regard each unique AS (identified by its ASN) as a vertex, and the P2C relationship between two vertices as a directed edge from provider to customer. Accordingly, a C2P relationship is viewed as a reversed P2C one, and a P2P relationship is represented via two edges in opposite directions.

We define two types of AS proximity to represent the similarity between two ASes according to their local connections/relationships with their neighbors.

Definition 2 (First-Order AS Proximity)**.** The first-order proximity between two ASes is their pairwise connection in the AS graph. For a pair of vertices (u, v), the first-order proximity between them is 1 if they are connected by an edge $e = (u, v) \in E$; otherwise it is 0.

Definition 3 (Second-Order AS Proximity)**.** The second-order proximity between two ASes is the similarity between their neighborhood network structures. Given vertices u, v, let $\boldsymbol{p_u} = \langle \boldsymbol{w}_{u,1}, \ldots, \boldsymbol{w}_{u,|V|} \rangle$ denote the first-order proximity of AS u with other ASes, the second-order proximity between u and v is quantified based on the consistency between $\boldsymbol{p_u}$ and $\boldsymbol{p_v}$.

While the concept of proximity has been proposed before [35], BEAM is the first to extend its interpretation to BGP semantics. As illustrated in Fig. 2(II.1), ASes C and D have high first-order proximity due to the P2P relationship (*i.e.,* two edges in the AS graph), and also high second-order proximity since they provide Internet transit services for a similar set of customers. We also confirm our interpretation via a real-world example: AS 8903 and AS 12541, owned by cloud service provider *Evolutio*, serve as each other's backup and hence have very similar routing roles. In terms of proximity, they have 25 common customers and their neighbor AS sets are highly similar (with a Jaccard index of 86.2%); thus, the proximity is consistent with routing role similarity.

As discussed in §2, the Internet topology exhibits hierarchy. Typically, a provider AS is considered on a higher level than its customers. Thus, we define AS hierarchy as follows:

Definition 4 (AS Hierarchy)**.** The hierarchy of an AS is its tendency to establish P2C relationship with other ASes. For two vertices u, v, if there exists a directed edge $e = (u, v)$, the hierarchy from u to v is positive.

In real world, AS 7018 (*AT&T*) lies on the top of the Internet since it establishes either P2C or P2P relationship with other ASes, while AS 140061 (*China Telecom*) is a stub AS without any customers. Thus, AS 7018 and AS 140061 have very different routing roles in BGP, which is aligned with the positive hierarchy between them.

To quantify the AS proximity and hierarchy, we embed ASes into low-dimensional representations.

Definition 5 (AS Embedding)**.** Given $G = (V, E)$, AS embedding is to map each vertex $v \in V$ into a low-dimensional vector space $\mathbb{R}^d$, *i.e.,* learn a mapping function $f_{G;\theta} : V \to \mathbb{R}^d$, where θ contains learnable parameters and $d \ll |V|$.

Finally, we define our BEAM model as follows:

Definition 6 (BGP Semantics Aware Network Embedding)**.** Given AS relationships, the BEAM model constructs the AS graph $G = (V, E)$ and performs AS embedding, such that the embedding vectors $\boldsymbol{x} = \{\boldsymbol{x_v} | v \in V, \boldsymbol{x_v} = f_{G;\theta}(v)\}$ preserve the first- and second-order proximity and the hierarchy of ASes.

3.3 AS Graph Construction

The first step of training BEAM is to construct the AS graph. We use the real-world CAIDA AS relationship dataset [36][3] to construct the AS graph G. A business relationship between two ASes can be denoted as a tuple (u, v, t), where u and v are two ASNs and $t \in \{\text{P2P}, \text{P2C}, \text{C2P}\}$ refers to the relationship type. For each tuple (u, v, t) in the CAIDA dataset, if $t = \text{P2C}$, we add a directed edge $e = (u, v)$ into E. If $t = \text{C2P}$, we add a directed edge $e = (v, u)$ into E. And if $t = \text{P2P}$, we add two directed edges $e = (u, v)$ and $e' = (v, u)$ into E.

The rationale for using AS business relationship to construct the AS graph is that it primarily determines how an AS chooses to update the routing paths received from neighbors, and how the new generated route announcements are propagated [30]. Hence, it has direct impacts on ASes' routing roles. Moreover, the AS business relationship is a relatively stable property determined by real-world commercial agreements between connected ASes. Thus, it is challenging to impersonate a specific AS without simultaneously changing or faking multiple AS relationships. Besides, unlike historical route announcement data, the AS business relationship dataset contains fewer incorrect entries [34].

[3] CAIDA is not the only source of AS business relationships. Other sources like TopoScope [37] are also available for training our system.

3.4 AS Embedding

With a constructed AS graph, we embed ASes into a vector space while preserving proximity and hierarchy. To this end, we design two distance functions to measure the difference between ASes regarding proximity and hierarchy, respectively.

Proximity Distance. The proximity distance, indicated by *p_score*, models the first-order proximity between ASes: a small distance between two ASes means that their proximity is large. Per Def. 2, given a pair of directly linked vertices (u,v) and another pair of vertices (u',v') without a direct edge, *p_score* should reflect the difference between their first-order proximity, *i.e.,* $p\_score(u,v) < p\_score(u',v')$. Therefore, we define *p_score* as follows:

$$p\_score(u,v) = (\boldsymbol{x_v} - \boldsymbol{x_u})^\intercal \big((\boldsymbol{x_v} - \boldsymbol{x_u}) \odot \boldsymbol{l}\big), \tag{1}$$

where $\boldsymbol{x_u}, \boldsymbol{x_v} \in \mathbb{R}^d$ denote the embedding vectors of u,v, respectively. $\boldsymbol{l} \in \mathbb{R}^d$ is a learnable weight vector for the d components. $\intercal$ and $\odot$ denote matrix transpose and Hadamard product, respectively. Intuitively, $\boldsymbol{l}$ projects the embedding vectors into a subspace intended for preserving the proximity. To explicitly preserve the first-order proximity, BEAM learns $\boldsymbol{x}, \boldsymbol{l}$ by decreasing the *p_score* of two vertices with an edge, while increasing the *p_score* of two vertices that are not directly connected. Since this training strategy also preserves the second-order proximity implicitly (elaborated later in this section), we do not define a dedicated distance function for the second-order proximity.

Hierarchy Distance. The hierarchy distance, indicated by *h_score*, quantifies the hierarchical difference between ASes. Per Def. 4, given a pair of vertices (u,v) where $(u,v) \in E, (v,u) \notin E$, and another pair of vertices (u',v') without a directed edge, *i.e.,* $(u',v') \notin E$, the *h_score* should reflect the difference between their hierarchy, *i.e.,* $h\_score(u,v) > h\_score(u',v')$. To this end, we design *h_score* as follows:

$$h\_score(u,v) = (\boldsymbol{x_v} - \boldsymbol{x_u})^\intercal \boldsymbol{r}, \tag{2}$$

where $\boldsymbol{x_u}, \boldsymbol{x_v} \in \mathbb{R}^d$ denote the embedding vectors of u,v, respectively. $\boldsymbol{r} \in \mathbb{R}^d$ is a learnable unit vector indicating the descending direction of hierarchy. Intuitively, $\boldsymbol{r}$ projects the embedding vectors into a subspace intended for preserving the hierarchy, and thus the *h_score* calculates the projected length of $\boldsymbol{x_v} - \boldsymbol{x_u}$ on the specific direction vector $\boldsymbol{r}$ such that it has two important properties, *i.e.,* $h\_score(u,v) = -h\_score(v,u)$ and $h\_score(u,v) = h\_score(u,w) + h\_score(w,v)$. To explicitly preserve the hierarchy of ASes, BEAM learns $\boldsymbol{x}, \boldsymbol{r}$ by increasing the *h_score* of two vertices with a directed edge (*i.e.,* a P2C relationship), while decreasing *h_score* of two vertices not directly connected. Since ASes with P2P relationship is connected by two edges in opposite directions, their *h_score* would approach zero under this training strategy, which is consistent with the BGP semantics that two peering ASes are typically on the same hierarchy of the Internet.

Training Objective. With the two distance functions, we design the training objective of BEAM to ensure that a trained BEAM preserves both proximity and hierarchy. We consolidate the two distance functions as follows:

$$score(u,v) = -p\_score(u,v) + h\_score(u,v). \tag{3}$$

Since a small *p_score* means large proximity and a large *h_score* means large hierarchy, we subtract *p_score* in Eq.(3) so that *score* increases monotonically with the difference between ASes in terms of proximity and hierarchy. This design allows BEAM to preserve the two routing characteristics better. Although *score* may become zero in some cases, it will not affect BEAM 's training since the training objective is to enlarge the difference of *score* between observed edges and nonexistent edges. Per Def. 2 and 4, given an observed edge (u,v) and a nonexistent edge (u',v'), our model should assign $score(u,v) > score(u',v')$ with the optimal $\boldsymbol{x}$, $\boldsymbol{l}$ and $\boldsymbol{r}$. To this end, we formulate the optimization problem as follows:

$$\arg\min_{\boldsymbol{x},\boldsymbol{l},\boldsymbol{r}} L = \sum_{\substack{(u,v)\in E \\ (u',v')\notin E}} -\log \sigma(score(u,v) - score(u',v')), \tag{4}$$

where E is the edge set of the AS graph, $\sigma(z) = \frac{1}{1+exp(-z)}$ is the sigmoid function, and L is the objective function to be minimized. We solve this problem via a fully connected neural network, which has an embedding layer and two linear layers. The embedding layer generates the embedding vectors ($\boldsymbol{x}$) that represent ASes' routing roles and the two linear layers project the embedding vectors into two subspaces that preserve proximity ($\boldsymbol{l}$) and hierarchy ($\boldsymbol{r}$), respectively. For each edge $(u,v) \in E$, we sample 10 negative (nonexistent) edges $(u',v') \notin E$ and each $((u,v),(u',v'))$ forms one training instance. The neural network generates the embedding vectors of u and v, computes the loss via Eq.(4), and uses SGD [38] to optimize itself. When the training is complete, the neural network learns the ASes' routing roles. We empirically set $d = 128$ and train the network for 1,000 epochs. The batch size is 1024 and the initial learning rate is 10^{-5}.

Computing Pairwise AS Difference. The pairwise AS difference represents their difference in the routing roles, which we define between the embedding vectors of two ASes using the BEAM model (including its parameters $\boldsymbol{x}$, $\boldsymbol{l}$, and $\boldsymbol{r}$):

$$\begin{aligned} D_{\boldsymbol{l},\boldsymbol{r}}(u,v) &= |p\_score(u,v)| + |h\_score(u,v)| \\ &= \underbrace{(\boldsymbol{x_v} - \boldsymbol{x_u})^\intercal ((\boldsymbol{x_v} - \boldsymbol{x_u}) \odot \boldsymbol{l})}_{\textit{the first-order proximity}} + \underbrace{|(\boldsymbol{x_v} - \boldsymbol{x_u})^\intercal \boldsymbol{r}|}_{\textit{the hierarchy}}. \end{aligned} \tag{5}$$

Note that this pairwise AS difference directly reflects the first-order proximity and the hierarchy between ASes. Moreover, per the definition of the second-order proximity (Def. 3), the pairwise AS difference between two vertices should be small if their neighbors and the business relationships with their neighbors are similar. In Appendix A, we prove a theorem

that this pairwise AS difference does preserve the second-order proximity between ASes. Thus, our BEAM model can preserve the first-order proximity, the second-order proximity and the hierarchy between ASes.

3.5 Embedding Results Analysis

We train BEAM with the CAIDA AS relationship dataset collected on 06/01/2018 to study the routing roles of ASes. The dataset contains 61,549 ASes and 439,981 business relationships, and is randomly selected from the CAIDA datasets collected before historical BGP incidents (see §5).

Visualizing Embedding Vectors. We visualize the embedding vectors computed by BEAM in a 3-D space to illustrate the overall characteristics of AS routing roles. Since BEAM learns the unit direction vector $\boldsymbol{r}$ that represents the descending direction of hierarchy, we decompose each embedding vector into two parts: the projection on $\boldsymbol{r}$, and the projection on the plane orthogonal to $\boldsymbol{r}$ (*i.e.,* the rejection). We use the length of the projection as the coordinate value of the Z-axis to represent the hierarchy level of each AS. We further transform the rejection into a 2-D space by the widely used dimension reduction method t-SNE [39], and hence obtain the coordinate values of the X-axis and the Y-axis.

We visualize all embedding vectors in Fig. 3(A), where each vertex represents a unique AS and the color of the vertex indicates the AS hierarchy level. We observe that a few vertices are densely located (*i.e.,* high proximity) on the highest and the lowest levels of the Internet, while more medium-level vertices are sparsely distributed (*i.e.,* low proximity). For a better illustration, we show the terrain plot of the same 3-D space in Fig. 3(B) via Inverse Distance Weight [40], which is an interpolation metric widely used for estimating the spatial distribution of scatter points. The observations are consistent with the BGP fact that a small number of ASes on the highest hierarchy level have similar routing roles, since they all provide transit services for other ASes and establish P2P relationships with the ASes on the same level to form "Internet backbone". On the contrary, it takes multiple C2P links for another small set of ASes to reach the backbone ASes. Thus, they lie on the lowest hierarchy level, forming several clusters with lower Z-axis coordinate values. The rest ASes are located on the medium hierarchy. They have different providers and customers, and serve diverse routing roles.

Routing Role Analysis. We further study the routing roles of specific ASes to check if the computed embedding vectors reflect their actual routing properties in the Internet. In particular, we choose 16 Tier-1 ASes (*e.g., AT&T*), 9 major yet not Tier-1 ASes (*e.g., Telstra*), and 50 other randomly selected stub ASes (*i.e.,* the ASes connected to only one other AS). We present the projections of their routing roles on both YZ-plane (Fig. 3(C)) and XY-plane (Fig. 3(D)). The results show that the Tier-1 ASes form dense clusters on the highest hierarchy level, the major yet not Tier-1 ASes form several dense clusters on the levels relatively lower than those of Tier-1 ASes, and the stub ASes are dispersed on the much lower levels. This confirms that the computed embedding vectors preserve both the proximity and hierarchy of ASes. Figure 3(C), however, does not exhibit distinct hierarchies, which is expected since the Internet topology is not strictly hierarchical [34]; for example, AS 4134 (*China Telecom*) has P2P relationship with the Tier-1 AS 1299 (*Arelion*), yet it also keeps a route with 2 consecutive C2P links to reach AS 1299.

Pairwise AS Difference Comparison. We measure the difference of routing roles between two ASes. We choose AS pairs in three categories: (i) We randomly sample a subset S_0 from all pairs of ASes with the P2P relationship while enforcing the following constraint. In particular, given an AS pair, we obtain the neighbor AS set of each AS in the pair. The AS pair is eligible only if the Jaccard index (*i.e.,* the similarity) of the two neighbor AS sets is between 0% and 10%. We also sample another subset S_1 without enforcing this constraint. (ii) We randomly sample six subsets (denoted as N_0 to N_5) from all pairs of ASes that are not connected directly, where the Jaccard indices of the two neighbor AS sets are between 0% and 10%, 0% and 20%, 20% and 40%, 40% and 60%, 60% and 80%, and 80% and 100%, respectively. (iii) We randomly sample seven subsets (denoted as H_0 to H_6), where the shortest path between two ASes in one pair is a direct P2P link, and 2, 3, 4, 5, and 6 consecutive P2C links, respectively. For instance, if two AS pairs (*e.g.,* AS1 and AS2, AS2 and AS3) are both in the P2C relationship, then AS1 and AS3 form two consecutive P2C links. Please see Appendix B for a summary of these sampled subsets.

For each sampled subset, we compute the difference of routing roles between two ASes and obtain Cumulative Distribution Function (CDF) curves. We make three types of comparisons among the CDF curves of different subsets. The comparisons of the P2P AS pairs (S_0), P2C AS pairs (H_0) and no-relationship AS pairs (N_0) show that S_0 has the smallest difference and N_0 exhibits the largest difference (see Fig. 3(E)). It reveals that BEAM preserves the first-order proximity. The comparisons among the subsets with different degrees of neighbor intersection (N_1 to N_5) show that if two ASes have more common neighbors (*i.e.,* high second-order proximity), they tend to have similar routing roles (see Fig. 3(F)). In Fig. 3(G), the difference of routing roles increases with the number of consecutive P2C links between two ASes, which is consistent with the nature of AS hierarchy.

4 The Anomaly Detection System

In this section, we develop a semantics aware routing anomaly detection system built upon BEAM[4].

[4]Our system is open source at this GitHub repository.

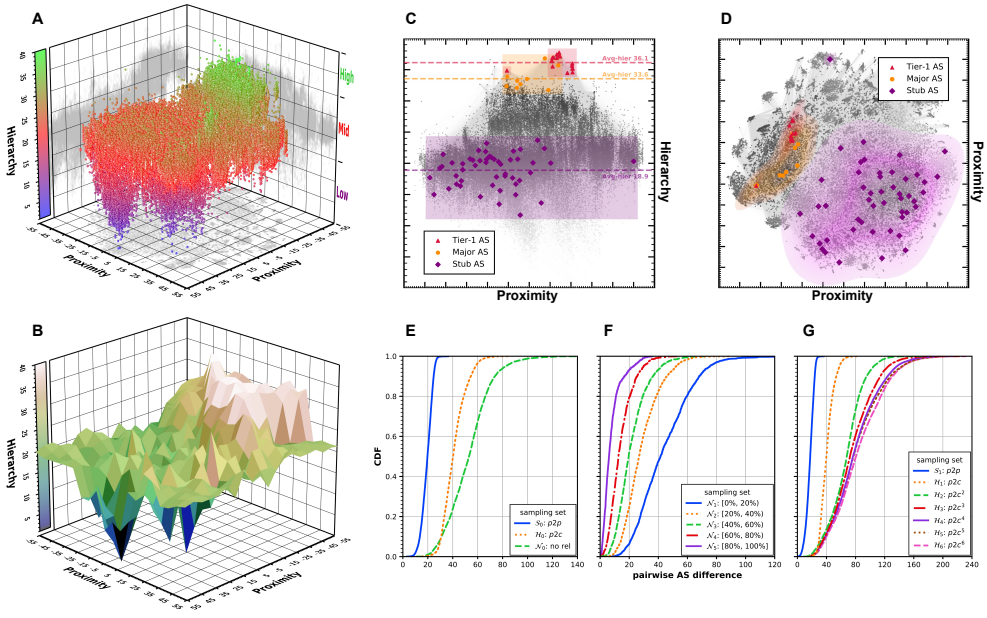

Figure 3: **Embedding results.** (**A**) All embedding vectors visualized in a 3-D space. The Z-axis represents the hierarchy level and the XY-plane reflects the proximity. (**B**) The terrain plot of the embedding vectors. The estimated spatial distribution shows the overall characteristics of AS routing roles. (**C**) The YZ-plane projection of the embedding vectors. The X-axis and the Y-axis of the projection plane indicate the proximity and the hierarchy, respectively. The areas with different colors illustrate the distribution of typical ASes (*i.e.,* vertices with different colors), and the dashed lines show their average hierarchy levels. (**D**) The XY-plane projection of the embedding vectors. (**E**) The routing role difference regarding the sampled subset S_0, H_0 and N_0. (**F**) The routing role difference regarding the sampled subset N_1 to N_5. (**G**) The routing role difference regarding S_1 and H_1 to H_6.

4.1 System Overview

Our routing anomaly detection system is designed to detect global-scale Internet routing anomalies. As shown in Fig. 4, our detection system consists of three components: the routing monitor, the BEAM engine, and the anomaly detector. The routing monitor establishes connections with global vantage points to capture route changes in real time. The BEAM engine utilizes our BEAM model to compute the *path difference scores* of the route changes, which quantifies the routing role difference between the new and the original routing paths. Based on the path difference scores, the anomaly detector identifies suspicious route changes, and groups the suspicious route changes sharing the same prefix into anomalous prefix events. This is necessary to identify the prefixes impacted by widespread routing anomalies. It further locates the ASes responsible for each anomalous prefix event and correlates the events caused by the same set of responsible ASes. Since the BEAM model used in our detection system is pretrained with AS relationship data (instead of labeled routing anomaly data), our detection is unsupervised.

4.2 BEAM Engine

Not all route changes are caused by routing anomalies. For instance, different ASes of the same organization may claim the ownership of a specific prefix simultaneously, thus generating two routing paths with different origin ASes. This is called multiple origin AS (MOAS) [41] and should not be considered as anomalous. Thus, the BEAM engine relies on *path difference score* to identify suspicious route changes.

To compute the path difference score for a route change, we design a method based on the dynamic time warping (DTW) algorithm [42], an effective way of measuring the overall difference between two ordered sequences of unequal length. Specifically, given two routing paths $S = \langle v_1, \ldots, v_m \rangle$ and $S' = \langle v'_1, \ldots, v'_n \rangle$, the DTW algorithm repeatedly selects a pair of ASes from S and S', and generates eligible sequences of AS pairs by satisfying the following conditions: (i) Each AS in S (S') should be paired with one or more ASes from S' (S). (ii) The first (last) AS in S should be paired with the first (last) AS in S'. (iii) Given $i < j$, if v_i and v_j from S are paired with v'_k and v'_l in S', then there keeps $k \leq l$; S and S' are symmetric. For example, if $S = \langle v_1, v_2, v_3 \rangle$, $S' = \langle v'_1, v'_2, v'_3 \rangle$, then $((v_1, v'_1), (v_2, v'_2), (v_3, v'_3))$ and $((v_1, v'_1), (v_1, v'_2), (v_2, v'_2), (v_3, v'_3))$ are both eligible, but $((v_1, v'_1), (v_1, v'_2), (v_2, v'_1), (v_3, v'_3))$ is ineligible because the two pairs (v_1, v'_2) and (v_2, v'_1) violate the condition (iii). For each eligible sequence, we sum the pairwise AS difference (given by BEAM) of its AS pairs. Then, we choose the minimum value from all summed values as the path difference

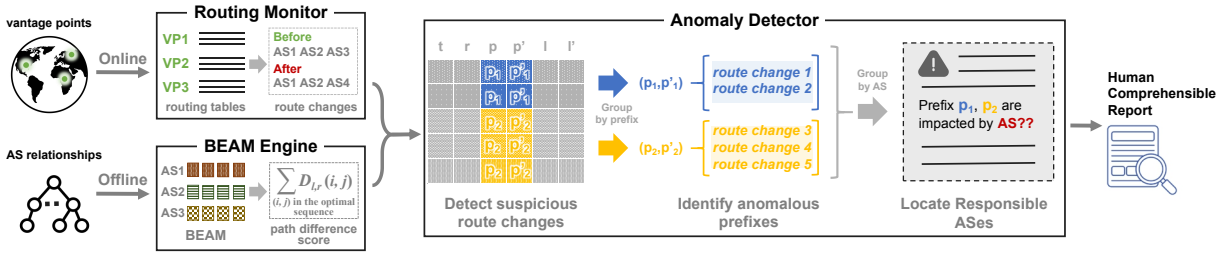

Figure 4: **The workflow of our routing anomaly detection system built upon BEAM.**

score. The sequence of AS pairs with the minimum value is referred to as *the optimal sequence*. Let $S[1:i]$ ($S'[1:j]$) denote the first i (j) ASes in S (S'). We apply dynamic programming to obtain the optimal sequence of AS pairs for $S[1:i]$ and $S'[1:j]$ based on prior states as follows:

- Given the optimal sequence for $S[1:i]$ and $S'[1:j-1]$, add a new AS pair $(S[i], S'[j])$ to the sequence.
- Given the optimal sequence for $S[1:i-1]$ and $S'[1:j]$, add a new AS pair $(S[i], S'[j])$ to the sequence.
- Given the optimal sequence for $S[1:i-1]$ and $S'[1:j-1]$, add a new AS pair $(S[i], S'[j])$ to the sequence.

We choose the operation yielding the minimal sum of pairwise AS difference as the optimal AS pair sequence for $S[1:i]$ and $S'[1:j]$. The induction base, *i.e.,* the optimal AS pair sequence for $S[1:1]$ and $S'[1:1]$, is trivial to compute. For details of this algorithm, please see our additional technical report [43].

4.3 Anomaly Detector

Detecting Suspicious Route Changes. The anomaly detector first identifies suspicious route changes caused by routing anomalies. Specifically, for a route change, the anomaly detector checks whether its path difference score is greater than a threshold th_d, which is dynamically computed using historical legitimate route changes (detailed in §5.2). If so, the route change is regarded as *suspicious*.

Identifying Anomalous Prefixes. It is necessary to prioritize the widespread routing anomalies captured by multiple vantage points. Towards this end, the anomaly detector groups the suspicious route changes that impact the same prefix into different *prefix events*, where each event is associated with a specific prefix and sorts the suspicious route changes by their occurrence time. For each event, the anomaly detector applies a sliding window to count the number of individual vantage points that observe suspicious route changes within the window. If this number is above a threshold th_v (detailed in §5.2), we consider the prefix event is associated with a widespread routing anomaly and regard the event as anomalous.

Locating Responsible ASes. The misbehaved ASes that are responsible for routing anomalies may impact multiple prefixes simultaneously. To obtain comprehensive information about the affected prefixes in each routing anomaly, the anomaly detector correlates all anomalous prefix events based on their responsible ASes. In particular, for each suspicious route change associated with an anomalous prefix event, our detector identifies the ASes that either appear in the new path or in the original path. Then, we compute the intersection of these ASes from all route changes as the responsible ASes for the anomalous prefix event. Given two prefix events, if their time ranges have overlaps and they have common responsible ASes, we consider they are correlated. Thereby, we can divide all anomalous prefix events into multiple sets, where each event only correlates with the other events in the same set (*i.e.,* no cross-set correlation). Finally, our anomaly detector treats each set as an individual routing anomaly and outputs an alarm that specifies both the affected prefixes and the responsible ASes. Due to space limitations, the implementation details are provided in our additional technical report [43].

5 Experimental Results

In this section, we evaluate the path difference scores and perform experiments with real-world BGP data. We also deploy our system at a large ISP to verify its effectiveness in practice.

5.1 Measuring Path Difference Score

We use real-world route announcements to analyze both legitimate and anomalous route changes in terms of path difference scores. We collect 18 reports on historical routing anomalies spanning from 2008 to 2021, including 15 BGP hijacking (2 prefix and 13 subprefix hijacking) and 3 BGP route leak incidents. For each anomaly, we obtain manually confirmed information (*e.g.,* the time of anomalies and the affected prefixes) from two authoritative sources, the Oracle blogs [44] and the BGPStream monitor [45]. Based on the information, we fetch all route announcements 12 hours before and after the anomalies from RouteViews [46] and obtain 18 datasets. The total number of collected route announcements is 11,861,377,951. In each dataset, we identify the anomalous route changes with the confirmed information of the anomaly. The anomalous route changes in our datasets cover both *origin change* (*i.e.,* two routing paths have different origin ASes)

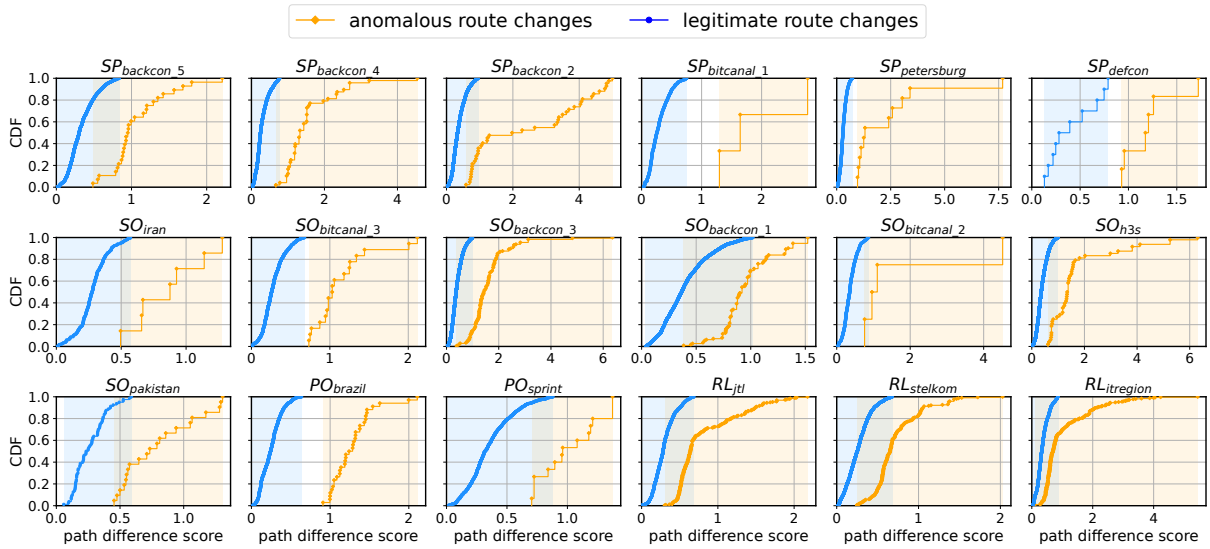

Figure 5: **Statistical comparisons of path difference scores between anomalous and legitimate route changes.**

and *path change* (*i.e.,* two routing paths share the same origin yet traverse different ASes). We name each dataset (*e.g.,* PO_{brazil}) accordingly. We also locate legitimate route changes such as origin changes incurred by the multiple origin AS issue [41]. See details in our additional technical report [43].

Figure 5 shows the path difference scores measured by the BEAM engine on the 18 datasets. The path difference scores of anomalous route changes are much higher than those of legitimate ones, indicating that routing anomalies would significantly change the routing roles of ASes on the routing paths. For instance, the anomalous route changes in $SO_{pakistan}$ changed the origin AS from AS 36561 (*YouTube*) to AS 17557 (*Pakistan Telecom*). These two origin ASes are significantly different in their neighbors, geographic locations and hierarchical levels in the Internet. In contrast, the different origin ASes in a legitimate route change often belong to the same organization and their neighbors are more similar (*e.g.,* the same upstream AS), resulting in much smaller routing role churn between the original and new routes.

5.2 Routing Anomaly Detection Results

We now validate the performance of our detection system. Since RouteViews archives the RIB data (*i.e.,* the snapshot of routing table) of global vantage points bi-hourly, for each dataset, we fetch the most recent RIB data before the confirmed anomaly to initialize the routing tables that our detection system monitors. Then we use the route announcements observed within two hours as input, which include all routes associated with the confirmed anomaly and enough legitimate route changes. We set the sliding window length to two hours for the same reason. The thresholds th_d and th_v are decided by historical data. Specifically, we use the observed path difference scores for all legitimate route changes two hours before the current window as a reference distribution, and set th_d as the knee point of its CDF curve. th_v is determined similarly. The calculation of the knee point is automated via kneed [47].

To systematically evaluate our system, we substitute the BEAM engine with other commonly used features or representation learning models to create variants of our detection system. In particular, we create 6 variants: ED uses the edit distance [48] to measure path difference; JI uses the Jaccard index of neighbor AS sets to measure the similarity of two ASes; Li, Ma, NV and SD uses the general network representation model Line [35], Marine [49], node2vec [50] and SDNE [51] to train embedding vectors, respectively, and utilize Euclidean distance of embedding vectors to measure the similarity of two ASes. All these variants are well-trained and use the same settings as our detection system. Moreover, we compare our system with two state-of-the-art ML-based BGP anomaly detection approaches: AV [28] and LS [22]. AV and LS identify anomalous route changes and the time intervals where anomalies occur, respectively. For fair comparisons, we apply our anomaly detector to aggregate their detection results into different alarms (see Appendix C). We do not compare our work with the active probing-based systems like [10, 15] because they heavily rely on the real-time probing results collected by many data-plane facilities. These real-time probing results are not available for our historical datasets.

Each detection system may raise multiple alarms for a dataset. Each alarm reports a potential routing anomaly. If any alarm matches the confirmed information, *i.e.,* the target prefix is reported as one of the anomalous prefixes and the misbehaved ASes are also identified as responsible, we consider the confirmed anomaly in this dataset as *detected*. Besides the confirmed anomaly, there could be other alarms

Table 1: **Detection results on the 18 real-world datasets.** The ✓/✗ indicates whether the confirmed anomaly of a dataset is *detected*. If *detected*, the number of all raised alarms (#Alarms) and that of false alarms (#FalseAlarms) are presented in bold.

Dataset	Detected									#Alarms(#FalseAlarms)								
	ED	JI	Li	Ma	NV	SD	LS	AV	Ours	ED	JI	Li	Ma	NV	SD	LS	AV	Ours
$SP_{backcon\_5}$	✓	✓	✓	✓	✓	✗	✗	✗	✓	**18(5)**	**12(3)**	**21(4)**	**15(3)**	**17(2)**	9(1)	62(31)	5(1)	**34(2)**
$SP_{backcon\_4}$	✓	✓	✓	✓	✓	✗	✓	✓	✓	**14(1)**	**8(1)**	**15(2)**	**13(1)**	**12(1)**	7(1)	**42(17)**	**22(6)**	**21(0)**
$SP_{backcon\_2}$	✓	✓	✓	✓	✓	✗	✓	✗	✓	**29(7)**	**21(7)**	**24(5)**	**25(7)**	**21(4)**	8(3)	**38(13)**	23(18)	**37(1)**
$SP_{bitcanal\_1}$	✗	✗	✗	✓	✓	✗	✗	✗	✓	16(0)	16(0)	14(0)	**18(0)**	**17(0)**	7(0)	67(36)	30(10)	**16(0)**
$SP_{petersburg}$	✓	✓	✓	✓	✓	✗	✓	✓	✓	**22(3)**	**14(0)**	**21(3)**	**20(1)**	**16(0)**	12(2)	**66(28)**	**37(16)**	**24(0)**
SP_{defcon}	✓	✓	✓	✓	✓	✗	✓	✓	✓	**7(2)**	**7(2)**	**9(3)**	**9(3)**	**9(3)**	2(2)	**28(10)**	**17(9)**	**7(1)**
SO_{iran}	✓	✓	✓	✓	✓	✗	✗	✗	✓	**15(1)**	**8(1)**	**24(5)**	**12(2)**	**16(3)**	0(0)	21(11)	19(10)	**31(2)**
$SO_{bitcanal\_3}$	✗	✗	✗	✗	✓	✗	✗	✗	✓	26(4)	24(3)	29(6)	25(3)	**26(5)**	7(0)	44(19)	17(8)	**40(1)**
$SO_{backcon\_3}$	✓	✓	✓	✓	✓	✗	✗	✗	✓	**32(8)**	**23(4)**	**27(6)**	**34(8)**	**34(9)**	6(1)	49(27)	19(9)	**35(5)**
$SO_{backcon\_1}$	✓	✗	✓	✓	✓	✗	✗	✗	✓	**19(6)**	16(4)	**35(14)**	**18(4)**	**17(7)**	0(0)	63(35)	25(11)	**18(3)**
$SO_{bitcanal\_2}$	✗	✓	✓	✓	✓	✗	✗	✗	✓	16(1)	**15(1)**	**17(2)**	**15(1)**	**16(1)**	12(2)	39(14)	29(8)	**24(0)**
SO_{h3s}	✓	✗	✓	✓	✓	✗	✓	✗	✓	**11(1)**	3(0)	**15(3)**	**12(2)**	**9(0)**	5(1)	**38(22)**	27(8)	**14(0)**
$SO_{pakistan}$	✓	✓	✓	✓	✓	✗	✗	✗	✓	**12(4)**	**8(2)**	**9(2)**	**10(4)**	**8(1)**	1(0)	26(14)	2(0)	**10(1)**
PO_{brazil}	✓	✓	✓	✓	✓	✗	✗	✓	✓	**30(5)**	**32(4)**	**30(5)**	**37(5)**	**25(3)**	11(2)	52(25)	**28(11)**	**51(1)**
PO_{sprint}	✗	✗	✗	✗	✗	✗	✓	✗	✓	20(0)	18(0)	16(0)	22(2)	19(2)	10(2)	**84(24)**	33(8)	**29(0)**
RL_{jtl}	✗	✗	✗	✗	✗	✗	✓	✗	✓	17(1)	16(1)	21(2)	16(2)	17(1)	6(3)	**60(40)**	21(11)	**46(5)**
$RL_{stelkom}$	✗	✓	✗	✗	✗	✗	✓	✗	✓	25(4)	**21(2)**	34(6)	26(3)	23(3)	4(0)	**284(225)**	17(8)	**43(3)**
$RL_{itregion}$	✗	✗	✗	✗	✗	✗	✓	✗	✓	25(0)	21(1)	23(0)	20(0)	21(3)	6(2)	**74(44)**	23(9)	**44(4)**
Overall	11/18	11/18	12/18	13/18	14/18	0/18	9/18	4/18	18/18	354(53)	283(36)	384(68)	347(51)	323(48)	113(22)	1137(635)	394(161)	524(29)

that indicate unrevealed routing anomalies or are simply false alarms. It is difficult to contact the operators to further confirm these potential anomalies since most anomalies occurred long ago. To verify these unknown alarms, we define four anomalous route change patterns that represent typical routing anomalies based on domain knowledge and authorized data such as RPKI validation states. If an alarm matches at least one pattern, we consider it a true alarm with high confidence, otherwise it is a *false alarm*. These patterns are as follows:

- P1 (Unauthorized Route Change): The origin ASes before and after the route change belong to different organizations and have different RPKI validation states, *i.e.,* one in the *invalid_ASN* state and the other in the *valid* state [7].
- P2 (Route Leak): The routing path before or after the route change violates the valley-free criterion [30].
- P3 (Path Manipulation): The routing path before or after the route change contains reserved ASNs or adjacent ASes that have no business relationship records between them [11].
- P4 (ROA Misconfiguration): The origin ASes before and after the change are from the same organization but have different RPKI validation states, *i.e.,* one in the *invalid_length* or *invalid_ASN* state and the other in the *valid* state [52].

These patterns are endorsed by the domain experts from a large ISP where we deployed our system. They also apply these patterns to verify our real-world detection results (see §5.5). Note that these patterns alone *should not* be used to detect routing anomalies directly because they cannot correlate massive route changes with the same root cause, which would result in too many false alarms. We discuss the rationale behind these patterns in our additional technical report [43].

Table 1 shows that all previously-confirmed 18 routing anomalies are correctly detected by our system within tens of alarms. Further, our system reports no false alarms for 6 datasets that cover the confirmed anomalies, and only 5 false alarms in the worst case. These false alarms are mainly related to route engineering practices such as AS prepending [53], while some involve stub ASes with limited connections. In contrast, the baselines cannot detect all these anomalies and raise more false alarms than ours (except for SD that cannot detect any anomalies). Besides, the baselines require many extra data to train the models, *e.g.,* AV needs RIB entries in every two hours and LS requires a large amount of training data to eliminate the negative impacts of label noises. Moreover, AV cannot detect transient anomalies and LS incurs high FP due to per-minute anomaly detection. Please see the details in Appendix C. Overall, our detection system outperforms these baselines by significant margins. In summary, (i) our system addresses the key challenges of ML-based detection methods and realize effective Internet routing anomaly detection. (ii) compared with our BEAM model, general network representation learning models are not able to effectively capture BGP semantics for routing anomaly detection.

5.3 Runtime Overhead

Our system runs on a Linux server with Intel Xeon E5-2650v4 (2.20GHz). We present its runtime overhead in Fig. 6. The X-axis displays the datasets in their chronological order, with

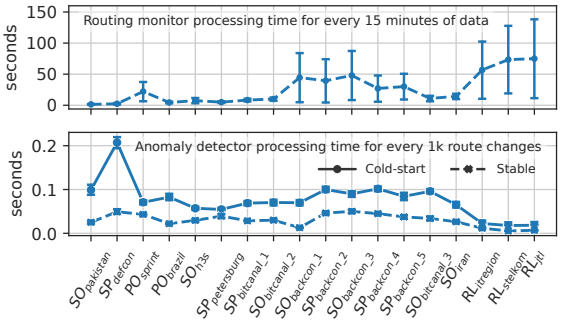

Figure 6: **The runtime overhead of our detection system.**

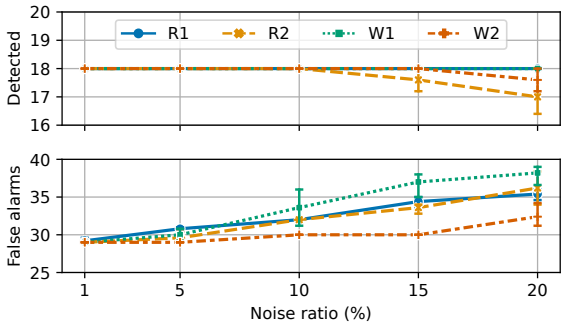

Figure 7: **Detection results given noisy AS relationships.**

the later datasets containing more ASes in operation. The number of ASes in our datasets increases from 27,588 to 73,014 over the entire period. The top figure shows the average time it takes to process every 15 minutes of data from each dataset. The error bar reflects the 95% confidence interval. The variance in processing time is mostly caused by the variance in the sizes of 15-minute data. The processing time is less than 100 seconds in most cases, indicating that the increase of ASes only slightly impacts our system's runtime overhead. The largest processing time (∼140s in RL_{jtl}) is still much smaller than 15 minutes, meaning that our system can effectively process the stream of global route announcements in real time. The bottom figure plots the average time it takes to process every 1,000 route changes. Due to the caching employed by our system (*e.g.,* the caching of path difference scores), the anomaly detector spends much less time at the steady state (∼0.05s per 1,000 route changes), compared with the cold start (∼0.2s).

5.4 Robustness Analysis

We analyze the robustness of our system given noisy AS relationship data, which is created by modifying or deleting the original AS relationships in the CAIDA dataset. We consider four types of noisy datasets. In particular, given a noise ratio r, we first randomly select r% AS relationships, and then flip their labels (*i.e.,* changing P2P to P2C and P2C to P2P) to produce the R1 dataset, or delete them to create the R2 dataset. R1 and R2 represent the noisy dataset caused by inaccurate and incomplete AS relationship inference, respectively. To create another two types of noisy datasets, we first select top-r% AS relationships with the fewest BGP routes that use their underlying AS-to-AS links and create the W1 dataset by flipping their labels and W2 by deleting them. These two types of noisy data are common in the Internet because the AS relationships serving fewer routes are more likely to be mislabeled due to their limited Internet visibility.

We train our system with each noisy dataset independently and evaluate its detection performance following the same steps in §5.2. We repeat each experiment for 5 times to avoid bias and plot the results in Fig. 7. The error bar shows the 95% confidence interval. Even when the noise ratio reaches 20%, our system still detects at least 17 true anomalies (18 in total) and only generates less than 40 false alarms across all datasets. Note that such a high noise ratio is rare in practice. These results demonstrate the robustness of our system under noisy AS relationship data.

5.5 Real-World Deployment

We deploy our detection system in the main operational AS of a large ISP to evaluate its performance in practice. Based on the customer cone size, the AS rank is in the global top 100 [54]. At the time of our deployment, the AS maintains live BGP sessions with about 500 neighbors, including 14 Tier-1 ASes. Thereby, the AS has a fairly comprehensive view on the Internet-wide routing paths. We set up our detection system in a server that receives real-time incoming route announcements from all BGP routers within the AS via the iBGP protocol. We train the BEAM engine of our system using the latest CAIDA AS relationship dataset collected at the time of our deployment, which includes 74,923 ASes and 505,927 AS business relationship records. The sliding window length of our system is one hour. We use the same parameter settings and the method of identifying false alarms as described in §5.2. To reduce the repetitive alarms raised in different time windows, we aggregate the alarms sharing the same anomalous prefix and responsible AS.

The system is online since Jaunary 1, 2023. We analyze the generated results from January 1 to February 1, 2023. In total, the system processes 152,493,303 live route announcements during this month, detects 5,106,442 route changes and raises 548 alarms. We show the alarms' impact and daily statistics in Table 2 and Fig. 8, respectively. On average, our system identifies 17.68 alarms per day. The domain experts of the ISP carefully verify the correctness of each alarm based on the patterns described in §5.2. They find that most alarms (497 out of 548) indicate real routing anomalies, *i.e.,* true alarms. These true alarms, not detected by the ISP's existing routing security mechanisms, include 84 unauthorized route changes

Table 2: **The overall impact of the detected anomalies.**

#Affected Routes	#Affected Prefixes	#Affected Origins
1,202	961	477

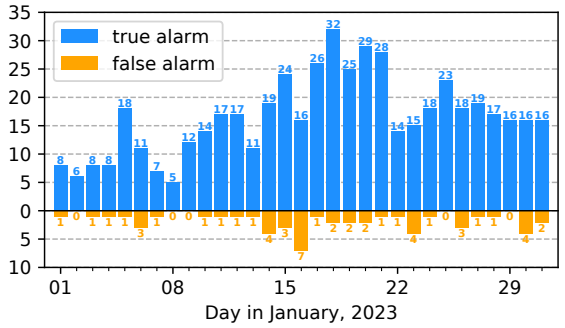

Figure 8: **Daily alarm number in real-world deployment.**

(P1), 123 route leaks (P2), 270 path manipulations (P3) and 20 ROA misconfiguration (P4). The interpretability of these alarms greatly facilitates the identification of anomaly sources (see §6 for detailed case study). More importantly, our system only raises an average of 1.65 false alarms per day. These false alarms can be further eliminated with minimal intervention (see discussions in §7). Overall, our detection system demonstrates promising results in real-world deployment.

We further investigate *how many anomalies are caught by an existing security mechanism but not our system*. First, we check whether our system misses the anomalies detected by the ISP's existing security mechanism. The ISP detects invalid routes based on its customers' IRRs. However, due to the incomplete coverage of prefixes, the ISP's security mechanism does not generate any alarm during our system's deployment. Thus, our system does not miss any alarm raised by the ISP itself. Next, we check whether there are real routing anomalies missed by both our system and the ISP itself. This requires a reliable source for BGP incidents. Although BGPstream is a promising candidate, we cannot use it in our paper because its vantage points are quite different from those of the ISP, which would introduce non-negligible experimental bias. Therefore, we use the RPKI validation results as a reference. Specifically, among all the RPKI-invalid announcements that are generated during our system's deployment, our system only misses about 2.25% of them and most of the missed ones are due to the limited Internet visibility, *i.e.,* only a few vantage points observe these invalid announcements. Overall, this result indicates our system has very low false negatives.

Ethics. We operate our detection system in compliance with the ISP's policy/agreement and under the close supervision of ISP's administrators. Our evaluation does not involve any sensitive or privacy data. We only collect results for analysis and do not interfere with Internet routing operations.

6 Case Study

In this section, we illustrate the interpretability of our detection results by analyzing four detected anomalies: three from historical events (Cases 1-3) and one from our real-world deployment (Case 4). Cases 1-3 each cover a different category of routing anomalies, *i.e.,* origin change, path change, and route leak. In each case, we compare a representative anomalous route change with a legitimate one that occurred on the same day. To interpret the difference between routing paths, we apply two visualization techniques: the heat map of the path difference scores and the embedding map of the routing roles. In particular, the heat map shows the path difference scores for the eligible sequences of AS pairs and marks the optimal sequence identified by DTW. The embedding map, generated by t-SNE [39], visualizes the routing roles of the ASes to illustrate the deviations between two paths.

Case 1. Figure 9(A) shows the origin change in $SO_{pakistan}$, where AS 17557 (*Pakistan Telecom*) hijacked a subprefix of AS 36561 (*YouTube*) by announcing 208.65.153.0/24. Figure 9(A)(I) sees the anomalous pattern that most ASes on the new path are different from those on the old path. The two origin ASes, AS 17557 and AS 36561, are far apart in the embedding map with few common neighbors, indicating very different routing roles. Thus, their route announcements would traverse quite different paths before they eventually converge on AS 5503. This pattern also appears in the heat map, where the path difference score rises sharply as the AS pairing operation proceeds, because the ASes from two paths are completely different after a few steps. In contrast, Fig. 9(A)(II) shows a legitimate route change on the same day with low path difference scores and almost overlapping paths in the embedding map, because the legitimate route update changes the origin from AS 6198 to AS 6197, both operated by *BellSouth Network* and with similar routing roles.

Case 2. Figure 9(B) shows the path change in $SP_{backcon\_2}$, where AS 203959 (*BackConnect*) hijacked a subprefix of AS 25761 (*Staminus Comm.*) by faking a nonexistent routing path to the real origin AS. The embedding map reveals that the fake path detours significantly from the real one, as AS 203959 is neither on the real path nor similar to any ASes on it in terms of routing roles. The heat map also indicates an upsurge in the path difference score in the AS pairing. The optimal sequence of AS pairs greatly deviates from the diagonal of the heat map because no AS is similar to AS 203959 in the counterpart. In contrast, the paths in the legitimate route change are similar in routing roles; we leave the analysis to the reader.

Case 3. Figure 9(C) shows the route leak in RL_{JTL}, where AS 36866 (*JTL*) received the route to 156.0.233.0/24 from its provider AS 8966 (*Emirates Tel.*) and leaked it to its another provider AS 37662 (*WIOCC*). The leaked route results in a much longer path through AS 37662 that detours significantly from the original one. It also violates the valley-free criterion. Accordingly, the heat map shows high path difference scores

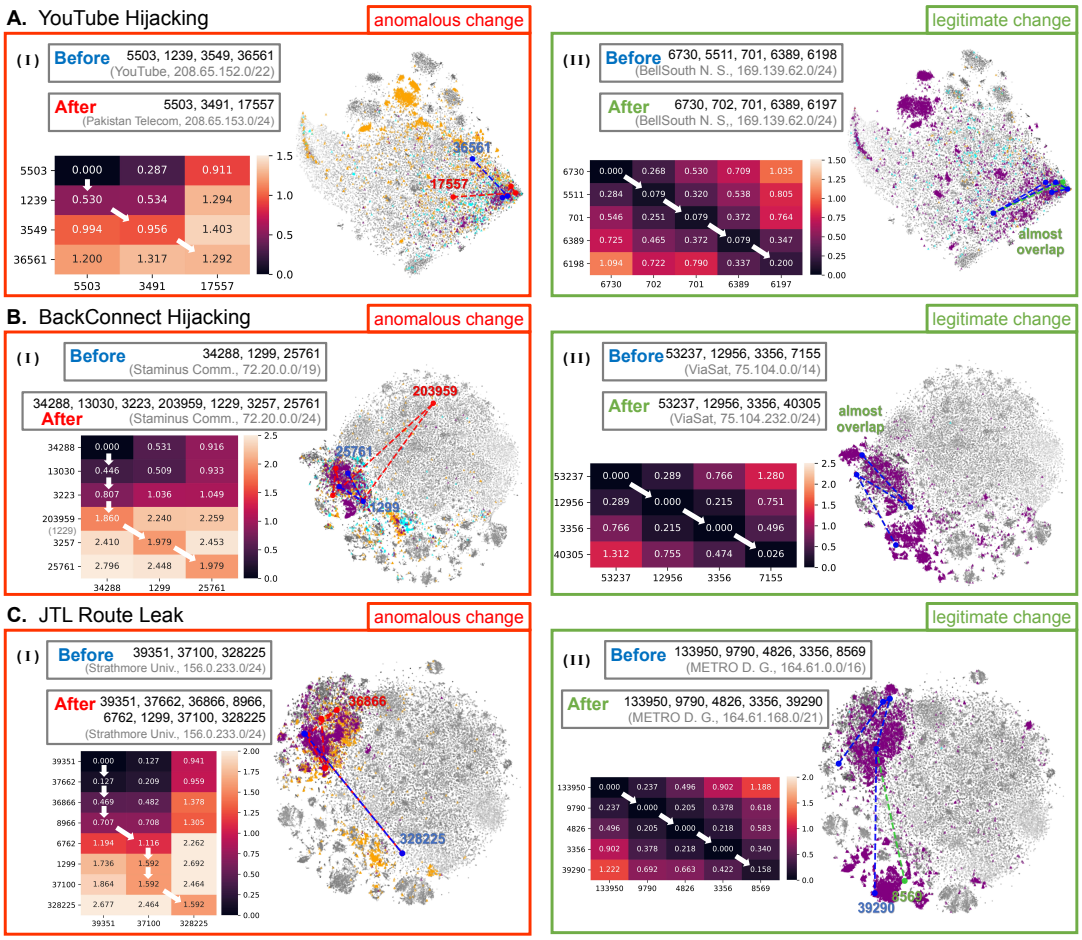

Figure 9: **Typical anomalous/legitimate route changes.** In each plot, the heat map (left) displays the path difference score at each AS pairing step of DTW in a top-to-bottom and left-to-right order. The white arrows show the optimal sequence of AS pairs. The embedding map (right) represents embedding vectors in a 2-D plane. The purple dots denote the common neighbors of the two routing paths, while the orange and cyan dots correspond to the exclusive neighbors for the new and old paths, respectively.

and the optimal sequence forms a hump-like pattern away from the diagonal. In contrast, the legitimate route change has minor path difference; we leave the analysis to the reader.

Case 4. We represent the first alarm reported by our system during its real-world deployment in Fig. 10. This alarm, labeled Alarm 0, starts at 01:04:40, January 1, 2023, and lasts about 1 hour and 24 minutes, affecting three prefixes and six routes observed by two individual vantage points. The top part of Fig. 10 provides an overview of this alarm. The bottom part of Fig. 10 further plots a representative route change event captured by this alarm observed from AS 6453. In this event, AS 42440 (*RDG-AS*) announces 185.88.179.0/24, which is owned by AS 201691 (*WEIDE*), without authorization (its RPKI validation state is *invalid_ASN*). Moreover, AS 42440 is also on the path before the route change, indicating a Type-5 route leak as described in RFC 7908 [55]. This pattern also appears in the heat map. The optimal sequence before AS 42440 follows the diagonal and the path difference score increases sharply after AS 42440, indicating a significant change of routing roles. The embedding map also demonstrates the clear difference between AS 42440 and AS 201691.

7 Discussion

Per-Prefix Threshold. Our system uses the same detection threshold values (*i.e.,* th_d and th_v) for all prefixes. Assigning an individual threshold for each prefix may gain better results. But it will incur significant computational overheads and large detection delays due to the large number of prefixes in the Internet (*i.e.,* over 940k IPv4 and 200k IPv6 prefixes in November 2023), which is inappropriate for online detection.
Reducing False Alarms. We can apply two heuristics to further reduce our system's false alarms. (i) Since most route hijacks and route leaks are short-lived [10], we can label the long-lived routes as normal. (ii) The different origin ASes in a legitimate MOAS event usually have similar routing

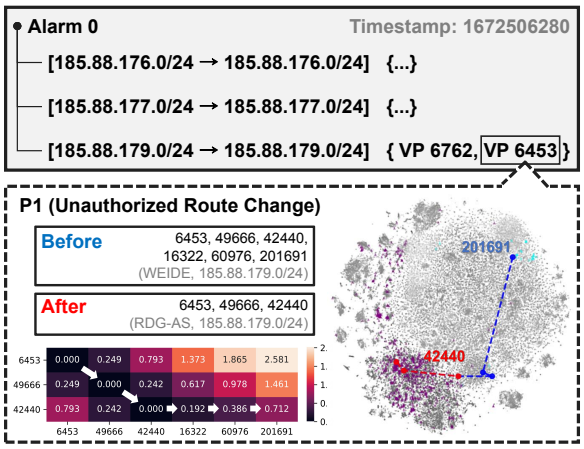

Figure 10: **An anomaly report in real-world deployment.**

characteristics [41] (*e.g.,* from the same organization) while the malicious ASes do not. We leave the application of both heuristics in our system to future work.

Adaptive Attacks. An adaptive attacker may attempt to bypass our system by mimicking the normal ASes' routing roles, yet it is very difficult in practice. Specifically, to imitate a normal AS's routing role, a malicious AS has to establish business relationships with the normal AS's neighbors, which is time-consuming and would reduce the malicious AS's stealthiness due to the possible scrutiny required for establishing neighboring relationships. Further, to make the routing roles of a forged path similar to those of a real path, the attacker has to control multiple ASes on the forged path. Our empirical study (presented in detail in Appendix D) reveals that it requires the attacker to control at least two ASes and establish hundreds of new AS relationships to make the routing roles of a forged path similar to those of a real path. This is considerably intractable in practical scenarios.

Evolving Routing Roles. AS routing roles evolve as ASes update routing policies. Our system is resilient against routing role evolution. Per §5, even if the time gap between training and detection is over 30 days (*i.e.,* the dataset update cycle), our system still performs well. Moreover, we can retrain BEAM with latest AS relationships to keep up with the routing role evolution. It takes ∼10 hours to train the model on our platform with GeForce RTX 2080 Ti, which is acceptable since CAIDA releases a new dataset roughly every month.

Detection with Unknown ASes. BEAM cannot learn routing roles of ASes whose relationships with other ASes are unknown (*i.e.,* unknown ASes). Fortunately, the existing study [34] has revealed AS relationships for most ASes absent in the dataset. For instance, there are only 368 unknown ASes in the most recent events that we analyze, which is only 0.5040% of the total ASes. Further, our measurement study discovered that some unknown ASes use the AS numbers that are reserved for private use [56]. The routing paths containing such unknown ASes should be regarded as anomalous.

8 Related Work

Traditional Routing Anomaly Detection. The existing studies consist of the control-plane based [11, 17, 18], the data-plane based [9, 14, 16] and the hybrid methods [10, 12, 13, 15]. The control-plane based methods maintain the normal/authoritative route information for each prefix and check if the newly received route contradicts it. The data-plane based methods identify routing anomalies by analyzing the reachability from multiple hosts to the target prefix. However, these traditional approaches require non-trivial manual investigation from network operators, *e.g.,* collecting the normal route information of each prefix, and deploying vast network probes to monitor the prefixes in the world, which incurs unacceptable operation overhead for deployment.

ML Based Routing Anomaly Detection. Machine learning is utilized to detect routing anomalies [19–29]. Shapira *et al.* [28] (AV) use unsupervised word embedding to model routes. But AV fails to detect transient anomalies due to its reliance on RIB snapshots and suffers from non-trivial retraining overheads. Dong *et al.* [22] and Hoarau *et al.* [27] perform supervised classification on BGP time series. Both require large-scale labeled datasets, which is hard to achieve in practice. Moreover, all these methods cannot provide interpretable results, incurring great manual efforts for validation.

Application of Network Representation Learning. Network representation learning (NRL) [57] aims to learn latent, low-dimensional representations of network vertices, while preserving network characteristics, *e.g.,* the structure information and vertex content, where the learnt representations can be used for downstream tasks. NRL has been applied in various domains, *e.g.,* social network analysis [58], recommendation system [59], and anomaly detection [60]. In this paper, we develop a BGP semantics aware NRL model to measure ASes' routing roles for routing anomaly detection.

9 Conclusion

In this paper, we present a routing anomaly detection system centering around a novel network representation learning model named BEAM. The core design of BEAM is to accurately learn the routing roles of Internet ASes by incorporating the BGP semantics. As a result, routing anomaly detection, given BEAM, is reduced to discovering unexpected routing role churns upon observing new route announcements. We implement a prototype of our routing anomaly detection system and extensively evaluate its performance using 18 real-world RouteViews datasets containing over 11 billion route announcement records. The results demonstrate that our system can detect all previously-confirmed routing anomalies within an acceptable number of alarms. We also perform one month of real-world detection at a large ISP and detect 497 true anomalies in the wild with only 1.65 daily false alarms on average, demonstrating the practical feasibility of our system.

Acknowledgments

We sincerely thank our Shepherd and all the anonymous reviewers for their valuable comments. This work is supported in part by the National Key R&D Program of China under Grant 2022YFB3102300 and NSFC under Grant 62132011. Qi Li is the corresponding author of the paper.

References

[1] M. Doug, "Large european routing leak sends traffic through china telecom," Oracle blog, 2019.

[2] S. Aftab, "A major bgp hijack by as55410-vodafone idea ltd," MANRS blog, 2021.

[3] C. Catalin, "Klayswap crypto users lose funds after bgp hijack," The Record blog, 2022.

[4] M. Lepinski and K. Sriram, "Bgpsec protocol specification," *RFC 8205, IETF*, 2017.

[5] P. v. Oorschot, T. Wan, and E. Kranakis, "On interdomain routing security and pretty secure bgp (psbgp)," *TISSEC*, vol. 10, no. 3, pp. 11–es, 2007.

[6] S. Kent, C. Lynn, and K. Seo, "Secure border gateway protocol (s-bgp)," *J-SAC*, vol. 18, no. 4, pp. 582–592, 2000.

[7] P. Mohapatra, J. Scudder, D. Ward, R. Bush, and R. Austein, "Bgp prefix origin validation," in *IETF RFC 6811*, 2013.

[8] W. Chen, Z. Wang, D. Han, C. Duan, X. Yin, J. Yang, and X. Shi, "Rov-mi: Large-scale, accurate and efficient measurement of rov deployment," in *NDSS*, 2022.

[9] C. Zheng, L. Ji, D. Pei, J. Wang, and P. Francis, "A lightweight distributed scheme for detecting ip prefix hijacks in real-time," *SIGCOMM CCR*, vol. 37, no. 4, pp. 277–288, 2007.

[10] P.-A. Vervier, O. Thonnard, and M. Dacier, "Mind your blocks: On the stealthiness of malicious bgp hijacks." in *NDSS*, 2015.

[11] P. Sermpezis, V. Kotronis, P. Gigis, X. Dimitropoulos, D. Cicalese, A. King, and A. Dainotti, "Artemis: Neutralizing bgp hijacking within a minute," *TON*, vol. 26, no. 6, pp. 2471–2486, 2018.

[12] J. Schlamp, R. Holz, Q. Jacquemart, G. Carle, and E. W. Biersack, "Heap: reliable assessment of bgp hijacking attacks," *J-SAC*, vol. 34, no. 6, pp. 1849–1861, 2016.

[13] X. Hu and Z. M. Mao, "Accurate real-time identification of ip prefix hijacking," in *S&P*. IEEE, 2007, pp. 3–17.

[14] J. Li, T. Ehrenkranz, and P. Elliott, "Buddyguard: A buddy system for fast and reliable detection of ip prefix anomalies," in *ICNP*. IEEE, 2012, pp. 1–10.

[15] X. Shi, Y. Xiang, Z. Wang, X. Yin, and J. Wu, "Detecting prefix hijackings in the internet with argus," in *IMC*, 2012, pp. 15–28.

[16] Z. Zhang, Y. Zhang, Y. C. Hu, Z. M. Mao, and R. Bush, "ispy: Detecting ip prefix hijacking on my own," in *SIGCOMM*, 2008, pp. 327–338.

[17] J. Li, D. Dou, Z. Wu, S. Kim, and V. Agarwal, "An internet routing forensics framework for discovering rules of abnormal bgp events," *SIGCOMM CCR*, vol. 35, no. 5, pp. 55–66, 2005.

[18] H. Yan, R. Oliveira, K. Burnett, D. Matthews, L. Zhang, and D. Massey, "Bgpmon: A real-time, scalable, extensible monitoring system," in *CATCH*. IEEE, 2009, pp. 212–223.

[19] M. Cheng, Q. Xu, L. Jianming, W. Liu, Q. Li, and J. Wang, "Ms-lstm: A multi-scale lstm model for bgp anomaly detection," in *ICNP*. IEEE, 2016, pp. 1–6.

[20] C. Testart, P. Richter, A. King, A. Dainotti, and D. Clark, "Profiling bgp serial hijackers: capturing persistent misbehavior in the global routing table," in *IMC*, 2019, pp. 420–434.

[21] M. Cheng, Q. Li, J. Lv, W. Liu, and J. Wang, "Multi-scale lstm model for bgp anomaly classification," *TSC*, 2018.

[22] Y. Dong, Q. Li, R. O. Sinnott, Y. Jiang, and S. Xia, "Isp self-operated bgp anomaly detection based on weakly supervised learning," in *ICNP*. IEEE, 2021, pp. 1–11.

[23] B. Al-Musawi, P. Branch, and G. Armitage, "Detecting bgp instability using recurrence quantification analysis (rqa)," in *IPCCC*. IEEE, 2015, pp. 1–8.

[24] N. M. Al-Rousan and L. Trajković, "Machine learning models for classification of bgp anomalies," in *HPSR*. IEEE, 2012, pp. 103–108.

[25] A. Lutu, M. Bagnulo, J. Cid-Sueiro, and O. Maennel, "Separating wheat from chaff: Winnowing unintended prefixes using machine learning," in *INFOCOM*. IEEE, 2014, pp. 943–951.

[26] T. Shapira and Y. Shavitt, "A deep learning approach for ip hijack detection based on asn embedding," in *NetAI*, 2020, pp. 35–41.

[27] K. Hoarau, P. U. Tournoux, and T. Razafindralambo, "Suitability of graph representation for bgp anomaly detection," in *LCN*. IEEE, 2021, pp. 305–310.

[28] T. Shapira and Y. Shavitt, "Ap2vec: an unsupervised approach for bgp hijacking detection," *TNSM*, vol. 19, no. 3, pp. 2255–2268, 2022.

[29] O. R. Sanchez, S. Ferlin, C. Pelsser, and R. Bush, "Comparing machine learning algorithms for bgp anomaly detection using graph features," in *Big-DAMA*, 2019, pp. 35–41.

[30] L. Gao, "On inferring autonomous system relationships in the internet," *TON*, vol. 9, no. 6, pp. 733–745, 2001.

[31] H. Ballani, P. Francis, and X. Zhang, "A study of prefix hijacking and interception in the internet," *SIGCOMM CCR*, vol. 37, no. 4, pp. 265–276, 2007.

[32] L. Prehn and A. Feldmann, "How biased is our validation (data) for as relationships?" in *IMC*, 2021, pp. 612–620.

[33] V. Giotsas, M. Luckie, B. Huffaker, and K. Claffy, "Inferring complex as relationships," in *IMC*, 2014, pp. 23–30.

[34] M. Luckie, B. Huffaker, A. Dhamdhere, V. Giotsas, and K. Claffy, "As relationships, customer cones, and validation," in *IMC*, 2013, pp. 243–256.

[35] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *WWW*, 2015, pp. 1067–1077.

[36] CAIDA. AS Relationships Dataset. Accessed Dec. 10, 2021. [Online]. Available: https://www.caida.org/catalog/datasets/as-relationships/

[37] Z. Jin, X. Shi, Y. Yang, X. Yin, Z. Wang, and J. Wu, "Toposcope: Recover as relationships from fragmentary observations," in *IMC*, 2020, pp. 266–280.

[38] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Stat.*, pp. 400–407, 1951.

[39] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *J. Mach. Learn. Res.*, vol. 9, no. 11, 2008.

[40] G. Y. Lu and D. W. Wong, "An adaptive inverse-distance weighting spatial interpolation technique," *Comput Geosci*, vol. 34, no. 9, pp. 1044–1055, 2008.

[41] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, S. F. Wu, and L. Zhang, "An analysis of bgp multiple origin as (moas) conflicts," in *SIGCOMM WS*, 2001, pp. 31–35.

[42] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *KDD workshop*, vol. 10, no. 16. Seattle, WA, USA:, 1994, pp. 359–370.

[43] "Additional technical and experimental report for bgp semantics aware network embedding." [Online]. Available: https://github.com/yhchen-tsinghua/routing-anomaly-detection/blob/master/doc/

[44] "Oracle blogs," Accessed Dec. 10, 2021. [Online]. Available: https://blogs.oracle.com/internetintelligence/

[45] BGPStream. All Events for BGP Stream. Accessed Dec. 10, 2021. [Online]. Available: https://bgpstream.com/

[46] University of Oregon Route Views Project. MRT format RIBs and UPDATEs. Accessed Dec. 10, 2021. [Online]. Available: http://routeviews.org/

[47] K. Arvai, "kneed." [Online]. Available: https://github.com/arvkevi/kneed

[48] E. S. Ristad and P. N. Yianilos, "Learning string-edit distance," *PAMI*, vol. 20, no. 5, pp. 522–532, 1998.

[49] M.-H. Feng, C.-C. Hsu, C.-T. Li, M.-Y. Yeh, and S.-D. Lin, "Marine: Multi-relational network embeddings with relational proximity and node attributes," in *ACM WWW*, 2019, pp. 470–479.

[50] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *SIGKDD*, 2016, pp. 855–864.

[51] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *SIGKDD*, 2016, pp. 1225–1234.

[52] Y. Gilad, A. Cohen, A. Herzberg, M. Schapira, and H. Shulman, "Are we there yet? on rpki's deployment and security," *Cryptology ePrint Archive*, 2016.

[53] R. K. Chang and M. Lo, "Inbound traffic engineering for multihomed ass using as path prepending," *IEEE Netw.*, vol. 19, no. 2, pp. 18–25, 2005.

[54] CAIDA. AS Rank. Accessed May. 25, 2023. [Online]. Available: https://asrank.caida.org/

[55] K. Sriram, D. Montgomery, D. McPherson, E. Osterweil, and B. Dickson, "Problem definition and classification of bgp route leaks," *RFC 7908, IETF*, 2016.

[56] J. Mitchell, "Autonomous system (as) reservation for private use," *RFC 6996, IETF*, 2013.

[57] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: A survey," *TBD*, vol. 6, no. 1, pp. 3–28, 2018.

[58] L. Liu, X. Li, W. K. Cheung, and L. Liao, "Structural representation learning for user alignment across social networks," *TKDE*, vol. 32, no. 9, pp. 1824–1837, 2019.

[59] Q. Tan, N. Liu, X. Zhao, H. Yang, J. Zhou, and X. Hu, "Learning to hash with graph neural networks for recommender systems," in *WWW*, 2020, pp. 1988–1998.

[60] Y. Fan, M. Ju, S. Hou, Y. Ye, W. Wan, K. Wang, Y. Mei, and Q. Xiong, "Heterogeneous temporal graph transformer: An intelligent system for evolving android malware detection," in *SIGKDD*, 2021, pp. 2831–2839.

A Preservation of the Second-Order Proximity

In practice, we derive the weight vector $\boldsymbol{l}$ (see Def. 2) from the softmax transformation of a learnable variable $\boldsymbol{l'} \in \mathbb{R}^d$, and ensures that the minimum value of its components is above α ($\alpha > 0$) to avoid trivial solutions:

$$\boldsymbol{l} = [l_0, \ldots, l_{d-1}]^\intercal = \big(softmax(\boldsymbol{l'})\big)_{\alpha+}. \tag{6}$$

Accordingly, $\boldsymbol{l}$ is differentiable with respect to $\boldsymbol{l'}$ and satisfies the following criteria:

$$\|\boldsymbol{l}\|_1 = \sum_{i=0}^{d-1} |l_i| = 1, \quad l_{min} = \min\{l_0, \ldots, l_{d-1}\} \geq \alpha. \tag{7}$$

In practice, we set $\alpha = \frac{1}{d} \times 10^{-6}$, a rather small positive number. Now, we prove the theorem that the pairwise AS difference does preserve the characteristics of the second-order proximity between ASes[5].

Theorem 1. *The distance function (5) preserves the second-order proximity.*

Proof. Given two vertices u, v ($u \neq v$) having similar neighbors and business relationships, there must exist at least one vertex w ($w \neq u, w \neq v$) such that $(u,w) \in E, (v,w) \in E$. Then after sufficient optimization of the objective function, there must exist two thresholds $\varepsilon_1, \varepsilon_2 > 0$ such that

$$0 \leq p\_score(u,w), p\_score(v,w) \leq \varepsilon_1,$$
$$h\_score(u,w), h\_score(v,w) \geq \varepsilon_2.$$

According to Equation (7), we have

$$p\_score(u,w) \leq \varepsilon_1 \tag{8}$$
$$\Longrightarrow (\boldsymbol{x_w} - \boldsymbol{x_u})^\intercal((\boldsymbol{x_w} - \boldsymbol{x_u}) \odot \boldsymbol{l}) \leq \varepsilon_1. \tag{9}$$
$$\Longrightarrow \|\boldsymbol{x_w} - \boldsymbol{x_u}\|_2 \leq \sqrt{\frac{\varepsilon_1}{l_{min}}} \leq \sqrt{\frac{\varepsilon_1}{\alpha}}. \tag{10}$$

Similarly, we have $\|\boldsymbol{x_w} - \boldsymbol{x_v}\|_2 \leq \sqrt{\frac{\varepsilon_1}{\alpha}}$. Thus $|h\_score(u,v)|$ is upper bounded:

$$\begin{aligned} |h\_score(u,v)| &= |h\_score(u,w) - h\_score(v,w)| \\ &\leq |h\_score(u,w)| + |h\_score(v,w)| \\ &= |(\boldsymbol{x_w} - \boldsymbol{x_u})^\intercal \boldsymbol{r}| + |(\boldsymbol{x_w} - \boldsymbol{x_v})^\intercal \boldsymbol{r}| \\ &\leq \|\boldsymbol{x_w} - \boldsymbol{x_u}\|_2 \cdot \|\boldsymbol{r}\|_2 + \|\boldsymbol{x_w} - \boldsymbol{x_v}\|_2 \cdot \|\boldsymbol{r}\|_2 \\ &\leq 2\sqrt{\frac{\varepsilon_1}{\alpha}}. \end{aligned}$$

Also, $p\_score(u,v)$ is upper bounded by $2\varepsilon_1$:

$$p\_score(u,v) \leq p\_score(u,w) + p\_score(v,w) \leq 2\varepsilon_1$$

Thus, we have $D_{\boldsymbol{l},\boldsymbol{r}}(u,v) = |p\_score(u,v)| + |h\_score(u,v)| \leq 2\varepsilon_1 + 2\sqrt{\frac{\varepsilon_1}{\alpha}}$. That is, there exists an upper bound for the distance between u, v in terms of the distance function (5). Moreover, when the objective function is sufficiently optimized, ε_1 is a rather small positive number approaching zero and α is a constant, and then the distance $D_{\boldsymbol{l},\boldsymbol{r}}(u,v)$ should approach zero, which means the two vertices u, v are at a close distance to each other in the low-dimensional vector space. □

[5]Note that this is a theoretical proof under ideal conditions. In the actual training process, the convergence of the embedding vectors would also be affected by initial values and sampling techniques.

B Sampled AS Pair Datasets

Notation in the table: *No rel* stands for no relationship. *Ngbr JI* refers to the Jaccard index of two neighbor AS sets.

Feature	Name	#AS pair	Sampling rule
1st-order proximity	S_0	10,000	P2P, Ngbr JI $\in [0\%, 10\%)$
	H_0	10,000	P2C, Ngbr JI $\in [0\%, 10\%)$
	N_0	10,000	no rel, Ngbr JI $\in [0\%, 10\%)$
2nd-order proximity	N_1	2,000	no rel, Ngbr JI $\in [0\%, 20\%)$
	N_2	2,000	no rel, Ngbr JI $\in [20\%, 40\%)$
	N_3	2,000	no rel, Ngbr JI $\in [40\%, 60\%)$
	N_4	2,000	no rel, Ngbr JI $\in [60\%, 80\%)$
	N_5	2,000	no rel, Ngbr JI $\geq 80\%$
hierarchy	S_1	10,000	with a direct P2P
	H_1	10,000	with a direct P2C
	H_2	10,000	with 2 consecutive P2C
	H_3	10,000	with 3 consecutive P2C
	H_4	10,000	with 4 consecutive P2C
	H_5	10,000	with 5 consecutive P2C
	H_6	10,000	with 6 consecutive P2C

C Comparison with ML-based Methods

We implement two baselines, *i.e.,* AV and LS, according to their papers [22, 28]. Here, we describe their experimental setup and compare them to our system in detail.

AV Setup. In accordance with the original setup in the paper [28], for each dataset, we train an AV detection model based on the most recent RIB snapshots collected from all vantage points before the confirmed anomaly occurs, and perform detection on the next RIB snapshots in the next two hours. To reduce the training time, the model is trained offline on all two-hour RIB snapshots in parallel. Note that this strategy is not realizable for an AV model deployed for real-world online detection task, because the model has to be trained on the RIB snapshot collected 2 hours earlier and perform detection on the latest RIB snapshot. We set the hyperparameters the same to the original paper. The outputs of the AV detection model are anomalous route changes. For fair comparisons, we further apply our anomaly detector (see §4.3) to aggregate these route changes into alarms, *i.e.,* identifying anomalous prefixes and locating responsible ASes.

LS Setup. The LS method aggregates all BGP announcements in each two minutes into a time interval and extracts

Table 3: **Comparison with ML-based methods.** *Vol.*, *Req.* and *Ann.* are short for Volume, Requirement and Announcement.

Method	Training				Detection	
	Training Data	Training Data Vol.	Retraining Req.	ML Model Type	Detection Data	Anomaly Type
Ours	AS relationships	∼500K	Monthly	Unsupervised	Ann.	Short & Long-lived
AV [28]	RIB entries	∼10M	Bihourly	Unsupervised	RIB entries	Long-lived
LS [22]	Ann. time series	∼100M	Unknown	Supervised	Ann. time series	Short & Long-lived

86 features. Note that LS is a supervised detection method. Thus, for each dataset, we use the other 17 datasets as the training data to train an LS model, and then classify the time intervals (*i.e.,* normal or anomalous) in this dataset. The labeling method of the training data and the parameters of the LS detection model are the same to the original paper. All route changes in the detected anomalous time intervals are considered anomalous. Finally, we also utilize our anomaly detector to aggregate these anomalous route changes into alarms.

The Comparison. We compare our system with AV and LS in Table 3. Specifically, since AV performs detection in the RIB snapshots of every two hours, it cannot detect transient routing anomalies that occur within the period between two RIB snapshots. Besides, AV requires frequent model updates because tens of millions of RIB entries are updated bihourly, which incurs significant retraining overheads. To the best of our knowledge, there is no technique that can be directly applied to accelerate the training of AV. By contrast, our system directly checks every route change collected from each real-time BGP announcement such that it can detect both transient and long-lived routing anomalies. Further, our system only requires the AS relationship data for training, which is much smaller than the RIB snapshots and more stable, ensuring a small retraining overhead.

LS characterizes announcements as time series and applies a supervised neural network to detect if each data point in the time series is associated with routing anomalies. However, each data point in the time series, *e.g.,* the data in a two-minute time interval, may include a large number of legitimate announcements, leading to much more false alarms than our system. Besides, since LS utilizes a supervised model, it requires a large amount of BGP anomaly data for training, *e.g.,* more than 2,000 BGP anomaly events [22], which is difficult to achieve in practice. By contrast, our system performs detection in an unsupervised manner and gets rid of the reliance on anomalous training data.

D An Empirical Study on Adaptive Attacks

We empirically study how many ASes an adaptive attacker needs to control and how many AS relationships need to be established to make the routing roles of a malicious path[6] similar to those of a legitimate path. Specifically, for each legitimate routing path from the 18 real-world datasets (described in §5.1), we start from the vantage point of the legitimate path and search for the malicious path that has the most overlapping segments with the legitimate path. Then, we choose the ASes that appear on the malicious path but not on the legitimate path. These ASes have to be controlled by the attacker to mimic the legitimate path ASes' routing roles. Next, we compare the neighbors of the malicious and the legitimate path and calculate the minimum number of relationships that need to be established to make 10% neighbors the same. In this study, we assume that having 10% common neighbors is sufficient to make the routing roles of a malicious path similar to those of the legitimate path. This reduces the difficulty for the attacker, such that the result is a lower-bound estimate.

We show the result in Fig. 11, where the top and bottom figures display the average number of ASes that must be controlled and the average number of AS relationships that need to be established, respectively. The error bar shows the 95% confidence interval. In most cases, an adaptive attacker has to control at least two ASes and establish hundreds of new AS relationships to make the routing roles of a malicious path and a legitimate path similar. Note that, the cost for an adaptive attacker to bypass our system in practice would be much higher than the above result, because our system monitors multiple vantage points simultaneously, *i.e.,* the attacker has to make the routing roles of multiple malicious routing paths similar to those of normal paths. Thus, it is very difficult for adaptive attackers to bypass our system's detection.

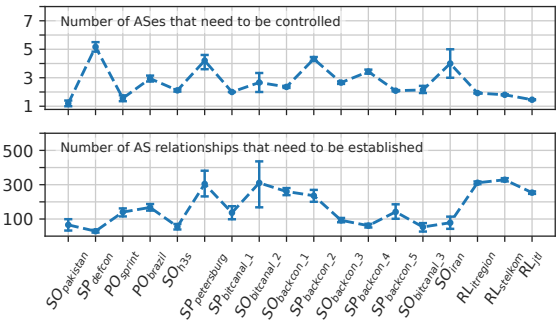

Figure 11: **The empirical estimate of the cost for an adaptive attacker to make the routing roles of a malicious path and a legitimate path similar.**

[6] A malicious path refers to a routing path that contains the known malicious AS in the BGP anomaly event.