# BML: An Efficient and Versatile Tool for BGP Dataset Collection

Kevin Hoarau, Pierre Ugo Tournoux, Tahiry Razafindralambo

# BML: An Efficient and Versatile Tool for BGP Dataset Collection

Kevin Hoarau
*Université de La Réunion, LIM, France*
kevin.hoarau@univ-reunion.fr

Pierre Ugo Tournoux
*Université de La Réunion, LIM, France*
pierre.tournoux@univ-reunion.fr

Tahiry Razafindralambo
*Université de La Réunion, LIM, France*
tahiry.razafindralambo@univ-reunion.fr

***Abstract*—The Border Gateway Protocol (BGP) is in charge of the route exchange at the Internet scale. Anomalies in BGP's behaviour can have several causes (*e.g.* mis-configuration, outage and attacks) and despite being rare, their consequences can threaten the Internet stability and reliability. The study of such anomalies requires the extraction of specific features and internet topology from BGP data. The literature shows that adhoc procedures and tools have been developed to extract specific features to train machine learning models for anomaly detection. In this paper we propose BML, a BGP dataset generation tool that extracts the majority of known features in the literature, the internet topology and that allows the user to build specific features from BGP data. We illustrate the use of BML on a BGP anomaly by extracting 32 synthetic features and 14 BGP's graphs features which allow a comprehensive understanding of the Border Gateway Protocol.**

***Index Terms*—BGP Anomaly, Machine Learning, Dataset.**

## I. Introduction

The Border Gateway Protocol (BGP) is the routing protocol standard on the Internet. A failure of the protocol could impact any service relying on the Internet. Such failures, namely BGP anomalies, happen for several reasons ranging from hardware failures to malicious attacks [1]. BGP anomalies and their detection are studied using BGP data collection projects such as [2], [3]. The study of BGP traces is fundamental to the understanding of the protocol and anomalies that could arise.

BGP data are stored as logs at different collecting points on the Internet and must be pre-processed to be useful. The BGP data pre-processing is time consuming due to the amount of data. Moreover, the data pre-processing method is always influenced by the anomalies to be studied or the period to be observed. There is no systematic and standard way to pre-process BGP data to extract useful information or to be used in machine learning tools.

Many approaches based on statistical features or graph features have been developed in the field of BGP anomaly detection. These features describing the behavior of BGP are extracted from the BGP data. However, the extraction of the features of interest is strongly related to the pre-processing method. Studying different features may lead to different time consuming pre-processing method. A unique pre-processing method should allow the extraction of all important features at once and therefore eases the study of BGP anomalies.

The authors of [4] developed a tool to build a BGP dataset. This tool focuses on generating datasets of the most commonly used BGP statistical features but cannot generate BGP's graph structure and embedings as used in [5]–[8]. Their tool also lacks versatility as it requires to restart the lengthy process of data collection in order to modify the generated features. Therefore, the tool developed in [4] is not suitable for machine learning (ML) approaches for anomaly detection in BGP as the ones described in [5]–[8]. The tool developed in this paper, called BML, allows the extraction of BGP synthetic features (as in [4]), BGP's graph structure and provides the users with the opportunity to craft their own features.

This paper focuses on the construction of BGP datasets to extract synthetic features and graph structure. We propose BML, a tool that automates the BGP data collection and preparation processes. BML only needs a set of timestamps that may relate to relevant BGP events and define the parameters that affect the trade-off between completeness, collection time and the storage footprint of the resulting dataset. Once the dataset is collected, the user can define and apply several data transformation functions to refactor and exploit the data.

We illustrate the usefulness of BML with a simple use case where we analyze the BGP behavior during a large scale anomaly that caused significant disruptions on the Internet and mainly in Japan. The results shows that the tool only needs a few minutes on a low-end computer and less than 30 lines of Python code to collect the relevant BGP data and extract the 32 BGP synthetic features and 14 BGP's graph features.

The remaining of this paper is structured as follows: section II provides a short description of BGP, its anomalies and a state of the art. Section III describes our tool (BML). Section IV is dedicated to a use case study, using BML. Finally, section V concludes and discusses our contribution.

## II. Background

### A. BGP in a nutshell

The Internet consists of Autonomous Systems (ASs) interconnected by Border Gateway Protocol (BGP). Most of the ASs are Internet Service Providers (identified by an ASN) that own IP prefixes [9]. ISPs operate BGP routers that maintain TCP connections with a set of BGP neighbors to exchange routing information with other ASes. Traffic is sent through

This project has received funding from the Région Réunion and the European Union - European Regional Development Fund (ERDF) as part of the INTERREG V - 2014-2020 program.

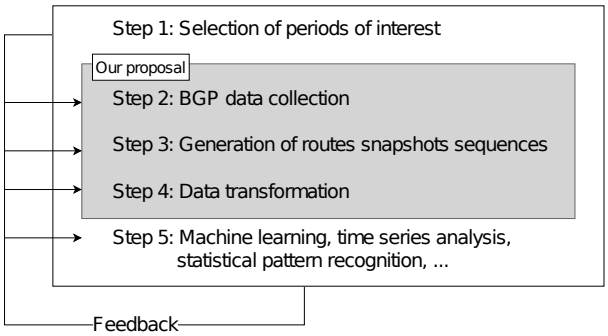


Fig. 1: Iterative workflow of BGP anomaly study.

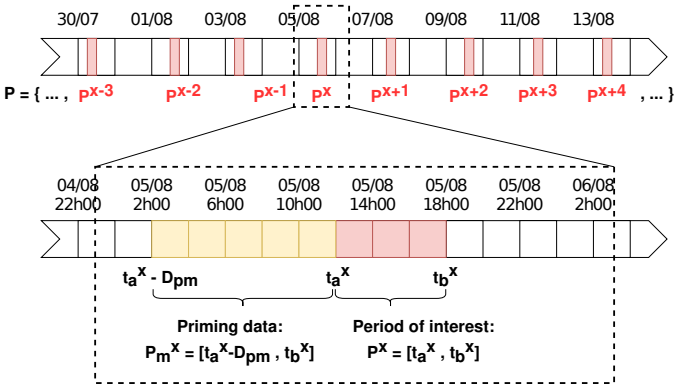


Fig. 2: BGP data collection.

routes learned by BGP. BGP incrementally updates its routes. When using BGP, a route to an IP prefix is identified by the set of ASes (namely the AS-PATH) that participate in the traffic forwarding which avoids routing loops [10].

### B. BGP Anomalies

BGP anomalies are failures or malfunctions of the routing protocol. They can be the result of a protocol vulnerability [11], a configuration error [12], an external event such as a hardware failure [13] or a worm spread [14]. Anomalies can cause instability or overload on the BGP routers. They can lead to invalid network topologies [9] that may result in the unreachability of some prefixes. They can also be used to impersonate some prefixes or for traffic interception [15]. To alleviate the effect of BGP anomalies, several works have been done on the detection of BGP anomalies [1] using time series analysis, statistical pattern recognition or machine learning.

### C. State of the art for BGP data collection

RouteViews [2] and RIPE RIS [3] projects have been collecting and archiving BGP data from different collectors distributed across the world since 2000. The collection of BGP routing information is the cornerstone for any analysis of the BGP protocol. Each of these collectors receives BGP updates from all its neighbors and updates its Routing Information Base (RIB) accordingly. Data from RouteViews and RIPE RIS projects require cleaning and time rearrangement from the different collectors. The BGPStream framework [16] has been designed to sanitize data from BGP routing information.

Once sanitized, the BGP data need transformation to extract useful information. In the literature, a widely adopted BGP data transformation is feature extraction. These features falls into different categories: i) volume features that captures BGP's stability and AS-PATH features that capture topological changes [17]–[21] and ii) graphs features that capture the underlying graph structure of BGP [8]. However, none of the work from the literature provide an systematic approaches to extract all of the features from BGP data.

In the literature, each contribution develops its own tool to extract BGP information. On one hand, in [5]–[8], BGP's graph structure and embedings are used to study BGP behavior. In these papers, the extraction processes are developed for the specific purpose of each paper and do not allow easy reuse. Moreover, these papers do not extract volume features and AS-PATH features (synthetic features). On the other hand, Fonseca *et al.* in [4] developed a tool for the construction of BGP datasets. Their representation of BGP's structure and dynamic is limited to synthetic features. The tool described in [4] is the closest to ours but does not capture the graph structure of BGP especially needed for machine learning (ML) models. Unlike other tools in the literature, BML can change the data representation and features without recollecting the data. BML developed in this paper fills the gap in the literature by providing both the extraction of volume features, AS-PATH features and the extraction of BGP's graphs features but also allows the user to define its own features.

## III. Our Proposal

Our proposal (BML) allows BGP specialists and machine learning practitioners to focus on developing and evaluating BGP or ML models by automating part of the BGP workflow (see Figure 1). Once the period of interest is defined in step ❶, in step ❷ BML automatically collects the data with a specific data structure (see Section III-A for more details). For each period of interest, BML extracts periodic snapshots of the BGP routes and produces a lossless yet storage efficient representation in step ❸ (see Section III-B for more details). During step ❹, BML proposes helper functions to ease the development of a data transformation that fits the requirements of the user or an ML model (see Section IV for an example). The output of this transformation function can be interpreted or used to train and test a ML model (step ❺).

The workflow presented in Figure 1 is iterative and relies on a try-and-error methodology. Our goal is to speed up the iterations of the model by ensuring that: first, constructing and updating a dataset only requires to define periods of interest and a set of parameters, and second, a change in an ML model and/or the data transformation does not require to recollect the BGP data.

### A. BGP data collection

In BML, an event that is worth including in the dataset is called a period of interest $P^x = [t_a^x, t_b^x]$ (notations are

| Symbol | Description |
|---|---|
| $P = \{P^1, ..., P^x, ..., P^N\}$ | a set of period of interest |
| $N$ | the number of periods of interest |
| $P^x = [t_a^x, t_b^x]$ | a period of interest starting at time $t_a^x$ and ending at time $t_b^x$ |
| $Pm^x = [t_a^x - D_{pm}, t_b^x]$ | the priming period for a period of interest $P^x$ |
| $D_{pm}$ | the duration of the priming data collection |
| $D = t_b^x - t_a^x$ | the duration of a period of interest |
| $U_{[t_i,t_j]} = \{u_1, u_2, ...\}$ | a time-ordered set of BGP updates messages whose timestamp $t_x$ are such that $t_i < t_x < t_j$ |
| $r_p = \{r_{p,0}, ..., r_{p,j}, ...\}$ | $r_p$ is a set of routes collected toward a prefix $p$ and $r_{p,j}$ the list of routes (*peer*:AS-PATH) obtained from a collector $j$ |
| $S_t = \{r_1, ..., r_p, ...\}$ | a snapshot of the routes at time $t$ |
| $S^x = \{S_{t_a^x}, S_{t_a^x+\Delta}, ..., S_{t_b^x}\}$ | the sequence of routes snapshots for a period of interest $P^x$ |
| $T(.)$ | a user-defined data transformation function |
| $T^x = T(S^x)$ | the sequence of transformed data for a period of interest $P^x$ |
| $\Delta$ | the interval for the extraction of routes snapshots |

TABLE I: Table of notations.

summarized in Table I). All of the BGP updates from the selected collectors (section II-C) whose timestamp lies in $[t_a^x, t_b^x]$ interval will be included in the dataset. However, at $t_a^x$ there is obviously not enough information to reflect the actual BGP Routing Information Base (RIB). Indeed, as the BGP protocol is incremental, we face a cold start as the routing updates collected between $t_a^x$ and $t_a^x + \epsilon$ do not cover all the routes nor they allow to reproduce BGP's graph at time $t_a^x + \epsilon$. To overcome this issue, we introduce a priming data collection period $Pm^x = [t_a^x - D_{pm}, t_a^x]$ where BGP data is collected before the period of interest (see Figure 2). BML can also leverage the most recent RIB dump available on the collector augmented with the updates received until $t_a^x$. However, the use of RIB dumps comes with twofold increase in the execution time and should only be used if relevant.

After the priming data collection, the BGP updates whose timestamp falls within $[t_a^x, t_b^x]$ are also collected.

*1) Data storage:* The priming period is stored as a route snapshot obtained from algortihm. 1. This representation allows a reduction of the storage space. The updates collected during the period of interest cannot be stored in such route dumps as their temporal dimension needs to be preserved. The default priming period is set to 10h and needs an average of 20MB while the period of interest requires 75MB of storage per hour.

*2) Data collection parameters:* The following parameters cannot be modified after the data collection. They have a significant impact on the execution time, storage and data quality, and as such, they need to be chosen carefully : i) List of periods of interest: the bound of each period of interest should be $t_b^x - t_a^x$ specified as well as the duration of the priming period $D_{pm}$ which should be large enough for to prevent cold start while considering its impact on the dataset size and collection time. ii) List of collectors: this parameter highly depends on the use case as in some cases, the location of the collectors may be important.

**Algorithm 1:** Routes snapshot

**Input:** $U$ an ordered set of updates, $R$ a dictionary of routes
**Output:** $R$ updated dictionary of routes
**begin**
  **foreach** *update u in U* **do**
    **if** $u_{type}$ = *announcement* **then**
      $R[u_{prefix}, u_{collector}, u_{peer}] \longleftarrow u_{as-path}$
    **end**
    **else if** $u_{type}$ = *withdrawal* **then**
      $R[u_{prefix}, u_{collector}, u_{peer}] \longleftarrow \emptyset$
    **end**
  **end**
**end**

### B. Data transformation

Once the data is collected it must be transformed to fit the goal of practitioners and the ML model requirement. Using algorithm 1, BML first generates a route snapshot every $\Delta$ seconds for the period of interest $P^x = [t_a^x, t_b^x]$ resulting in a set of $\frac{t_b^x - t_a^x}{\Delta}$ routes snapshots denoted as $S^x = \{S_{t_a^x}, S_{t_a^x+\Delta}, ..., S_{t_b^x}\}$. An example of such a snapshot is given in Figure 3.

To complete the data transformation, a feature specific data transformation function $T(.)$ is applied to the sequence of routes snapshot $S^x = \{S_{t_a^x}, S_{t_a^x+\Delta}, ..., S_{t_b^x}\}$. The result is the representation sequence $T^x = T(S^x)$. Alternatively, the data transformation can take as input the update sequence instead of the corresponding routes snapshot. This might be required for some transformation such as the computation of the number of routes announcements made within the period of interest.

*1) Examples of data transformation functions:* Various features and data representation have been or may be used by ML models for BGP analysis:

```
{
"195.252.70.0/24":{
  "rrc19":{
     "37468":"37468 5483 6700",
     "37697":"37697 37468 8400 6700 6700",
     "57695":"57695 328383 327782 37100 8400 6700 6700",
     "37640":"37640 8400 8400 8400 8400 8400 6700 6700"
  },
  "rrc00":{
     "202365":"202365 49697 5483 6700",
     "396503":"396503 6939 5483 6700",
     ...
  },
...
}
```

Fig. 3: Sample of a JSON routes snapshot which maps every $(prefix, collector, peer)$ triples observed to an AS-PATH $\{AS_0, AS_1, ..., AS_j\}$. We can see that the collector "rrc19" has four routes to the prefix "195.252.70.0/24" and that one of these routes is announced by its peer "37468" with the AS-PATH "37468 5483 6700".

| Type of feature | | Number of features | Total | Used in |
|---|---|---|---|---|
| Statistical | Volume | 15 | 32 | [4], [17]–[21] |
| | AS-PATH | 17 | | |
| Graph | Centrality | 6 | 14 | [8] |
| | Clique | 2 | | |
| | Clustering | 3 | | |
| | Distance | 1 | | |
| | Topology | 2 | | |

TABLE II: Features extracted by BML (by default).

- Statistical features: *e.g.* number of prefixes or routes, average hop count of the routes [19], [21], [22].
- Graphs: as BGP reflects the internet topology, a route snapshot can easily be represented as a graph whose structure may differ depending on the ML model [5]–[7].
- Graph features: while many ML models can't exploit graph structured data, they can consume synthetic metrics derived from BGP graphs such as the centrality metrics [8] or adjacency matrix.

*2) Data transformation parameters:* The data transformation step requires only two inputs from the user :

- The snapshotting interval $\Delta$: defines the interval between each route snapshot in each period of interest.
- The data transformation function $T(.)$: takes as input a routes snapshot and produces a representation adequate for the ML model.

Note that if these parameters need to be modified, the practitioner only needs to re-execute the data transformation step and not the data collection steps. This allows quick iterations of the BGP workflow and gains in productivity.

## IV. Use case

In this section, we illustrate how BML may benefit ML practitioners and other BGP specialists by studying the BGP anomaly induced by the Google leak[1] that occurred on the 25th of August 2017. At 03:22 UTC, the AS 15169 owned by Google accidentally started to announce several routes to prefixes it does not own and thus became a transit provider. This error caused significant disruptions on the Internet and mainly in Japan.

BML is configured to collect the period of interest P=[03:00;04:00] with a default priming period of 10 hours from the RIPE RIS collector `rrc06` which is located in Japan. BML integrates 46 predefined transformation functions including 32 statistical features that are mostly used in the anomaly detection literature (please refer to Table II). The entire pipeline of the dataset definition, collection, and transformation takes only 20 lines of Python code as shown in Figure 4. The code shows how simple it is to extract features with BML since the variables in this code are mostly descriptions.

During our test case, the data collection took 4 minutes and 21 seconds to complete[2]. We set $\Delta = 1$ minute. The changes of the settings regarding the data transformation *e.g.* the snapshotting interval $\Delta$ or the features, only take a few seconds since BML will not recollect the data.

```
from BML.data import Dataset
from BML.transform import DatasetTransformation
from BML import utils

################
# Data collection

folder = "dataset/"
dataset = Dataset(folder)
dataset.load({
 "PrimingPeriod": 10*60*60, # 10 hours of priming data
 "IpVersion": [4], # only IPv4 routes
 "Collectors": ["rrc06"], # rrc06: at Otemachi, Japan
 "PeriodsOfInterests":
  [{
   "name": "GoogleLeak",
   "label": "anomaly",
   "start_time": utils.getTimestamp(2017, 8, 25, 3, 0, 0),
      # August 25, 2017, 3:00 UTC
   "end_time": utils.getTimestamp(2017, 8, 25, 4, 0, 0)
      # August 25, 2017, 4:00 UTC
  }]
})
# run the data collection
utils.runJobs(dataset.getJobs(), folder+"collect_jobs")

####################
# Data transformation

# features extraction every minute
datTran = DatasetTransformation(folder,
            "BML.transform", "Features", 1)
# run the data transformation
utils.runJobs(datTran.getJobs(), folder+"transform_jobs")
```

Fig. 4: BML code to collect the data of the Google leak use case.

Figure 5 shows a plot of the 32 statistical features re-scaled using min-max normalization. Among the plotted features, *nb_A* and *nb_W* are respectively the number of route announcements and withdrawals received during the snapshotting interval. First, between 03:22 UTC and 03:27 UTC a lot of announcements are received due to the routes leak. Then, between 03:31 UTC and 03:36 UTC a second wave of withdrawals and announcements arrived due to Google team fixing the error. While this work doesn't focus on feature evaluation, the data generated by BML clearly shows that these features are relevant for anomaly detection.

BML also implements data transformation that generates a graph structure on which are computed a total of 14 graph features. Using $\Delta = 1$ minute, this transformation took 7 minutes and 47 seconds to complete[2]. The computation of these graph features are much more complex compared to the statistical features. Figure 6 shows a plot of these features re-scaled using min-max normalization. The impact of the BGP anomaly seems to be better rendered with the graph features than with the statistical features.

[1] https://bgpmon.net/bgp-leak-causing-internet-outages-in-japan-and-beyond/

[2] During our test case, we used a computer with the following specification: 3.7GHz/8 cores processor with 64 Gb of RAM running Ubuntu 18.04

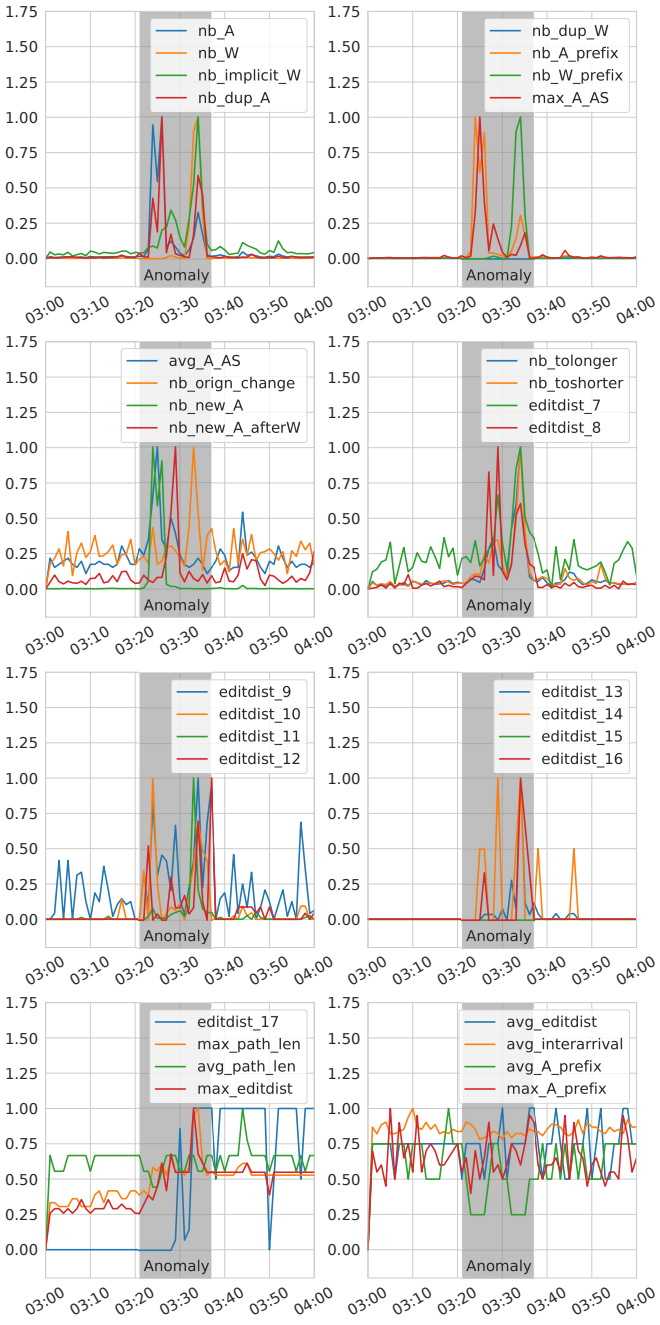


Fig. 5: Statistical features during Google leak.

Modern ML models tend to rely on multiple features. To illustrate the relevance of BML in this context, we computed a principal component analysis (PCA) on all of the 32 statistical features (resp. all of the 14 graph features) which are then plotted on two dimensions. Figure 7 shows the 2d projection of the PCA which allows to visualize the separability of the data during the normal behavior of BGP and during an anomaly. The data within the interval 03:22 UTC and 03:37 UTC are labeled as anomalous. We can see that the non-anomalous data tend to be projected close to each other whereas the anomalous data are more randomly distributed. The non-anomalous projection generated from the graph features appears to be more compact than the one generated from the statistical features. With the ability of BML to extract several features, the development of ML models relying on multiple features is simplified.

A major benefit of BML is the ability for a user to easily define its own data transformation function. To demonstrate this functionality of BML, suppose we want to compute the number of announcements received during an interval where the Google AS (AS 15169) is in the AS-PATH. As shown on Figure 8 this transformation can be implemented in only 10 lines of Python code. We computed this transformation every minute and it took only 7 seconds to complete[2]. Figure 9 shows a plot of this transformation and we observe a similar pattern as the statistical features.

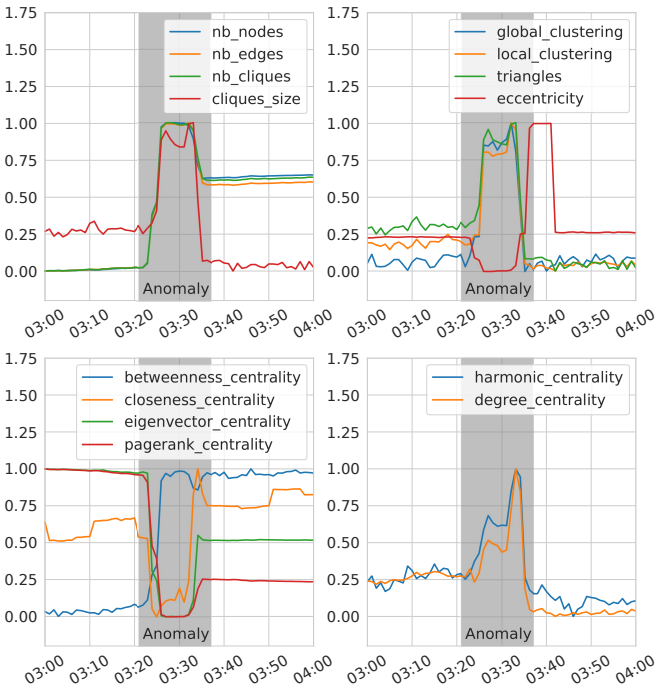


Fig. 6: Graph features during Google leak.

## V. Perspective and Conclusion

This paper introduced BML[3], a tool for BGP dataset's collection used to study BGP anomalies. BML eases the development of machine learning models by automating the dataset collection for BGP researchers and allowing the ML-practitioners to focus on the training and optimization of their models. While similar tools were limited to the collection of statistical features dataset, BML stores the data in a loss-less yet affordable data structure. BML then allows to generate 32 of the most used statistical features and 14 graph features. Users can also leverage the data transformation helper function to create their custom features without the need to recollect

[3]https://github.com/KevinHoarau/BML

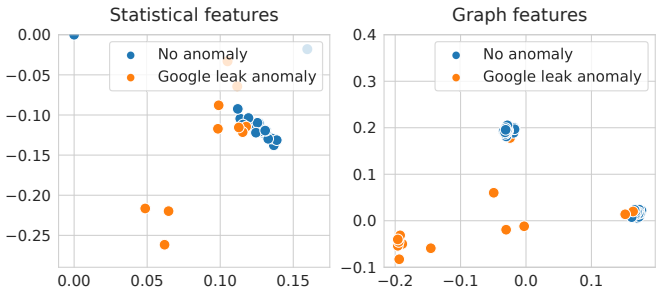


Fig. 7: 2D projection of features using PCA (Principal Component Analysis).

```python
from BML.transform import BaseTransform

class GoogleRoutes(BaseTransform):

 computeRoutes = False

 def transforms(self, index, routes, updates):
  n = 0
  for update in updates:
    if update["type"]=='A':
      if "15169" in update["fields"]["as-path"]:
        n += 1
  return(n)
```

Fig. 8: Example of user-defined data transformation.

the data. We illustrated BML's capability on the BGP anomaly induced by the Google leak of 2017. It included a principal component analysis that showed that machine learning tools using multiple features may be promising, emphasizing the need to easily generate a wide variety of features in a short amount of time. Tools such as BML may also help researcher to share datasets and promote reproducibility and replicability.

It is worth noting that BML is the first data collection tool that allows to generate graph representation. These have recently been shown promising to enhance the performance of BGP anomaly detection models and our evaluation on Google's BGP leak showed similar insight.

## References


[1] B. Al-Musawi, P. Branch, and G. Armitage, "BGP anomaly detection techniques: A survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 377–396, FebJan 2017. [Online]. Available: http://dx.doi.org/10.1109/comst.2016.2622240

[2] RouteViews, "Routeviews - university of oregon route views project." [Online]. Available: http://www.routeviews.org/routeviews/

[3] RIPE, "Routing information service (RIS)." [Online]. Available: https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris/routing-information-service-ris

[4] P. Fonseca, E. S. Mota, R. Bennesby, and A. Passito, "Bgp dataset generation and feature extraction for anomaly detection," in *2019 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2019, pp. 1–6.

[5] P. Goyal, N. Kamra, X. He, and Y. Liu, "Dyngem: Deep embedding method for dynamic graphs," *arXiv preprint arXiv:1805.11273*, 2018.

[6] S. Mahdavi, S. Khoshraftar, and A. An, "Dynamic joint variational graph autoencoders," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2019, pp. 385–401.


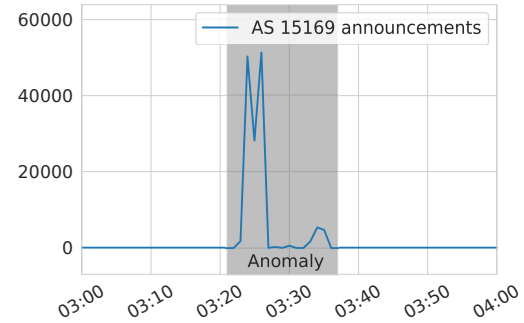


Fig. 9: AS 15169 announcements during Google leak.


[7] Y. Han and Y. Shen, "Partially supervised graph embedding for positive unlabelled feature selection." in *IJCAI*, 2016, pp. 1548–1554.

[8] O. R. Sanchez, S. Ferlin, C. Pelsser, and R. Bush, "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*, 2019, pp. 35–41.

[9] L. Gao, "On inferring autonomous system relationships in the internet," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 733–745, Dec 2001.

[10] S. H. Y. Rekhter, T. Li, "A border gateway protocol 4 (bgp-4)," Network Working Group, IETF, RFC 4271, 2006. [Online]. Available: https://tools.ietf.org/html/rfc4271

[11] K. Butler, T. R. Farley, P. McDaniel, and J. Rexford, "A survey of bgp security issues and solutions," *Proceedings of the IEEE*, vol. 98, no. 1, pp. 100–122, 2009.

[12] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding bgp misconfiguration," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 3–16, 2002.

[13] J. H. Cowie, A. T. Ogielski, B. Premore, E. A. Smith, and T. Underwood, "Impact of the 2003 blackouts on internet communications," *Preliminary Report, Renesys Corporation (updated March 1, 2004)*, 2003.

[14] L. Wang, X. Zhao, D. Pei, R. Bush, D. Massey, A. Mankin, S. F. Wu, and L. Zhang, "Observation and analysis of bgp behavior under stress," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, 2002, pp. 183–195.

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

[16] C. Orsini, A. King, D. Giordano, V. Giotsas, and A. Dainotti, "Bgpstream: a software framework for live and historical bgp data analysis," in *Proceedings of the 2016 Internet Measurement Conference*, 2016, pp. 429–444.

[17] Q. Ding, Z. Li, P. Batta, and L. Trajković, "Detecting bgp anomalies using machine learning techniques," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2016, pp. 003 352–003 355.

[18] Y. Li, H. J. Xing, Q. Hua, X. Z. Wang, P. Batta, S. Haeri, and L. Trajković, "Classification of bgp anomalies using decision trees and fuzzy rough sets," in *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2014, pp. 1312–1317.

[19] M. Cheng, Q. Li, J. Lv, W. Liu, and J. Wang, "Multi-scale lstm model for bgp anomaly classification," *IEEE Transactions on Services Computing*, 2018.

[20] M. Cosovic, S. Obradovic, and E. Junuz, "Deep learning for detection of bgp anomalies," in *International Work-Conference on Time Series Analysis*. Springer, 2017, pp. 95–113.

[21] I. O. de Urbina Cazenave, E. Köşlük, and M. C. Ganiz, "An anomaly detection framework for bgp," in *2011 International Symposium on Innovations in Intelligent Systems and Applications*, June 2011, pp. 107–111.

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