Apprentissage automatique pour la détection d'anomalies dans les graphes issus des données réseau

Kévin Hoarau

UNIVERSITÉ DE LA RÉUNION

Thèse de Doctorat de l'Université de la Réunion

Spécialité

INFORMATIQUE/MATHÉMATIQUE

présentée par

M. Kévin HOARAU

pour obtenir le grade de

DOCTEUR de l'Université de la Réunion

Apprentissage automatique pour la détection d'anomalies dans les graphes issus des données réseau.

Soutenue publiquement le 9 septembre 2022 devant le jury composé de

Pr. Géraldine TEXIER	**Rapporteur**
Dr. HDR. Prométhée SPATHIS	**Rapporteur**
Pr. Emmanuel LOCHIN	**Examinateur**
Dr. HDR. Vania CONAN	**Examinateur**
Pr. Etienne PAYET	**Directeur de thèse**
Pr. Dali KAAFAR	**Co-directeur de thèse**
Dr. Pierre Ugo TOURNOUX	**Co-encadrant de thèse**
Dr. HDR. Tahiry RAZAFINDRALAMBO	**Co-encadrant de thèse**

Cette thèse a reçu le soutien financier de la Région Réunion et de l'Union Européenne (Fonds Européen de Développement Régional - FEDER) dans le cadre du Programme Opérationnel de Coopération Territoriale.

REGION REUNION
www.regionreunion.com

À ma fille

Remerciements

Je tiens à remercier M. Pierre-Ugo Tournoux pour la qualité de son encadrement, mais aussi pour ses qualités humaines qui ont fait de lui un soutien sans failles tout au long de cette thèse. Je le remercie également de s'être toujours rendu disponible et de m'avoir apporté un environnement de travail des plus agréable. Enfin, je remercie à la fois, M. Pierre-Ugo Tournoux et M. Tahiry Razafindralambo pour tout le temps qu'ils m'ont accordé, leurs contributions à mes travaux de thèse, et leurs conseils avisés.

Je remercie M. Étienne Payet d'avoir accepté de diriger cette thèse, mais aussi pour son soutien et tous les bons conseils qu'il m'a apportés avant, pendant, et j'espère après ma thèse.

Je remercie M. Dali Kaafar pour avoir codirigé cette thèse et pour m'avoir accueilli au sein de son unité de recherche durant 4 mois. Cette expérience en plus des bénéfices qu'elle a apportés à mes travaux de thèse, m'a également beaucoup apporté sur le plan personnel et professionnel. Plus généralement, je remercie l'ensemble de l'équipe de l'Optus Macquarie University Cyber Security Hub pour son accueil.

Je tiens à remercier, l'ensemble de l'équipe du Laboratoire d'Informatique et de Mathématiques de l'Université de la Réunion pour m'avoir permit d'effectuer cette thèse dans de bonnes conditions.

Je remercie également l'équipe de l'École Supérieure d'Ingénieurs Réunion Océan Indien dans laquelle j'ai occupé le poste d'ATER durant deux ans. Cette expérience m'a permis de me conforter dans mon projet de carrière d'enseignant-chercheur.

Je remercie M. Dominique Gay pour le temps qu'il m'a accordé durant mon stage de fin de master et qui m'a conforté dans mon choix de poursuivre mes études dans un doctorat. Je remercie également, M. Pascal Anelli, M. Jean Diatta, M. Remy Courdier et M. Denis Payet pour m'avoir soutenu dans ce choix.

Je tiens à remercier M. Thierry Rakotoarivelo et M. Emmanuel Lochin d'avoir pris part aux comités de suivi de thèses et pour la pertinence de leurs remarques et conseils.

Je remercie, mes collègues doctorants et amis pour leur soutien, leurs discussions passionnées et leur bonne humeur. Merci Fonenantsoa, Irène, Mandimby, Nathan, Réhan, Richard, Taher et Tahina.

Je tiens à remercier mes parents qui m'ont toujours encouragé dans tous mes projets professionnels et personnels. Je remercie également mes deux grandes soeurs pour leur bienveillance et leur soutien. Je tiens également à remercier mes beaux-parents pour

toutes les heures de baby-sitting sans lesquelles cette thèse n'aurait pu exister.

Enfin, je remercie ma femme et ma fille pour tout le bonheur qu'elles m'apportent au quotidien et qui m'a permis de garder le moral malgré les difficultés qu'une thèse apporte. Je remercie également ma femme pour les nombreuses heures passées à la relecture de ce manuscrit.

Résumé

L'analyse des réseaux, de leurs protocoles et applications est impactée par l'évolution rapide des méthodes d'apprentissage automatique. Par ailleurs, bien que les données de ce domaine d'application soient intrinsèquement liées aux représentations sous forme de graphe, ce sont des représentations tabulaires qui sont généralement utilisées par les techniques d'apprentissage automatique ce qui, par conséquent, ne permet pas de représenter toute la complexité de ces données. Cette thèse s'intéresse à l'exploitation des graphes des données réseau à l'aide de techniques d'apprentissage automatique. Il est notamment proposé d'intégrer et d'évaluer les avancées récentes dans le domaine des *Graph Neural Networks* (GNN). Le cadre applicatif retenu est celui de la détection d'anomalies dans le *Border Gateway Protocol* (BGP), protocole qui génère des graphes massifs et complexes dans lesquels les anomalies sont difficilement décelables. Ce protocole constitue l'épine dorsale de l'Internet ce qui justifie que ses anomalies aient été largement étudiées par la communauté, que ce soit via des règles expertes ou des méthodes d'apprentissage automatique classiques.

Un travail préalable identifie que contrairement aux principaux domaines d'application de l'apprentissage automatique, il n'y a pas de jeux de données de référence pour l'étude des anomalies BGP. En outre, la construction de ces derniers apparaît pénible et constitue un frein à la recherche dans ce domaine. Ainsi, BML, un outil pour la construction de jeu de données BGP est proposé. La première contribution de cette thèse met en exergue le fait que l'exploitation d'attributs extraits d'un graphe BGP permet d'y détecter une anomalie avec des performances conformes à l'état de l'art. C'est le cas pour les anomalies de grande échelle (*accuracy* de 88%) mais il permet également d'améliorer significativement les performances sur les anomalies de petite échelle ($+18\%$ d'*accuracy*). Dans une seconde contribution, la composante temporelle est intégrée par l'utilisation d'un réseau de neurones récurrent (RNN). À partir d'une séquence de graphes BGP, une série temporelle d'attributs est extraite puis consommée par ce modèle. Cependant, il apparaît que la perte d'information induite par l'extraction d'attributs du graphe BGP nuit aux performances. Dans une dernière contribution, ce problème est contourné par l'utilisation d'un GNN qui exploite directement les graphes sans étape préalable d'extraction d'attributs. Par construction, ce modèle offre également une granularité fine qui a permis de détecter une anomalie au niveau d'un AS avec une *accuracy* de 96% sur des évènements de grande échelle. À notre connaissance, il s'agit du premier modèle basé sur un GNN pour la détection d'anomalies BGP.

Ces travaux ont mis en évidence la pertinence des représentations sous forme de graphe pour l'analyse des données issues de BGP. Néanmoins, les GNN ouvrent davantage de perspectives que celles étudiées dans cette thèse. Notamment, l'identification du noeud à l'origine d'une attaque ou encore la prédiction de l'impact d'une anomalie.

Mots-clés :
Apprentissage automatique, Détection d'anomalie, Graph Neural Networks, BGP

Abstract

The analysis of networks, their protocols and applications is impacted by the fast evolution of machine learning methods. Moreover, although the data in this application domain are intrinsically related to graph representations, tabular representations are generally used by machine learning techniques, which consequently do not allow to represent all the complexity of these data. This thesis focuses on the exploitation of graphs of network data using machine learning techniques. In particular, it is proposed to integrate and evaluate recent advances in the field of Graph Neural Networks (GNN). The application domain is the detection of anomalies in the Border Gateway Protocol (BGP), a protocol that generates massive and complex graphs in which anomalies are difficult to detect. This protocol is the backbone of the Internet, which justifies that its anomalies have been widely studied by the community, either via expert rules or classical machine learning methods.

A prior work identifies that unlike the main application domains of machine learning, there are no baseline datasets for the study of BGP anomalies. Moreover, the construction of these datasets appears laborious and is a barrier to the research in this area. Therefore, BML, a tool for the construction of BGP datasets is proposed. The first contribution of this thesis highlights the fact that the exploitation of features extracted from a BGP graph allows to detect an anomaly with performances in accordance with the state of the art. This is the case for large scale anomalies (accuracy of 88%) but it also allows to significantly improve the performances on small scale anomalies ($+18\%$ of accuracy). In a second contribution, the temporal component is integrated by using a recurrent neural network (RNN). From a sequence of BGP graphs, a temporal series of features is extracted and consumed by this model. However, it appears that the loss of information induced by the extraction of features from the BGP graph is detrimental to the performance. In a last contribution, this problem is overcome by using a GNN that directly exploits the graphs without any feature extraction step. By construction, this model also provides a fine granularity that made it possible to detect an anomaly at the level of an AS with an accuracy of 96% on large scale events. To our knowledge, this is the first GNN-based model for BGP anomaly detection.

This work has highlighted the relevance of graph representations for the analysis of BGP data. Nevertheless, GNNs open more perspectives than those studied in this thesis. In particular, the identification of the node at the origin of an attack or the prediction of the impact of an anomaly.

Keywords:
Machine Learning, Anomaly detection, Graph Neural Networks, BGP

Table des matières

Chapitre 1

Introduction

Sommaire

Ce chapitre introduit les principaux défis soulevés par cette thèse. Il présente tout d'abord le contexte de cette thèse qui est l'application des techniques d'apprentissage automatique aux graphes. La problématique étant de déterminer l'intérêt des représentations sous forme de graphe par rapport aux représentations habituellement utilisées pour les données réseau. Puis il présente le cas d'application, la détection d'anomalies sur le *Border Gateway Protocol* (BGP). Les contributions qui vont de l'élaboration d'un outil de collecte et de transformation de données jusqu'à la proposition d'une architecture de réseaux de neurones pour la détection d'anomalies BGP sont également introduites. Enfin, l'organisation de ce document est présentée.

1.1 Contexte

La collecte de données massives, l'augmentation des capacités de calculs et des percées algorithmiques ont permis des progrès significatifs dans le domaine de l'apprentissage automatique durant cette dernière décennie. C'est plus particulièrement les techniques d'apprentissage profond ou *deep learning* qui ont amené des progrès importants en termes de performance dans divers domaines tels que la vision par ordinateur [1] et la reconnaissance du langage naturel [2]. Ces succès étant principalement dus à l'apparition de nouvelles architectures de réseaux de neurones avec notamment les réseaux de neurones convolutifs et les réseaux de neurones récurrents.

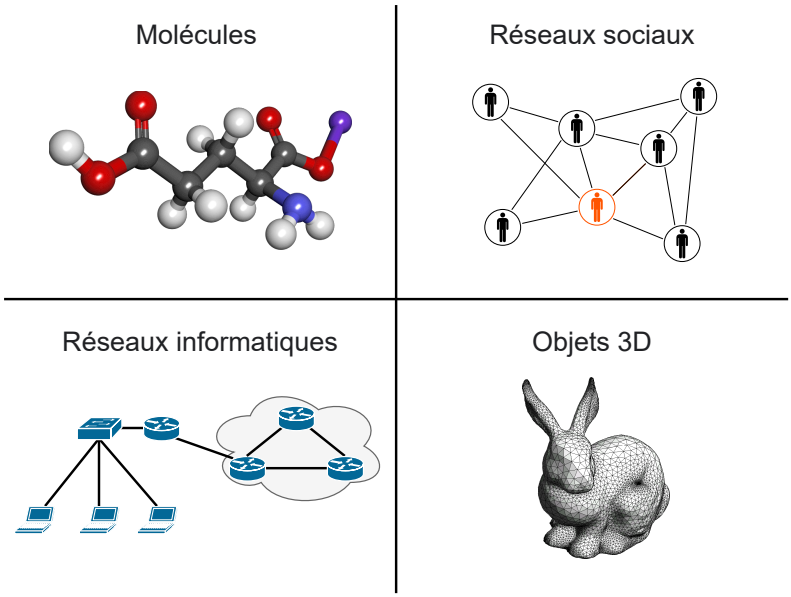


FIGURE 1.1 – Exemples de données non-euclidiennes

Les techniques d'apprentissage automatique sont particulièrement adaptées aux données dites euclidiennes, c'est-à-dire possédant une structure sous-jacente sous forme de grille. Les données tabulaires où chaque colonne correspond à un attribut et chaque ligne à une observation sont les plus fréquentes. Les images sont également un exemple de données euclidiennes qui peuvent être vues comme une grille à 2 dimensions. Toutefois, dans de nombreux domaines, des données dites non-euclidiennes [3] doivent être manipulées. Les graphes et variétés sont les exemples les plus courants de données non-euclidiennes. De telles données sont présentes dans divers domaines, par exemple, les réseaux sociaux, les réseaux de télécommunications ou l'infographie pour la représentation d'objet 3D (voir figure 1.1).

Cette thèse s'intéresse aux graphes qui est la forme la plus courante de données non-euclidiennes. Afin d'appliquer les techniques d'apprentissage automatique aux graphes il est nécessaire d'extraire des représentations qui leur sont adaptées. Une première solution consiste à extraire des métriques issues de la théorie des graphes [4] ou bien définies grâce à des connaissances expertes sur le domaine d'application [5]. Récemment, des approches de *representation learning* [6] ont émergées dont le but est de traiter le problème

d'extraction de représentations comme un problème d'apprentissage automatique plutôt que comme un pré-traitement. L'objectif étant d'apprendre un plongement *embedding* où chaque noeud du graphe est transformé en un vecteur de dimension d. Ce plongement pouvant être appris de manière à ce que les relations géométriques dans l'espace de plongement reflètent la structure du graphe (voir figure 1.2). Il peut également être appris en fonction de la tâche finale dans une architecture de bout-en-bout. Ce type d'architecture de réseaux de neurones pouvant ingérer des données représentées sous forme de graphes est désigné sous le terme *Graph Neural Network* (GNN) (voir section 2.5.1) et reçoit de plus en plus d'attention.

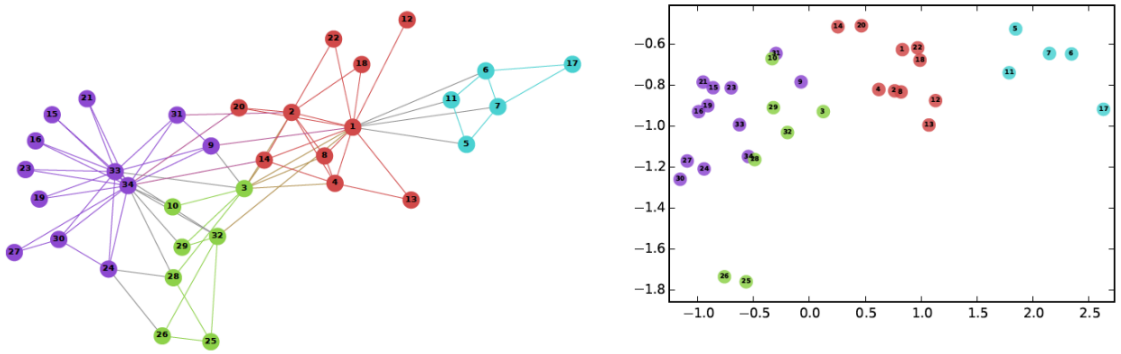


FIGURE 1.2 – Exemple de plongement de graphe [6]. **À gauche :** graphe du réseau social *Zachary's karate club*, où les noeuds sont connectés si les individus correspondants sont amis. Les noeuds partagent la même couleur s'ils appartiennent à la même communauté. **À droite :** visualisation 2D des plongements des noeuds générés à l'aide de la méthode DeepWalk [7]. Les distances entre les noeuds reflètent la similarité dans le graphe d'origine.

1.2 Problématique

Dans le cadre des réseaux informatiques, les graphes sont une représentation particulièrement adaptée. En effet, il permettent de représenter les entités qui composent un réseaux ainsi que leur interactions. Par exemple, les équipements d'un réseau local (hôtes, commutateurs, routeurs, etc) peuvent être représentés sous forme de graphe. A l'échelle de l'Internet, le graphe BGP (voir chapitre 2.2.3) permet de représenter les liens entres les réseaux (*i.e* Fournisseurs d'accès à Internet, Universités, etc) qui le composent.

Cependant, l'application des techniques d'apprentissage automatique aux réseaux tire rarement profit de cette représentation. Généralement, des attributs spécifiques au type de réseau étudié sont extraits (cf. chapitre 3.3.4). Un exemple d'attribut est le nombre d'annonce de route émisses dans une fenêtre temporelle fixe. L'extraction de ce type d'attributs implique naturellement une perte d'information par rapport à une représentation complexe des réseaux sous forme de graphes. De plus, les graphes étant une représentation plus générique que ces attributs spécifiques, ils permettent le transfert des solutions développées d'un domaine d'application à un autre. Par exemple, une même architecture de réseaux de neurones pourrait être utilisé sur des réseaux différents. Grâce à l'apprentissage par transfert [8], il est envisageable de pré-entraîner un modèle sur un réseau pour lequel on dispose de beaucoup de données avant de l'appliquer à un autre réseau.

Cette thèse se concentre sur la détection d'anomalies dans les données issues des réseaux. L'approche proposée consiste à exploiter les techniques d'apprentissage automatique et plus précisément de tirer profit des avancées récentes dans le domaine des *Graph Neural Networks* (GNN). Ainsi, deux questions sont soulevées :

1. Quel est l'apport des représentations sous forme de graphe par rapport aux représentations habituellement utilisées ?
2. Les GNN permettent-ils de tirer profit de ces représentations sous forme de graphes ?

1.3 Cas d'application : le protocole BGP

Il est difficilement envisageable de vouloir apporter une réponse générique aux questions soulevées ci-dessus pour l'ensemble des réseaux informatiques. Ainsi, cette thèse se restreint à un cas largement étudié avec les représentations classiques en dépit d'une structure en graphe évidente, le *Border Gateway Protocol* (BGP). Pour ce domaine d'application spécifiquement, elle vise à répondre aux questions soulevées ci-dessus.

Le *Border Gateway Protocol* (BGP) est l'épine dorsale d'Internet permettant l'interconnexion de l'ensemble des entités indépendantes qui le compose. Chacune de ces entités, que ce soit par exemple une entreprise ou un fournisseur d'accès à Internet, est appelée un système autonome ou *autonomous system* (AS). Un AS est identifié par un numéro unique, son ASN (*Autonomous System Number*) et dispose de connexions avec d'autres AS clients, fournisseurs, etc (cf. figure 1.3). Une des forces de BGP étant ainsi qu'il permet aux AS d'implémenter leur propre politique de routage en fonction des accords commerciaux avec les AS voisins. Par exemple, un AS ne souhaite probablement pas annoncer à un fournisseur une route provenant d'un autre de ses fournisseurs, car il se verrait alors facturé pour du trafic pour lequel il n'obtient aucun revenu. Deux autres raisons de l'adoption de BGP en tant que protocole de routage global sont son passage à l'échelle dû à son fonctionnement incrémental et le fait qu'il évite l'apparition de boucle de routage. Ce dernier point étant supporté par le fait que BGP est un protocole de routage à vecteur de chemin c'est-à-dire que chaque route est décrite par la séquence des AS à traverser pour atteindre la destination. Ainsi, si un AS voit son propre ASN dans une route qui lui est annoncée alors il n'utilisera pas cette route, car cela créerait une boucle de routage. Une conséquence de cette propriété de BGP est que chaque routeur BGP peut utiliser l'ensemble des routes dont il dispose pour constituer un graphe. En collectant des routes à partir de plusieurs routeurs, il est alors possible d'obtenir une vision partielle de la topologie globale du réseau BGP sous forme de graphe.

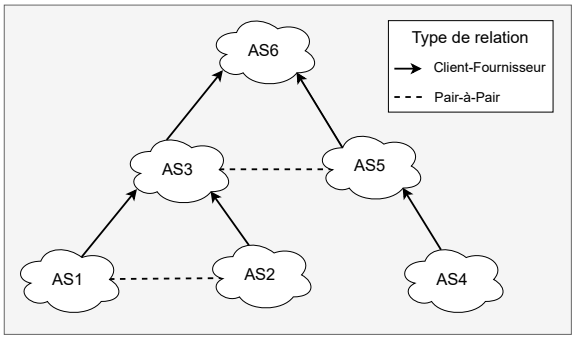


FIGURE 1.3 – Représentation schématique d'un réseau BGP

Toutefois, malgré sa position centrale dans l'interconnexion des réseaux au niveau mondial, BGP souffre de vulnérabilités intrinsèques. Notamment, il est difficile de vérifier en temps réel si une annonce de route provient bien de l'AS propriétaire du préfixe annoncé. Ce type d'attaque étant connu sous le terme de détournement de préfixe ou *prefix hijacking*. Bien que des solutions, telles que la *Resource Public Key Infrastructure* (RPKI),

soient de plus en plus adoptées, elles ne permettent pas de prévenir tous les problèmes de sécurité et d'instabilité dont BGP peut être victime. Ces problèmes sont généralement réunis sous le terme d'anomalies BGP qui regroupe à la fois des évènements intentionnels et non intentionnels tels que des défaillances matérielles. Afin d'être en mesure de réagir au plus vite face à ces anomalies, de nombreux travaux se sont intéressés à leur détection. D'une part, des solutions ad hoc ont été proposées principalement pour la détection des détournements de préfixe. D'autre part, des techniques d'apprentissage automatique ont été utilisées en s'intéressant principalement à des anomalies de plus grande échelle. Ces dernières s'appuyant généralement sur des attributs similaires qui sont des mesures statistiques telles que le nombre de routes annoncées dans un intervalle de temps fixe. Plus récemment, de nouveaux types d'attributs ont été proposés avec l'extraction de mesures à partir du graphe BGP telles que des métriques issues de la théorie des graphes. Bien que cette nouvelle approche offre des performances intéressantes sur les anomalies étudiées, elle n'a pas été comparée avec les attributs statistiques plus généralement utilisés. Cela est notamment la conséquence de l'absence de jeu de données de référence dans le domaine et donc de la difficulté de reproductibilité des résultats.

Objectifs

Dans le contexte qui vient d'être présenté, les objectifs de cette thèse sont donc les suivants :

1. Faciliter et encourager la reproductibilité des résultats pour la détection des anomalies BGP.
2. Identifier l'intérêt des représentations des données BGP sous forme de graphe par rapport aux attributs généralement utilisés.
3. Déterminer si les GNN permettent de tirer profit des représentations des données BGP sous forme de graphe.

1.4 Contributions

Les principales contributions de cette thèse répondent directement aux objectifs énoncés ci-dessus.

Tout d'abord, un premier travail propose BML, un outil permettant de faciliter la construction et l'analyse de jeux de données BGP. Il automatise les tâches fastidieuses du processus de construction d'un jeu de données et met à disposition un ensemble de transformations pertinentes pour ce type de données. Par effet de bord, BML facilite grandement la reproductibilité des résultats pour la détection des anomalies BGP. L'outil, ainsi que divers jeux de données d'anomalies BGP ont été rendu publiquement disponible durant cette thèse.

La seconde contribution de cette thèse évalue la pertinence des représentations des données BGP sous forme de graphes pour la détection d'anomalies. Des outils statistiques et des méthodes d'apprentissage automatique sont utilisés pour comparer les attributs fréquemment utilisés dans la littérature et ceux issus du graphe BGP. Cette comparaison est faite aussi bien sur des anomalies de grandes échelles (*e.g.* fuites de routes) que sur des anomalies de plus petites échelles (*e.g.* détournement de préfixes). Ce travail met en évidence l'intérêt des représentations sous forme de graphes qui offrent à la fois de bonnes performance sur les anomalies de grande échelle (*accuracy* de 88%) tout en les améliorant sur des anomalies de petite échelle (+18% d'*accuracy*).

Une troisième contribution propose et évalue un modèle de réseau de neurones récurrent (RNN) permettant détecter une anomalie dans une séquence d'attributs du graphe BGP. A défaut d'améliorer significativement les performances, ce travail permet de mettre en évidence que les données d'apprentissage semblent être la principale limitation pour l'amélioration des performances du modèle proposé. Deux pistes d'améliorations sont identifiées. Premièrement, l'entraînement des modèles sur de plus gros jeux de données. Et deuxièmement, l'utilisation de représentations moins bruitées que les attributs du graphe.

Enfin, dans la dernière contribution le graphe BGP est directement consommé par un *Graph Neural Networks* (GNN). Cela permet de supprimer la coûteuse étape de génération des attributs mais également la perte d'information qui en découle. Le GNN permet de générer les séquences de plongements des AS, lesquels sont ensuite classifiés avec un MLP. L'évaluation montre une *accuracy* de 96%, suggérant ainsi que les GNN constituent un outil pertinent pour la détection d'anomalies BGP. Par ailleurs, en dépits de la complexité du graphe BGP appliqué en entrée, les plongements appris par le GNN mettent clairement en exergue la présence d'anomalies.

1.5 Organisation du document

Cette thèse s'articule en 7 chapitres dont cette introduction en est le premier et la conclusion le dernier.

Dans le second chapitre, un état de l'art est effectué en présentant tout d'abord quelques caractéristiques fondamentales du protocole BGP. Puis, quelques définitions sur la théorie des graphes et le graphe BGP sont présentées. Enfin, après un tour d'horizon sur les anomalies BGP et la littérature autour de leurs détections, les techniques d'apprentissage profond sur les graphes dont font partie les GNN sont abordées.

Le troisième chapitre présente BML, un outil élaboré durant cette thèse. Tout d'abord, les besoins auxquels répond cet outil et son principe de fonctionnement sont détaillés. Ce chapitre définit également les différentes transformations de données implémentées dans BML telles que les attributs statistiques et les attributs du graphe BGP. Pour finir, des cas d'application concrets de BML sont proposés et détaillés dans leur mise en oeuvre.

Dans le quatrième chapitre, une étude de la pertinence des représentations sous forme de graphe de données BGP par rapport aux attributs statistiques est présentée. Pour cela, un jeu de données est construit à l'aide de BML. Ce jeu de données incluant à la fois des anomalies de grande et de plus petite échelle. Une analyse statistique et des visualisations de ce jeu données sont ensuite présentées. Enfin, plusieurs algorithmes d'apprentissage automatique sont comparés et une conclusion sur l'intérêt de l'exploitation des graphes BGP pour la détection d'anomalies est faite.

Le cinquième chapitre aborde l'exploitation de la composante temporelle des données BGP pour la détection des anomalies. Dans ce contexte, il se concentre sur les attributs du graphe BGP et sur des anomalies de petite échelle à savoir des détournements de chemins. Ce travail permet de conclure que malgré l'utilisation d'un modèle plus complexe permettant de tirer profit de la composante temporelle des données, les performances ne sont pas augmentées par rapport à un modèle plus simple présenté au chapitre 4. Ainsi, deux pistes d'améliorations peuvent être identifiées. Premièrement, la collecte de jeux de données de grandes dimensions c'est-à-dire avec plus d'évènements d'anomalies BGP. Deuxièmement, l'utilisation de représentation moins sujette au bruit avec par exemple l'exploitation du graphe BGP sans étape d'extraction d'attributs, mais aussi en s'orientant vers des approches localisées visant à détecter une anomalie sur un sous-ensemble du graphe.

Dans le sixième chapitre, BGNN, un modèle de réseau de neurones basé sur un GNN pour la détection d'anomalies dans une séquence de graphes BGP est présenté. Ce modèle prend en entrée une séquence de graphes BGP et utilise une approche localisée afin de détecter si un AS a été victime d'une anomalie dans cette séquence. Le modèle est évalué sur plusieurs évènements de grande échelle, ce qui permet de démontrer la capacité des GNN d'identifier une anomalie dans un graphe BGP. De plus, une analyse du modèle permet de visualiser les motifs capturés par ce dernier. Enfin, les performances en matière de temps de détection de BGNN sont évaluées.

Le septième et dernier chapitre conclut cette thèse et présente quelques perspectives.

Chapitre 2

Etat de l'art

Sommaire

Ce chapitre présente un état de l'art recouvrant les deux axes principaux de cette thèse : la détection d'anomalies BGP et l'application des techniques d'apprentissage automatique aux graphes. Le protocole BGP, ses différents types d'anomalies et les solutions existantes pour leur détection sont alors développés. Il présente également quelques rappels sur les graphes et sur l'extraction du graphe BGP avant d'effectuer un tour d'horizon sur l'application des techniques d'apprentissage profond aux graphes.

2.1 Le protocole BGP

Border Gateway Protocol

Le réseau Internet est composé d'un ensemble de sous-réseaux indépendants appelés systèmes autonomes ou AS (*Autonomous Systems*). Chaque AS correspondant à une entité administrative telle qu'un fournisseur d'accès à Internet, une université ou une grande entreprise. Le protocole BGP pour *Border Gateway Protocol*, est le protocole de routage permettant aux AS d'échanger des informations de routage. Au cours du temps, BGP a connu plusieurs modifications et en est actuellement à sa quatrième version qui est définie dans la RFC 4271 [9].

Cette section effectue un tour d'horizon du protocole BGP afin d'en présenter les aspects nécessaires à la compréhension de ces travaux de thèse.

2.1.1 Préfixes et ASN

Dans le réseau BGP, chaque AS est identifié par un numéro d'AS ou *Autonomous System Number* (ASN). Depuis la RFC 6793 [10] les ASN sont encodés sur quatre octets. L'*Internet Assigned Numbers Authority* (IANA) qui est en charge de l'allocation des ASN a réservé les plages 64512 à 65534 et 4200000000 à 4294967294 pour des usages privés. Chaque AS dispose également d'une ou de plusieurs plages d'adresses IP appelées préfixes. Depuis sa version 4, BGP dispose d'un ensemble de mécanismes pour le support du *Classless Inter-Domain Routing* (CIDR) [11]. La notation CIDR est donc utilisée pour définir la plage d'adresse couverte par un préfixe. Ainsi, le préfixe 192.168.1.0/24 correspond à l'ensemble des adresses IP comprises entre 192.168.1.0 et 192.168.1.255. Comme pour les ASN, c'est l'IANA qui est en charge de l'allocation des préfixes.

Afin d'alléger sa tâche, l'IANA délègue l'allocation des ASN et des préfixes à cinq Registres Internet Régionaux (RIR) : l'*African Network Information Center* (AFRINIC) pour le continent africain, l'*Asia Pacific Network Information Center* (APNIC) pour l'Asie et la région Pacifique, l'*American Registry for Internet Numbers* (ARIN) pour les États-Unis et le Canada, le *Latin America and Caribbean Network Information Center* (LACNIC) pour l'Amérique latine et les Caraïbes et le *Réseaux IP Européens Network Coordination Center* (RIPE NCC) pour l'Europe et le Moyen-Orient.

2.1.2 Caractéristiques principales

Quelques caractéristiques principales de BGP le différencie d'autres protocoles de routage. Tout d'abord, BGP est un protocole à vecteur de chemin c'est-à-dire qu'une route vers un préfixe est définie par l'ensemble des AS qui participent à l'acheminement du trafic. Cet ensemble, appelé AS-PATH est construit durant le processus de propagation de

la route durant lequel chaque AS traversé va ajouter son ASN à l'AS-PATH. Un avantage majeur de ce mécanisme est qu'il empêche l'apparition de boucles de routage étant donné qu'un AS va ignorer toute route contenant son propre ASN.

Deuxièmement, la mise à jour des routes BGP fonctionne par incrémentation. En effet, une fois que l'ensemble des routes a été échangé par les routeurs BGP, seules des mises à jour de route sont transmises. Cela permet d'économiser de la bande passante. De plus, depuis la version 4 du protocole, le mécanisme d'agrégation permet d'économiser encore plus de bande passante et de limiter la taille des tables de routage en combinant plusieurs routes en une seule [9].

Enfin, BGP étant un protocole pour l'interconnexion de différentes entités commerciales, son rôle en plus d'assurer le fonctionnement du réseau est de respecter les accords commerciaux établis entre ces entités. Ces accords commerciaux sont généralement classés dans trois catégories [12] : *customer-to-prodiver* (c2p) où un AS client paie un AS fournisseur pour acheminer son trafic, *peer-to-peer* (p2p) quand deux AS acheminent mutuellement leur trafic gratuitement et *sibling-to-sibling* (s2s) lorsque deux AS appartiennent à la même entité. Afin que les opérateurs d'AS puissent implémenter ces accords commerciaux, BGP leur permet de définir leur politique de routage par le biais de règles d'importation et d'exportation des routes.

2.1.3 Fonctionnement de BGP

Routing Information Base

Un routeur BGP conserve l'ensemble des informations de routage à sa disposition dans sa *Routing Information Base* (RIB) qui est divisée en trois parties distinctes : l'Adj-RIBs-In, la Loc-RIB et l'Adj-RIBs-Out. L'Adj-RIBs-In contient les informations de routage non traitées en provenance des routeurs BGP voisins. La Loc-RIB correspond aux routes utilisées par le routeur pour l'acheminement du trafic. Ces routes sont sélectionnées en appliquant la politique de routage locale à l'Adj-RIBs-In. L'Adj-RIBs-Out contient les informations de routage sélectionnées pour la propagation vers les routeurs BGP voisins en fonction de la politique de routage locale.

Pour assurer la fiabilité de l'échange d'information entre deux routeurs, BGP s'appuie sur le protocole TCP (*Transmission Control Protocol*) [13]. Ainsi, les deux routeurs établissent une session TCP sur le port 179. BGP dispose de deux modes de fonctionnement : *Exterior BGP* (eBGP) et *Interior BGP* (iBGP). Une session eBGP est établie entre deux routeurs appartenant à des AS différents. Ces routeurs sont connectés au travers de liaisons point-à-point ou dans des points d'échange Internet ou *Internet eXchange Point* (IXP). Les IXP sont des installations utilisant des équipements de niveau 2 et qui permettent à plusieurs AS de se connecter ensemble. Une session iBGP est utilisée pour la communication entre deux routeurs au sein d'un même AS. En plus d'iBGP, les routeurs au sein d'un même AS utilisent généralement un protocole de routage interne ou *Interior*

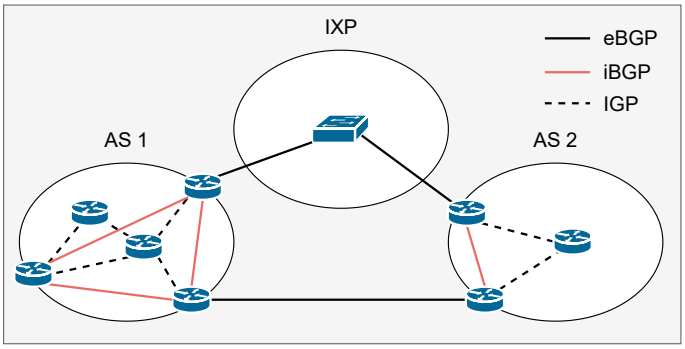


FIGURE 2.1 – Exemple de réseau BGP

gateway protocol (IGP) tel que OSPF (*Open Shortest Path First*). L'IGP est utilisé pour le routage interne à l'AS et pour la découverte des préfixes de destination. La figure 2.1 illustre un exemple de réseau BGP avec les différents types de connexions.

2.1.4 Le message UPDATE

Lorsque deux routeurs communiquent via une session BGP, ils disposent de quatre types de messages : OPEN, UPDATE, NOTIFICATION et KEEPALIVE. Le message OPEN est le premier message envoyé après l'établissement de la connexion TCP. Si le message est accepté, un message KEEPALIVE est envoyé périodiquement pour maintenir la connexion ouverte. Le message NOTIFICATION est envoyé à la suite d'une erreur et la connexion est interrompue juste après son envoi.

Message UPDATE

Le message UPDATE est le plus important, car il permet l'échange des informations de routage. Un seul message UPDATE peut être utilisé pour simultanément annoncer un ensemble de routes qui partagent des mêmes attributs et invalider plusieurs routes. Il se divise donc en deux parties : une liste de préfixes qui ne sont plus atteignables (Withdrawn Routes) et une séquence d'attributs décrivant les routes nouvellement disponibles (Path Attributes).

Les attributs d'un message UPDATE sont classés dans quatre catégories : *well-known mandatory*, *well-known discretionary*, *optional transitive* et *optional non-transitive*. Un attribut *well-known* doit être pris en charge par toutes les implémentions de BGP et doit obligatoirement être présent s'il est *mandatory* et que l'UPDATE contient une annonce de route. Un attribut *optionnel* n'est pas nécessairement reconnu par toutes les implémentations de BGP. Cependant, s'il est *transitif*, alors il doit être propagé même s'il n'est pas reconnu. Voici quelques-uns des attributs les plus connus :

ORIGIN : est un attribut *well-known mandatory* qui est généré par l'AS à l'origine de l'annonce. Cet attribut ne doit donc pas être modifié durant la propagation de la route.

Les valeurs possibles sont 0 si la route provient d'un IGP, 1 si elle provient d'un EGP (*Exterior gateway protocol*), BGP étant le seul EGP en utilisation ou 2 pour INCOMPLETE si la route vient d'une autre source comme une route statique.

AS_PATH : est un attribut *well-known mandatory* liste tous les AS qui ont participé à la propagation de l'annonce. Cette liste peut être composée de deux types d'éléments : des AS_SETs ou des AS_SEQUENCEs. Une AS_SEQUENCE est une liste ordonnée d'AS contrairement à un AS_SET. Lorsqu'un routeur BGP propage une route vers le routeur d'un AS voisin, il doit donc ajouter son ASN au début de l'AS_PATH. Si le premier élément de l'AS_PATH est une AS_SEQUENCE alors il y ajoute son ASN. À l'inverse, si c'est un AS_SET, il ajoute à l'AS_PATH une nouvelle AS_SEQUENCE contenant son ASN.

MULTI_EXIT_DISC : est un attribut *optionnel non-transitif* utilisé pour indiquer à un AS voisin le lien à favoriser si plusieurs connexions existent entre les deux AS. Si plusieurs routes équivalentes existent pour un même préfixe, la route avec la valeur la plus faible sera favorisée.

LOCAL_PREF : est un attribut *well-known discretionary* qui doit être inclus dans toutes les annonces qu'un routeur BGP propage vers les routeurs du même AS. Il permet d'indiquer aux routeurs du même AS le niveau de préférence d'une route parmi toutes les routes disponibles pour un même préfixe. Plus la valeur est élevée, plus la route est préférée. C'est un attribut très utile pour l'implémentation des politiques de routage. Par exemple, un AS accordera des valeurs plus élevées aux routes passant par ses clients plutôt qu'à celles passant par ses fournisseurs.

2.1.5 Processus de décision

Lorsqu'un routeur BGP reçoit un message UPDATE, il met à jour sont Adj-RIB-In et démarre son processus de décision. Le rôle de ce processus est d'appliquer la politique de routage de l'AS à l'Adj-RIB-In afin de déterminer les routes à utiliser localement et celles à annoncer aux AS voisins. Il est également en charge de l'agrégation des routes et de la réduction des informations de routage. Le processus de décision se déroule en trois phases :

Phase 1 : La phase 1 a pour mission de déterminer la valeur de l'attribut LOCAL_PREF de chaque route en fonction de la politique de routage de l'AS. Si l'annonce provient d'un routeur appartenant au même AS, alors la valeur peut être reprise.

Phase 2 : La phase 2 est en charge du choix de la meilleure route parmi toutes les routes disponibles pour une destination. La route retenue sera ensuite installée dans la Loc-RIB. Si l'attribut NEXT_HOP indique une adresse non atteignable ou si l'attribut AS_PATH contient une boucle de routage alors la route est ignorée. Si plusieurs routes sont disponibles pour un même préfixe alors une procédure de *tie-breaking* est déclenchée

Ordre	Comparaison
1	Valeur de LOCAL_PREF la plus élevée
2	Taille de l'AS_PATH la plus petite
3	Valeur de ORIGIN la plus petite
4	Valeur de MULTI_EXIT_DISC la plus petite
5	Préférence des routes eBGP sur les routes iBGP
6	Coût IGP le plus faible
7	Identifiant du routeur le plus faible

TABLE 2.1 – Procédure de *tie-breaking*

et différentes comparaisons sont effectuées afin de choisir une seule route. L'ordre de ces comparaisons est indiqué dans le tableau 2.1.

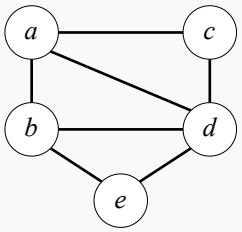
Phase 3 : Le rôle de la phase 3 est d'appliquer la politique de routage de l'AS à la Loc-RIB afin de déterminer les routes qui doivent être propagées vers les routeurs voisins et donc incluses dans Adj-RIBs-Out. C'est également elle qui est en charge des mécanismes d'agrégation de route et des informations de routage. En effet, étant donné que BGP utilise la notation CIDR [11], il est possible pour un routeur BGP d'agréger plusieurs routes en une seule afin de réduire la quantité d'information à transmettre. Un AS-SET peut alors être créé afin de synthétiser les AS-PATHs de toutes les routes.

2.2 Définitions préliminaires sur les graphes

2.2.1 L'objet graphe

L'objet graphe

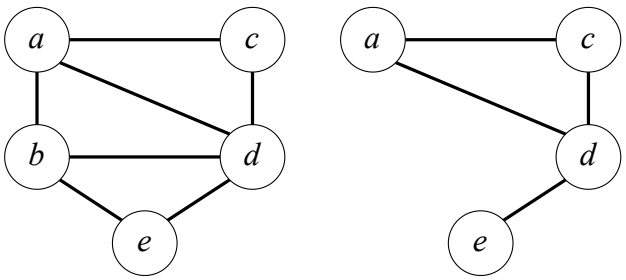
Un graphe est généralement défini comme un couple $G = (V, E)$ où V est un ensemble de sommets également appelé noeud et E est un ensemble d'arêtes également appelé lien. Un lien $e \in E$ est défini par $e = \{u, v\}$ où $u \in V$ et $v \in V$. Un lien où $u = v$ est appelé une boucle. Un graphe est non orienté si le lien $\{u, v\}$ est équivalent au lien $\{v, u\}$ et orienté dans le cas contraire.



La figure ci-dessus représente le graphe suivant :

$$G = (V, E),$$
$$\text{où } V = \{a, b, c, d, e\},$$
$$\text{et } E = \{\{a, b\}, \{a, c\}, \{a, d\}, \{b, d\}, \{b, e\}, \{c, d\}, \{d, e\}\}$$

Dans cette thèse le terme graphe est utilisé pour désigner un graphe non orienté. Deux noeuds u et v sont adjacents si $u, v \in E$. Ont dit également que v est un voisin de u et réciproquement. Un lien $e = \{u, v\}$ est incident à u et v. Le degré d'un noeud correspond au nombre de liens qui lui sont incidents. Un sous-graphe G' de G est un graphe $G' = (V', E')$ tel que $V' \subseteq V$ et $E' \subseteq E$. Le sous-graphe induit par l'ensemble des noeuds $V' \subseteq V$ est le graphe $G' = (V', E')$ où E' contient tous les liens de E tels que $e' = \{u, v\}$ où $u \in V'$ et $v \in V'$.



(A) Graphe $G(V, E)$ (B) Sous-graphe de G

FIGURE 2.2 – Exemples de graphes

Le graphe représenté sur la figure 2.2a possède 5 noeuds et 7 liens. Le degré du noeud d vaut 4. La figure 2.2b représente le sous-graphe de G induit à partir de l'ensemble des noeuds $V' = \{a, c, d, e\}$.

2.2.2 Représentations matricielles des graphes

En théorie algébrique des graphes, un graphe peut être représenté par différentes formes de matrices. Une première représentation matricielle est la matrice d'adjacence définie par :

$$A[u,v] = \begin{cases} 1 & si\ \{u,v\} \in E, \\ 0 & sinon. \end{cases} \tag{2.1}$$

Ainsi la matrice d'adjacence du graphe G représentée à la figure 2.2a est :

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array} \tag{2.2}$$

Une autre représentation utile est la matrice des degrés qui est une matrice diagonale où l'élément $D[u,u]$ correspond au nombre de liens incidents au noeud u. Ainsi, la matrice des degrés du graphe G est :

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{pmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix} \end{array} \tag{2.3}$$

A partir de la matrice d'adjacence A et de la matrice des degrés il est possible de calculer la matrice laplacienne telle que $L = D - A$ ou sa forme normalisée telle que $L' = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$. Ainsi, la matrice laplacienne du graphe G est :

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{pmatrix} 3 & -1 & -1 & -1 & 0 \\ -1 & 3 & 0 & -1 & -1 \\ -1 & 0 & 2 & -1 & 0 \\ -1 & -1 & -1 & 4 & -1 \\ 0 & -1 & 0 & -1 & 2 \end{pmatrix} \end{array} \tag{2.4}$$

Une difficulté que peut poser la manipulation des représentations sous forme de matrice d'un graphe est le fait qu'elles dépendent de l'ordre des noeuds. En effet, si l'on

construit la matrice d'adjacence du graphe G en changeant les étiquettes des noeuds du graphe, on obtient un résultat différent. Cela est problématique car il existe rarement de notion d'ordre dans les graphes issues des réseaux. Une solution est alors de calculer le spectre de ces matrices qui est un invariant de graphes. Ainsi, on s'intéresse généralement au spectre de la matrice d'adjacence et de la matrice laplacienne normalisée. Le spectre d'une matrice est l'ensemble de ses valeurs propres :

$$\lambda_1 \leq \lambda_1 \leq \cdots \leq \lambda_n \tag{2.5}$$

Pour une matrice A, un scalaire λ est une valeur propre de A s'il existe un vecteur x non nul tel que :

$$Ax = \lambda x \tag{2.6}$$

On dit de x qu'il est le vecteur propre associé à la valeur propre λ. Ces vecteurs sont notamment utilisés pour le calcul de certaines propriétés du graphe telle que la connectivité algébrique (cf. section).

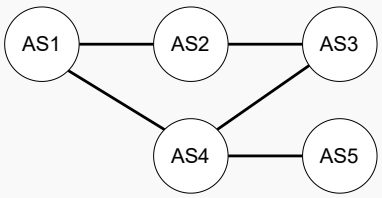
2.2.3 Le graphe BGP

Le graphe BGP

Le réseau BGP peut être représenté sous la forme d'un graphe où les AS sont les noeuds du graphe et les liens représentent les relations entre les AS [12]. A partir d'un ensemble de route BGP il est possible d'extraire le graphe BGP correspondant. Ainsi, les liens AS1–AS2 , AS2–AS3, AS1–AS4, AS4–AS5 et AS4–AS3 peuvent être inférés à partir de l'ensemble des routes suivantes :

Prefixe	AS_PATH
10.0.0.1/24	AS1 - AS2 - AS3
10.0.0.2/24	AS1 - AS4 - AS5
10.0.0.3/24	AS1 - AS4 - AS3

On obtient alors le graphe ci-dessous :



La construction de ce graphe nécessite d'avoir un point de vue omniscient sur l'ensemble du réseau BGP, ce qui ne peut être obtenu qu'en agrégeant les données de l'ensemble des AS et qui n'est donc pas possible en pratique. Cependant, BGP étant un protocole de routage à vecteur de chemin, il est possible pour un routeur BGP de d'inférer une vision locale qui sera un sous graphe du réseau BGP. En effet, à partir de l'attribut AS_PATH d'une route BGP qui décrit la succession des AS à traverser pour rejoindre une destination, il est possible d'inférer des liens entre les AS consécutifs. En combinant plusieurs routes, un graphe peut donc être construit. Afin d'avoir une vision plus globale du graphe BGP, il est possible de collecter les routes à partir de plusieurs routeurs BGP. En reprenant l'exemple de l'encadré ci-dessus et en supposant qu'un deuxième point de collecte voisin de l'AS5 donne les routes suivantes :

Prefixe	**AS_PATH**
10.0.1.1/24	AS5 - AS4 - AS3
10.0.1.2/24	AS5 - AS4 - AS1
10.0.1.3/24	AS5 - **AS4 - AS2** - AS1

Un nouveau lien AS4–AS2 qui n'était pas visible dans les routes précédentes est découvert. Cette approche est largement utilisée dans la littérature malgré le fait qu'elle fournit une vue incomplète du graphe BGP [14], [15].

2.3 Les anomalies BGP

De l'importance de BGP dans le fonctionnement d'Internet découle l'existence d'une grande quantité de travaux qui s'intéressent aux problèmes de sécurité et d'instabilité dont il peut être victime.

Anomalies BGP

Dans la littérature, le terme d'anomalies BGP est couramment utilisé pour désigner les incidents ayant lieux sur ce protocole. Ce terme englobe une grande variété d'incidents allant de défaillance matérielles à des attaques ciblées sur un préfixe ou un AS. Les conséquences des anomalies BGP ont également divers niveau de gravité. Certaines anomalies provoquant une surcharge des routeurs BGP tandis que d'autres rendent possible la prise de contrôle d'un préfixe ou le détournement de trafic.

Dans [16], Al-Musawi et al. propose une taxonomie des anomalies BGP en fonction de leurs causes. Cependant, une anomalie peut selon le contexte avoir différentes conséquences plus ou moins importantes. C'est pour cette raison que cette section présente les anomalies BGP à la fois du point de vue de leurs causes et des conséquences qu'elles peuvent avoir. Ainsi, la figure 2.3 propose une taxonomie non exhaustive des anomalies BGP qui regroupent les causes et les conséquences les plus communes de la littérature. De plus, le tableau 2.2 dresse une correspondance entre les causes des anomalies BGP et leurs conséquences possibles.

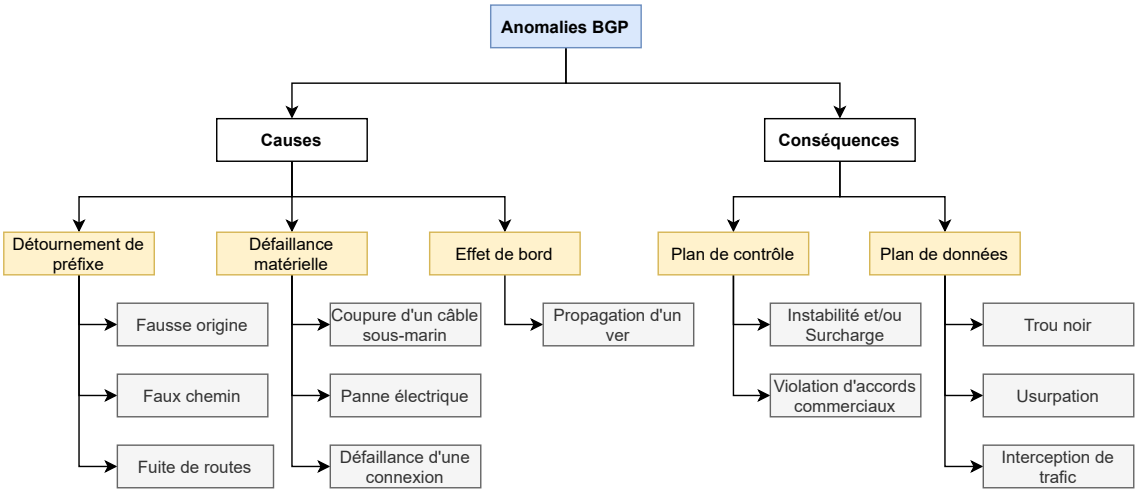


FIGURE 2.3 – Taxonomie non exhaustive des anomalies BGP

2.3.1 Du point de vue des causes

Détournement de préfixe

Le protocole BGP a été développé sur un principe de confiance mutuelle entre les AS et est par conséquent très vulnérable face à des annonces de routes incorrectes. Ces

Causes	Conséquences				
	Plan de contrôle		Plan de données		
	Instabilité et/ou surcharge	Violation d'accords commerciaux	Trou noir	Usurpation	Interception de trafic
Détournement de préfixe	✓	✓	✓	✓	✓
Défaillance matérielle	✓		✓		
Effet de bord	✓				

TABLE 2.2 – Correspondance entre les causes des anomalies BGP et leurs conséquences possibles

annonces pouvant être volontaires afin de conduire des activités malicieuses ou dues à des erreurs de configuration de la part des opérateurs d'AS. Dans la littérature, les termes de détournement de préfixe ou *prefix hijacking* [17]-[19] et de fuite de routes ou *route leaks* [20] sont couramment utilisés pour désigner ce type d'anomalies BGP. Dans la taxonomie proposée, le terme de détournement de préfixe est utilisé et la distinction est faite entre trois types de détournement de préfixe : les fausses origines, les faux chemins et les fuites de routes.

Fausse origine : un détournement d'origine survient lorsqu'un AS annonce être l'AS d'origine d'un préfixe qu'il ne possède pas. Cela peut alors créer une situation de concurrence avec les annonces faites par l'AS légitime. Cette situation de conflit est appelée *Multiple Origin AS* (MOAS) dans la littérature. Du point de vue du préfixe concerné, le réseau BGP est scindé en deux parties, celle qui utilise la route légitime et celle qui utilise la route erronée. Un AS choisissant une route plutôt que l'autre en fonction de sa politique de routage. Cependant, l'apparition d'un MOAS ne permet pas de détecter à elle seule une anomalie. En effet, elle peut être avoir une cause légitime telle que le *multihoming* [21]. De plus, si un AS annonce une route vers un sous préfixe du préfixe annoncé par l'AS légitime, alors aucun MOAS n'apparaît. Et comme un routeur choisira toujours la route la plus spécifique pour acheminer son trafic, alors tous les routeurs recevant l'annonce erronée vont utiliser cette route. Dans le cadre d'activités malicieuses, un AS peut également éviter de créer un MOAS et de lever des soupçons en annonçant une route vers un préfixe qui n'est actuellement annoncé par aucun AS. Ce préfixe pouvant soit être non alloué ou dormant c'est-à-dire alloué à un AS mais non annoncé par ce dernier [22].

Faux chemin : un détournement de chemin survient lorsqu'un AS annonce avoir une route vers un préfixe alors que celle-ci n'existe pas dans la réalité. Ici, aucun MOAS n'apparaît puisque c'est le chemin qui est erroné et non pas l'origine de la route. L'AS-PATH de la route contiendra alors au moins une paire d'AS consécutifs qui ne sont pas connectés dans la réalité. Dans le cadre d'une attaque, McArthur et al. [23] ont montré qu'il

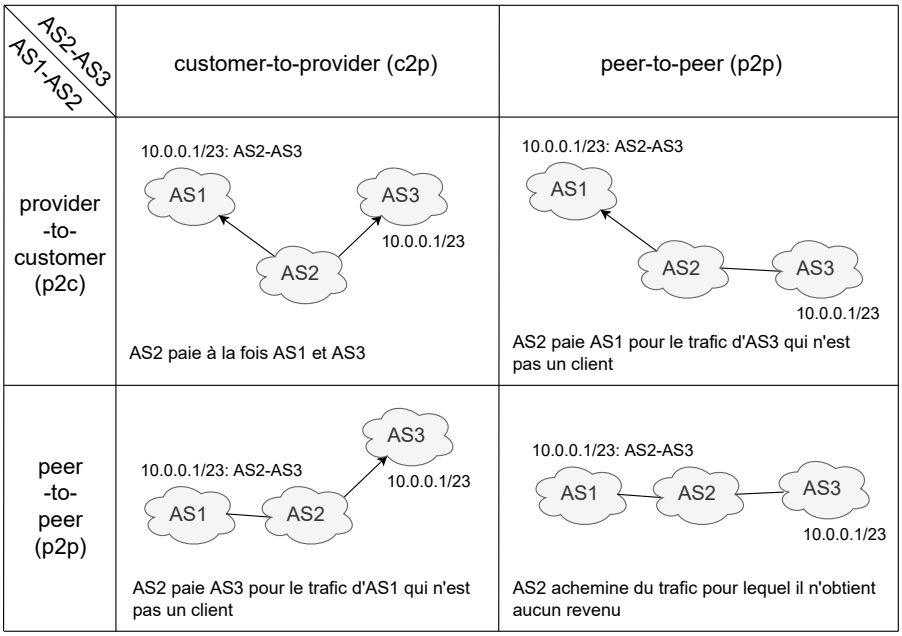


FIGURE 2.4 – Exemples de violation de la règle des chemins sans vallée

est possible pour l'AS attaquant d'ajuster la portée de l'attaque en ajustant la taille de la fausse route annoncée. Permettant ainsi de conduire une attaque plus ou moins discrète. Comme pour les fausses origines, le faux chemin peut être annoncé pour un sous préfixe ce qui résultera à l'adoption de la route par tous les routeurs recevant l'annonce.

Fuite de routes : la RFC 7908 [20] définie une fuite de routes BGP ou *route leak* comme une propagation d'annonces de routes au-delà leur portée prévue. Ces annonces ne respectant donc pas les politiques de routage d'au moins un des AS impliqué dans la propagation de la route. Ce type d'anomalie se différencie donc des faux chemins du fait que les AS impliqués sont bien connectés dans la réalité. Dans la littérature, on dit de ce type d'annonces qu'il viole la règle des chemins sans vallée ou *Valley-Free rule* [12]. Cette règle implique qu'après avoir traversé un lien *prodiver-to-customer* (p2c) ou *peer-to-peer* (p2p), une route ne doit pas traverser un lien *customer-to-provider* (c2p) ou p2p. En d'autres termes, un AS ne devrait pas acheminer du trafic pour lequel il n'obtient pas de revenu. La figure 2.4 donne quatre exemples de violation de la règle des chemins sans vallée.

Défaillance matérielle

La robustesse d'Internet vient de la redondance des connexions qui rend généralement la défaillance d'une connexion peu perceptible. Cependant, en fonction de l'importance du lien, une défaillance peut avoir un impact significatif sur le fonctionnement de BGP en provoquant un volume important de modification de routes. De plus, des défaillances plus importantes comme la coupure d'un câble sous-marin [24] ou une panne de courant à large échelle [25] peuvent même complètement déconnecter certaines parties d'Internet. En 2008 [24], la coupure d'un câble sous-marin en mer Méditerranée à

provoqué la disparition de plus 30% des préfixes en Égypte, au Koweït et au Soudan. En 2003 [25], une panne électrique touchant le nord-est des États-Unis et le Canada a impacté 1 700 organisations parmi lesquelles plus d'un millier ont été impactés pendant plus de quatre heures.

Effet de bord

Des anomalies affectant des éléments de bordure du réseau peuvent également avoir un impact sur le fonctionnement de BGP. En effet, il a été observé que la corruption de serveurs web par des vers informatiques peut impacter la stabilité de BGP. Les évènements les plus connus sont les attaques des vers Nimda, Code Red II et Slammer qui ont causé une surcharge de routage au niveau mondial [26], [27].

2.3.2 Du point de vue des conséquences

Il existe un large éventail de conséquences possible des anomalies BGP avec différent degré de préjudice. Du point de vue des conséquences, cette thèse fait la distinction entre deux catégories d'anomalies. Premièrement, celles qui affectent le plan de contrôle c'est-à-dire les messages UPDATE échangés par les routeurs BGP et leurs tables de routages. Deuxièmement, celles qui affectent le plan de données c'est-à-dire les données échangées sur le réseau et qui impactent donc plus directement les utilisateurs finaux.

Plan de contrôle

Les anomalies dont les conséquences se situent au niveau du plan de contrôle affectent le fonctionnement du protocole BGP. Ce type d'anomalies peut donc impacter le réseau BGP sans pour autant avoir un impact visible sur les services s'appuyant sur le réseau. Deux types de conséquences peuvent exister sur le plan de contrôle :

Instabilité et/ou surcharge : Le premier type de conséquences au niveau du plan de contrôle est l'instabilité et/ou la surcharge des routeurs BGP. Toutes les anomalies BGP peuvent avoir ce type de conséquence en fonction de leur échelle. Même des évènements non liés à BGP, telle que l'attaque de serveur web par des vers informatiques ont déjà eu ce type d'impact sur le plan de contrôle [26], [27].

Violation d'accords commerciaux : Le second type de conséquences au niveau du plan de contrôle est la violation d'accords commerciaux. Un AS pouvant alors recevoir du trafic qu'il n'est pas supposé prendre en charge. Ce type de conséquences peut survenir suite à des détournements de préfixe. Cela, s'est produit en 2017 [28] lorsqu'un AS contrôlé par Google a accidentellement fait fuiter un grand nombre de routes vers des préfixes qu'il ne possède pas. Comme les routes étaient plus spécifiques, elles ont été largement adoptées, causant des congestions et des pannes.

Plan de données

Au niveau du plan de données, les anomalies BGP peuvent avoir trois types de conséquences :

Trou noir : une anomalie crée un trou noir si le trafic est acheminé vers une destination où il n'obtiendra pas de réponse. Cela peut se produire à la suite d'une défaillance matérielle si aucune route de remplacement n'a été trouvée. Les détournements de préfixe de type fausse origine et faux chemin provoquent également fréquemment un trou noir car la route annoncée n'existe pas dans la réalité. Cela est également le cas pour les fuites de routes car même si la route existe dans la réalité, le fait qu'elle ne respecte pas les accords commerciaux peut conduire au rejet du trafic de la part de l'AS floué. En 2008 [29], un détournement de préfixe causé par le gouvernement pakistanais a engendré un trou noir rendant le site YouTube indisponible mondialement pendant plusieurs minutes. Le gouvernement souhaitant bloquer l'accès à YouTube dans le pays a annoncé une route plus spécifique et a involontairement fait fuiter l'annonce en dehors du pays. La route étant plus spécifique, tous les routeurs la recevant se sont mis à l'utiliser et à envoyer le trafic à destination de YouTube au Pakistan.

Usurpation : une usurpation arrive lorsqu'un AS répond au trafic à destination d'un préfixe qui ne lui appartient pas. Ce type d'anomalie n'a pas de raison d'arriver de façon involontaire et est donc la conséquence d'une attaque utilisant un détournement de préfixe. Même si un détournement de chemin ou une fuite de routes ne change pas l'AS de destination, l'attaquant peut choisir de répondre au trafic au lieu de le transmettre au prochain AS de l'AS-PATH. Une première motivation à conduire ce type d'attaque est de compromettre les services associés au préfixe usurpé. Le service Spamhaus, qui permet de savoir si une adresse IP est connue pour envoyer du spam, a été la cible d'une attaque de ce type en 2013 [30]. L'attaquant a alors mis en place un faux service qui répondait toujours positivement peu importe l'adresse IP entrainant le marquage comme spam de nombreux emails durant l'attaque. Une seconde motivation pour l'attaquant est de pouvoir conduire des activités malicieuses sans que l'on puisse remonter vers lui [22]. Un exemple connu est celui des *fly-by spammers* [31] qui utilisent des préfixes qui ne leur appartiennent pas pour envoyer des spams dans une courte période de temps. La furtivité de l'attaque évitant que les adresses IP soient mises sur liste noire.

Interception de trafic : l'interception de trafic est probablement la conséquence la plus insidieuse des anomalies BGP. Comme pour l'usurpation, elle est le résultat d'une attaque volontaire et ciblée. Une interception de trafic se produit lorsqu'un AS détourne le trafic d'un préfixe qui ne lui appartient pas tout en étant capable de l'envoyer vers son véritable AS de destination [18], [32]. L'attaque est donc invisible du point de vue des services qui continue de fonctionner normalement. Cependant, si les échanges ne sont pas sécurisés, l'attaquant peut alors lire et modifier le trafic à la volée. Ce type d'anomalie est volontaire et résulte donc d'un détournement de préfixe. En 2013 [33], Renesys a mis en

lumière des exemples d'interception de trafic ayant eu lieu la même année démontrant ainsi l'existence de ce type d'attaque dans la nature.

2.3.3 La détection plutôt que la sécurisation

Les limites de la sécurisation de BGP

Pour adresser le problème des anomalies sur le protocole BGP de nombreux travaux [16] se sont intéressés à leur détection. Cependant, plutôt que de détecter les anomalies il serait préférable d'éviter leurs apparitions. Il existe deux raisons principales qui expliquent pourquoi la détection des anomalies BGP est indispensable.

- Certaines anomalies ne sont pas liés à des failles de sécurité du protocole BGP
- L'adoption des solutions de sécurisation de BGP est trop lente

La première limitation à la sécurisation de BGP découle de la nature même de certaines anomalies BGP. En effet, les défaillances matérielles et effets de bord ne sont pas liés à des failles de sécurité du protocole BGP. Ainsi, même si des efforts sont effectués en ajoutant par exemple plus de redondance, des anomalies risquent toujours de survenir. La détection étant donc la seule option afin de permettre aux opérateurs d'agir le plus rapidement possible.

À l'inverse, les anomalies causées par des détournements de préfixe sont liées à un manque de sécurisation du protocole BGP. Cela amène donc la seconde raison qui motive de la détection des anomalies : le manque d'adoption des solutions de sécurisation de BGP. En effet, plusieurs solutions ont été apportées afin d'améliorer la sécurité de BGP [34], [35]. Cependant, les limitations des solutions proposées et l'effort nécessaire à leur déploiement freinent leur mise en place par les opérateurs d'AS [36]. La solution la plus adoptée actuellement est le *Resource Public Key Infrastructure* (RPKI) qui a été proposé en 2012 par le *Secure Inter-Domain Routing Working Group* (SIDR WG) de l'*Internet Engineering Task Force* (IETF). RPKI est décrit dans la RFC 6480 [37] et est actuellement déployé pour environs 37% des préfixes [38] (en juin 2022). RPKI s'appuie le chiffrement asymétrique afin de créer des *Route Origination Authorizations* (ROA) qui certifient qu'un AS est autorisé à être l'origine d'un préfixe. Cependant, RPKI permettant de valider l'origine des routes elle prévient uniquement les détournements d'origines. Afin de sécuriser également le chemin, le SIDR WG a proposé BGPsec qui a été définie en 2017 dans la RFC 8205 [39]. Ainsi, une route validée avec BGPsec assure que chaque AS listé dans l'AS_PATH a explicitement autorisé l'annonce de la route à l'AS suivant rendant ainsi impossibles les détournements de chemin. Cependant, cette propriété ne prévient pas les fuites de routes. Pour adresser ce problème il a été proposé l'utilisation de l'attribut DOWN_ONLY [40] permettant d'éviter les violations de la règle des chemins sans vallée (cf. figure 2.4). Cependant, cette solution est efficace que dans le cas de fuite de routes involontaire et ne protège donc pas des attaques.

2.4 Détection des anomalies BGP

Cette section présente un tour d'horizon des solutions proposées dans la littérature afin de détecter les anomalies BGP. Cette présentation n'est cependant pas exhaustive mais recouvre les travaux en lien avec cette thèse. Ainsi, trois axes de la littérature sont présentés. Tout d'abord, les travaux s'intéressant aux détournements de préfixes seront abordés. En effet, du fait des conséquences importantes que ce type d'anomalie peut avoir sur le plan de données (cf. table 2.2), elle constitue une part importante de la littérature. Ensuite, les solutions basées sur l'apprentissage automatique et sur les représentations sous forme de graphes seront présentées étant donné qu'il s'agit de l'approche adoptée dans cette thèse. Toutes ces solutions ont souvent en commun le fait qu'elles nécessitent l'accès aux données de contrôle du protocole, autrement dit, aux informations de routages échangés par les routeurs BGP. Ainsi, les différentes options pour la collecte de ces données sont tout d'abord introduites.

2.4.1 Collecte de données BGP

Archives de données BGP

L'étude des anomalies BGP et leur détection requièrent l'accès aux données de routage échangées durant leur occurrence. Heureusement, dès 1999 et 2001, les projets Ripe RIS [41] et Route Views [42], ont respectivement commencé la collecte et l'archivage de données BGP.
Ces projets ont ainsi mis en place plusieurs collecteurs de routes BGP à travers le monde. Un collecteur étant un routeur BGP connecté à plusieurs autres routeurs BGP appelés pairs appartenant à différents AS. À intervalles réguliers le collecteur génère une archive de tous les messages UPDATE reçus à partir de ses différents pairs depuis le dernier archivage.

Les archives sont générées toutes les 5 minutes pour Ripe RIS et toutes les 15 minutes pour Route Views. De plus, toutes les 8 heures pour Ripe RIS et toutes les 2 heures pour Route Views, tout le contenu de la table RIB du collecteur est également archivé. Actuellement, le projet Ripe RIS compte 26 collecteurs et le projet Route Views en compte 35. Les données BGP sont archivées au format MRT (*Multi-threaded Routing Toolkit*) décrit dans la RFC 6396 [43] et sont rendues accessibles au public au travers d'une interface web.

Bien que les données collectées par les projets Ripe RIS et Route Views soient très utiles, leurs exploitations ne sont pas très pratiques. En effet, pour étudier une anomalie BGP à partir de différents points de collecte, il est nécessaire d'identifier les différents fichiers aux formats MRT à récupérer puis d'utiliser un outil tel que bgpdump [44] pour transformer les données dans un format lisible par un humain. Enfin, les messages UPDATE récupérés à partir de différents points de collecte doivent être réordonnés temporellement afin de reconstruire la séquence d'échange des messages. Le framework BGPStream prend en charge cette complexité en permettant d'accéder aux données BGP de

différents points de collecte sous forme de flux comme si elles étaient reçues en temps réel. Pour cela, BGPStream s'appuie sur les données des projets Ripe RIS et Route Views en récupérant automatiquement les archives nécessaires et en réordonnant les données temporellement. De plus, l'outil dispose d'un mode de fonctionnement en temps réel facilitant ainsi le déploiement d'un outil de détection d'anomalies BGP.

BGPmon [45] lancé en 2009 est un autre outil de collection de données BGP en temps réel dont le rôle est de simplifier la collecte des données BGP et sa distribution aux applications souhaitant y avoir accès. Pour cela, l'outil s'appuie sur une architecture *publish-subscribe* où les applications clientes reçoivent un flux de données BGP aux formats XML. De plus, les collecteurs BGPmon implémentent uniquement les fonctions nécessaires à la collecte des données BGP les rendant plus légers et permettant un plus grand nombre de connexions avec des pairs.

Certains AS proposent également l'accès public à certaines de leurs informations de routage au travers de serveur *looking glass*. Ces serveurs permettent d'interroger directement les routeurs BGP par le biais d'une interface web [46] ou telnet [42]. Ils permettent donc d'obtenir des informations spécifiques et en temps réel sur certaines routes BGP. Cependant, le fait que ces *looking glass* n'utilisent pas la même interface rend leur exploitation fastidieuse.

2.4.2 Détection des détournements de préfixes

Cette section se concentre sur la détection des détournements de préfixe. Dans cette étude bibliographique, divers travaux sont présentés. Ces travaux couvrent un grand champ d'approches pour la détection de ces anomalies, le déploiement de la solution proposée et l'évaluation des performances.

Approche pour la détection

Les approches proposées dans la littérature pour les détections de détournements de préfixes peuvent être regroupées en trois catégories :

Plan de contrôle : Les approches basées sur le plan de contrôle [47]-[50] suivent passivement les données échangées par les routeurs BGP afin de repérer les éventuels détournements de préfixes. Ces travaux s'appuient généralement sur les données collectées par les projets Route Views et Ripe RIS [47], [48], [50]. Cependant, dans [49] les auteurs ont utilisé les données du projet BGPmon qui permet un fonctionnement en temps réel grâce à son flux en direct de données BGP.

Une première approche pour détecter les différents types de détournement de préfixe est d'identifier leurs conséquences sur le plan de contrôle. Ainsi, un détournement d'origine peut être détecté par l'apparition d'un conflit MOAS. Un détournement de chemin est détectable par l'apparition d'une paire d'AS consécutifs dans un AS_PATH alors que ces AS ne sont pas connectés dans la réalité. Enfin, une violation de la règle des chemins sans vallée permet d'identifier une fuite de route. Certains travaux [47], [48] considèrent

uniquement la détection des fausses origines alors que d'autres, plus récents, prennent également en compte les faux chemins [49]. Cependant, aucun d'entre eux s'intéresse aux fuites de routes ce qui peut être expliqué par le fait que les accords commerciaux entre les AS ne sont pas connus publiquement. Une autre approche est proposée dans [51] où un potentiel détournement de préfixe est détecté en mesurant la variation d'une route vers un préfixe par rapport à des préfixes similaires.

Une limitation principale des systèmes de détections basés sur le plan de contrôle est leur important nombre de faux positifs dû à la confusion avec des pratiques d'ingénierie de trafic telles que l'IP anycast et le multi-homing. Pour surmonter ce problème certains travaux [48] proposent d'avoir recourt à du filtrage. Cependant, cette solution requière que les AS maintiennent à jour ces règles de filtrage qui sont elles aussi une potentielle source d'erreurs.

Plan de donnée : Bien que le détournement de préfixe soit une anomalie qui provient d'une vulnérabilité de sécurité de BGP, ses conséquences sont également observables sur le plan de données (cf. table 2.2). Les approches pour la détection des détournements de préfixes basées sur de plan de données [17], [19], [52], [53] s'appuient sur ce constat en effectuant une surveillance active des préfixes. Un avantage par rapport au plan de contrôle est qu'elles sont moins sensibles aux pratiques d'ingénierie de trafic.

La première conséquence possible sur plan de données est l'apparition d'un trou noir où toutes les adresses IP couvertes par le préfixe impacté ne sont plus accessibles. De tels événements sont assez faciles à détecter en vérifiant l'accessibilité d'un ensemble d'adresses IP couvertes par le préfixe [19]. Cependant, une telle approche de détection est inefficace face des attaques plus élaborées telles que l'usurpation. Dans un scénario d'usurpation, l'attaquant au lieu de jeter le trafic du préfixe attaqué peut y répondre à sa place. Une solution consiste à effectuer le contrôle d'accessibilité à partir du réseau de l'AS attaqué et à destination de plusieurs adresses IP [52]. Avec cette approche, le trafic d'accusé de réception atteindra l'attaquant au lieu de l'AS légitime. Une autre approche de détection d'usurpation consiste à vérifier les hôtes de destination en prenant une empreinte des services qu'ils proposent [17]. Cela rendra alors la tâche de l'attaquant plus complexe car il devra être en mesure de copier cette empreinte. Cependant, cette approche est inefficace dans le cas d'une interception de trafic où l'attaquant est en capacité de transférer le trafic vers la destination légitime. Dans [53], les auteurs proposent une solution pour détecter à la fois l'interception de trafic et l'usurpation en surveillant les changements dans le nombre de sauts vers le préfixe.

Cependant, aucun des systèmes de détections basés sur le plan de données ne couvre les trois types d'anomalies. De plus, ces approches sont sensibles à d'autres anomalies BGP telles que les défaillances matérielles créant ainsi des faux positifs.

Hybride : Les approches hybrides [17], [19], [50] tentent de tirer parti des deux approches précédentes. L'idée générale de ces propositions est d'identifier un éventuel détournement de préfixe grâce à la surveillance passive du plan de contrôle et d'éliminer les faux positifs avec des mesures actives sur plan de données.

Argus [19] utilise une mesure de corrélation entre l'observation d'une anomalie et l'accessibilité du préfixe correspondant. Pour chaque événement suspect, plusieurs routeurs vérifient s'ils sont concernés par l'événement et si les IP appartenant au préfixe concerné sont accessibles. La corrélation de ces observations sur plan de données et sur le plan de contrôle est ensuite utilisée pour déterminer s'il y a un détournement de préfixe en cours. L'un des principaux points forts d'Argus est la surveillance du plan de contrôle qui couvre les trois types de détournement de préfixe. Malheureusement, la mesure sur plan de données limite la détection aux événements entraînant un trou noir. Dans [17], les auteurs proposent un système de détection ayant pour objectif de réduire le nombre de faux positifs d'une détection sur plan de contrôle en utilisant l'empreinte de l'hôte final. La surveillance sur le plan de contrôle couvre tous les types de détournement de préfixe, mais la deuxième étape de détection basée sur l'empreinte de l'hôte final peut être contournée par une interception de trafic. Heap [50] propose une approche pour enquêter sur les alarmes de détournement de préfixe produits par des systèmes de détection afin de filtrer les faux positifs. Le système s'appuie sur trois sources de données pour identifier les événements faussement positifs causés par l'ingénierie de trafic : l'*Internet Routing Registry* (IRR) qui est utilisé pour obtenir des informations sur les relations entre les AS impliqués dans l'événement, les informations sur la topologie du réseau et une analyse active des clés publiques *Secure Socket Layer* (SSL) et *Transport Layer Security* (TLS) des hôtes finaux. Cependant, le système ne couvre que les événements de détournement de sous-préfixe et peut être contourné par interception.

Les approches hybrides sont intéressantes car elles permettent de réduire le taux de faux positifs de la détection sur le plan de contrôle en appliquant une vérification du plan de données. Cependant, certains des inconvénients de la surveillance du plan de données demeurent tels que la possibilité pour un attaquant de déjouer certaines vérifications. De plus, la détection sur plan de données des interceptions de trafic est difficile.

Stratégie de déploiement

Quelle que soit l'approche utilisée pour la détection, il existe plusieurs stratégies pour le déploiement d'un système de détection de détournement de préfixe. Deux caractéristiques importantes de cette stratégie de déploiement sont la localisation du détecteur et les points de référence utilisés.

Il existe deux grandes catégories d'approche pour le fonctionnement du système de détection : auto-exploitation par l'AS en vue de surveiller les préfixes qui lui appartiennent ou le recours à une tierce partie. Les approches auto-déployées [49], [52] proposent de déployer le détecteur de détournement de préfixe à l'intérieur de l'infrastructure du propriétaire du préfixe. Le système peut effectuer une surveillance active [52] et/ou passive [49] avec plusieurs points de référence mais le traitement des données et le déclenchement des alarmes sont effectués par l'AS. Un avantage majeur de cette approche est la robustesse de la notification. Cependant, l'AS doit affecter les ressources nécessaires pour le déploiement et la maintenance du système. Les approches opérées par des tiers peuvent être classées en deux types. Premièrement, le système de détection

peut surveiller un ensemble de préfixes [48], [51], [53] qui appartiennent aux AS qui ont souscrit au service. Lorsqu'un détournement de préfixe est détecté par le tiers, il en informera l'AS correspondant via des canaux privés (*e.g.* email). Deuxièmement, le tiers surveille tous les préfixes annoncés sur Internet [17], [19], [47] et déclenche des alertes de détournement de préfixe via des canaux publics (*e.g.* site web). L'avantage des approches opérées par des tiers est la facilité de déploiement pour l'AS qui n'a qu'à s'inscrire au service ou à consulter un canal public. Cependant, en cas de piratage, les alarmes peuvent ne pas atteindre l'AS ou être filtrées lors d'une attaque élaborée.

Les détournements de préfixes sont des événements qui divisent Internet en deux parties : l'une qui est affectée par l'anomalie et l'autre qui continue d'observer la route légitime. Ainsi, la visibilité d'un événement de détournement de préfixe pour un système de détection est directement liée au nombre et à l'emplacement des points de référence utilisés. C'est la raison pour laquelle la majorité des travaux qu'ils se basent sur le plan de contrôle ou sur le plan de données utilisent plusieurs points de référence. Cependant, dans [50] un seul point de référence est utilisé car la proposition considère uniquement les détournements de sous-préfixe qui ne divise pas Internet car une route plus spécifique est toujours préférée par un routeur BGP. Pour la surveillance passive du plan de contrôle, les points de référence largement utilisés sont les différents collecteurs de route des projet Route Views et Ripe RIS. Une autre solution employée dans [19] consiste à utiliser des *looking glass* BGP. Pour la surveillance active du plan de données, une approche consiste à effectuer des mesures depuis l'infrastructure de l'AS à destination d'un ensemble de points de référence [52]. À l'inverse, d'autres travaux effectuent des mesures à partir de plusieurs points de référence vers une IP appartenant préfixe surveillé. Dans [53], les auteurs utilisent le réseau Planetlab [54] pour effectuer de telles mesures. Les noeuds du réseau Planetlab leur permettent d'effectuer des vérifications actives à destination des IP surveillées.

Évaluation des performances

Lorsqu'on propose un système de détection d'anomalies, une étape essentielle est d'évaluer ses performances. Dans la littérature sur le détournement de préfixe, deux approches principales sont utilisées pour l'évaluation des systèmes de détection. La première s'appuie sur des données issues du monde réel et la seconde utilise un environnement simulé.

Pour évaluer leurs travaux, les auteurs de [17], [19], [26], [50], [52], [53] ont choisi de déployer leur système dans le monde réel et d'observer les événements rapportés. Une difficulté avec cette approche est le manque de données étiquetées. En effet, lorsqu'un détournement de préfixe est détecté, il est difficile de dire si l'événement est un faux positif ou non. La solution proposée dans [53] est de supposer que tous les événements rapportés sont des faux positifs, ce qui permet d'évaluer la limite supérieure des faux positifs. Dans [19], les auteurs valident une partie des événements signalés à l'aide des enregistrements *Route Origin Authentication* (ROA) et des *Internet Routing Registry* (IRR). Afin

de surmonter le manque de données étiquetées, une solution consiste à rejouer les événements connus de détournement de préfixe. Dans [47], [51], les données extraites lors d'événements passés de détournement de préfixe bien connus sont utilisées pour l'évaluation. Une autre solution utilisée dans [49], [52] consiste à générer des détournements de préfixe sur des AS contrôlés par les expérimentateurs. Le procédé consiste à avoir deux AS, un attaquant et une victime, et à détourner un préfixe de la victime à partir de l'attaquant. Avec ces deux approches, il est possible de savoir si des anomalies connues ou générées sont détectées. Cependant, il existe peut être d'autres anomalies qui ne sont pas détectables pas le système, ces approches permettent donc uniquement l'évaluation de la borne inférieure du taux de faux négatif.

Dans [48], [49], [52], les performances du détecteur de détournement de préfixe proposé sont évaluées en simulation. Cet environnement contrôlé permet de générer des détournements de préfixes et de mesurer à la fois les taux de faux positifs et de faux négatifs. Cependant, la simulation doit être la plus réaliste possible pour minimiser l'écart de performance avec un déploiement réel.

Construire une simulation BGP à l'échelle Internet n'est pas une tâche facile et au moins deux obstacles majeurs doivent être surmontés. Premièrement, l'extraction de la topologie BGP et deuxièmement, l'inférence des politiques de routage des AS. La topologie BGP est composée du graphe des AS qui représente les interconnexions entre les AS et les relations entre les AS qui sont les accords commerciaux entre chaque paire d'AS. Le graphe des AS peut être extrait en combinant des instantanés de table de routage et des messages de mises à jour BGP à partir de plusieurs points de collecte [48], [52]. Les projet Route Views et Ripe RIS sont de bons candidats pour une telle collecte de données. Cependant, l'inférence de relation est une tâche plus complexe et une approche couramment adoptée est l'algorithme de Gao [12]. Cet algorithme repose sur l'hypothèse que les routes BGP doivent respecter la règle des chemins sans vallées (cf. 2.4). Ainsi, le but de l'algorithme est de trouver l'ensemble des relations qui maximisent le nombre de chemins respectant cette règle. Dans [49], les auteurs utilisent les données sur les relations entre AS extraites par CAIDA avec une méthodologie similaire [55]. Avec le protocole BGP, chaque AS peut définir sa propre politique de routage qu'il garde généralement privée. Cependant, les bonnes pratiques Gao-Rexford décrites dans [56] reflètent les pratiques courantes d'ingénierie de trafic appliquées par les AS et qui garantissent une convergence et une stabilité globales de BGP. Ces directives sont couramment appliquées pour extraire les politiques de routage des AS à partir d'une topologie BGP.

Discussion

Dans cette section les différentes approches existantes pour la détection, le déploiement et l'évaluation de ces solutions de détection de détournements de préfixe ont été présentées. Afin, de comparer de manière synthétique les travaux présentés, sept critères ont été identifiés :

Types de détournement de préfixe			Travaux								
Causes		Conséquences	[47]	[48]	[19]	[52]	[49]	[50]	[51]	[17]	[53]
Préfixe exacte	Fausse origine	Trou noir	✓	✓	✓	✓	✓		✓	✓	
		Usurpation	✓	✓		✓	✓		✓	✓	✓
		Interception	✓	✓			✓		✓		✓
	Faux chemin	Trou noir			✓	✓	✓		✓	✓	
		Usurpation				✓	✓		✓	✓	✓
		Interception					✓		✓		✓
	Fuite de routes	Trou noir			✓	✓			✓	✓	
		Usurpation				✓			✓	✓	✓
		Interception							✓		✓
Sous préfixe	Fausse origine	Trou noir		✓	✓		✓	✓	✓	✓	
		Usurpation		✓			✓	✓	✓	✓	
		Interception		✓			✓		✓		
	Faux chemin	Trou noir			✓		✓	✓	✓	✓	
		Usurpation					✓	✓	✓	✓	
		Interception					✓		✓		
	Fuite de routes	Trou noir			✓			✓	✓	✓	
		Usurpation						✓	✓	✓	
		Interception							✓		

TABLE 2.3 – Types de détournement de préfixe détectés

Types de détournements de préfixes détectés : le tableau 2.3 présente les différents types de détournements de préfixes détectés dans les travaux présentés. Une seule des solutions proposées [51] couvre théoriquement tous les cas possibles d'événements. Il y a plusieurs explications à ce manque de couverture des types de détournement de préfixe. Depuis le plan de contrôle, les fuites de routes sont difficiles à détecter car les politiques de routage des AS ne sont pas accessibles au public. Depuis le plan des données, les attaques aboutissant à une interception sont difficiles à détecter puisqu'elles n'impactent pas les services.

Précision : la détection de détournement de préfixe est sujette à un nombre élevé de faux positifs en raison de sa confusion possible avec les pratiques légitimes d'ingénierie de trafic et avec d'autres anomalies telles qu'une défaillance matérielle. Pour surmonter ce problème, certains travaux proposent des approches hybrides [17], [19], [50] mais elles peuvent être contournées par une attaque plus élaborée. Une autre solution consiste à filtrer les alertes [48], [49] mais cela nécessite la mise en d'une configuration pour le filtrage et sa mise à jour.

Facilité de déploiement : on peut supposer que la facilité de déploiement d'un système de détection est une condition importante pour son adoption par les AS. Les solutions les plus faciles à déployer sont celles opérées par des services tiers [17], [19], [47], [48], [51], [53], mais peuvent manquer de robustesse en ce qui concerne la notification des AS victimes. Les approches plus difficiles à déployer sont celles auto-déployé [49], [52] et les solutions qui nécessitent de maintenir une configuration [48], [49], qui offrent respectivement une meilleure robustesse et de meilleures performances.

Types d'anomalies (Causes)		Travaux										
		[57]	[58]	[59]	[60]	[61]	[62]	[63]	[64]	[65]	[66]	[67]
Détournement de préfixe	Fausse origine		✓									
	Faux chemin											
	Fuite de routes		✓						✓	✓	✓	
Défaillance matérielle	Coupure d'un câble sous-marin										✓	
	Panne électrique		✓								✓	✓
	Défaillance d'une connexion											
Effet de bord	Propagation d'un ver	✓		✓	✓	✓	✓	✓	✓		✓	✓

TABLE 2.4 – Types d'anomalies présents dans les jeux de données

Délai de détection : la majorité des travaux réalisent un délai de détection de l'ordre de grandeur de la minute. Cependant, dans [19], [49], les systèmes proposés peuvent détecter les détournements de préfixe en quelques secondes.

Identification des réseaux impliqués : lorsqu'un détournement de préfixe se produit, une information précieuse est de savoir quels sont les AS d'origines et propagateurs de la fausse route. Cependant, aucun des travaux présentés ne considère ce problème.

Notification de la victime : lors d'un détournement de préfixe, la notification de l'AS victime peut être difficile car l'alarme peut être redirigée vers un trou noir ou vers l'attaquant. Ce problème est résolu par les approches auto-déployé [19], [49], mais parmi les solutions opérées par des tiers ce problème n'est pris en compte que dans [48].

Atténuation automatique : dans [49] les auteurs proposent deux techniques pour atténuer automatiquement l'impact un détournement de préfixe. C'est un avantage par rapport aux autres travaux où les opérateurs d'AS doivent inspecter manuellement l'incident et trouver des contre-mesures.

2.4.3 Approches basées sur l'apprentissage automatique

Cette section se concentre sur la partie de la littérature qui utilise l'apprentissage automatique pour détecter les anomalies BGP.

Jeux de données utilisés

Avant d'appliquer une technique d'apprentissage automatique à un problème, il est nécessaire de disposer d'un jeu de données pour entraîner et tester le modèle. Cependant, il n'existe pas de consensus autour d'un jeu de données pour les anomalies BGP

que les chercheurs peuvent utiliser pour évaluer et comparer leurs travaux. En conséquence, chaque article utilise son propre jeu de données. Cependant, les auteurs suivent généralement une approche similaire pour la construction de leur jeu de données qui est composée de quatre étapes : le choix des anomalies à inclure dans le jeu de données, la collecte des données de BGP, l'extraction d'attributs et l'étiquetage.

Choix des anomalies : Le tableau 2.4 montre les types d'anomalies qui sont considérées dans les travaux examinés. Ce tableau liste les anomalies du point de vue de leurs causes uniquement car les travaux présentés utilisent tous les données du plan de contrôle et ne considèrent donc pas les conséquences des anomalies sur le plan de données. Il est à noter que 9 travaux (80% des travaux présentés) incluent des anomalies provoquées par des effets de bord dans leur jeu de données. Ce choix est discutable car ces anomalies ne sont pas directement liées à BGP et ont un impact limité. En effet, le tableau 2.2 montre qu'une anomalie provoquée par un effet de bord ne peut entraîner qu'une instabilité de BGP et/ou une surcharge des routeurs BGP. Trois articles prennent également en compte les événements de défaillance matérielle. Enfin, quatre travaux incluent des événements de fuite de routes.

Collecte des données BGP : Pour construire des ensembles de données qui capturent le comportement de BGP lors d'anomalies passées, il est nécessaire d'avoir accès à un historique des données du plan de contrôle de BGP. Pour cela, la majorité des travaux s'appuient sur les projets de collecte de données BGP Route Views et Ripe RIS. Cependant, ces sources de données introduisent un délai pour la disponibilité des données et donc pour la détection en temps réel des anomalies. BGPmon, qui fournit un flux en temps réel de données BGP, est une solution potentielle à ce problème, mais n'est utilisé dans aucun des articles présentés.

Extraction d'attributs : À partir des données BGP brutes collectées pour chaque anomalie, il est nécessaire d'extraire des attributs pouvant être exploités par des algorithmes d'apprentissage automatique. Dans les articles présentés, ces attributs sont calculés à partir d'un ensemble de messages UPDATE capturés dans fenêtre temporelle de k minutes (généralement $k = 1$). Ces attributs qui sont calculés sur des données BGP agrégées se répartissent généralement en deux catégories : ceux qui caractérisent le volume de données BGP et ceux calculés à partir des AS-PATH des messages UPDATE. Les attributs de volume tels que le nombre d'annonces et de retraits de routes visent à capturer les changements dans la stabilité de BGP. Les attributs d'AS-PATH tels que la longueur moyenne des AS-PATH et la distance d'édition maximale visent à capturer les changements topologiques. Ces attributs sont présentés en détail dans le chapitre 3. Au total, 39 attributs sont utilisés avec parmi eux 15 attributs qui sont utilisés dans au moins 80% des travaux. Le tableau 2.5 montre l'utilisation de ses quinze attributs dans les travaux. À la fin de cette étape, le jeu de données BGP lié à un événement peut être vu comme une matrice $M \in \mathbb{R}^{N \times T}$ où N est le nombre d'attributs calculés et T correspond à la durée d'observation (avec un pas de temps de k minutes). Ainsi, une ligne $M_{n,*}$ est une série temporelle

Attributs		Travaux										
		[57]	[58]	[59]	[60]	[61]	[62]	[63]	[64]	[65]	[66]	[67]
Volume	Nb. annonces	✓	✓	✓	✓	✓	✓	Nd	✓	✓	✓	✓
	Nb. retrait	✓	✓	✓	✓	✓	✓	Nd	✓	✓	✓	✓
	Nb. préfixes annoncés	✓	✓	✓	✓	✓	✓	Nd	✓	✓	✓	✓
	Nb. préfixes retirés	✓	✓	✓	✓	✓	✓	Nd	✓	✓	✓	✓
	Nb. annonces dup.	✓	✓	✓	✓	✓	✓	Nd		✓		✓
	Nb. retraits dup.	✓	✓	✓	✓	✓	✓	Nd		✓		✓
	Nb. retraits implicites	✓	✓	✓	✓	✓	✓	Nd		✓		✓
	Nb. paquets IGP	✓	✓	✓	✓	✓	✓	Nd	✓	✓		
	Nb. paquets EGP	✓	✓	✓	✓	✓	✓	Nd	✓	✓		
	Nb. paquets incomplets	✓	✓	✓	✓	✓	✓	Nd	✓	✓		
AS-PATH	Longeur AS-PATH [1]	✓	✓	✓	✓	✓	✓	Nd	✓	✓	✓	
	Max. AS-PATH	✓	✓	✓	✓	✓	✓	Nd	✓	✓	✓	
	Long. AS-PATH unique [1]	✓	✓	✓	✓	✓	✓	Nd	✓	✓	✓	
	Distance d'édition [1]	✓	✓	✓	✓	✓	✓	Nd	✓	✓		
	Distance d'édition max.	✓	✓	✓	✓	✓	✓	Nd	✓	✓		

[1] Moyenne.
[Nd] Non disponible : information non-présente dans l'article.

TABLE 2.5 – Quinze attributs utilisés dans 80% des travaux

qui représente l'évolution d'un attribut et une colonne $M_{*,t}$ est une vue de tous les attributs au temps t.

Étiquetage : Une difficulté majeure pour l'étude des anomalies BGP est le manque de données étiqueté. En effet, afin d'entraîner et d'évaluer les performances d'un modèle d'apprentissage automatique, les auteurs des articles présentés doivent étiqueter leur jeu de données. Pour chaque ensemble d'attributs calculés à un temps t, il est nécessaire d'attribuer une étiquette telle que "1" si une anomalie était en cours et "0" dans le cas contraire. Malheureusement, en raison du manque d'information sur l'occurrence des anomalies BGP, il est difficile d'étiqueter avec précision les jeux de données d'anomalies. Dans la littérature, les auteurs affectent généralement arbitrairement les étiquettes aux données en fonction de la période au cours de laquelle l'anomalie a lieu. Par exemple, toutes les données du jour de l'événement sont étiquetées "1" et les données des jours précédents et suivants sont étiquetés "0".

Méthodologies appliquées

Parmi les travaux examinés, différentes techniques ont été utilisées par les auteurs pour chaque étape du processus d'entraînement de leurs modèles d'apprentissage automatique. Ces choix méthodologiques sont présentés ci-après, selon quatre critères : le pré-traitement appliqué aux données, les méthodes de sélection d'attributs, les algorithmes d'apprentissage automatique utilisés et le type de classification effectué.

Pré-traitements : Au lieu d'alimenter directement leur algorithme d'apprentissage automatique avec leur jeu de données, certains articles appliquent des techniques de pré-traitement afin d'améliorer les performances de leur modèle. Un pré-traitement qui est

couramment utilisé lors dans le domaine de l'apprentissage automatique est la normalisation des données. En effet, un problème fréquent lorsque plusieurs attributs sont utilisés pour entraîner algorithme d'apprentissage automatique est que ces attributs ne varient pas dans les mêmes plages de valeurs. En fonction des algorithmes cela peut biaiser l'apprentissage en donnant plus d'importance à certains attributs ou impacter la vitesse convergence de l'algorithme. Ainsi, les techniques de normalisation permettent d'éviter ces problèmes en ré-ajustant les valeurs des attributs. Une technique couramment utilisée est la cote Z ou *Z-score* qui transforme les attributs dans une distribution avec une moyenne de zéro et une variance unitaire :

$$z = \frac{x - \mu}{\sigma} \tag{2.7}$$

où μ est la moyenne de l'attribut, σ l'écart type. Trois travaux utilisent des techniques de normalisation durant l'étape de pré-traitement [57], [65], [66]. Une autre technique de pré-traitement utilisé dans deux travaux [59], [65] est la discrétisation. La discrétisation d'un attribut consiste à transformer un attribut continu dans une forme discrète. Cela permet notamment de réduire le nombre de degrés de liberté d'un modèle et ainsi d'éviter le sur-apprentissage.

Sélection d'attributs : La sélection d'attributs est une technique couramment utilisée en apprentissage automatique qui consiste à sélectionner un sous-ensemble d'attributs à partir de l'ensemble des attributs initiaux. Elle permet de filtrer les attributs redondants et non pertinents. De plus, cela permet également de réduire la complexité du modèle et ainsi d'éviter le fléau de la dimension ou *curse of dimensionality*. Parmi les travaux présentés, deux types de méthodes sont utilisés. Tout d'abord, les méthodes de type *filter* sont utilisées dans [57]-[61], [64]. Ce type de méthodes sont indépendantes du modèle utilisé pour la classification et mesurent l'utilité pour d'un attribut par rapport à l'étiquette à prédire. Différentes mesures peuvent être utilisées telles de la corrélation ou l'information mutuelle. Dans [58], [59], [64] les auteurs ont également utilisé des méthodes de type *wrapper* qui intègrent le modèle dans le processus de sélection. Avec les méthodes de type *wrapper*, les attributs sont ajoutés ou supprimés de manière itérative au sous-ensemble d'attributs en évaluant leur impact sur les performances du modèle. Un inconvénient majeur de ce type de méthodes est une augmentation du temps d'apprentissage du modèle.

Algorithmes d'apprentissage automatique : Les machines à vecteurs de support ou *Support Vector Machine* (SVM) est une une classe d'algorithmes d'apprentissage automatique largement utilisé et réputé pour ses bonnes performances sur des tâches de classification variée. L'objectif d'un classificateur SVM est de trouver le meilleur hyperplan qui divise l'ensemble de données en fonction des classes à prédire. Parmi les articles présentés, six proposent d'utiliser un SVM pour la détection des anomalies BGP [57], [59], [61], [63], [64], [66]. Un défaut des SVM est qu'ils sont moins efficaces sur les jeux de données bruités qui contiennent des classes qui se chevauchent comme cela peut être le cas ici en

raison de l'imprécision du processus d'étiquetage des données.

Un autre algorithme très utilisé [58]-[60], [63], [64], [66] est le classificateur Naive Bayes (NB) qui est une famille de classificateurs qui reposent sur l'hypothèse que les caractéristiques sont indépendantes les unes par rapport aux autres. Les NB sont connus pour être simples à mettre en œuvre et rapides en calcul, mais peuvent conduire à de mauvais résultats si l'hypothèse d'indépendance est violée.

Dans [58], [59], [62], [66], [67] les auteurs ont choisi d'utiliser des algorithmes tels que le C4.5 pour construire un arbre de décision. L'un des avantages des arbres de décision est que le modèle est interprétable visuellement, mais ils sont sujets au sur-apprentissage. Les auteurs de [63], [64] ont utilisé des méthodes d'ensemble avec leurs algorithmes d'arbre de décision, tels que Adaboost et Random Forest, qui sont connus pour offrir de meilleures performances qu'un unique arbre de décision.

Les récents succès des algorithmes d'apprentissage profond ont attiré l'attention de la communauté de l'apprentissage automatique et certains des travaux examinés tentent également de les appliquer au problème de la détection des anomalies BGP. Dans [61], [63], [64], les auteurs ont utilisé des réseaux de réseaux neuronaux récurrents ou *Recurrent Neural Network* (RNN) et *Long Short-Term Memory* (LSTM). Ces algorithmes font partie d'une classe de réseaux de neurones artificiels conçus pour traiter des séquences de données. Ils sont utiles pour exploiter la dimension temporelle des données, ce qui peut être le cas des données d'anomalie BGP. D'autres architectures de réseaux de neurones artificiels telles que le *MultiLayer Perceptron* (MLP) [64], [65] et l'*Extreme Learning Machine* (ELM) [62] ont également été appliquées. Un inconvénient majeur des réseaux de neurones artificiels et de l'apprentissage profond est la difficulté d'interprétation du modèle entrainé et le temps nécessaire pour l'entraînement et l'ajustement des hyperparamètres du modèle.

Type de classification : Tous les articles présentés utilisent une approche de classification binaire où chaque enregistrement issu du jeu de données est classé comme "1" si une anomalie est prédite et "0" dans le cas contraire. Cependant, certains articles vont plus loin en appliquant une approche multi-classes. Dans [57], [60], les auteurs considèrent quatre classes où chaque classe est l'un des jeux de données d'apprentissage. Cependant, leurs jeux de données ne sont composés que de trois anomalies causées par des effets bords et d'un ensemble de données sans aucune anomalie. Dans [64] cinq classes d'événements sont utilisées parmi lesquelles trois anomalies causées par des effets bords, une anomalie causée par une fuite de route et un ensemble de données sans aucune anomalie.

Discussion

Pour conclure ce tour d'horizon de la littérature sur les détecteurs d'anomalies BGP basés sur l'apprentissage automatique, cette section discute des limites des travaux présentés et des orientations possibles pour les travaux futurs dans ce domaine.

Limites des travaux présentés : Deux types de limites peuvent être identifiés dans les travaux présentés. Le premier est lié aux jeux de données utilisés pour entraîner et évaluer les modèles et le second concerne les méthodologies qui sont appliquées dans les articles.

Un premier problème avec les jeux de données utilisés par les travaux examinés est le manque d'information sur les occurrences d'anomalies BGP. Cela conduit à un étiquetage arbitraire et approximatif des enregistrements qui peut affecter les performances du modèle. De plus, cela empêche l'automatisation complète du processus d'étiquetage ce qui explique le faible nombre d'anomalies présent dans les jeux de données. Évidemment, cela a un impact sur la généralisation du modèle. Une seconde limitation vient de l'agrégation de données qui sont collectées à partir de plusieurs points de collecte pendant k minutes. Ces données agrégées limitent les anomalies observables aux grands événements. En effet, un événement situé dans une petite portion d'Internet est peu susceptible d'être visible une fois les données agrégées. Un autre problème vient de la granularité des attributs extraits qui empêchent une interprétation fine. En effet, même si de tels attributs statistiques peuvent révéler l'existence d'une anomalie, certaines informations telles que les réseaux affectés sont impossibles à inférer.

Les choix méthodologiques opérés dans les articles pour la classification des anomalies présentent également certaines limites, notamment en ce qui concerne les types d'anomalies considérés. La majorité des travaux considèrent les anomalies causées par des effets de bords qui ont des impacts limités. Des anomalies plus nuisibles telles que le détournement de préfixe et la défaillance matérielle qui peuvent entraîner un trou noir sont également prises en compte dans certains travaux, mais seuls deux articles [58], [66] considèrent les deux à la fois. Un deuxième aspect limitant des méthodologies appliquées est l'approche multi-classes. Dans tous les articles qui évaluent leur modèle avec une approche multi-classe [57], [60], [64], les classes à prédire sont les différentes anomalies utilisées pour entraîner le modèle. Même si une telle approche peut être utile pour évaluer la distinction entre différentes anomalies, elle n'évalue pas la performance du modèle sur de nouvelles anomalies. Il serait plus utile de classer les événements par leurs types avec des classes génériques telles que : détournement de préfixe, pannes matérielles et effet de bord. Une limitation supplémentaire est qu'aucun des travaux n'aborde le problème de l'identification des réseaux impliqués dans l'anomalie et sa propagation alors qu'il s'agit d'un critère important pour qu'un détecteur d'anomalie BGP soit utile à un opérateur d'AS. Un autre critère important est la capacité d'un modèle à fonctionner en temps réel. Cependant, un seul article [66] aborde ce sujet.

Directions possibles : Avant de passer en revue les directions possibles qui pourraient être prises pour surmonter les limitations identifiées, une liste d'exigences clés pour un détecteur d'anomalies BGP est proposée en s'appuyant sur les travaux d'Al-Musawi et al. dans [16] :

— Détection de divers types d'anomalies : parmi les articles examinés, un seul [66] couvre toutes les causes d'anomalies listées dans la taxonomie proposée (c.f. figure 2.3).

— Identification du type d'anomalie : même si certains articles expérimentent l'approche multi-classes, ils considèrent tous des classes qui correspondent aux anomalies présentent dans le jeu de données. Aucun des articles n'aborde le problème de l'identification du type d'anomalie observé.
— Identification des réseaux impliqués : pour être utile en pratique, un critère important pour un détecteur d'anomalie BGP est sa capacité à identifier les réseaux impliqués dans une d'anomalie. C'est-à-dire, l'AS à l'origine de l'anomalie, les AS qui ont propagés l'anomalie et l'AS victime. Cependant, aucun des travaux examinés ne propose de solution à ce problème.
— Détection en temps réel : pour que l'administrateur d'un AS puisse réagir plus rapidement, un détecteur d'anomalie BGP doit fonctionner en temps réel avec un laps de temps le plus court possible entre l'occurrence de l'anomalie et sa détection. Cependant, un seul article [66] considère cette question dont le principal obstacle est probablement le délai pour que les données BGP soient disponibles à partir de la source de données utilisée.

Un problème récurrent dans les travaux présentés est le manque d'informations précises sur l'occurrence d'anomalies BGP qui limite la diversité des événements utilisés dans les jeux de données. Une solution pour obtenir des données précises consiste à utiliser des informations provenant d'entités de gouvernance telles que les *Route Origination Authorization* (ROA) ou les *Internet Routing Registry* (IRR). Les ROA qui font partie du mécanisme *Resource Public Key Infrastructure* (RPKI) certifient quel AS est à l'origine d'un préfixe. Les bases de données IRR sont gérées par les registres Internet régionaux ou *Regional Internet Registry* (RIR) et contiennent des informations gérées par les administrateurs des AS à propos de leurs connexions et politiques de routage. Cependant, l'imprécision des IRR due à leur manque de maintenance et la faible adoption de RPKI (37% des préfixes [38] en juin 2022) limitent la partie des données BGP qui peuvent être étiquetées en s'appuyant ces dernières. Une autre solution consiste à générer des anomalies BGP dans un environnement simulé qui produira des données BGP étiquetées avec précision. Cependant, la difficulté est double. Premièrement, la topologie simulée doit être réaliste avec la topologie de l'Internet actuel en termes d'interconnexions, de politiques de routage et de pratiques d'ingénierie de trafic. Deuxièmement, les anomalies simulées doivent également être réalistes et couvrir le plus grand nombre d'anomalies possibles. Ces deux conditions sont difficiles à remplir et si elles sont violées, elles conduiront à un modèle qui ne correspond pas au problème réel de la détection des anomalies BGP. Enfin, le manque d'informations précises sur l'occurrence d'anomalies BGP peut également être surmonté en utilisant des techniques non supervisées. De telles techniques sont susceptibles d'être de bons candidats pour couvrir un large éventail de types d'anomalies, mais pas pour l'identification du type de l'anomalie.

Une limitation importante des travaux examinés est le type d'attribut utilisé. Ces attributs statistiques sont extraits à partir de données agrégées à partir de plusieurs points de collecte. De telles données limitent la granularité des anomalies observables et empêchent

l'identification des réseaux impliqués. Pour résoudre ces problèmes, il est nécessaire d'exploiter des représentations plus complexes des données BGP. Les graphes représentant la topologie du réseau BGP sont de bons candidats car ils permettent de conserver beaucoup plus d'information par rapport aux attributs utilisés dans les travaux présentés.

2.4.4 Approches basées sur les graphes

Plusieurs travaux ont proposé des solutions afin de prendre en compte la topologie du réseau BGP pour la détection des anomalies [4], [68]-[70]. Pour cela, un graphe représentant les connexions entre les AS est construit à partir de l'ensemble des routes (cf section 2.2). Dans [68] les auteurs introduisent une métrique pour la détection des anomalies BGP basé sur une visualisation hiérarchique du graphe BGP. Les auteurs ont montré que leur approche était susceptible de détecter à la fois des anomalies de grande échelle telles que des fuites de route que des anomalies de plus petite échelle telles que des détournements de préfixe. Une autre métrique pour la détection des anomalies BGP est proposée dans [69] où les auteurs proposent une approche géométrique pour l'analyse du graphe BGP. Ainsi, ils proposent de détecter une anomalie BGP au travers de la déformation géométrique qu'elle provoque sur le graphe BGP.

Dans [70] et [4], les auteurs s'appuient sur des métriques calculées sur des graphes BGP afin d'extraire des attributs adaptés aux algorithmes d'apprentissage automatiques. Ainsi, dans [4] des métriques de la théorie des graphes telles que la centralité et le coefficient de clustering sont utilisés tandis que dans [70] une métrique spécifique à BGP appelée *AS Hegemony* [71] est utilisée.

2.5 Apprentissage profond sur les graphes

Geometric Deep Learning

Les approches pour appliquer l'apprentissage profond aux graphes peuvent être classées en deux catégories. Premièrement, les *Graph Neural Networks* (GNN) [72] visent à adapter les techniques d'apprentissage profond afin qu'elles puissent prendre en entrée des graphes. La deuxième catégorie regroupe les méthodes de plongement de graphes ou *graph embedding* [73] qui projettent le graphe dans un espace vectoriel de faible dimension compatible avec les algorithmes traditionnels d'apprentissage profond.
Le terme *Geometric Deep Learning* [3] a émergé pour désigner des techniques d'apprentissage profond qui peuvent être appliquées à des domaines non euclidiens tels que les graphes et les variétés ou *manifolds*. Les domaines d'application sont variés tels que les réseaux sociaux, les modèles 3D d'objets et la science moléculaire.

2.5.1 Graph Neural Networks

Les travaux sur les GNN sont motivés par l'incapacité des architectures conventionnelles de réseaux de neurones à être appliquées à des données complexes appelées données non euclidiennes. La principale motivation derrière les GNN est d'avoir un fonctionnement équivalent aux réseaux neuronaux convolutif ou *Convolutional Neural Networks* (CNN) mais pour des données non euclidiennes. Les GNN visent à obtenir des propriétés des CNN telles que l'invariance de décalage et le partage de paramètres. Initialement, deux visions différentes se sont confrontées dans la littérature pour la création de GNN. La première repose sur des mécanismes d'agrégation spatiale alors que la seconde repose sur la définition de l'opération de convolution sur des graphes dans le domaine spectral.

Convolution sur les graphes

Le principal fondement théorique des GNN est la généralisation de l'opération de convolution habituellement appliquée à des données représentées sous forme de grille (par exemple des images) à des données non euclidiennes [3] telles que des graphes ou des variétés. De manière similaire à la convolution traditionnelle, l'idée derrière la convolution sur les graphes est de calculer une représentation pour chaque noeud en fonction de son voisinage. L'adaptation d'une telle opération est intéressante, car elle a le potentiel de tirer profit de la structure du graphe, mais aussi de bénéficier des propriétés des CNN telles qu'un nombre de paramètres du modèle indépendant de la taille des données d'entrée. La littérature sur la convolution sur les graphes peut être classée en deux catégories [72] : les méthodes spatiales et les méthodes spectrales.

Les méthodes spatiales : Les méthodes spatiales ont constitué la première approche afin appliquer la convolution sur des graphes proposés par Scarselli et al. [74]. Dans leur papier, Scarselli et al. proposent un modèle récurrent où chaque couche de convolution t c'est-à-dire chaque récurrence du modèle calcule une sortie appelée représentation cachée. Cette représentation est définie comme :

$$h_n^t = f_w(l_n, l_{co[n]}, h_{ne[n]}^{t-1}, l_{ne[n]}) \tag{2.8}$$

où h_n^t est la représentation cachée du nœud n, l_n est le vecteur des attributs du nœud n, $l_{co[n]}$ est le vecteur des attributs des arêtes de n, $h_{ne[n]}^{t-1}$ est le vecteur des représentations cachées des voisins de n à la couche précédente, $l_{ne[n]}$ est le vecteur des attributs des voisins de n et w sont les poids pour la fonction $f(\cdot)$. La fonction $f(\cdot)$ est généralement un réseau de neurones. Cependant, $f(\cdot)$ doit être une application contractante pour assurer qu'un état stationnaire sera atteint après un nombre fini T de récurrences. Ainsi, lorsque le point fixe est atteint, la sortie est calculée à partir des représentations cachées. Des travaux plus récents [75] utilisent des *Gated Recurrent Units* (GRU) et une rétro-propagation du gradient dans le temps avec un nombre T de récurrence fixe. Avec cette approche, les auteurs suppriment le besoin d'atteindre l'état stationnaire et donc la contrainte que la fonction $f(\cdot)$ soit une application contractante. Cependant, cela consomme plus de mémoire vive, car tous les états intermédiaires doivent être stockés pour le calcul du gradient.

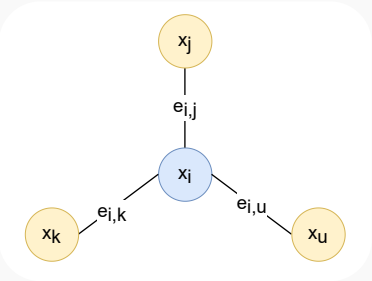
Les méthodes spectrales : Les méthodes spectrales ont été introduites dans [76] et exploitent la définition de l'opération de convolution dans le domaine spectral. Un modèle largement utilisé dans la littérature est le *Graph Convolutional Networks* (GCN) proposé par Kipf et al. [77]. Dans un GCN, la sortie $Z \in \mathbb{R}^{N \times F}$ d'une couche de convolution sur un graphe est définie comme :

$$Z = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Theta \tag{2.9}$$

où $X \in \mathbb{R}^{N \times C}$ est la matrice des attributs où chaque noeud $n \in \mathbb{N}$ dispose de C attributs, $\tilde{A} = A + I_N$ est la matrice d'adjacence, $\tilde{D}^{-\frac{1}{2}} = \sum_j \tilde{A}_{ij}$ est la matrice des degrés et $\Theta \in \mathbb{R}^{C \times F}$ est une matrice de poids contenant F filtres. Même si la convolution a lieu dans le domaine spectral, l'approche définie par Kipf et al. [77] est localisée dans l'espace. Autrement dit, chaque scalaire $z_{ij} \in Z$ correspond à l'application d'un filtre $f \in F$ sur le voisinage à un saut d'un nœud $n \in N$. Cependant, un inconvénient majeur de la convolution spectrale est que les filtres appris lors de l'entraînement du modèle dépendent de la structure du graphe. Un modèle entraîné doit donc toujours être appliqué sur un graphe de même structure, seuls les attributs peuvent changer.

Message Passing Neural Networks (MPNN)

Récemment, les méthodes de convolution spectrales et spatiales ont été unifiées sous des paradigmes d'agrégation de voisinage ou de passage de messages [78], [79]. Dans ces paradigmes, pour chaque noeud du graphe, les états de ses voisins sont agrégés et utilisés pour calculer le nouvel état du noeud. En empilant plusieurs couches de cette même opération, les informations peuvent se propager d'un saut à chaque couche.



Dans l'exemple ci-dessus, le nouvel état du noeud x_i est calculé à l'aide des fonctions *message*, *aggregation* et *update*, tel que :

$$\begin{aligned} x_i = \mathit{update}(x_i, \\ \mathit{aggregation}(\mathit{message}(x_i, x_j, e_{i,j}), \\ \mathit{message}(x_i, x_k, e_{i,k}), \\ \mathit{message}(x_i, x_u, e_{i,u})) \\ ) \end{aligned}$$

Note : le MPNN tel qu'il est proposé dans [79] s'applique sur des graphes non-dirigés c'est-à-dire que $e_{i,k} = e_{k,i}$.

Pour décrire un MPNN de façon plus formelle, considérons un graphe d'entrée où un noeud i est défini par son vecteur de caractéristiques $x_i \in \mathbb{R}^F$ (où F est le nombre de caractéristiques du noeud). Un lien entre le noeud i et le noeud j est désigné par son vecteur de caractéristiques $e_{i,j} \in \mathbb{R}^{\mathbb{D}}$ (où D est le nombre de caractéristiques du lien). Le vecteur de caractéristiques d'un noeud est mis à jour au niveau de la couche k par :

$$x_i^k = \gamma(x_i^{k-1}, \psi_{j \in N(i)}(\phi^k(x_i^{k-1}, x_j^{k-1}, e_{i,j}))) \tag{2.10}$$

où :

— ϕ : la fonction *message* est une fonction différentiable telle qu'un *Multi Layer Perceptron* (MLP)

— ψ : la fonction *aggregation* est une fonction différentiable et invariante par permutation telle que la somme, la moyenne ou le maximum.

— γ : la fonction *update* est une fonction différentiable comme un MLP

— $N(i)$: est l'ensemble des voisins du noeud i

Sous cette définition les différents modèles de GNN peuvent être définis par leurs fonctions ϕ, ψ et γ spécifiques. De plus, plusieurs unités de GNN peuvent être utilisées dans la même couche afin d'avoir une sortie multicanal. Cette sortie multicanal produit donc un vecteur pour chaque noeud du graphe appelé plongement du noeud ou *node's embedding*.

Architectures des GNN

À l'instar des architectures conventionnelles de réseaux de neurones et des CNNs dont ils reprennent des principes, les GNN peuvent avoir des architectures très variées. Ces architectures se différencient principalement en fonction des blocs de calcul utilisés et de la tâche d'apprentissage.

Blocs de calculs : L'opération de convolution utilisée dans un GNN requière d'agréger les états des noeuds voisins pour calculer le nouvel état d'un noeud. Lorsque la taille du graphe augmente et que le nombre liens augmente, cela peut entraîner une explosion du nombre de calculs à effectuer. Pour éviter ce problème, plusieurs travaux proposent des mécanismes d'échantillonnage. Dans [80], [81], les auteurs proposent proposent d'échantillonner le voisinage d'un noeud lors de l'agrégation. Cela permet ainsi de réduire la taille du voisinage des noeuds. Une autre approche consiste à échantillonner les noeuds entre chaque couche du GNN [82]. Ainsi, à chaque couche du GNN un ensemble de noeuds est échantillonné. Lors du calcul du nouvel état d'un noeud, seuls les états des voisins faisant partie des noeuds échantillonnés sont utilisés. Enfin, certains travaux proposent d'échantillonner des sous-graphes [83]. À chaque couche du GNN un sous-graphe est échantillonné et seuls les noeuds appartenant à ce sous-graphe sont utilisés lors de l'agrégation.

Dans les CNN, une couche de convolution est généralement suivie d'une couche de *pooling* afin d'extraire des informations plus génériques. Deux catégories d'approches sont proposées dans la littérature afin d'effectuer une opération de *pooling*. La première est le *direct pooling* également appelée *readout* dont l'objectif est de calculer une représentation globale à partir de l'ensemble des noeuds du graphe. L'approche la plus simple consiste à calculer cette valeur en utilisant une opération telle que la moyenne, le maximum ou la somme. Dans [84], les auteurs proposent l'opération *SortPooling* qui ordonne les noeuds en fonction de la structure du graphe puis applique un CNN pour calculer la représentation globale. La seconde catégorie d'approches utilisées pour effectuer une opération de *pooling* dans les GNN est le *hierarchical pooling*. Avec cette approche, la structure du graphe est utilisée pour calculer des représentations hiérarchiquement [85]-[87]. L'approche consiste généralement à partitionner le graphe en agrégeant les noeuds les plus proches. Pour chaque partition une représentation est calculée et l'opération est répétée afin d'obtenir une représentation hiérarchique.

Le mécanisme d'attention [88] est connu pour le gain de performance qu'il permet d'obtenir notamment pour la reconnaissance du langage naturel. L'idée derrière ce mécanisme est de permettre au modèle de se concentrer sur une partie des données pour

produire sa prédiction. L'importance à accorder aux différentes parties des données étant apprise durant l'entraînement du modèle. Certains travaux proposent des solutions afin de transposer ce mécanisme aux graphes en permettant à un noeud d'affecter des poids à ses voisins [89], [90].

Tâches d'apprentissage : Dans la littérature, les GNN sont généralement utilisés pour trois tâches d'apprentissage. La première est l'apprentissage supervisé pour la classification de graphes [84], [87]. Le modèle doit donc prendre en entrée un graphe et produire en sortie la classe du graphe. Pour cela, un *readout* est généralement utilisé afin de calculer une représentation globale. Un *Multi-Layer Perceptron* (MLP) est ensuite appliqué à cette représentation pour produire la classe à prédire.

La seconde tâche d'apprentissage pour lequel les GNN peuvent être utilisés est l'apprentissage semi-supervisé pour la classification des noeuds [77]. Dans ce type de problème de classification, l'entrée est généralement un graphe dont seulement un sous-ensemble des noeuds est étiqueté. L'objectif est de prédire l'étiquette des noeuds non étiquetés.

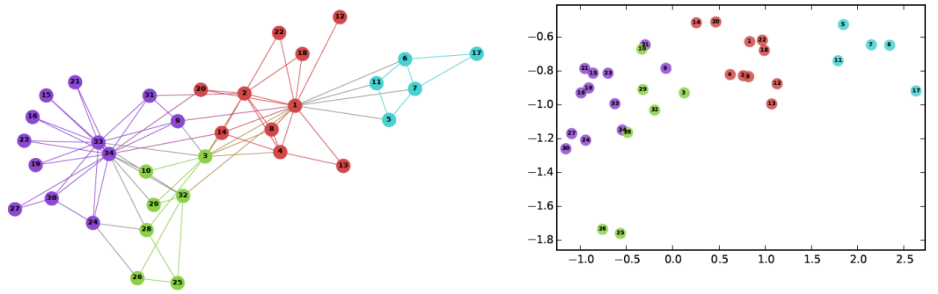
La dernière tâche d'apprentissage est l'apprentissage non-supervisé pour la génération ou le plongement de graphe. Pour cela, une solution courante est l'utilisation d'une architecture auto-encodeur [91] composé de deux parties : un encodeur et un décodeur. L'encodeur utilise plusieurs couches de convolution pour produire une représentation latente du graphe qui est ensuite donnée en entrée au décodeur dont l'objectif est de reconstruire de graphe d'entrée. Le GNN peut ainsi être entraîné de façon non-supervisé en minimisant l'erreur de reconstruction. La partie encodeur peut ensuite être utilisée pour produire un plongement du graphe tandis que la partie décodeur peut être utilisée pour générer de nouveaux graphes appartenant à la même distribution que celle des données d'apprentissage.

2.5.2 Plongement de graphes

Dans le domaine de l'apprentissage automatique, les techniques de plongement ou d'*embedding* sont largement utilisées afin de représenter des données complexes sous forme de vecteurs de dimension relativement faibles. Ces techniques sont largement utilisées avec les textes où chaque mot sera représenté par un vecteur encodant sa sémantique. Les mots sémantiquement proches seront donc proches dans l'espace de plongement. Ainsi, le plongement d'un graphe est la représentation de ce dernier sous la forme d'un vecteur ou d'un ensemble de vecteurs. Le plongement doit également conserver certaines propriétés du graphe telles que sa topologie ou la similarité des noeuds.

Plongement de graphe

En général, le plongement d'un graphe consiste à encoder chaque noeud du graphe sous la forme d'un vecteur $Y_i \in \mathbb{R}^d$ où d est la dimension de l'espace de plongement. Le graphe étant alors représenté par une matrice $Y \in \mathbb{R}^{n \times d}$ où n correspond au nombre de noeuds du graphe.



Exemple de plongement de graphe [6]

La figure ci-dessus donne un exemple de plongement de graphe dans un espace à deux dimensions. À gauche, le graphe du réseau social *Zachary's karate club*, où les noeuds sont connectés si les individus correspondants sont amis. Les noeuds partagent la même couleur s'ils appartiennent à la même communauté. À droite, la visualisation 2D des plongements des noeuds générés à l'aide de la méthode DeepWalk [7]. Les distances entre les noeuds reflètent la similarité dans le graphe d'origine.

Parfois, le plongement est effectué au niveau du graphe entier et consiste donc à représenter le graphe par un vecteur $Y \in \mathbb{R}^d$. Cela est utile lorsque les tâches subséquentes concernent le graphe entier, par exemple dans le cadre de la classification de graphes. De façon formelle le plongement d'un graphe est défini comme une application :

$$f : v_i \mapsto Y_i \in \mathbb{R}^d, \text{ où } d \ll n \tag{2.11}$$

Dans la littérature les méthodes de plongement de graphes sont classées dans trois catégories : les méthodes basées sur la factorisation, celles basées sur des marches aléatoires et celles basées sur l'apprentissage profond.

Factorisation

Les méthodes de plongement de graphes basées sur la factorisation s'appuient sur les représentations sous forme de matrice d'un graphe (cf section 2.2) et sur leur factorisation. Les matrices utilisées pouvant être par exemple, la matrice d'adjacence ou la matrice Laplacienne. Par exemple, la méthode *Locally Linear Embedding* (LLE) [92] s'appuie sur la matrice d'adjacence en partant de l'hypothèse que le plongement d'un noeud est une combinaison linéaire des plongements de ses noeuds voisins. Ainsi, pour un noeud i le

poids d'un noeud j dans le calcul de cette combinaison linéaire est l'élément W_{ij} de la matrice d'adjacence du graphe. Le plongement du noeud i est défini comme :

$$Y_i = \sum_j W_{ij} Y_j \tag{2.12}$$

Ainsi, le plongement $Y \in \mathbb{R}^{n \times d}$ peut être obtenu en minimisant la fonction objectif suivante :

$$\Phi(Y) = \sum_i |Y_i - \sum_j W_{ij} Y_j|^2 \tag{2.13}$$

Des contraintes d'optimisation sont également définies afin d'éviter les solutions dégénérées [92]. Il existe une variété de méthodes de plongement de graphes basées sur la factorisation dans la littérature qui apportent des propriétés différentes en redéfinissant la fonction objectif $\Phi(Y)$.

Marche aléatoire

Une marche aléatoire de longueur l sur un graphe est un processus en l étapes qui commence sur un noeud v_0. À chaque étape, un déplacement est effectué vers un des voisins du noeud courant. Si le graphe est pondéré, la probabilité de choisir un voisin est proportionnelle au poids du lien avec ce voisin. Sinon, tous les voisins ont la même probabilité d'être choisis. Inspiré par le succès de [2] pour la reconnaissance du langage naturel, DeepWalk [7] utilise l'algorithme SkipGram [2] pour le plongement des noeuds du graphe. Pour un noeud v_i appartenant à une marche aléatoire $W = [v_0, \ldots, v_i, \ldots, v_l]$, l'objectif de prédire les noeuds précédents et suivants v_i à partir du plongement du noeud. Le plongement Y_{v_i} du noeud v_i est donc obtenu en maximisant la probabilité :

$$P_i = P(\{v_{i-w}, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{i+w}\} | Y_{v_i}) \tag{2.14}$$

où w est la taille de la fenêtre de prédiction. Le plongement $Y \in \mathbb{R}^{N \times D}$ peut être obtenu en minimisant la fonction objectif suivante :

$$\Phi(Y) = \sum_i -\log P_i \tag{2.15}$$

D'autres méthodes [93], [94] proposent des améliorations de DeepWalk. Par exemple en utilisant une approche différente pour la recherche des marches aléatoires.

Apprentissage profond

Avec les avancés récentes dans le domaine de l'apprentissage profond, certains travaux ont proposé d'utiliser ces méthodes pour le plongement de graphe. Le modèle *Structural deep network embedding* [95] (SDNE) utilise une architecture auto-encodeur qui prend en entrée la matrice d'adjacence du graphe et produit un plongement du graphe. Le modèle est entraîné de façon non-supervisé en reconstruisant la matrice d'adjacence à

partir du plongement du graphe. La fonction objectif résultante est :

$$\mathcal{L}_{2nd} = \sum_i \|(\hat{A}_i - A_i) \odot b_i\|_2^2 \tag{2.16}$$

où A_i est le vecteur d'adjacence du noeud i, $\hat{A}_i$ est la reconstruction produite par l'auto-encodeur et $\odot$ le Le produit de Hadamard. Le vecteur d'adjacence étant généralement de nature éparse, le terme b_i introduit un biais donnant plus d'importance aux éléments non nuls. Afin que le plongement du graphe capture également la structure locale, le modèle est également entraîné de façon à ce que les noeuds voisins soient proches dans l'espace de plongement. La fonction objectif suivante est alors définie :

$$\mathcal{L}_{1st} = \sum_{i,j} A_{i,j} \|Y_i - Y_j\|_2^2 \tag{2.17}$$

Afin de satisfaire ces deux objectifs le modèle est entraîné en minimisant la fonction objectif mixe :

$$\mathcal{L}_{mix} = \mathcal{L}_{2nd} + \alpha \mathcal{L}_{1st} + v\mathcal{L}_{reg} \tag{2.18}$$

où $\mathcal{L}_{reg}$ est un terme de régularisation pour éviter le surapprentissage du modèle.

Plutôt que d'utiliser un auto-encodeur classique, il est possible d'utiliser un GNN pour produire le plongement du graphe. Ainsi, dans [91] les auteurs proposent le modèle *Variational Graph Auto-Encoder* (VGAE) où un GCN [77] à deux couches est utilisé pour produire un plongement Y du graphe. La partie décodeur reconstruit la matrice d'adjacence en effectuant le produit des plongements des noeuds :

$$\hat{A} = \sigma(YY^T) \tag{2.19}$$

où $\sigma(.)$ est la fonction d'activation sigmoïde. Au lieu d'utiliser un auto-encodeur classique, un auto-encodeur variationnel est utilisé. Ce type d'auto-encodeur plutôt que d'encoder un noeud comme un point dans l'espace de plongement va l'encoder sous la forme d'une distribution. Le modèle est entraîné un minimisant la fonction objectif :

$$\mathcal{L} = \mathbb{E}_{q(Y|X,A)}[\log p(A|Y)] - KL[q(Y|X,A)\|p(Y)] \tag{2.20}$$

où $\mathbb{E}_q(.)[\log p(.)]$ est l'erreur de reconstruction et $KL[q(.)\|p(.)]$ est un terme de régularisation pour contrôler la distribution du plongement produit.

2.5.3 Applications à des domaines connexes

Dans la littérature, des travaux ont appliqué les GNN et les méthodes de plongement de graphes à des thématiques proches du sujet de cette thèse. Ainsi, plusieurs travaux proposent des méthodes de plongement de graphes adaptées aux graphes dynamiques [96]-[98] et qui ont été évaluées sur des jeux de données de graphes BGP. Ces méthodes ont été évaluées sur des tâches de prédiction de lien. Pour cela, un graphe BGP dont une partie des liens entre les AS ont été cachés est donné en entrée au modèle et l'objectif est

de prédire ces liens cachés. Dans [96] le plongement du graphe s'appuie sur une architecture auto-encodeur traditionnelle tandis que dans [97] un *Graph Auto-Encoder* [91] est utilisé. Un plongement de graphes basé sur des marches aléatoires est utilisé dans [98] où les auteurs proposent une modification de la méthode *node2vec* [93].

Des travaux ont également appliqué des GNN [99], [100] et des méthodes de plongement de graphes [98], [101] à des problématiques de détection d'anomalies dans des graphes. Dans [99], les auteurs proposent un classement des noeuds non-supervisé à l'aide d'une architecture auto-encodeur basée sur un GCN [77]. Le modèle est entraîné de façon à reconstruire la structure du graphe et la matrice des attributs des noeuds à partir des plongements des noeuds. Dans [100], les auteurs proposent un modèle de bout-en-bout pour la détection de liens anormaux. L'approche s'appuie sur un GCN [77] ainsi que sur un mécanisme d'attention. Un plongement de graphes basé sur une architecture auto-encodeur est utilisé dans [96] alors que des marches aléatoires sont utilisées dans [101] et [98].

2.6 Conclusion

Dans cet état de l'art, le protocole BGP ainsi que les différents types d'anomalies auxquels il est vulnérable ont été présentés. Les causes de ces anomalies sont variées et elles peuvent être d'origine malicieuse ou due à des erreurs de configuration. Leurs conséquences sont également très variable allant d'une surcharge des routeurs BGP jusqu'à des perturbations importantes sur le plan de données (*e.g.* indisponibilité d'un service, usurpation, interception de trafic). Même si des efforts sont mis en oeuvre pour améliorer la sécurité de BGP, ils restent insuffisants et il est donc indispensable de travailler sur des solutions pour la détection des anomalies BGP. La littérature s'est beaucoup penchée sur le problème de la détection des détournements de préfixes en proposant des solutions ad-hoc. Cependant, ces solutions souffrent de limitations telles que la faible précision de la détection ou la difficulté de déploiement ou de maintenance. Une autre ligne de recherche s'est concentrée sur l'utilisation de techniques d'apprentissage automatique pour la détection d'anomalies BGP. Une grande partie de ces travaux suivent une méthodologie similaire consistant à extraire des attributs statistiques à partir des données BGP. Cependant, ces travaux sont généralement limités par les jeux de données utilisés qui couvrent un sous-ensemble des anomalies BGP. De plus, ces jeux de données n'étant pas standardisés les résultats sont difficilement reproductibles. D'autres travaux, ont proposé d'exploiter le graphe BGP pour y détecter des anomalies. Le potentiel de ce type d'approche a été présenté notamment grâce aux avancés récentes dans le domaine de l'apprentissage profond sur les graphes notamment les *Graph Neural Networks* (GNN).

Suite à ce travail bibliographique, les trois objectifs principaux de cette thèse ont été identifiés. Pour rappel, ces objectifs sont :

1. Faciliter et encourager la reproductibilité des résultats pour la détection des anomalies BGP.
2. Identifier l'intérêt des représentations des données BGP sous forme de graphe par rapport aux attributs généralement utilisés.
3. Déterminer si les GNN permettent de tirer profit des représentations des données BGP sous forme de graphe.

La contribution présentée dans le chapitre 3 traite le premier objectif tandis que les chapitres 4 et 5 se concentrent sur le deuxième. Enfin, le dernier objectif est exploré dans le chapitre 6.

Chapitre 3

Construction de jeux de données BGP pour l'apprentissage automatique

Sommaire

Ce chapitre présente un travail préliminaire à l'application des techniques d'apprentissage automatique à BGP, la construction de jeu de données BGP. L'outil résultant, BML, permet d'automatiser le processus de construction d'un jeu de données BGP et offre différentes transformations des données qui sont compatibles avec les algorithmes d'apprentissage automatique conventionnels. Les caractéristiques de BML et son fonctionnement sont présentés puis sa facilité d'utilisation est démontrée au travers de deux cas d'applications.

Un pre-requis à l'utilisation des techniques d'apprentissage automatique est d'avoir à sa disposition un jeu de données regroupant d'un ensemble d'observation du problème considéré. Dans le cadre de cette thèse un tel jeu de données contiendrait des observations du protocole BGP au cours de différentes anomalies BGP. Cependant, il n'existe pas de consensus autour d'un jeu de données de référence d'anomalies BGP. En effet, dans la littérature les travaux utilisant des techniques d'apprentissage automatique pour entraîner un modèle pour la détection des anomalies BGP construisent généralement leur propre jeu de données. Ce travail fastidieux peut donc freiner l'émergence de nouveaux travaux et rend difficile la reproduction des résultats. Afin de répondre à ces problèmes, un outil nommé BML a été construit durant cette thèse et rendu accessible publiquement. Cet outil permet de simplifier et d'accélérer la construction de jeux de données BGP. À partir d'une configuration initiale, il permet de construire automatiquement un jeu de données BGP pouvant être utilisé notamment pour l'entraînement de modèles d'apprentissage automatique. Cette section présente le fonctionnement et les caractéristiques de BML ainsi que des cas d'utilisations. L'outil BML présenté dans cette section a fait l'objet d'une publication [102].

3.1 Présentation de BML

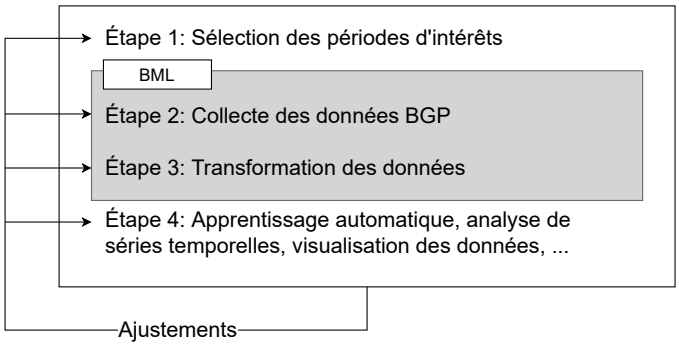


FIGURE 3.1 – Processus de construction d'un jeu de données BGP

BML est un outil pour la construction de jeux de données BGP qui a été élaboré durant cette thèse. Bien que cet outil ait été pensé en priorité pour l'application des techniques d'apprentissage automatique aux données BGP, son usage ne se limite pas à cette problématique. En effet, BML peut être utilisé pour la construction de jeux de données avec des objectifs divers tels que l'analyse statistique des données BGP, ou la visualisation de ces données.

3.1.1 Processus de construction d'un jeu de données BGP

La figure 3.1 décrit le processus de construction d'un jeu de données BGP dans lequel BML s'inscrit. Ce processus est composé de 4 étapes.

Étape 1 : La première est la sélection des périodes d'intérêt. Dans BML, une période d'intérêt correspond à une fenêtre temporelle qui intéresse l'utilisateur et qui sera donc incluse dans le jeu de données. Ainsi, une période d'intérêt P^x est définie par :

$$P^x = [t_a^x, t_b^x] \tag{3.1}$$

où t_a^x et t_b^x sont les *timestamps* correspondant au début et à la fin de la période d'intérêt.

Étape 2 : La seconde étape est la collection des données BGP. Pour chaque période d'intérêts, BML va automatiquement récupérer les données BGP collectées par différents routeurs BGP dans la fenêtre temporelle correspondante.

Étape 3 : La troisième étape est la transformation des données. Pour cette étape, l'utilisateur doit spécifier une fonction de transformation $T(.)$ qui prend en entrée des données BGP et Δ une période d'échantillonnage. Ainsi, BML va découper les données BGP avec un intervalle de temps Δ et appliquer à chaque segment de données BGP la transformation $T(.)$. Le résultat de cette opération est alors une séquence :

$$[T_1, T_2, \ldots, T_{n-1}, T_n] \tag{3.2}$$

où le nombre d'échantillons est $n = \lfloor \frac{t_b^x - t_a^x}{\Delta} \rfloor$. À la fin de cette, étape le travail de BML est terminé et l'utilisateur dispose de son jeu de données. Une représentation d'un jeu de données BGP d'anomalies BGP est donnée ci-dessous :

Id	**Attributs**	**Classe**
P^1	$[T_1, T_2, \ldots, T_{59}, T_{60}]$	Anomalie
P^2	$[T_1, T_2, \ldots, T_{59}, T_{60}]$	Anomalie
...	...	...
P^{499}	$[T_1, T_2, \ldots, T_{59}, T_{60}]$	Normal
P^{500}	$[T_1, T_2, \ldots, T_{59}, T_{60}]$	Normal

Ce jeu de données contient 500 périodes d'intérêt avec pour chacune d'entre-elles 60 échantillons de transformation.

Étape 4 : La dernière étape du processus est l'utilisation du jeu de données sur la tâche souhaitée par l'utilisateur telle que l'entraînement d'un modèle d'apprentissage automatique. Cependant, cette étape n'est généralement pas définitive. En effet, en fonction des résultats obtenus, l'utilisateur voudra généralement effectuer certains ajustements tels que l'ajout de périodes d'intérêt, le choix des routeurs BGP à utiliser pour la collecte des données ou la période d'échantillonnage. Ce processus est donc un processus itératif qui prend fin lorsque l'utilisateur obtient des résultats satisfaisants. Dans la conception de BML, un intérêt particulier a donc été porté à la nature itérative de ce processus afin de rendre chaque nouvelle itération la plus simple et la plus rapide possible.

3.1.2 Caractéristiques de BML

L'objectif de BML est d'offrir une solution générique pour la construction de jeux de données BGP. De plus, face au volume important de données BGP, une attention particulière a été portée à l'efficacité de la solution afin de rendre possible la collection de jeux de données de grande dimension. Ainsi, lors de sa conception différentes caractéristiques nécessaires ont été identifiées. Ces caractéristiques sont présentées brièvement ci-dessous avant d'être détaillées dans les sections suivantes.

Flexibilité de la transformation : Afin d'être le plus générique possible, une première condition est de ne pas limiter l'utilisateur à un ensemble restreint de transformations possibles. Ainsi, BML permet à l'utilisateur de définir sa propre fonction de transformation ou d'utiliser celles déjà implémentées. Cette fonction de transformation peut être définie à partir d'une fonction de transformation générique ou à partir des fonctions de transformation déjà implémentées.

Paramétrable : En plus de sa flexibilité au niveau de la transformation, BML offre divers paramètres pour la collection et la transformation des données. Ces paramètres peuvent notamment être utilisés pour établir un compromis entre quantité de données à analyser et la charge de calcul.

Évite la re-collection des données : La collecte des données BGP est une étape qui implique le téléchargement et l'analyse d'un important volume de données et qui par conséquent peut être chronophage. BML a donc été construit de façon à éviter d'avoir à exécuter cette étape plusieurs fois. Ainsi, après une itération si les ajustements impliquent l'étape 3 ou 4 alors il ne sera pas nécessaire de ré-exécuter les étapes précédentes. Cela implique donc de conserver les données BGP.

Limitation du besoin en stockage : Avec la conservation des données BGP arrive le problème du stockage. Pour limiter les besoins en stockage différents mécanismes ont été mis en place afin de stocker uniquement les données nécessaires pour l'exécution des étapes 3 et 4, mais aussi en compressant les données.

Parallélisation et distribution : Afin d'exploiter le plus possible la puissance de calcul disponible, BML permet de paralléliser diverses opérations à la fois pour la collecte et la transformation des données. Sur des systèmes multi-cœurs cela permet de réduire significativement les temps d'exécutions. De plus, BML offre la possibilité de fragmenter un jeu de données afin de répartir les tâches de collection ou de transformation de données sur plusieurs machines.

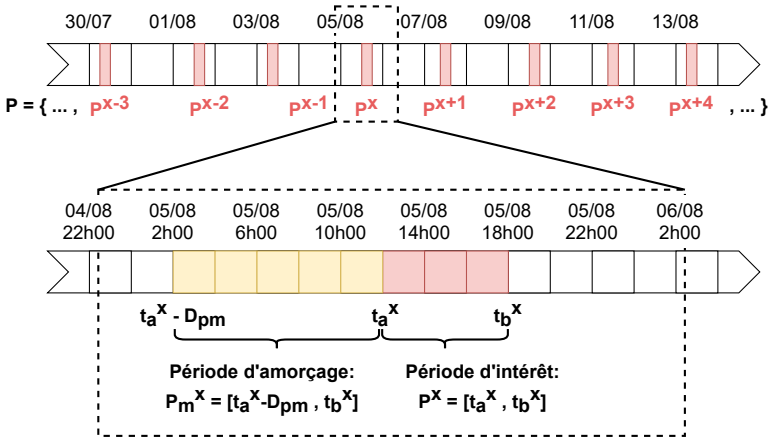


FIGURE 3.2 – Collecte de données BGP avec BML

3.2 Collecte des données BGP

Pour chaque période d'intérêt P^x, l'objectif de BML est de récupérer des données BGP collectées durant une fenêtre temporelle $[t_a^x, t_b^x]$. Pour cela, il est donc nécessaire d'avoir accès à des données historiques du protocole. Ainsi, BML s'appuie sur les projets de collecte et d'archivage de données BGP Ripe RIS et Route Views (cf section 2.4.1). Ces projets permettent de récupérer des messages UPDATE collectés par différents routeurs BGP ainsi que des *dumps* de leurs *Routing Information Base* (RIB). Ces routeurs sont appelés des collecteurs. Afin de faciliter, la récupération et l'analyse de ses données BML utilise le framework BGPstream (cf section 2.4.1). Cependant, BPG étant un protocole incrémental, seules des mises à jour de routes sont échangées par les routeurs. Cela amène donc à un problème de démarrage à froid dans la collecte de données car peu de données sont disponibles à $t_a^x + \epsilon$, le début de la période d'intérêt. Pour surmonter ce problème BML permet de définir une *priming period* ou période d'amorçage afin de collecter des données BGP avant la période d'intérêt :

$$Pm^x = [t_a^x - D_{pm}, ta^x] \tag{3.3}$$

où D_{pm} est la durée de la période d'amorçage fixée pour toutes périodes d'intérêt du jeu de données. La figure 3.2 illustre la collecte de données BGP pour une période d'intérêt. Durant la période d'amorçage, BML offre deux modes de fonctionnement : la collecte des messages UPDATE uniquement ou la collecte des *dumps* des RIBs les plus récentes qui sont ensuite augmentées avec les messages UPDATE reçus jusqu'à t_a^x. La 2ème option offre une vue plus complète des routes BGP mais augmente considérablement le temps d'exécution de la collecte des données. Après la collecte des données d'amorçage, les messages UPDATE capturés durant la période d'intérêt sont récupérés.

3.2.1 Stockage des données

Le réseau BGP connecte plus de 70 K AS et qui échangent des informations de routage pour plus de 890 K préfixes [103] (en septembre 2021). Cela implique donc d'importants

Algorithme 1 : Instantané des routes

Entrées : U une séquence de messages UPDATE, R un instantané des routes
Sorties : R instantané des routes mis à jour
début
  pour chaque *update* u *dans* U **faire**
    si u_{type} = *annonce* **alors**
      $R[u_{prefixe}, u_{collecteur}, u_{pair}] \longleftarrow u_{as\_path}$
    fin
    sinon si u_{type} = *retrait* **alors**
      $R[u_{prefixe}, u_{collecteur}, u_{pair}] \longleftarrow \emptyset$
    fin
  fin
fin

volumes de données de contrôle échangés par les routeurs BGP. Afin, d'éviter de re-collecter les données BGP entre deux itérations du processus de construction du jeu de données (voir figure 3.1) il est nécessaire de stocker efficacement ces données.

Période d'amorçage : La première solution pour la réduction de l'espace de stockage a été introduite sur les données d'amorçage. Ces données étant utilisées pour avoir un état initial des routes BGP, la dimension temporelle présente dans la séquence des messages UPDATE n'est pas nécessaire. Pour supprimer cette composante temporelle, un mécanisme d'agrégation des messages UPDATE a été proposé afin de conserver uniquement l'état initial des routes. Ce mécanisme est détaillé dans l'algorithme 1. À partir d'une séquence de messages UPDATE, cet algorithme produit un instantané des routes BGP. Un exemple d'instantané des routes est donné avec la figure 3.3. Si les RIBs ont également été collectés alors ils peuvent être utilisés pour construire un instantané initial qui est également fourni en entrée de l'algorithme et mis à jour avec les messages UPDATE. Durant la conception de BML, des expérimentations ont donc été menées afin de déterminer le gain potentiel de l'élimination de la composante temporelle dans les données d'amorçage. Ainsi, des messages UPDATE ont été collectés pendant 24 heures à partir de 26 collecteurs. L'espace requis pour stocker la séquence des messages UPDATE a été comparé à celui requis pour stocker l'instantané des routes construit à partir de cette séquence. La figure 3.4 présente les résultats en fonction de la durée de la période d'amorçage (de 1 à 24 heures). La figure montre aussi l'espace requis lorsqu'une compression sans perte (gzip) des données est utilisée. On peut voir que l'agrégation des données d'amorçage dans un seul instantané des routes permet de réduire la taille du jeu de données avec un facteur variant entre 60 pour 2 heures d'amorçage et 79 pour 24 heures. Si les données sont compressées, le ratio chute à 41 pour 2 heures d'amorçage et 52 pour 24 heures. Cela montre l'intérêt significatif d'agréger les données d'amorçage dans un instantané représentant les routes connues au temps t_a^x c'est-à-dire le début de la période d'intérêt. Cette option (avec compression) a donc été retenue pour le stockage des données d'amorçage.

```
{
"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",
     ...
  },
...
}
```


FIGURE 3.3 – Exemple d'instantané des routes qui fait correspondre chaque triplet $(prefixe, collecteur, pair)$ observé à un AS_PATH $\{AS_0, AS_1, ..., AS_j\}$. On peut voir que le collecteur "rrc19" dispose de quatre routes vers le préfixe "195.252.70.0/24" et qu'une de ces routes est annoncée par son pair "37468" avec l'AS_PATH "37468 5483 6700".

Période d'intérêt : Pour le stockage des données collectées durant la période d'intérêt, le mécanisme utilisé pour les données d'amorçage n'est pas directement applicable puisque la composante temporelle des données est indispensable. En effet, ces données seront échantillonnées avec un intervalle Δ lors de l'étape de transformation. Cependant, il est possible d'utiliser l'algorithme 1 pour générer un instantané des routes toutes les Δ minutes. On obtient alors une séquence d'instantanés des routes :

$$S^x = \{S_{t_a^x}, S_{t_a^x+\Delta}, ..., S_{t_b^x}\} \tag{3.4}$$

L'espace requis en utilisant cette approche a été comparé à celui requis pour stocker les séquences des messages UPDATE pour une période d'intérêt de 1 heure en utilisant 10 heures de données d'amorçage. La figure 3.5 montre cette comparaison en fonction de la valeur de Δ avec et sans compression des données. On observe que pour des valeurs de Δ inférieures à 7 minutes si les données sont compressées et 11 minutes dans le cas contraire, le stockage de la séquence des messages UPDATE est plus efficace. Ainsi, le choix a été fait de stocker la séquence (compressée) des messages UPDATE puisque cette solution est plus efficace sur des valeurs de Δ inférieurs à 11 minutes et qu'elle n'introduit pas de perte d'information. En effet, si l'utilisateur souhaite réduire la période d'échantillonnage Δ entre deux itérations du modèle, il ne sera pas nécessaire de re-collecter les données.

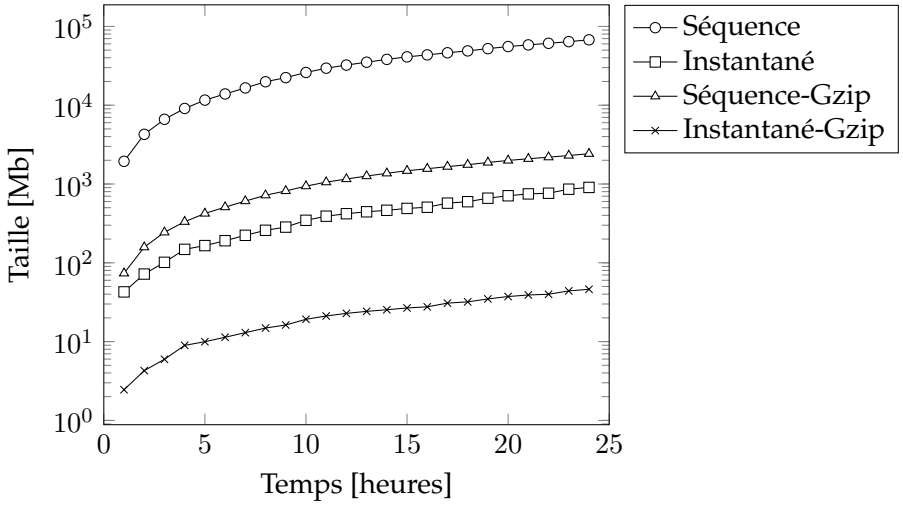


FIGURE 3.4 – Comparaison des options de stockage des données d'amorçage

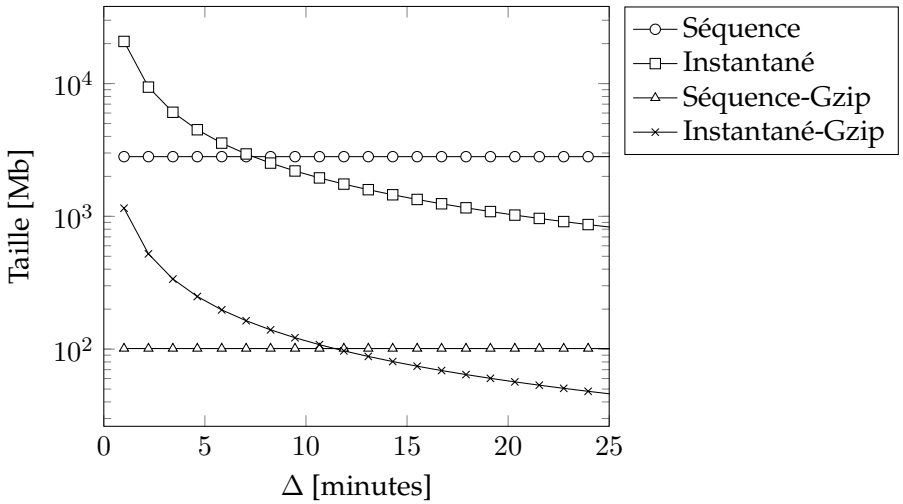


FIGURE 3.5 – Comparaison des options de stockage pour la période d'intérêt

3.2.2 Paramètres de collecte

La collecte des données BGP peut être ajustée à partir d'un ensemble de paramètres P qui doivent être fourni à BML. Parmi ces paramètres de collecte, on retrouve notamment :

- `Projects` : définit le ou les projets d'archivages à utiliser pour la récupération des données parmi Route Views et Ripe RIS
- `Collectors` : permet de filtrer les collecteurs à utiliser pour la transformation
- `IpVersion` : permet de filtrer les routes à utiliser pour la transformation en fonction de la version du protocole IP
- `PrimingPeriod` : définit la durée de la période d'amorçage
- `UseRibsPriming` : permet de choisir si les RIBs doivent également être utilisés pour la période d'amorçage
- `UseRibsData` : permet de choisir si les RIBs doivent également être utilisés pour la période d'intérêt

Les valeurs des paramètres de collecte peuvent être définies par l'utilisateur à deux niveaux : au niveau du jeu de données ou au niveau d'une période d'intérêt. La définition d'un paramètre au niveau du jeu de données sera utilisée pour la collecte de toutes les périodes d'intérêt. À l'inverse, la définition d'un paramètre au niveau des périodes d'intérêt permet de spécifier des valeurs différentes de ce paramètre pour la collecte de chacune des périodes d'intérêt.

3.2.3 Parallélisation

Pour chaque période d'intérêt $P^x = [t_a^x, t_b^x]$ du jeu de données, BML doit effectuer les étapes suivantes lors de la collecte des données :

1. télécharger les données de la période d'amorçage $[t_a^x - D_{pm}, t_a^x]$ à partir de plusieurs collecteurs et les ré-agencer temporellement à l'aide de bgpstream
2. calculer un instantané $S_{t_a^x}$ pour la période d'amorçage puis le compresser
3. télécharger les données de la période d'intérêt $[t_a^x, t_b^x]$ à partir de plusieurs collecteurs et les ré-agencer temporellement à l'aide de bgpstream puis les compresser

Afin d'accélérer l'étape de collecte de données, il est possible de paralléliser la construction d'un jeu de données $\{P^1, \ldots, P^x, \ldots, P^N\}$ en répartissant les périodes d'intérêt sur plusieurs processus ou sur plusieurs machines. Pour cela, BML décompose la construction d'un jeu de données un ensemble de tâches unitaires appelées *jobs*. Ces *jobs* peuvent être sauvegardés au format JSON (voir figure 3.6) pour être exécutés ultérieurement ou répartis sur plusieurs machines. Lors de l'exécution, BML permet de définir le nombre de processus à utiliser pour l'exécution en parallèle des *jobs*.

3.2.4 Estimation de la durée de collection et de l'espace de stockage

Lors de la planification de la collecte d'un jeu de données BGP, il est utile d'estimer à l'avance l'espace de stockage ainsi que le temps nécessaire pour collecter les données. Grâce à BML, le stockage nécessaire peut être estimé en fonction de la durée de la période

```
[
  {
    'includes': 'from BML.data.dataset import processSample',
    'target': 'processSample',
    'args': ('anomaly', 1434094980, 1434102180, 'TM', ... ),
    ...
  },
  {
    'includes': 'from BML.data.dataset import processSample',
    'target': 'processSample',
    'args': ('anomaly', 1396459500, 1396466700, 'IndoSat', ... ),
    ...
  },
  ...
]
```

FIGURE 3.6 – Exemple de décomposition de la construction d'un jeu de données avec BML en un ensemble de tâches unitaires appelées *jobs*. Une tâche est définie entre autres par "target" qui est la fonction à exécuter et "args" les arguments à passer à cette fonction.

d'amorçage D_{pm}, la durée de la période d'intérêt D et le nombre de périodes d'intérêts N :

$$S_{dataset} = (S_{routes}(D_{pm}) + S_{updates}(D)) \times N \tag{3.5}$$

$S_{routes}(D_{pm})$ est la taille moyenne d'un instantané compressé lorsqu'une période d'amorçage D_{pm} est utilisée et $S_{updates}(D)$ est la taille moyenne d'un ensemble compressé de messages UPDATE collectée pendant un temps D. Les valeurs de $S_{routes}(D_{pm})$ et $S_{updates}(D)$ doivent être déterminées lors d'expériences préliminaires ou approximées en utilisant le résultat des expériences décrites dans la section 3.2.1 (figure 3.4). Sur la base de ces expériences, on observe que même avec 5 000 périodes d'intérêt de 2 heures et 10 heures de période d'amorçage, l'estimation de l'espace de stockage est d'environ 900 Go, ce qui est acceptable. De plus, la figure 3.4 montre que $S_{routes}(t)$ est beaucoup plus petit que $S_{updates}(t)$ ce qui implique que la durée de la période d'amorçage D_{pm} a beaucoup moins d'impact sur la taille de l'ensemble de données que la durée de la période d'intérêt D.

Le temps nécessaire à la collecte des données peut être estimé en fonction de la durée de la période d'amorçage D_{pm}, la durée de la période d'intérêt D et le nombre de périodes d'intérêts N :

$$T_{dataset} = (D + D_{pm}) \times T_h \times N \tag{3.6}$$

T_h est le temps moyen requis pour la collecte de 1 heure de données et doit être déterminé lors des expériences préliminaires. En effet, cette valeur dépend fortement de l'environnement d'exécution tel que la bande passante du réseau. De plus, comme cet

outil permet de paralléliser et de distribuer la collecte d'un jeu de données sur plusieurs machines, ce temps moyen de collecte peut être considérablement réduit.

En utilisant les données des expériences décrites dans la section 3.2.1 il a été mesuré $T_h = 11.5$ minutes. Ces expériences ont également été renouvelés mais en utilisant une machine multi-coeur avec 16 processus en parallèle et il a été mesuré $T_h = 1,4$ minutes dans ce contexte.

Ainsi, la collecte de données avec 1 000 périodes d'intérêt de 2 heures et 10 heures de période d'amorçage durerait approximativement 87 jours en utilisant 1 processus. En utilisant 16 processus en parallèle, la même collecte de données durerait approximativement 10 jours. Il s'agit d'un facteur de réduction de 8.7 même si 16 processus sont utilisés, ce qui est dû au fait que des goulots d'étranglement tels que la bande passante du réseau et l'accès au disque apparaissent.

3.3 Transformation des données

Une fois les données collectées pour toutes les périodes d'intérêt du jeu de données, BML peut être utilisé pour transformer ces données dans un format spécifié par l'utilisateur. Pour cela, l'utilisateur devra choisir une fonction de transformation déjà existante dans BML ou implémenter sa propre fonction de transformation. Cette fonction de transformation $T(.)$ afin de transformer chaque période d'intérêt en une séquence :

$$T^x = \{T_{t_a^x}, T_{t_a^x+\Delta}, \ldots, T_{t_b^x}\} \tag{3.7}$$

Cette section définit de façon générique le type de fonction de transformation attendu par BML et donne quelques exemples de fonction de transformation déjà implémentés.

3.3.1 Définition générique

Dans BML une fonction de transformation est définie par :

$$T_t = T(S_t, U_{[t-\Delta,t]}, P) \tag{3.8}$$

où $t \in [t_a^x, t_a^x + \Delta, \ldots, t_b^x]$, S_t est l'instantané des routes calculées au temps t et $U_{[t-\Delta,t]}$ est la séquence des messages UPDATE collectée entre $[t - \Delta, t]$ et P un ensemble de paramètres de transformation. S_t donne un état des routes au temps t tandis que $U_{[t-\Delta,t]}$ permet d'observer l'évolution des routes entre $t - \Delta$ et t. Un cas particulier est $t = t_a^x$ puisque $S_{t_a^x}$ est l'instantané calculé à partir des données d'amorçage et $U_{[t_a^x-\Delta,t_a^x]} = \emptyset$.

Toutes fonctions de transformations dans BML doit pendre en entrée à la fois S_t et $U_{[t-\Delta,t]}$ même si une seule de ces entrées est utilisée. Pour accélérer l'exécution il est possible d'empêcher le calcul de S_t dans ce cas on obtiendra $S_t = S_{t_a^x}$ quelque soit t. Enfin pour économiser en mémoire, il est également possible d'éviter le chargement de $S_{t_a^x}$ et dans ce cas on obtiendra $S_t = \emptyset$ quelque soit t.

La classe `BaseTransform` implémente cette définition générique d'une fonction de transformation dans BML. Ainsi, toute implémentation d'une fonction de transformation dans BML doit hériter de cette classe (voir la figure 3.7) ou d'un de ces descendants et redéfinir la méthode :

```
transforms(self, index, routes, updates)
```

Cette méthode doit produire en sortie un objet sérialisable en JSON. De plus les méthodes :

```
preProcess(self)
postProcess(self, transformedData)
```

peuvent être redéfinies pour appliquer un pré-traitement ou post-traitement.

```python
from BML.transform import BaseTransform

class MyTransform(BaseTransform):

    def transforms(self, index, routes, updates):
        ...
        return(data)

    def preProcess(self):
        ...

    def postProcess(self, transformedData):
        ...
        return(transformedData)
```

FIGURE 3.7 – Squelette d'implémentation d'une fonction de transformation dans BML

3.3.2 Paramètres de transformation

Lors de la transformation d'un jeu de données, un ensemble de paramètres P doit être fourni à BML. Ces paramètres sont composés de paramètres par défauts tandis que d'autres sont spécifiques à certaines fonctions de transformation. Lors de l'implémentation d'une fonction de transformation, il est donc possible de définir de nouveaux paramètres. Parmi les paramètres par défaut, on retrouve notamment :

— `Period` : la valeur de l'intervalle d'échantillonnage Δ
— `Collectors` : permet de filtrer les collecteurs à utiliser pour la transformation
— `IpVersion` : permet de filtrer les routes à utiliser pour la transformation en fonction de la version du protocole IP

Comme pour la collecte des données, les valeurs des paramètres de transformations peuvent être définies par l'utilisateur à deux niveaux : au niveau du jeu de données ou au niveau d'une période d'intérêt. Un paramètre défini au niveau du jeu de données sera utilisé pour la transformation de toutes les périodes d'intérêt alors qu'un paramètre défini au niveau des périodes d'intérêt permet de spécifier des valeurs différentes de ce paramètre pour la transformation de chacune des périodes d'intérêt.

3.3.3 Parallélisation

Comme pour la collecte des données, il est possible de réduire le temps de transformation d'un jeu de données en parallélisation certaines exécutions. BML offre deux options à l'utilisateur pour paralléliser la transformation de son jeu de données.

Premièrement, comme pour la collecte des données, il est possible de paralléliser la transformation d'un jeu de données $\{P^1, \ldots, P^x, \ldots, P^N\}$ en répartissant les transformations des périodes d'intérêt sur plusieurs processus ou sur plusieurs machines. Là encore, BML décompose la transformation du jeu de données en un ensemble *jobs* (voir figure 3.6) qui peuvent être sauvegardées au format JSON pour être exécutés ultérieurement ou répartis sur plusieurs machines.

Afin, d'exploiter encore plus efficacement la puissance de calcul disponible, BML permet également de paralléliser la transformation d'une période d'intérêt. Ainsi, le calcul des transformations $\{T_{t_a^x}, T_{t_a^x+\Delta}, \ldots, T_{t_b^x}\}$ peut s'effectuer en parallèle. Pour cela, BML calcule tout d'abord l'ensemble des instantanés $\{S_{t_a^x}, S_{t_a^x+\Delta}, \ldots, S_{t_b^x}\}$ et des séquences de messages UPDATE $\{U_{[t_a^x-\Delta,t_a^x]}, U_{[t_a^x,t_a^x+\Delta]} \ldots, U_{[t_b^x-\Delta,t_b^x]}\}$. Ce calcul n'étant pas parallélisé. Puis, le calcul des transformations $\{T_{t_a^x}, T_{t_a^x+\Delta}, \ldots, T_{t_b^x}\}$ est effectué en parallèle. Ainsi, cette approche permet de réduction significative des temps de calcul lorsque de la fonction de transformation $T(S_t, U_{[t-\Delta,t]}, P)$ prend un temps non négligeable par rapport au temps de calcul de S_t et $U_{[t-\Delta,t]}$. Ce mécanisme de parallélisation est implémenté dans la classe `BaseTransformParallelized`. Ainsi, une fonction de transformation peut être rendue parallélisable simplement en héritant de la classe `BaseTransformParallelized` à la place de la classe `BaseTransform`.

3.3.4 Transformations implémentées

Une des caractéristiques principales de BMl est sa flexibilité par rapport aux fonctions de transformation qu'il est possible d'implémenter. Cependant, afin de faire économiser du temps aux utilisateurs et d'encourager la reproductibilité des résultats, BML propose par défaut plusieurs fonctions de transformations. Ces dernières peuvent être directement utilisées ou être la base de l'implémentation de nouvelles fonctions de transformation. Cette section présente certaine de ces fonctions de transformation.

Extraction d'attributs statistiques

Dans la littérature, l'extraction d'attributs statistiques a été largement adoptée pour la construction de jeu de données BGP pour l'apprentissage automatique (voir la section 2.4.3). Ainsi, BML implémente une fonction de transformation permettant d'extraire les attributs statistiques les plus utilisés dans la littérature. Au total, 32 attributs statistiques sont extraits par BML :

- `nb_A` : nombre d'annonces de route
- `nb_W` : nombre de retraits de route
- `nb_dup_A` : nombre d'annonces dupliquées. Une annonce est dupliquée si un pair (c.-à-d. un routeur voisin) annonce une route qui est déjà connue pour ce pair avec le même AS_PATH
- `nb_dup_W` : nombre de retraits dupliqués. Un retrait est dupliqué si un pair retire une route qu'il a déjà retirée
- `nb_implicit_W` : nombre de retrait implicite de route. Un retrait est implicite si un pair annonce une route qui est déjà connue pour ce pair avec un AS_PATH différent. L'ancienne route devient donc implicitement invalide.
- `nb_A_prefix` : nombre de préfixes dont une route a été annoncée au moins une fois
- `nb_W_prefix` : nombre de préfixes dont une route a été retirée au moins une fois
- `max_A_prefix` : nombre maximum de routes annoncées pour un préfixe

— `avg_A_prefix` : nombre moyen de routes annoncées pour un préfixe
— `max_A_AS` : nombre maximum de routes annoncées par un AS
— `avg_A_AS` : nombre moyen de routes annoncées par un AS
— `nb_orign_change` : nombre de changement d'AS d'origine observés
— `nb_new_A` : nombre de routes annoncées pour la première fois
— `nb_new_A_afterW` : nombre de routes annoncées après avoir été retirées
— `avg_interarrival` : temps moyen entre les messages UPDATE
— `max_path_len` : longueur maximum d'un AS_PATH
— `avg_path_len` : longueur moyenne d'un AS_PATH
— `max_editdist` : distance d'édition maximum d'un AS_PATH
— `avg_editdist` : distance d'édition moyenne d'un AS_PATH
— `editdist_k` : nombre d'occurrences d'une distance d'édition égale à k avec $k \in [7, 17]$ (11 attributs)
— `nb_tolonger` : nombre de changements d'AS_PATH vers un AS_PATH plus court
— `nb_toshorter` : nombre de changements d'AS_PATH vers un AS_PATH plus long

Extraction du graphe BGP

Le réseau BGP peut être représenté sous la forme d'un graphe où les AS sont les noeuds du graphe et les liens représentent les relations entres les AS (voir la section 2.2.3). Cette thèse s'intéresse particulièrement à ce type de représentation du réseau BGP. Ainsi, dans cette section, deux algorithmes sont proposés pour l'extraction du graphe BGP et implémentés dans BML.

L'algorithme 2 permet l'extraction du graphe BGP tel qu'il a été défini dans la section 2.2.3 c'est-à-dire un graphe non orienté qui représente les relations entre les AS. L'algorithme prend en entrée un instantané R des routes BGP et produit en sortie le graphe BGP correspondant G. On notera que dans cet algorithme l'ajout d'un noeud ou d'un lien au graphe est ignoré si le noeud ou le lien est déjà présent dans le graphe.

Algorithme 2 : Extraction du graphe BGP

Entrées : R un instantané des routes
Sorties : G un graphe BGP
début
  $G \longleftarrow$ un graphe vide
  pour chaque *AS_PATH* a *dans* R **faire**
    pour chaque *paire d'AS* (u, v) *consécutifs dans* a **faire**
      ajouter le noeud u a G
      ajouter le noeud v a G
      ajouter le lien (u, v) a G
    fin
  fin
fin

Le graphe extrait par l'algorithme 3 est une extension de celui extrait par l'algorithme précédent permettant l'extraction d'un graphe pondéré. Chaque, noeud et lien du graphe possèdent donc un poids w qui lui est propre. Ce poids est calculé à partir d'un ensemble de préfixes P tel que :

$$w = \sum_{p \in P} W(p) \tag{3.9}$$

où $W(p)$ est une fonction calculant le poids d'un préfixe p. Pour un noeud l'ensemble des préfixes P utilisés pour le calcul de son poids w est l'ensemble des préfixes dont il est l'origine dans au moins une route. Pour un lien, cet ensemble est l'ensemble des préfixes pour lesquels au moins une route passe par ce lien. Ainsi, la fonction $W(p) = 1$ permet de calculer pour chaque noeud le nombre de préfixes dont il est l'origine et pour chaque lien le nombre de préfixes transportés sur ce lien. Cependant, il peut être intéressant d'associer des poids différents aux préfixes. Par exemple, $W(p)$ peut être définie comme le nombre d'adresses IP couvertes par le préfixe p. Ainsi, un lien transportant 2 préfixes "/24" aurait le même poids qu'un lien transportant un seul préfixe "/23".

Algorithme 3 : Extraction du graphe BGP pondéré

Entrées : R un instantané des routes
Sorties : G un graphe BGP
début
  $G \longleftarrow$ un graphe vide
  pour chaque *préfixe p dans* R **faire**
    $A \longleftarrow$ l'ensemble des AS_PATHs pour le préfixe p
    $w \longleftarrow$ le poids W(p) du préfixe p
    pour chaque *paire d'AS* (u, v) *consécutifs dans* A *et sans doublons* **faire**
      ajouter le noeud u a G
      ajouter le noeud v a G
      ajouter le lien (u, v) a G
      mettre à jour le poids du lien (u, v) en lui ajoutant w
    fin
    pour chaque *pour chaque AS d'origine o dans* A *et sans doublons* **faire**
      mettre à jour le poids du noeud o en lui ajoutant w
    fin
  fin
fin

Extraction d'attributs à partir d'un graphe

Le graphe BGP est une représentation riche qui est néanmoins complexe à analyser. Afin de réduire cette complexité tout en tirant profit de la représentation sous forme de graphe des données BGP, il est possible de calculer un ensemble de métriques à partir du graphe BGP. Ces métriques issues de la théorie des graphes [104]-[106] peuvent ainsi être utilisées pour calculer des attributs. En fonction des métriques utilisées, les attributs obtenus peuvent être de deux types : ceux pour lesquels une valeur est calculée pour chaque noeud du graphe et ceux pour lesquels cette valeur est définie pour tout le graphe.

Au total, BML permet l'extraction de 31 attributs d'un graphe à partir d'un graphe BGP construit avec l'algorithme 2.

Attributs au niveau des noeuds :

— `degree` : le degré $deg(v)$ d'un noeud v est le nombre liens adjacents a ce noeud

— `degree_centrality` : la centralité de degré d'un noeud v est la fraction des noeuds qu'il a pour voisin :

$$\frac{deg(v)}{|V|-1}$$

— `betweenness` : la centralité intermédiaire ou *betweenness centrality* est le nombre de fois qu'un noeud v est sur le chemin le plus court entre deux autres noeuds :

$$\sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

où $\sigma(s,t)$ est le nombre du plus court chemin de s à t et $\sigma(s,t|v)$ est le nombre de ces chemins passant par v.

— `closeness` : la centralité de proximité ou *closeness centrality* est l'inverse de la somme de la longueur des chemins les plus courts entre un noeud v et tous les autres noeuds du graphe :

$$\frac{|V|-1}{\sum_{u \in V} d(v,u)}$$

où $d(v,u)$ est la longueur du chemin le plus court entre v et u.

— `pagerank` : le pagerank est un algorithme utilisé par le moteur de recherche Google afin de mesurer la popularité d'une page Web *c-à-d* un noeud dans le graphe du Web. Cette valeur est proportionnelle au nombre de fois qu'un utilisateur passerait par cette page en parcourant aléatoirement le graphe du Web.

— `eigenvector` : centralité de vecteur propre d'un noeud v ou *eigenvector centrality* est l'élément x_v du vecteur x définie par :

$$Ax = \lambda x$$

où A est la matrice d'adjacence du graphe et λ est la plus grande valeur propre de A.

— `load` : la *load centrality* est une variante de la centralité intermédiaire introduite dans [107].

— `harmonic` : la centralité harmonique ou *harmonic centrality* est la somme des inverses des chemins les plus courts entre un noeud v et tous les autres noeuds du graphe :

$$\sum_{u \in V} \frac{1}{d(v,u)}$$

— `node_clique_number` : pour un noeud v une clique maximum est le plus grand sous-graphe complet contenant v. Le nombre de cliques du noeud v ou *node clique number* est la taille de la clique maximum contenant v c'est-à-dire son nombre de

noeuds.

— `number_of_cliques` : pour un noeud v plusieurs cliques maximum contenant ce noeud peuvent exister. Le nombre de cliques est donc le nombre de cliques maximum contenant le noeud v.

— `triangles` : cet attribut calcule le nombre de triangles contenant le noeud v :

$$T(v)$$

— `clustering` : le coefficient d'agglomération ou *clustering coefficient* d'un noeud v quantifie à quel point le voisinage du noeud est proche de former un graphe complet. Ainsi, il est défini comme la proportion des triangles existant parmi tous les triangles possibles :

$$\frac{2T(v)}{deg(v)(deg(v)-1)}$$

— `square_clustering` : le *square clustering* d'un noeud v est la proportion des carrés existants parmi tous les carrés possibles et donne la probabilité que deux voisins de v partagent un voisin commun autre que v.

— `eccentricity` : l'excentricité d'un noeud v est le plus long des plus courts chemins qui vont de ce noeud à tous les autres noeuds du graphe :

$$\max_{u \in V} d(v,u)$$

— `average_shortest_path_length` : cet attribut indique la taille moyenne des plus courts chemins qui vont d'un noeud v à tous les autres noeuds du graphe :

$$\frac{1}{|V|-1} \sum_{u \in V} d(v,u)$$

où $\sigma(v,u)$ est le nombre de plus courts chemins de v à u.

— `local_efficiency` : l'*efficiency* d'une paire de noeuds dans un graphe est l'inverse de la longueur du plus chemin entre ces noeuds. La *local efficiency* d'un noeud v est calculée à partir de G' le sous-graphe induit par l'ensemble des voisins de v. Ainsi, *local efficiency* du noeud v est la moyenne des *efficiency* de toutes les paires de noeuds du graphe G' :

$$\frac{1}{|V|(|V|-1)} \sum_{u,w \in V} \frac{1}{d(u,w)}$$

— `average_neighbor_degree` : l'*average neighborhood degree* d'un noeud v est la moyenne des degrés des voisins de ce noeud :

$$\frac{1}{|N(v)|} \sum_{u \in N(v)} deg(u)$$

où $N(v)$ est l'ensemble de voisins du noeud v.

Attributs au niveau du graphe :

— `nb_of_nodes` : cet attribut indique le nombre de noeuds $|V|$ du graphe

— `nb_of_edges` : cet attribut indique de nombre de liens $|E|$ du graphe

— `diameter` : le diamètre d'un graphe est l'excentricité maximale des noeuds du graphe. En d'autres termes, il s'agit du plus long des plus courts chemins du graphe.

$$max_{u,v \in V} d(u, v)$$

— `assortativity` : l'assortativité mesure la tendance des noeuds à se connecter à des noeuds de degré similaire. Cette mesure est effectuée en calculant la corrélation de Pearson entre le degré des noeuds adjacents :

$$\frac{cov(X, Y)}{\sigma_X \sigma_Y}$$

où $X = deg(U)$, $Y = deg(V)$ et $(U_i, V_i) \in E$.

— `largest_eigenvalue` : cette valeur également appelée le rayon spectral est la plus grande valeur propre λ_n de la matrice d'adjacence du graphe :

$$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$$

— `algebraic_connectivity` : la connectivité algébrique λ_2 est la deuxième plus petite valeur propre de la matrice laplacienne du graphe.

— `effective_graph_resistance` : pour le calcul de l'*effective graph resistance*, le graphe est vu comme un circuit électrique ou un lien (u, v) correspondent à une résistance $r_{uv} = 1$ Ohm. L'*effective graph resistance* est la somme des résistances entre toutes les paires de noeuds du graphe. Cette valeur peut être calculée à partir du spectre de la matrice laplacienne du graphe :

$$n \sum_{i=2}^{n} \frac{1}{\lambda_i}$$

— `symmetry_ratio` : le *symmetry ratio* est une mesure de robustesse qui est définie comme :

$$\frac{\epsilon}{D + 1}$$

où ϵ est le nombre de valeurs propres distinctes de la matrice d'adjacence et D est le diamètre du graphe.

— `natural_connectivity` : la *natural connectivity* est une mesure qui évalue la robustesse basée sur la somme des parcours fermés. Un parcours fermé de taille k étant un chemin de longueur k dont les extrémités sont un même point. La *natural connectivity* peut être calculée comme une moyenne du spectre de la matrice d'adjacence :

$$ln \left(\frac{\sum_{i=1}^{n} e^{\lambda_i}}{n} \right)$$

— `node_connectivity` : la *node connectivity* correspond au nombre minimum de noeuds qui doivent être supprimés afin de déconnecter un graphe :

$$\min_{v,w \in V} \kappa(v, w)$$

où $\kappa(v, w)$ est le nombre minimum de noeuds appartenant à $V - \{v, w\}$ dont la suppression détruirait tous les chemins entre v et w.

— `edge_connectivity` : l'*egde connectivity* correspond au nombre minimum de liens qui doivent être supprimés afin de déconnecter un graphe :

$$\min_{v,w \in V} \lambda(v, w)$$

où $\lambda(v, w)$ est le nombre minimum de liens appartenant à E dont la suppression détruirait tous les chemins entre v et w.

— `weighted_spectrum_3` : le *weighted spectrum* est une mesure de différentes propriétés topologiques d'un graphe en fonction d'un paramètre N. Sa valeur est liée aux N-cycles du graphe. Un N-cycle d'un graphe étant un sous-graphe formant une chaîne fermée de taille n. Par exemple, le nombre de 3-cycles correspond au nombre de triangles du graphe. Le *weighted spectrum* est calculé à partir du spectre de la matrice laplacienne du graphe :

$$\sum_{i=1}^{n} (1 - \lambda_i)^N$$

Pour cet attribut N est fixé à 3 est la valeur du *weighted spectrum* est alors liée au coefficient d'agglomération ou *clustering coefficient*.

— `weighted_spectrum_4` : Cet attribut est le *weighted spectrum* pour $N = 4$, la valeur est alors liée au nombre de chemins disjoints du graphe.

— `percolation_limit` : le seuil de percolation correspond au ratio des noeuds qui doivent être supprimés aléatoirement pour désintégrer un graphe en sous-graphes déconnectés. Cette valeur peut être calculée à partir des degrés de noeuds $\delta_1, \delta_2, \ldots, \delta_n$:

$$1 - \frac{1}{\frac{\langle \kappa_0^2 \rangle}{\langle \kappa_0 \rangle} - 1}$$

où :

$$\langle \kappa_0 \rangle = \frac{\delta_1 + \delta_2 + \cdots + \delta_n}{n}$$

$$\langle \kappa_0^2 \rangle = \frac{\delta_1^2 + \delta_2^2 + \cdots + \delta_n^2}{n}$$

— `nb_spanning_trees` : un arbre couvrant ou *spanning tree* d'un graphe est un sous-graphe contenant tous les n noeuds du graphe, $n - 1$ liens et aucune boucle. Pour un même graphe, plusieurs arbres couvrants peuvent exister. Le nombre d'arbres couvrant d'un graphe peut être calculé à partir du spectre de la matrice

laplacienne tel que :

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

3.4 Cas d'applications

Dans cette section, deux cas d'applications concrets de BML sont présentés. Tout d'abord, afin de mettre l'accent sur la polyvalence de cet outil, son utilisation dans le cadre d'une l'étude d'une anomalie BGP est présentée. Le second cas d'application est la construction d'un jeu données de grande taille collecté dans le cadre de cette thèse.

3.4.1 Étude d'une anomalie BGP

Cette section illustre comment BML peut être utile pour l'analyse rapide d'évènements liés à BGP. Pour cela, elle propose d'étudier la fuite de routes provoquée par Google le 25 août 2017[1]. À 3 heures 22 UTC, l'AS 15169 appartenant à Google a accidentellement commencé à annoncer plusieurs routes vers des préfixes qu'il ne possède pas et est ainsi devenu un fournisseur de transit. Cette erreur a causé des perturbations importantes sur Internet et principalement au Japon.

Extraction d'attributs statistiques

Afin d'analyser cet évènement, BML a été configuré pour collecter des données BGP pour une période d'intérêt de 3 heures à 4 heures UTC. Une période d'amorçage de 10 heures a été utilisée en utilisant uniquement le collecteur `rrc06` du projet Ripe RIS puisqu'il est situé au Japon qui est le pays qui a été le plus impacté. Afin d'accélérer la collecte et la transformation des données, les RIBs n'ont pas été utilisés.

Tout d'abord, 32 attributs statistiques implémentés dans BML ont été extrait toutes les 2 minutes. L'ensemble du code pour la collecte et la transformation du jeu de données ne prend que 20 lignes environ comme le montre la figure 3.9.

Pour cette expérimentation, une machine disposant d'un processeur de 3.7 GHz/16 coeurs avec 64 Go de RAM sous Ubuntu 18.04 a été utilisé. La collection des données a duré 4 minutes et 21 secondes tandis que l'extraction des attributs statistiques a pris 9 secondes.

La figure 3.8 montre un graphique des 32 attributs statistiques où les valeurs ont été redimensionnées à l'aide d'une normalisation *min-max* :

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{3.10}$$

Parmi les attributs extraits, `nb_A` et `nb_W` sont respectivement le nombre d'annonces de routes et de retraits reçus pendant l'intervalle Δ. Entre 3 heures 22 et 3 heures 27, de nombreuses annonces sont reçues en raison de la fuite de routes. Puis, entre 3 heures 31 et 3 heures 36, une deuxième vague de retraits et d'annonces est arrivée en raison de la correction de l'erreur par Google. Ainsi, ces résultats montrent clairement que la surveillance des attributs statistiques est pertinente pour la détection d'anomalies BGP.

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

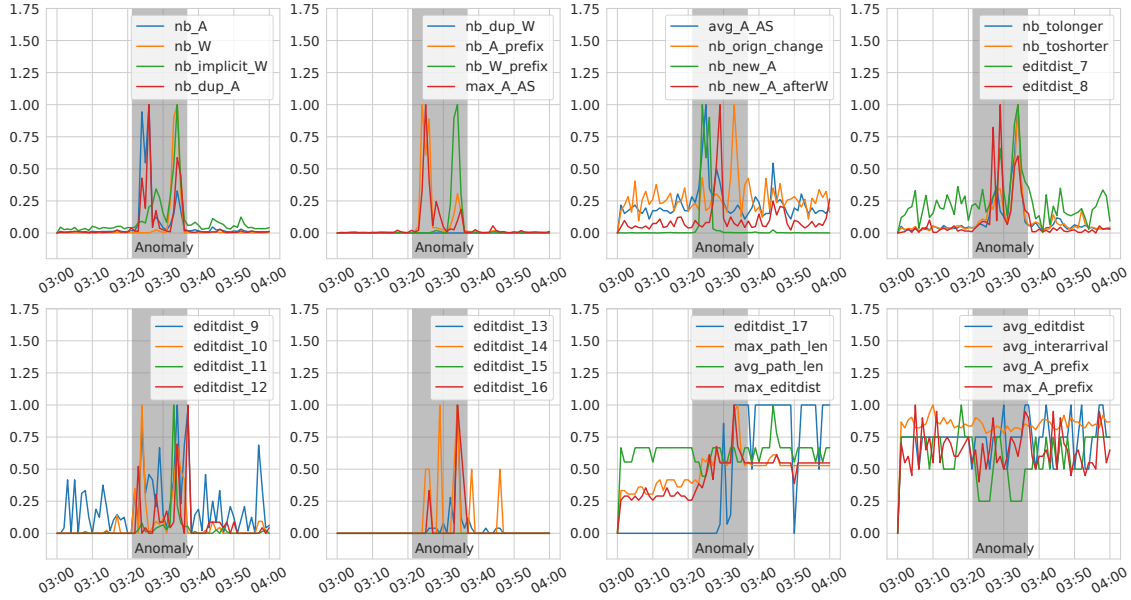


FIGURE 3.8 – Attributs statistiques durant la fuite de routes de Google

Étude du graphe BGP

BML implémente également des transformations qui permettent de reconstruire le graphe BGP et de calculer un ensemble d'attributs à partir de ce graphe. Dans cette expérimentation, 14 attributs ont été extraits à partir du graphe BGP. La transformation des données a duré 7 minutes et 47 secondes. La figure 3.10 montre un graphique de ces 14 attributs redimensionnés à l'aide d'une normalisation *min-max*. On peut voir que cette anomalie a également impacté de manière significative le graphe BGP.

Définition d'une nouvelle fonction de transformation

Un avantage majeur de BML est la possibilité pour un utilisateur de définir facilement sa propre fonction de transformation des données BGP. Pour illustrer cette fonctionnalité, l'hypothèse est faite que l'utilisateur souhaite visualiser le nombre d'annonces dans lesquelles l'AS 15169 appartenant à Google se trouve dans l'AS_PATH. Comme le montre la figure 3.12, cette transformation peut être implémentée en seulement 10 lignes de code avec BML. Le calcul de cet attribut n'a pris que 7 secondes pour s'exécuter. La figure 3.11 montre un graphique de cet attribut et un motif similaire à celui des caractéristiques statistiques est visible.

3.4.2 Collecte d'un jeu de données de grande taille

Un des facteurs limitant l'application des techniques d'apprentissage automatique à la détection des anomalies BGP est le manque de jeu de données (cf section 2.4.3). En effet, il n'existe actuellement aucun jeu de données de référence pour l'étude de ces anomalies. Cette thèse souhaite adresser cette problématique en proposant à la communauté un jeu de données de grande taille contenant des anomalies BGP récentes. Un tel jeu de

Défaillance matérielle	Temps de début de l'anomalie
	Temps de fin de l'anomalie
	AS concerné
	Nombre de préfixes impactés
Fausse origine	Temps de début de l'anomalie
	AS victime
	AS attaquant
	Préfixe de la victime
	Préfixe annoncé
	Nombre de routeurs ayant observé l'anomalie
Fuite de routes	Temps de début de l'anomalie
	AS d'origine
	AS responsable de la fuite
	Préfixe
	Nombre de routeurs ayant observé l'anomalie

TABLE 3.1 – Description des anomalies issues de *bgpstream*

données permettrait d'une part l'entraînement de modèles plus complexes qui nécessite un volume de données important. Et d'autre part, il encouragerait la reproductibilité des résultats et offrirait un socle commun pour la comparaison des différents travaux dans la thématique.

Obtention et sélection des anomalies BGP

Pour la construction d'un jeu de données d'anomalies BGP, il est essentiel d'avoir des informations précises sur des anomalies passées. Ces informations peuvent être utilisées pour définir un ensemble de périodes d'intérêt qu'il est possible d'inclure dans le jeu de données. Cependant, il n'y a pas de consensus autour d'un ensemble d'anomalies BGP. Au lieu de cela, certaines anomalies BGP à grande échelle sont parfois signalées dans des articles de blog qui nécessitent du travail humain pour extraire des informations afin de créer un ensemble de données. Pour surmonter ce problème, la solution adoptée consiste à s'appuyer sur le système de détection d'anomalies BGP *bgpstream* [108] qui est connu pour l'envoi d'alerte sur un compte Twitter. Ce travail s'appuis sur l'outil *bgpstream* car il fournit un grand nombre d'anomalies BGP récentes pouvant facilement être récupérées sous forme structuré grâce à des techniques de *web scraping*.

En s'appuyant sur les données disponibles sur le site web de *bgpstream* il a été possible de récupérer 4438 anomalies étant survenues entre le 22 mars 2021 et le 8 octobre 2021. Ces anomalies sont classées par *bgpstream* dans trois catégories qui dans la taxonomie proposée (cf figure 2.3) correspondent à : défaillance matérielle (2877 anomalies), détournement de préfixe par fausse origine (1112 anomalies) et détournement de préfixe par fuite de routes (449 anomalies). Il est à noter que ces anomalies comportent des doublons puisqu'un même évènement peut être reporté plusieurs fois par *bgpstream* si, par exemple, il implique plusieurs préfixes. Le tableau 3.1 donne les informations qui ont été récupérées en fonction du type d'anomalie.

Pour la construction d'un premier jeu de données, il a été choisit de construire un jeu de données balancé, c'est-à-dire contenant le même nombre d'anomalies par types. Il a été déterminé que 150 évènements permettait d'assurer cette balance. L'obtention de cette valeur est présentée dans la section suivante. Pour sélectionner ces 150 anomalies, uniquement les anomalies sur le réseau IPv4 sont tout d'abord conservés . Ensuite, sont regroupés les évènements correspondants à une même anomalie. Par exemple pour les fausses origines, si un AS détourne plusieurs préfixes appartenant à un ou plusieurs AS alors autant d'évènements seront détectés par *bgpstream*. Ainsi, sont regroupé toutes les anomalies ayant le même AS attaquant dans une fenêtre d'une heure. Cette opération réduit le nombre de fausse origine de 1075 à 642. Une nouvelle étape de filtrage est effectuée afin de garantir le non-chevauchement des anomalies et pour cela uniquement les anomalies dont les temps de début sont espacés d'au moins 5h30 sont conservées. Enfin, afin de sélectionner les 150 anomalies, le nombre de routeurs ayant observé l'anomalie a été choisi comme critère et permettant ainsi de conserver les 150 anomalies qui se sont le plus propagées.

Sélection des périodes sans-anomalies

Afin d'entraîner un modèle d'apprentissage automatique pour la détection des anomalies BGP, il est utile d'inclure dans le jeu de données à la fois des exemples d'anomalies, mais également des exemples de données BGP capturées lors de son fonctionnement normal. Ainsi, il est nécessaire de sélectionner des périodes sans-anomalies à inclure dans le jeu de données. Une approche simpliste consiste à sélectionner ces périodes aléatoirement. Cependant, cette approche n'est pas adaptée car elle ne garantit pas l'absence d'anomalie à l'instant sélectionné. Afin de d'éviter de cette situation, une méthodologie a été définie afin de sélectionner des périodes sans-anomalies uniquement dans les espaces entre les anomalies connues. Pour déterminer ces espaces, les 4438 anomalies survenues entre le 22 mars 2021 et le 8 octobre 2021 ont été utilisés. Dans chacun de ces espaces, le nombre de périodes pouvant être sélectionnées a été limité à 4 afin d'éviter qu'un ratio trop important des périodes sans-anomalies soit extrait dans un seul espace. En effet, cela pourrait introduire un biais dans les données qu'il est souhaitable d'éviter. De plus, une période peut être sélectionnée uniquement si elle commence au minimum 3 heures après une anomalie connue et si elle se termine au minimum 1 heure avant une anomalie connue. En appliquant ces règles, il a été possible d'extraire 150 périodes sans-anomalies.

Collecte des données

Pour les 600 exemples du jeu de données, l'évolution de BGP a été capturée durant 2 heures. Dans le cas des anomalies cette fenêtre est centrée sur le début de l'anomalie avec donc 1 heure de données avant et après l'apparition de l'anomalie. Pour la collecte des données BGP avec BML, les collecteurs `rrc04` et `rrc05` du projet Ripe Ris [41] ont été utilisé pour leur large adoption dans la littérature. Une période d'amorçage de 10 heures a été utilisée afin de garantir qu'au moins un RIB dump soit capturé pour chaque

collecteur. Dans l'environnement d'expérimentations (c'est-à-dire une machine disposant d'un processeur de 3.7 GHz/16 coeurs avec 64 Go de RAM sous Ubuntu 18.04), la collecte des données a duré environ 14 heures.

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

#################
# Collecte

folder = "dataset/"
dataset = Dataset(folder)

dataset.setParams({
    "PrimingPeriod": 10*60, # période d'amorçage de 10 heures
    "IpVersion": [4], # uniquement des routes IPv4
    "Collectors": ["rrc06"], # rrc06: à Otemachi, Japon
})

dataset.setPeriodsOfInterests([
    {
    "name": "GoogleLeak",
    "label": "anomaly",
    "start_time": utils.getTimestamp(2017, 8, 25, 3, 0, 0),  # 25 août 2017 à 3h00 UTC
    "end_time": utils.getTimestamp(2017, 8, 25, 4, 0, 0)  # 25 août 2017, 4h00 UTC
    }
])

# exécute la collecte des données
utils.runJobs(dataset.getJobs(), folder+"collect_jobs")

####################
# Transformation

from BML.transform import DatasetTransformation

datTran = DatasetTransformation(folder, "BML.transform", "Features")

datTran.setParams({
        "global":{
            "Period": 1, # attributs calculés toutes les minutes
        }
    })

# exécute la transformation des données
utils.runJobs(datTran.getJobs(), folder+"transform_jobs")
```

FIGURE 3.9 – Code BML pour la collecte et la transformation des données de la fuite de routes de Google

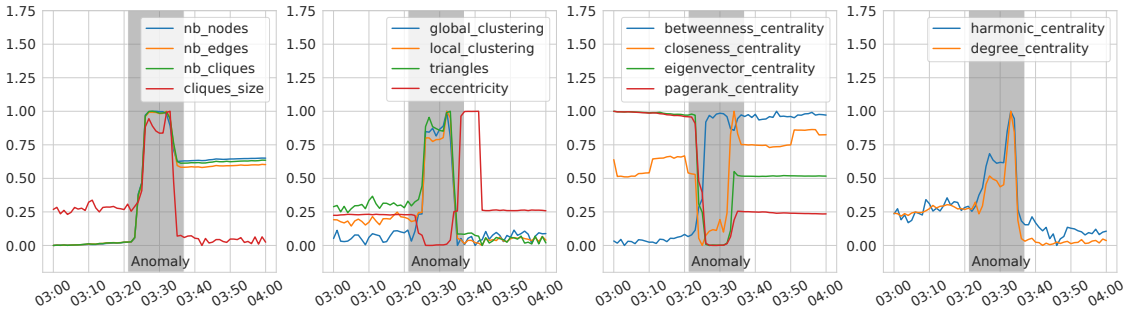


FIGURE 3.10 – Attributs extraits à partir du graphe BGP durant la fuite de routes de Google

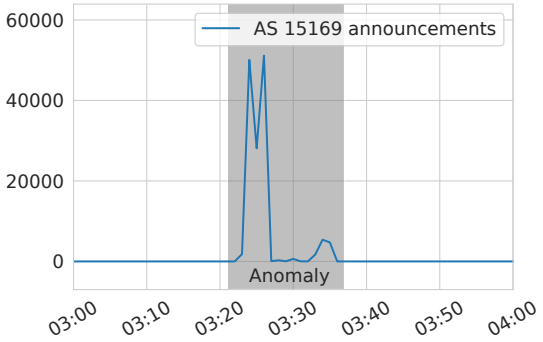


FIGURE 3.11 – Annonces de routes de l'AS 15169 durant la fuite de routes de Google

```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)
```

FIGURE 3.12 – Exemple de fonction de transformation définie par l'utilisateur

3.5 Conclusion

Dans ce chapitre, un travail préliminaire à l'application des techniques d'apprentissage automatique à BGP a été présenté. Il s'agit de la conception de BML, un outil pour la construction de jeux de données BGP. Ce travail était indispensable en raison de l'absence de jeux de données de référence dans le domaine. BML permet d'automatiser la construction de tels jeux de données qui sont de surcroît compatibles avec les algorithmes d'apprentissage automatique grâce à des méthodes de transformations de données déjà implémentées.

Ce chapitre a introduit BML en présentant son principe de fonctionnement et en détaillant certaines de ses caractéristiques qui lui procurent son efficacité en stockage et en temps d'exécution. Ensuite, le mécanisme générique de transformation des données BGP a été présenté en sus des transformations déjà implémentées dans l'outil. Enfin, la dernière section a présenté deux cas d'applications de BML. Le premier s'intéressant à l'étude d'une anomalie BGP de grande échelle et le second à la construction d'un jeu de données comportant un grand nombre d'anomalies BGP. L'outil a par ailleurs servi de base pour les contributions présentées dans les chapitres suivants ainsi que pour les travaux d'autres équipes [109].

Chapitre 4

Pertinence des représentations sous forme de graphes

Sommaire

Ce chapitre présente un travail portant sur l'évaluation de l'intérêt du graphe BGP par rapport à des attributs statistiques qui sont fréquemment utilisés dans la littérature. Un jeu de données a spécialement été construit incluant trois types d'anomalies BGP. La visibilité des anomalies dans les attributs statistiques et dans des attributs issus du graphe BGP est d'abord comparée avec une approche statistique. Puis, différents algorithmes d'apprentissage automatique sont évalués pour les deux groupes d'attributs.

Comme cela a été présenté dans l'état de l'art (section 2.4.3), les attributs statistiques extraits à partir des données BGP ont largement été adoptés dans la littérature pour l'application des techniques d'apprentissage automatique à la détection des anomalies BGP. Dans cette thèse, l'hypothèse que les représentations sous forme de graphe des données BGP peuvent permettre d'améliorer les performances de détection pour certains types d'anomalies est soutenue. Les résultats apportés dans les travaux de Sanchez et al. [4] confortent cette hypothèse. En effet, dans leurs travaux, les auteurs ont montré que des attributs extraits à partir du graphe BGP permettent la détection d'anomalies BGP à large échelle.

Cependant, à notre connaissance, il n'existe dans la littérature aucune comparaison des attributs statistiques et des attributs extraits à partir du graphe BGP. Cette section se concentre sur cette problématique afin de déterminer les bénéfices de l'utilisation des représentations sous forme de graphes des données BGP par rapport aux attributs statistiques. Plus précisément, elle propose de comparer de manière équitable les performances de détection de différents algorithmes d'apprentissage automatique en fonction du type d'anomalie considéré. Les résultats présentés dans cette section ont fait l'objet d'une publication [110].

4.1 Construction d'un jeu de données

La première étape pour la mise en oeuvre d'une comparaison des attributs statistiques et de ceux basés sur les graphes a été la construction d'un jeu de données contenant différents types d'anomalies BGP.

4.1.1 Évènements d'anomalies BGP

Afin d'être certains que les anomalies incluses dans le jeu de données soient représentatives de leur type, des anomalies documentées et qui ont donc fait l'objet d'une analyse d'expert ont été incluses. Ainsi, le jeu de données inclut 4 évènements de grande échelle, 17 évènements de détournement de préfixe par fausse origine et 14 évènements de détournement de préfixe par faux chemin.

Les 31 événements de détournement de préfixe ont été reportés dans des billets de blogs [111] et utilisés dans [70] à des fins de classification d'anomalies tandis que les événements de grande échelle ont été utilisés dans [4] pour la détection d'anomalies BGP basés sur des attributs du graphe BGP.

4.1.2 Collecte de données

Pour la collecte de données BGP et l'extraction des attributs, BML (voir section 3) a été utilisé. Pour tous les événements, les données ont été collectées une heure avant et une heure après le début estimé de l'événement. Par conséquent, pour chaque événement, 2 heures de données BGP sont utilisées. Les données sont collectées à partir des collecteurs `rrc04` et `rrc05` situés à Genève et à Vienne. Ces collecteurs ont été choisis pour leur

utilisation intensive dans la littérature [4], [64], [65]. BGP étant un protocole incrémental, il est nécessaire de collecter des données pendant une période d'amorçage (voir section 3.2) avant la fenêtre de temps de 2 heures. En utilisant les collecteurs du projet Ripe RIS, les dumps des tables RIB sont disponibles toutes les 8 heures. Une période d'amorçage de 10 heures a été utilisée pour assurer qu'au moins un dump RIB soit collecté ce qui permet d'avoir une vue complète des routes disponibles sur un collecteur. Les messages UPDATE reçus entre le dump RIB et la fenêtre d'observation sont utilisés pour mettre à jour les routes.

4.1.3 Extraction des attributs

Pour tous les événements du jeu de données, BML a été utilisé pour extraire 32 attributs statistiques (voir section 3.3.4) et 31 attributs basés sur le graphe BGP (voir section 3.3.4). Ces attributs ont été extraits toutes les 2 minutes, ce qui donne 60 échantillons de 63 attributs par événement.

Le calcul des attributs sur le graphe BGP est lourd en utilisation du processeur et de la mémoire vive. Afin de rendre ces attributs calculables en un temps raisonnable dans l'environnement d'expérimentation [1], deux options ont été explorées afin de réduire les temps de calcul. Tout d'abord, il est proposé de réduire la dimension du réseau BGP en extrayant le k-core du graphe. En théorie des graphes, le k-core d'un graphe est un sous-graphe maximal qui contient des noeuds de degré k ou plus. La motivation étant d'extraire la partie la plus connectée du graphe car il est supposé que cette dernière doit supporter la majeure partie du trafic et donc être la plus critique. La valeur de k est choisie de façon à conserver 2% des noeuds du graphe.

Cependant, il est à noter cette réduction de dimension du graphe induit une perte d'information potentiellement significative. Ainsi, la deuxième option pour la réduction des temps de calcul est l'extraction d'un sous-ensemble d'attributs sur le graphe du réseau BGP d'origine. Ce sous-ensemble est composé de 17 attributs qu'il est possible de calculer en un temps raisonnable dans la contexte de cette expérimentation.

Un problème récurrent lorsque l'on manipule des données avec plusieurs attributs est d'avoir des attributs qui n'évoluent pas dans la même plage de valeur. Cela peut entraîner un biais dans l'analyse des données ultérieure en donnant plus de poids à certains attributs. Pour éviter cela, tous les attributs ont été normalisés à l'aide de la cote Z qui transforme les attributs en une distribution de moyenne nulle et de variance unitaire :

$$x' = \frac{x - \mu}{\sigma} \tag{4.1}$$

4.1.4 Étiquetage des données

Pour chaque évènement BGP, une période d'intérêt de 120 minutes est considérée durant laquelle des échantillons de 63 attributs sont extraits toutes les 2 minutes. Un

1. Pour cette expérimentation, une machine disposant d'un processeur de 3.7 GHz/16 coeurs avec 64 Go de RAM sous Ubuntu 18.04 a été utilisée

échantillon E_i est extrait à un instant noté t_i, avec i le numéro d'échantillon dans la période d'intérêt. Le résultat est alors un ensemble de 60 échantillons $[E_1, E_{60}]$ prélevés sur la période d'intérêt $[t_1, t_{60}]$. Les périodes d'intérêts sont centrées sur le début estimé de l'anomalie. Ainsi, l'anomalie est a priori perceptible à partir de l'échantillon E_{30} collecté à t_{30} soit 60 minutes après le début de la période d'intérêt. L'hypothèse est faite que les 30 premiers échantillons $[E_1, E_{30}]$ ne contiennent pas d'anomalie et sont étiquetés « 0 » et les échantillons $[E_{31}, E_{60}]$ contiennent des anomalies et sont donc étiquetés « 1 ». Cependant, cet étiquetage induit deux biais :

— L'instant t_{31} correspondant au début de l'anomalie est rapporté par les experts ayant analysé l'évènement. Il est cependant possible qu'il y ait une imprécision de quelques minutes et que certains échantillons soient mal étiquetés. Les détournements de préfixe sont davantage sujets à ces imprécisions. Lors des représentations graphiques des résultats, le décalage peut être perceptible entre le début escompté de l'anomalie et sa visualisation dans les métriques.

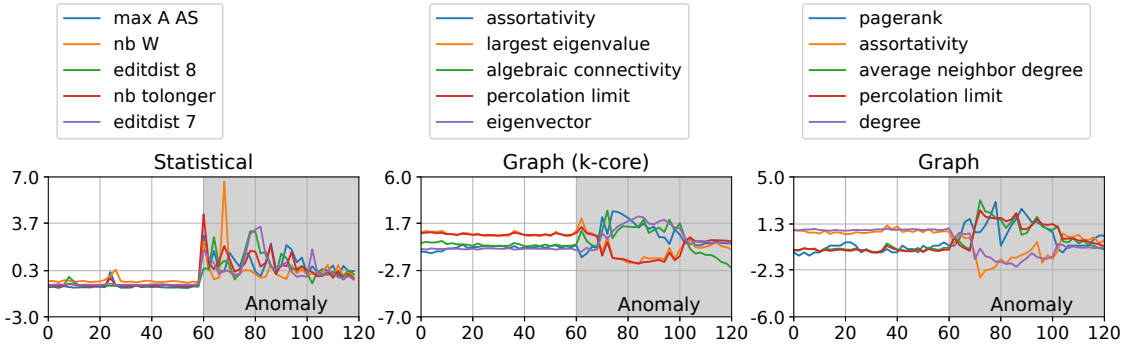
— Il est possible que l'anomalie ne soit plus perceptible avant un instant $t_f < t_{60}$. Ainsi, les échantillons $[E_f, E_{60}]$ seront étiquetés « 1 » alors qu'ils auraient dû être étiquetés « 0 ».

Ces erreurs d'étiquetage sont a priori variables en fonction des évènements et de leur type. Il n'est pas à exclure que l'entraînement supervisé puisse être impacté par des échantillons incorrectement étiquetés. Un biais peut être également induit durant la phase de validation des méthodes supervisées comme non-supervisées, entraînant une sous-évaluation des performances du modèle.

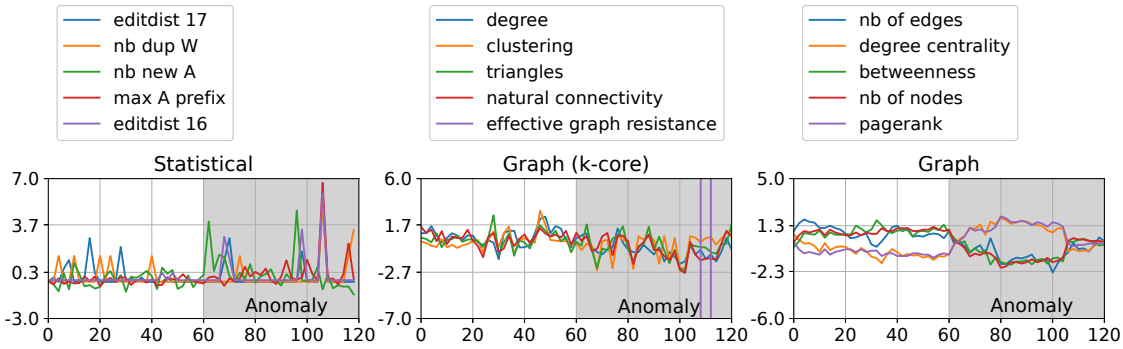
4.2 Analyse statistique et visualisation

4.2.1 Données brutes

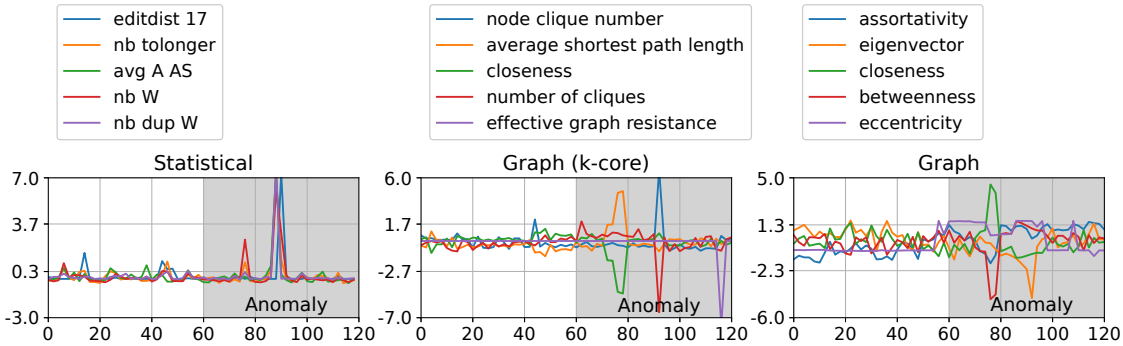
Avant toute analyse, cette section propose de visualiser l'évolution des attributs pendant la période d'intérêt. Afin de limiter la taille des figures, un évènement par type d'anomalie a été sélectionné pour lequel sont visualisés 5 attributs statistiques et 5 attributs du k-core du graphe BGP et 5 attributs du graphe d'origine 4.1. Les événements et les attributs affichés ont été sélectionnés arbitrairement pour la visibilité de l'anomalie. Par conséquent, ils doivent être considérés comme le scénario dans le meilleur des cas et ne sont pas nécessairement représentatifs des autres évènements.



(A) Évènement de grande échelle (TTNet)



(B) Détournement d'origine (Iran)



(C) Détournement de chemin (Enzu)

FIGURE 4.1 – Visualisation des données brutes

Sur l'événement de grande échelle (TTNet), l'anomalie est bien visible à la fois avec les

attributs statistiques et ceux issus des graphes BGP. Pour le détournement d'origine (Iran) l'impacte de l'anomalie est bien visible sur les attributs du graphe BGP d'origine mais est difficilement identifiable sur les attributs statiques ou du k-core du graphe. Concernant le détournement de chemin (Enzu), on peut voir que l'anomalie est visible sur les trois types d'attributs. Cependant, on observe également un décalage d'environs 15 minutes entre le début escompté de l'anomalie et son apparition dans les données, cela étant dû à l'étiquetage inexact de ce type d'événement. Pour conclure, cette visualisation montre que seuls les attributs issus du graphe d'origine permettent de distinguer les anomalies sur les trois évènements sélectionnés. Cependant, il est impossible de généraliser à partir de l'observation de ces trois évènements uniquement.

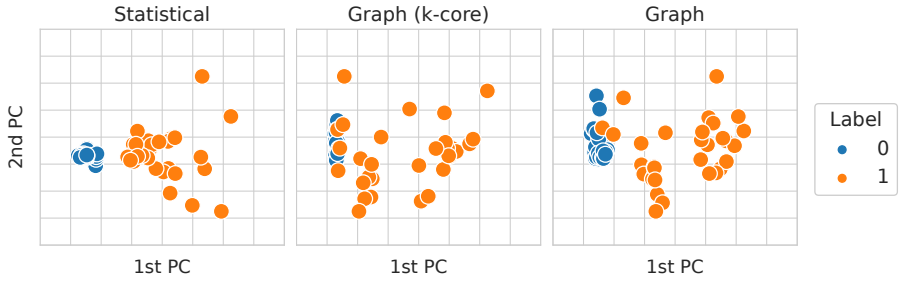
4.2.2 Visualisation synthétique

Afin d'arriver à une visualisation plus synthétique des données, une analyse en composantes principales (ACP) ou *principal component analysis* (PCA) a été utilisée afin de réduire la dimension des données. Pour chaque évènement et chaque type d'attribut une ACP est appliquée à la séquence $[E_1, E_{60}]$ et seules les 2 premières composantes principales (CP) sont conservées. On obtient alors 60 points dans un espace à 2 dimensions dont 30 sont étiquetés « 0 » et les 30 autres « 1 ». La figure 4.2 montre ainsi la projection 2d des trois évènements utilisés dans les sections précédentes.

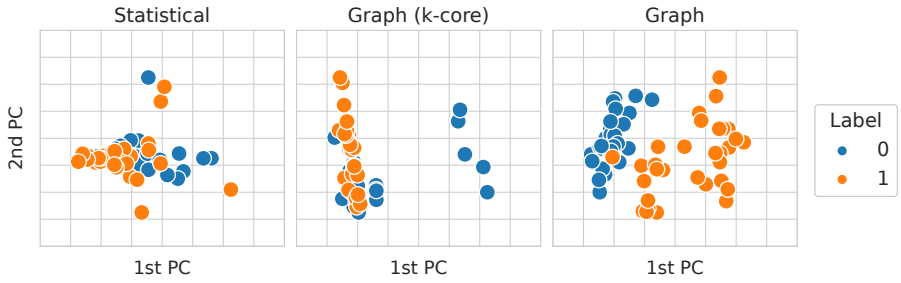
La figure 4.2 permet d'avoir un aperçu de la séparabilité entre les attributs extraits durant une anomalie et ceux extraits sans anomalies. Pour l'évènement de grande échelle (TTNet), on peut voir que pour les trois types d'attributs, les points sans anomalies ont tendance à être regroupés tandis que les points avec anomalie sont plus dispersés. Pour les évènements de détournement (Iran et Enzu), on observe que les attributs statistiques sont peu séparables car les points avec et sans anomalies se confondent. Les attributs du graphe et du k-core du graphe sont un peu plus séparables sur ces évènements. Cependant, cette appréciation visuelle ne permet pas d'affirmer de manière objective d'un évènement et plus facilement séparable avec un certain type attribut. De plus, comme pour la visualisation des données brutes il est impossible de tirer des conclusions générales. Pour cela, il est nécessaire de définir un critère de mesure de la séparabilité des données qui puisse être mesuré sur la totalité des évènements.

Pour mesurer la séparation entre les données avec et sans anomalies, le coefficient de silhouette a été utilisé, ce dernier est couramment utilisé pour évaluer la partition issue d'un algorithme de partitionnement. Pour chaque point, le coefficient de silhouette évalue la proximité du point par rapport aux points de la même partition et sa distance par rapport aux points des autres partitions. Ici, deux partitions sont présentes, une contenant 30 points sans anomalies et l'autre contenant les 30 points avec anomalies. Le coefficient de silhouette d'un point i est alors défini comme :

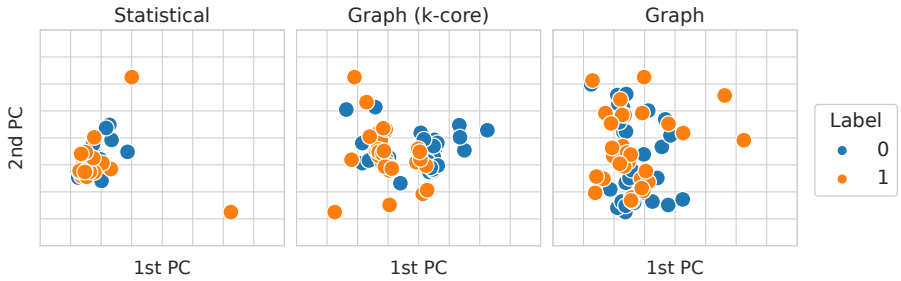
$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{4.2}$$



(A) Évènement de grande échelle (TTNet)



(B) Détournement d'origine (Iran)



(C) Détournement de chemin (Enzu)

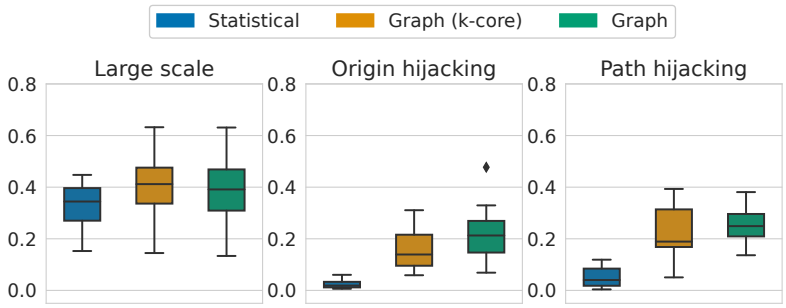
FIGURE 4.2 – Visualisation synthétique par projection 2d

où $a(i)$ est la distance moyenne entre i et tous les points de la même partition et $b(i)$ est la plus petite distance moyenne entre i et les points de toutes les autres partitions. Le coefficient de silhouette pour la partition est ensuite calculé comme la valeur moyenne sur tous les points. Le coefficient de silhouette varie de -1 à 1 où 1 est la meilleure valeur, 0 est obtenu sur des partitions qui se chevauchent et -1 lorsque des données ont été affectées aux mauvaises partitions.

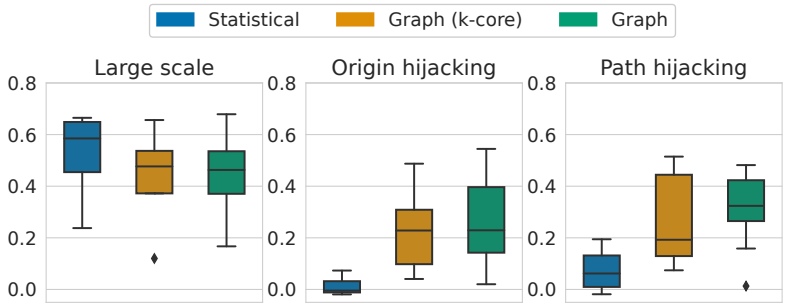
Le coefficient de silhouette donne une valeur unique pour évaluer la séparation dans les données d'un événement. Par conséquent, il est possible de l'utiliser comme métrique pour comparer la séparabilité des attributs statistiques et ceux issus du graphe. Ainsi, pour chaque type d'attribut, le coefficient de silhouette a été calculé sur l'ensemble des événements. La figure 4.3 montre la distribution de cette métrique en fonction du type d'attributs et du type d'anomalie. Le coefficient de silhouette est calculé avec et sans ACP, c'est-à-dire respectivement, en projetant les données en 2 dimensions et en conservant les données dans leur dimension d'origine. De manière générale, on observe que la réduction de la dimension des données à l'aide d'une ACP tend à améliorer la séparabilité des données. Cela peut s'expliquer par la présence d'attributs bruités dont l'effet est atténué

par l'ACP.

En se concentrant sur les résultats avec ACP, il est visible sur la figure 4.3 que les données sont légèrement mieux séparées en utilisant des attributs statistiques pour les événements de grande échelle. Cependant, pour les détournements d'origine et de chemin, les coefficients de silhouette des attributs statistiques sont proches de 0, ce qui signifie que les partitions se chevauchent. Les attributs du graphe et du k-core du graphe donnent de meilleurs résultats avec une valeur médiane autour de 0.2. Ces résultats pourraient probablement être améliorés avec un étiquetage plus précis des événements de détournement de préfixe. À partir de ces résultats, il est possible de s'attendre à attendre à obtenir des performances légèrement meilleures en utilisant des attributs statistiques au lieu des attributs issus du graphe pour la détection d'événements de grande échelle. Cependant, pour les détournements d'origine et de chemin, la détection devrait être presque aléatoire à partir des attributs statistiques, tandis que les attributs issus du graphe pourraient fournir de meilleurs résultats.



(A) Sans ACP



(B) Avec ACP

FIGURE 4.3 – Coefficient de silhouette

4.3 Application d'algorithmes d'apprentissage automatique

Cette section compare les avantages des attributs statistiques et issus du graphe pour la détection des anomalies BGP. Elle présente un pipeline d'apprentissage automatique pour classer un échantillon E_i appartenant à une séquence $[E_1, E_{60}]$ comme anormal ou normal.

4.3.1 Environnement d'expérimentation

Le pipeline d'apprentissage automatique est implémenté à l'aide de la bibliothèque *scikit-learn*. Les implémentations sont disponibles en ligne [2].

Algorithmes d'apprentissage automatique

Différents algorithmes d'apprentissage automatique peuvent conduire différents résultats sur la même tâche de classification, les performances de le pipeline sont donc évalués sur 5 algorithmes d'apprentissage automatique. Des algorithmes d'apprentissage automatique bien connus ont été utilisés, lesquels ont déjà été utilisés pour la détection d'anomalies BGP basées à la fois sur des attributs statistiques et issus du graphe BGP [4], [64], [66] : une machine à vecteurs de support ou *support-vector machine* (SVM) avec un noyau radial, un perceptron multicouche ou *multi-layer perceptron* (MLP) avec 3 couches cachées, un classificateur bayésien naïf ou *naive Bayes* (NB), un arbre de décision ou *decision tree* (DT) et un classificateur k-plus proches voisins ou *k-nearest neighbors* (KNN).

Réduction de dimension

Une bonne pratique pour l'entraînement d'un modèle d'apprentissage automatique consiste à réduire la dimension de l'espace des attributs. Cela permet de filtrer les informations redondantes et non pertinentes. De plus, cela empêche le sur-apprentissage et évite le fléau de la dimension [112] qui pourrait être un problème en raison de la taille limitée du jeu de données. Une analyse en composantes principales (ACP) a été utilisée pour projeter les caractéristiques dans un espace à 2 dimensions comme décrit dans la section précédente.

Validation croisée

Pour évaluer les performances des modèles, une approche de validation croisée avec un schéma *Leave One Group Out* (LOGO) a été utilisée. Pour un jeu de données composé de n événements, le modèle est entraîné à partir des données provenant de $n-1$ événements et l'évènement restant est utilisé comme ensemble de test indépendant. De manière itérative, chaque événement est utilisé comme ensemble de test et la moyenne des performances peut être calculée sur les n itérations. Cette approche permet d'utiliser une grande partie des données dans l'ensemble d'apprentissage tout en produisant

2. https://github.com/KevinHoarau/BGP_stat_vs_graph

des résultats statistiquement robustes. Dans les expérimentations, l'*accuracy* a été utilisée pour évaluer les performances des modèles :

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.3}$$

où TP, TN, FP and FN viennent de la matrice de confusion a savoir le nombre de vrais positifs ou *true positive* (TP), vrai négatifs ou *true negative* (TN), faux positifs ou *false positive* (FP) et faux négatifs ou *false negative* (FN).

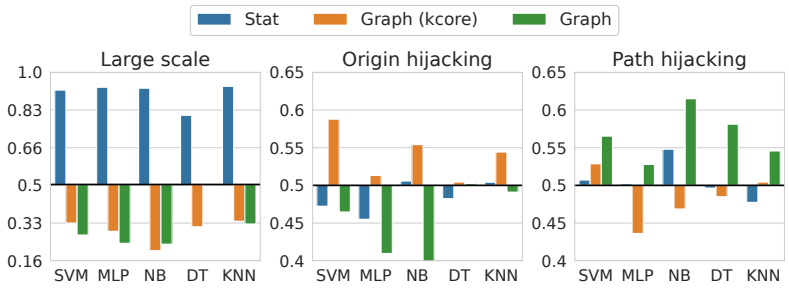
Ajustement des hyperparamètres

Un aspect très important lors de l'entraînement de modèles d'apprentissage automatique est l'ajustement des hyperparamètres des différents modèles. Ces hyperparamètres sont, par exemple, le taux d'apprentissage du MLP, la profondeur maximale de l'arbre de décision ou le nombre de voisins pour le KNN. Pour trouver la meilleure combinaison de paramètres pour chaque algorithme d'apprentissage automatique, la fonction *GridSearchCV* de *scikit-learn* a été utilisée. Cette fonction permet de définir une grille de valeurs d'hyperparamètres pour chaque algorithme et de rechercher de manière exhaustive la meilleure combinaison à partir de cette grille. Pour chaque combinaison possible, une validation croisée est effectuée sur l'ensemble d'apprentissage pour éviter le sur-entraînement. Pour cette étape, une validation croisée avec 3 partitions qui divise les événements en 3 sous-ensembles a été utilisée. Pour une itération, les données de 2 sous-ensembles sont utilisées pour l'entraînement et le sous-ensemble restant est utilisé comme ensemble de test.

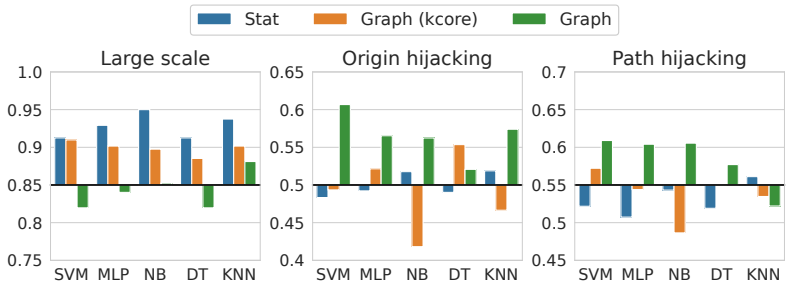
4.3.2 Performance des modèles d'apprentissage automatique

Pour évaluer les performances des modèles, le pipeline a été appliqué séparément pour chaque type d'anomalie. La figure 4.4 montre les performances des 5 modèles avec et sans ACP c'est-à-dire avec et sans réduction de la dimension des données.

De manière générale, on observe que les modèles offrent de meilleures performances lorsqu'une ACP est utilisée pour réduire la dimension des données. L'analyse des résultats se concentre donc sur cette approche pour . Comme cela a été pressenti dans les sections précédentes, les attributs statistiques surpassent les attributs du graphe BGP sur les événements de grande échelle (*accuracy* de 95% en utilisant le modèle NB), mais offrent des performances proches d'un classificateur aléatoire sur les événements de détournement d'origine et de chemin. Cependant, les attributs issus du graphe offrent une valeur d'*accuracy* de 90% sur les événements de grande échelle en utilisant le k-core et 85% avec le graphe d'origine. Sur les événements de détournement d'origine et de chemin, les attributs issus du graphe d'origine sont la meilleure option car une précision de 60% est obtenue à l'aide du modèle SVM. Sur la base de ces résultats, l'ACP sera utilisée afin de réduire la dimension des données pour le reste des expériences. De plus, les attributs issus du graphe d'origine seront utilisés car ils offrent des résultats stables sur les différents types d'événements par rapport aux attributs extraits sur le k-core du graphe.



(A) Sans ACP



(B) Avec ACP

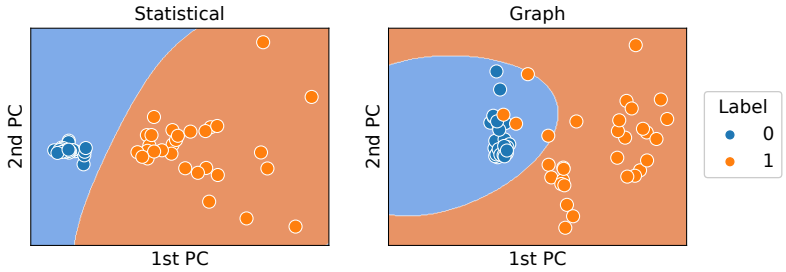
FIGURE 4.4 – Accuracy (voir éq. 4.3) des modèles ML

4.3.3 Détection d'anomalies

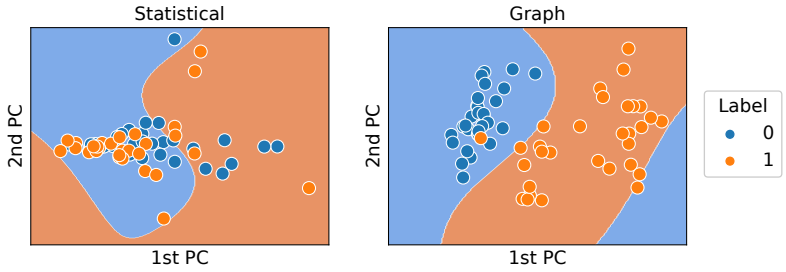
L'approche retenue à la section précédente est l'utilisation d'une ACP afin de réduire la dimension des données à deux dimensions. Les modèles d'apprentissage automatique opérants alors dans un espace à deux dimensions, il est possible de visualiser leurs frontières de décision pour une région de cet espace. Ainsi, pour un évènement donné, il est possible de projeter cet évènement en 2d et déterminer une région qui contient tous les points de l'évènement. À partir de cette région, il est possible de définir une grille de points et de leur appliquer le modèle afin d'obtenir la classe prédite pour chaque point. En prenant des points suffisamment proches une cartographie de la frontière de décision du modèle peut être effectuée.

La figure 4.5 montre la frontière de décision du modèle SVM pour chacun des trois évènements utilisés dans la section 4.2 (TTNet, Iran et Enzu). Les points de l'évènement ont une couleur en fonction de leur étiquette tandis que le fond de la figure est colorié en fonction de la prédiction du modèle. Sur l'évènement de grande échelle (TTNet), on peut voir que les attributs statistiques permettent de classer parfaitement les données de cet évènement avec une frontière de décision simple. Les attributs issus du graphe ont également une frontière de décision relativement simple avec uniquement 4 points mal classifiés par le modèle. Cette observation coïncide bien avec les résultats quantitatifs de la section précédente. Pour les détournements de préfixe (Iran et Enzu), on observe que les frontières de décision sont plus simples avec les modèles utilisant les attributs issus du graphe et sépare mieux les données ce qui correspond également aux résultats de la section précédente.

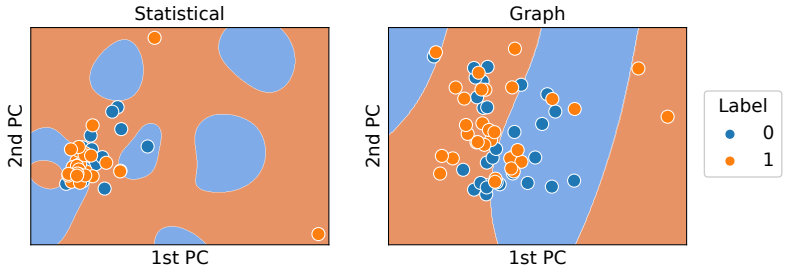
Il est visible sur la figure 4.5 que pour un évènement donné, même si la majorité des



(A) Évènement de grande échelle (TTNet)



(B) Détournement d'origine (Iran)



(C) Détournement de chemin (Enzu)

FIGURE 4.5 – Frontière de décision du modèle SVM

points est correctement classifiée par le modèle, quelques points peuvent être mal classifiés résultant en des faux positifs ou des faux négatifs. Cela peut être la conséquence de l'étiquetage inexact de ces données ou d'une mauvaise classification du modèle. Pour atténuer ce problème, il est possible d'utiliser plusieurs points afin de détecter une anomalie uniquement si une partie importante de ces points est classée comme anormale. Pour évaluer les performances d'un tel schéma de détection d'anomalies, chaque événement est divisé en deux sous-ensembles : l'ensemble négatif et positif. Le sous-ensemble négatif est composé des 30 points étiquetés « 0 » tandis que le sous-ensemble positif est composé des 30 points étiquetés « 1 ». Ensuite, pour chaque sous-ensemble, tous les points sont classés par le modèle et une anomalie est détectée uniquement si le nombre de points classés comme anormaux est supérieur à un seuil.

La figure 4.6 montre les performances d'une telle approche pour la détection d'anomalies BGP en fonction de la valeur du seuil. Le modèle SVM a été utilisé pour la classification individuelle des points. On peut voir que les meilleures performances en termes d'*accuracy* sont obtenues en utilisant un seuil de 15 à 20 échantillons. Pour les événements de grande échelle utilisant les attributs statistiques, une valeur d'*accuracy* de 100% contre 88% pour les attributs issus du graphe est obtenue. En revanche, pour les événements

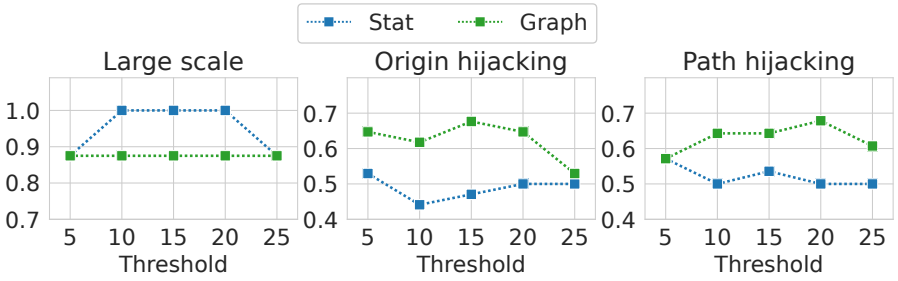


FIGURE 4.6 – Accuracy de la détection d'anomalies (voir eq. 4.3) en fonction du seuil c'est-à-dire le nombre d'échantillons requis pour être classé en anomalie par le SVM.

de détournement d'origine et de chemin, les attributs issus du graphe donnent une valeur d'*accuracy* de 68% tandis que pour les attributs statistiques, les performances sont équivalentes à un classifieur aléatoire c'est-à-dire environ 50%.

4.4 Conclusion

L'objectif du travail présenté dans cette section est la comparaison des attributs statistiques et des attributs extraits à partir du graphe BGP pour la détection des anomalies BGP. Cela a consisté à comparer de manière équitable ces deux types attributs en fonction du type d'anomalie considéré. Trois types d'anomalies ont ainsi été étudiés dans ce travail : des évènements de grande échelle, des détournements d'origines et des détournements de chemins. Une première étape d'analyse des données à permis de construire différentes visualisations des données mais aussi d'extraire une mesure quantitative de la séparabilité des échantillons étiquetés normaux et anormaux. Dans un second temps, différents algorithmes d'apprentissage automatique ont été évalués afin tout d'abord de classer un unique échantillon d'attributs puis de détecter une anomalie à partir d'une séquence d'attributs. Les étapes d'analyse de données et d'application d'algorithme d'apprentissage automatique ont mené à des observations similaires.

Pour les anomalies de grande échelle, les attributs statistiques offrent une meilleure séparabilité des données et de meilleures performances de détection. Cependant, les attributs issus du graphe BGP offrent également de bonnes performances sur ce type d'évènement avec une valeur d'*accuracy* (voir équation 4.3) de 88%. Ce travail a également montré que sur des évènements de plus petite échelle, c'est-à-dire des détournements de préfixes, les attributs statistiques ne permettent de discriminer les données normales et anormales. Les attributs issus du graphe quant à eux le peuvent et offrent une valeur d'*accuracy* de 68%. Ces résultats préliminaires sur les anomalies BGP de petite échelle sont intéressants car ils démontrent l'intérêt des représentations du graphe BGP pour la détection d'anomalie. Ici, l'exploitation du graphe BGP avec des métriques génériques permet à la fois d'atteindre des performances satisfaisantes sur les anomalies de grande échelle et de rendre possible la détection de certaines anomalies de plus petite échelle. Ces résultats motivent donc l'approfondissement d'approches pour la détection d'anomalies BGP basées sur des représentations sous forme de graphe.

Chapitre 5

Réseau de neurones récurrent pour l'analyse de séries temporelles BGP

Sommaire

En s'appuyant sur les résultats du chapitre précédent, ce chapitre propose de tirer profit de la composante temporelle des données BGP. Elle considère alors le problème de la détection d'une anomalie dans une série temporelle d'attributs du graphe BGP. Après une analyse statistique de l'impact des anomalies sur les attributs, un modèle de réseau de neurones récurrent (RNN) est proposé et évalué.

Le chapitre précédent à mis en évidence l'intérêt des représentations basées sur le graphe BGP pour la détection d'anomalie. Plus particulièrement, ce type de représentations semble être le plus approprié pour les anomalies de petite échelle. De plus, les expérimentations du chapitre précédent ont également montré que le fait de classifier une séquence d'échantillons plutôt qu'un échantillon pris individuellement permettait d'augmenter les performances du modèle (+8% sur l'*accuracy*). Ainsi, ce travail propose un modèle capable de s'appuyer sur la dimension temporelle des données BGP afin de détecter une anomalie.

Dans ce chapitre, un modèle de réseau de neurones récurrent (RNN) permettant détecter une anomalie dans une séquence d'attributs du graphe BGP est proposé et évalué. De plus, une analyse des données et du modèle est présentée afin de déterminer les attributs prédicteurs d'une anomalie. Les performances du modèle sont également analysées afin de dégager des perspectives d'améliorations. Les résultats présentés dans cette section ont fait l'objet d'une publication [113].

5.1 Construction d'un jeu de données

Le jeu de données comprend 28 enregistrements dont 14 positifs et 14 négatifs. Les enregistrements positifs sont extraits lors d'occurrences de détournement de chemin. Les enregistrements négatifs ont été prélevés arbitrairement 24h avant chaque enregistrement positif. Il n'y a pas d'anomalies connues dans les enregistrements négatifs. Pour chaque enregistrement, 2 heures de données BGP sont collectées et sont échantillonnées toutes les deux minutes pour générer un total de 60 instantanés du graphe BGP. À partir de chaque graphe, 13 attributs sont extraits. Il en résulte que pour chaque enregistrement du jeu de données, une série temporelle multivariée $X = x^1, ..., x^{60}$ est extraite où x^t est un vecteur de dimension 13.

5.1.1 Évènements d'anomalies BGP

Ce travail s'appuie sur 14 événements de détournement de chemin bien documentés et qui ont donc fait l'objet d'une analyse d'expert [111]. Ces évènements ont également été utilisés au chapitre précédent (voir chapitre 4). Dans [70] les auteurs se sont également appuyés sur ces évènements à des fins de classification des anomalies.

5.1.2 Collecte de données

Pour la collecte de données et l'extraction des attributs, BML (voir chapitre 3) a été utilisé. Pour tous les enregistrements positifs, les données ont été collectées une heure avant et une heure après le début estimé de l'événement. Pour les enregistrements négatifs, les données ont été collectées un jour avant chaque événement pendant 2 heures. Par conséquent, pour chaque enregistrement, 2 heures de données BGP sont utilisés. Comme au chapitre précédent (voir chapitre 4), les données sont collectées à partir des collecteurs

`rrc04` et `rrc05` situés à Genève et à Vienne et une période d'amorçage (voir section 3.2) 10 heures a été utilisée.

5.1.3 Extraction des attributs

Pour tous les enregistrements du jeu de données, BML a été utilisé pour extraire un instantané du graphe BGP toutes les 2 minutes. Les enregistrements de deux heures donnent alors des séquences de 60 graphes. Sur chacun de ces graphes, 13 (voir tableau 5.1) attributs sont calculés. Ces derniers peuvent être divisés en deux catégories (voir chapitre 3) :

— Au niveau des noeuds (8 attributs) : ces métriques sont calculées pour tous les noeuds du graphe et en utilisant la valeur moyenne comme attribut. Il comprend des métriques de centralité qui mesurent l'importance d'un noeud dans un graphe. D'autres métriques telles que le coefficient d'agglomération ou *clustering coefficient* et le degré mesurent la connectivité d'un noeud.

— Au niveau du graphe (5 attributs) : les métriques sont définies globalement et fournissent une valeur unique pour un graphe entier. Elles sont généralement utilisées pour évaluer la robustesse structurelle d'un réseau [5].

Type of metric	**Metric**
Node Level	Average neighbors degree Closeness centrality Clustering coefficient Degree Degree centrality Eigenvector centrality Pagerank Number of triangles
Graph Level	Assortativity Diameter Number of edges Number of nodes Percolation limit

TABLE 5.1 – Attributs extraits (voir chapitre 3)

Les valeurs de ces 13 attributs n'évoluent pas nécessairement dans la même plage de valeurs. Cela peut entraîner un biais dans l'analyse des données ultérieure en donnant plus de poids à certains attributs. Pour éviter cela, tous les attributs sont normalisés à l'aide de la cote Z qui transforme les attributs en une distribution de moyenne nulle et de variance unitaire (voir équation 4.1).

5.2 Analyse de l'impact des détournements de chemins

5.2.1 Visualisation des données brutes

Afin d'avoir un aperçu de la façon dont les attributs extraits varient lors d'un détournement de chemin, il sont visualisés sur un événement du jeu de données qui a été sélectionné arbitrairement pour la visibilité de l'anomalie. Par conséquent, il doit être considéré comme le meilleur scénario et pas nécessairement représentatif des autres enregistrements du jeu de données. La figure 5.1 montre la série temporelle des attributs pour cet événement (France).

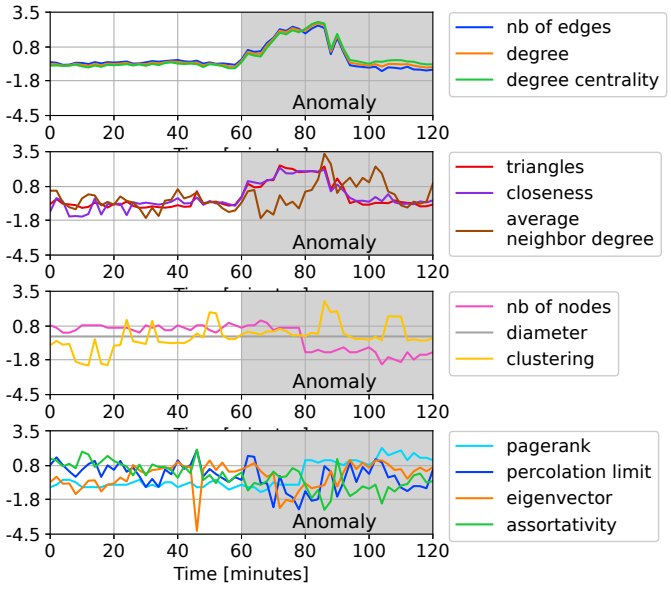


FIGURE 5.1 – Évolution de la série temporelle des attributs du graphe BGP avant et après l'apparition d'un détournement de chemin (événement France)

Sur cet événement, il est visible que certaines fonctionnalités sont plus impactées que d'autres par l'anomalie mais aussi qu'à la fois les attributs au niveau des noeuds et du graphe sont impactés. L'augmentation du nombre de liens et des degrés indique que le graphe est plus connexe pendant l'anomalie. Ceci peut s'expliquer par l'apparition des faux liens dans le graphe.

5.2.2 Analyse de l'impact des anomalies par attributs

Afin d'avoir une visualisation synthétique de l'impact d'une anomalie sur les différents attributs, ce travail propose de compter le nombre de valeurs aberrantes pour chaque attribut. Les valeurs aberrantes sont identifiées dans la série temporelle d'un attribut $X_f = x_f^1, ..., x_f^{60}$ en utilisant la méthode de l'écart interquartile ou *Inter-Quartile Range* (IQR). Cette méthode suppose que tout point de données inférieur à la limite inférieure (*lower*) ou supérieure à la limite *upper* est considéré comme une valeur aberrante :

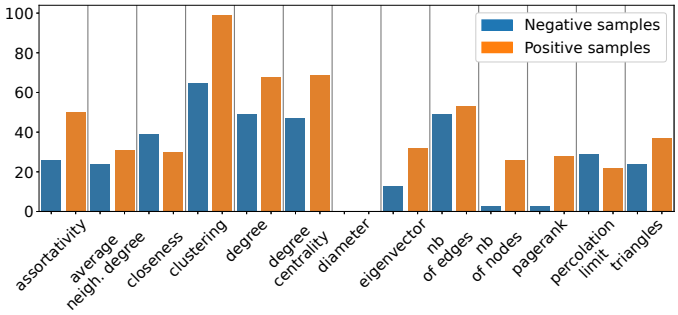


FIGURE 5.2 – Nombre de valeurs aberrantes par attribut

$$\begin{aligned} \textit{lower} &= (Q1 - 1.5 * IQR) \\ \textit{upper} &= (Q3 + 1.5 * IQR) \end{aligned} \tag{5.1}$$

où $Q1$ et $Q3$ sont les premier et troisième quartiles et $IQR = Q3 - Q1$.

La figure 5.2 montre la somme des valeurs aberrantes pour chaque attribut sur tous les enregistrements positifs et négatifs. Parmi les 13 attributs, 10 (7 au niveau du noeud et 3 au niveau du graphe) comptent plus de valeurs aberrantes dans les enregistrements positifs, ce qui indique qu'elles sont impactées par les anomalies. L'augmentation du nombre de valeurs aberrantes pour des attributs tels que le coefficient d'agglomération ou *clustering coefficient* et le degré indique que la connectivité du graphe est affectée par les anomalies.

5.3 Détection d'anomalies avec un RNN

5.3.1 Architecture du modèle RNN

Le problème que ciblé dans ce travail est un problème de classification de séquences. Pour une séquence d'entrée $X = x^1, ..., x^{60}$ le modèle doit produire une sortie $Y = 1$ si une anomalie est détectée dans la séquence ou $Y = 0$ sinon. Pour attaquer ce problème, une architecture de réseau de neurones récurrents ou *recurrent neural network* (RNN) à mémoire court et long terme est utilisé. Ce type d'architecture est désigné sous le terme *long short-term memory* (LSTM). Dans un RNN, pour chaque élément x^t un état caché h^t est produit en utilisant un ensemble partagé de paramètres θ et l'état caché de l'élément précédent h^{t-1} :

$$h^t = f(h^{t-1}, x^t; \theta) \tag{5.2}$$

Les RNN sont ainsi capables de tirer parti de la dimension temporelle des données. Les LSTM sont une amélioration des RNN qui permettent de capturer les dépendances à long terme en résolvant le problème de la disparition des gradients.

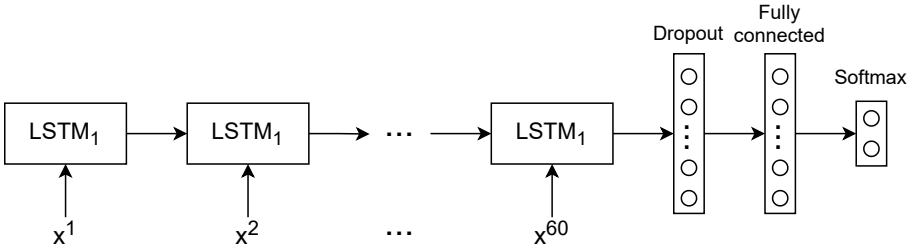


FIGURE 5.3 – Architecture du modèle RNN

L'architecture du modèle RNN est représentée sur la figure 5.3 et les hyperparamètres utilisés lors de l'entraînement du modèle sont listés dans le tableau 5.2. Tout d'abord, la séquence d'entrée x^t est transmise à un LSTM à une couche qui produit une sortie à 32 dimensions. Ensuite, une couche de *dropout* est utilisée pour aider à la généralisation du modèle. Pour chaque échantillon d'apprentissage et pour chaque *epoch*, la couche de *dropout* met à zéro au hasard 25% du vecteur à 32 dimensions, ce qui empêche le sur-apprentissage du modèle. Enfin, un réseau de neurones entièrement connecté à une seule couche avec une fonction d'activation *softmax* est utilisé pour prédire la classe de sortie.

Hyperparameter	**Value**
LSTM units	32
LSTM layers	1
Fully connected layers	1
Dropout rate	0.25
Nb. epochs	50
Learning rate	0.1

TABLE 5.2 – Model hyperparameters

5.3.2 Métriques d'évaluation

Pour évaluer les performances du modèle les métriques suivantes sont utilisées :

Accuracy : est utilisée pour évaluer la performance globale du classificateur pour les enregistrements positifs et négatifs.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision : évalue le nombre de vrais positifs parmi tous les enregistrements classés comme positifs.

$$Precision = \frac{TP}{TP + FP}$$

Recall : évalue le nombre d'enregistrements classés comme positifs parmi tous les enregistrements positifs.

$$Recall = \frac{TP}{TP + FN}$$

F1 score : est la moyenne harmonique de la *precision* et du *recall*.

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

où TP, TN, FP and FN viennent de la matrice de confusion a savoir le nombre de vrais positifs ou *true positive* (TP), vrais négatifs ou *true negative* (TN), faux positifs ou *false positive* (FP) et faux négatifs ou *false negative* (FN).

5.3.3 Résultats expérimentaux

Les expériences ont été implémentées à l'aide des bibliothèques *PyTorch* et *scikit-learn*. Les implémentations sont disponibles en ligne [1]. L'évaluation des performances du modèle est faite en utilisant un schéma de validation croisée avec 3 partitions. Le jeu de données est divisé en 3 sous-ensembles et de manière itérative un sous-ensemble est utilisé comme ensemble de test et le reste comme ensemble d'apprentissage. Avec cette approche, une grande partie des données peut être utilisée pour l'apprentissage (66%) tout en produisant des résultats statistiquement robustes. Les expériences ont été répétées 33 fois afin de calculer l'écart type des mesures de performance mesurées.

Les résultats de la validation croisée sont rapportés dans le tableau 5.3. Le modèle atteint une performance de classification correcte avec une valeur d'*accuracy* moyenne de 67%, une valeur de *recall* moyenne (65%) et une valeur de *precision* moyenne (72%). On voit également que les performances sont stables sur les 3 partions de test, notamment l'*acuracy* qui ne varient que de 0.01.

Il est également intéressant de noter que l'*accuracy* obtenue (67%) est équivalente à celle obtenue au chapitre précédent (voir chapitre 4) qui était de 68%. Cette valeur avait

1. https ://github.com/KevinHoarau/BGP_LSTM

Test fold	**Accuracy**		**F1 score**		**Precision**		**Recall**	
	Mean	Std dev	Mean	Std dev	Mean	Std dev	Mean	Std dev
1	0.68	0.10	0.59	0.16	0.83	0.18	0.49	0.19
2	0.67	0.13	0.64	0.20	0.67	0.17	0.65	0.26
3	0.67	0.11	0.71	0.10	0.65	0.12	0.82	0.17
All	**0.67**	**0.11**	**0.65**	**0.16**	**0.72**	**0.18**	**0.65**	**0.25**

TABLE 5.3 – Résultats de la validation croisée (sur 13 attributs)

été obtenue sur les mêmes évènements en utilisant une approche classifiant également une séquence d'attributs. Bien que cette approche soit moins expressive que l'architecture proposée dans ce travail, des performances équivalentes sont obtenues. Ainsi, on peut émettre l'hypothèse que les données ne permettent pas de discriminer certains des évènements étudiés. Il serait alors opportun d'explorer des représentations du graphe BGP entraînant moins de perte d'information que les attributs actuels qui sont mesurés sur l'ensemble du graphe.

Réduction du nombre d'attributs

Comme indiqué précédemment, tous les attributs ne sont pas affectés de la même manière par une anomalie. La sélection d'attributs est couramment utilisée dans la communauté de l'apprentissage automatique pour réduire la complexité des modèles. La réduction de la complexité est importante si le modèle est utilisé pour la détection d'anomalies en temps réel. De plus, la sélection d'attributs empêche le sur-apprentissage et évite le fléau de la dimension qui pourrait être un problème en raison de la taille limitée du jeu de données. Les k attributs les plus importants sont sélectionnés en utilisant la technique des gradients intégrés [114]. Cette méthode calcule l'importance de chaque attribut d'un réseau de neurones. Elle est appliquée à l'aide de la bibliothèque *captum* qui implémente des méthodes d'interprétabilité et de compréhension pour les modèles *PyTorch*.

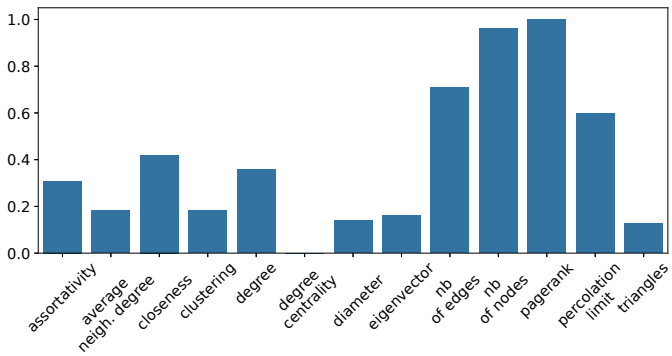


FIGURE 5.4 – Importance des attributs selon la méthode des gradients intégrés [114]

La figure 5.4 montre l'importance de chaque attribut qui est moyenné sur les trois modèles issus de la validation croisée. Quatre attributs ressortent : le nombre de liens, le

nombre de noeuds, le *pagerank* et la limite de percolation. Ces résultats coïncident partiellement avec ceux issus de l'analyse de données dans la section 5.2 (figure 5.2). En effet, il était possible de deviner que le nombre de noeuds et le *pagerank* étaient de bons prédicteurs en raison de l'écart important dans leur nombre de valeurs aberrantes entre les enregistrements positifs et négatifs. Cependant, l'importance du nombre de liens et de la limite de percolation tend à montrer que le modèle repose sur des motifs plus complexes à l'intérieur de ces attributs que le simple nombre de valeurs aberrantes.

Afin de confirmer que ces 4 attributs sont suffisants pour détecter les détournements de chemin dans les données, le modèle a été entraîné en utilisant uniquement ces attributs en entrée. Les résultats de la validation croisée sont rapportés dans le tableau 5.4.

Test fold	**Accuracy**		**F1 score**		**Precision**		**Recall**	
	Mean	Std dev	Mean	Std dev	Mean	Std dev	Mean	Std dev
1	0.66	0.11	0.56	0.18	0.80	0.19	0.46	0.20
2	0.66	0.15	0.67	0.15	0.66	0.15	0.69	0.17
3	0.69	0.14	0.71	0.13	0.68	0.14	0.77	0.18
All	**0.67**	**0.13**	**0.65**	**0.17**	**0.72**	**0.17**	**0.64**	**0.23**

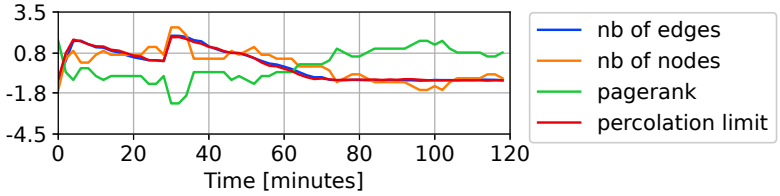
TABLE 5.4 – Résultats de la validation croisée (sur 4 attributs)

En utilisant cet ensemble de quatre attributs, le modèle fournit approximativement les mêmes performances qu'avec les 13 attributs de départ. La valeur d'*accuracy* moyenne reste la même (67%), la valeur de *recall* est de 64% avec 4 caractéristiques contre 65% avec 13 caractéristiques tandis que la valeur de *precision* est inchangé (72%). Ces performances peuvent s'expliquer par le fait les attributs non pertinents et qui introduisent du bruit lors de l'apprentissage du modèle ont été supprimés. Cette réduction du nombre d'attributs conduit également à un pipeline plus efficace dans un environnement de production. Premièrement, cela permet d'entraîner un modèle moins complexe qui est plus efficace en termes de calcul. Deuxièmement, moins d'attributs doivent être extraits pour classer un nouvel enregistrement.

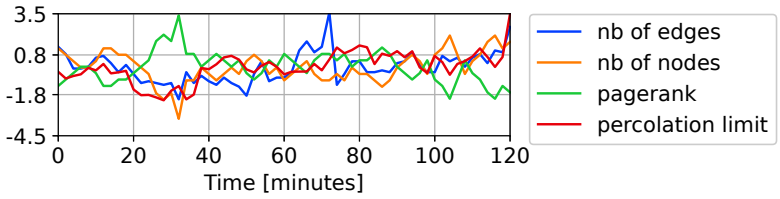
Analyse d'erreurs de classification

Pour avoir un aperçu des erreurs de classification commises par le modèle, les données brutes d'un faux positif et d'un faux négatif sont visualisées, c'est-à-dire respectivement, d'un enregistrement négatif classé positif et d'un enregistrement positif classé négatif.

La figure 5.5a montre un faux positif où l'on peut voir une anomalie à partir de $x = 25$ minutes. De tels motifs peuvent être causés par des anomalies moins nuisibles que les détournements de chemin tels que la défaillance d'une liaison physique. Cependant, comme ce motif est probablement différent de celui causé par détournement de chemin, le fait d'avoir plus de données d'apprentissage pourrait éviter de telles erreurs de classification. La figure 5.5b montre un événement faux négatif. Ces données sont bruitées et en essayant d'identifier manuellement les anomalies dans ces données, il est



(A) Faux positif : Torg (24h avant l'évènement)



(B) Faux négatif : Petersburg

FIGURE 5.5 – Exemple d'enregistrements mal classifiés

possible d'en détecter à $x = 30$, 40 et 70 minutes. En raison de son niveau de bruit, cet événement est probablement trop différent des anomalies de l'ensemble d'apprentissage. Encore une fois, avec plus de données d'entraînement, il serait peut-être possible d'éviter de telles erreurs de classification en introduisant plus d'événements similaires à celui-ci. D'autre part, une représentation du graphe BGP moins sujette au bruit par rapport aux attributs utilisé dans ce travail pourrait également permettre d'éviter ce type d'erreurs de classification.

5.4 Conclusion

Ce chapitre propose un modèle de réseau de neurones récurrent (RNN) permettant détecter une anomalie dans une séquence d'attributs du graphe BGP. Ce modèle a été évalué sur un ensemble de 13 attributs du graphe BGP. Puis, une analyse du modèle à permis d'identifier un sous-ensemble de 4 attributs permettant d'atteindre des performances équivalentes tout en réduisant la complexité du modèle. Cette réduction de complexité permettant à la fois d'éviter des problèmes de sur-apprentissage mais aussi d'obtenir un pipeline plus efficace dans un contexte de mise en production. De plus, en mettant en perspective ce sous-ensemble d'attributs avec une analyse statistique des données, il a été mis en évidence que le modèle s'appuie sur des motifs plus complexes que ceux préalablement identifiés.

L'analyse d'erreurs de classification du modèle a permis d'identifier que les données d'apprentissage semblent être la principale limitation pour l'amélioration des performances du modèle. Tout d'abord, la taille limitée du jeu de donnée implique que certains motifs d'anomalies sont peu représentés dans le jeu de données. De plus, les attributs du graphe de par leur nature agrégée sont très bruités rendant difficile la distinction entre le bruit et les anomalies. Cette limitation due au jeu de données utilisé semble également se confirmer dans les performances atteintes par le modèle (67%) qui sont équivalentes à celles obtenues au chapitre précédent (68%) quand bien même le modèle dispose d'une plus grande expressivité.

En conclusion, deux pistes principales sont identifiées pour l'amélioration des performances des modèles pour la détection d'anomalies BGP. Premièrement, l'entraînement des modèles sur un plus gros jeu de données. Et deuxièmement, l'utilisation de représentations moins bruitées que les attributs du graphe. Ce dernier point peut se traduire, par exemple, par l'utilisation de représentations plus riches du graphe BGP mais aussi en se reposant sur des approches localisées visant à détecter des anomalies sur un sous-ensemble du graphe.

Chapitre 6

Détection d'anomalies BGP basée sur les Graph Neural Networks

Sommaire

Ce chapitre propose de s'affranchir de la coûteuse étape d'extraction d'attributs à partir du graphe BGP mais également la perte d'information qui en découle. Pour cela, des modèles basés sur des *Graph Neural Networks* (GNN) qui consomment directement des graphes BGP sont proposés. Un travail préliminaire exploitant une architecture *auto-encoder* a démontré la faisabilité de cette approche pour la détection d'anomalies BGP. Ensuite, BGNN est présenté, il s'agit d'un modèle basé sur un GNN pour la détection localisée d'une anomalie dans une séquence de graphes BGP. BGNN génère les séquences de plongements des AS, lesquels sont ensuite classifiés avec un *multilayer perceptron* (MLP).

Les travaux présentés dans les sections précédentes ont montré l'intérêt des représentations sous forme de graphes des données BGP pour la détection d'anomalies. Ces expérimentations ont permis d'identifier qu'une approche localisée est une piste à explorer pour l'amélioration les performances des modèles d'apprentissage automatique proposés. En effet, une approche localisée permettrait de considérer uniquement les portions du graphe BGP concernées par une anomalie durant l'entraînement. Un modèle ainsi entraîné pourrait alors être déployé pour la surveillance spécifique d'un AS ou bien pour une surveillance globale du réseau en appliquant le modèle sur un ensemble d'AS réparti sur le graphe BGP.

Les Graph Neural Networks (GNN) sont particulièrement adaptés dans le cadre d'une approche localisée puisqu'ils permettent de produire des plongements des noeuds d'un graphe pouvant ensuite être consommés par des modèles d'apprentissage automatique. Par ailleurs, un travail préliminaire effectué en début de thèse avait mis en évidence la visibilité d'une anomalie BGP sur le plongement d'un AS produit par un GNN. Ce travail est présenté dans la première section de ce chapitre. La section suivante présente BGNN, un modèle basé sur un GNN pour la détection localisée d'une anomalie dans une séquence de graphes BGP.

À notre connaissance, BGNN est le premier modèle basé sur un GNN pour la détection d'anomalies BGP. Ce travail a fait l'objet d'une publication [115].

6.1 Travail préliminaire avec les GNN

Cette section présente un travail préliminaire qui a été effectué en première partie de thèse afin de valider l'approche consistant à utiliser les GNN pour la détection des anomalies BGP. Les objectifs de ce travail préliminaire sont, d'une part de valider la faisabilité technique de cette approche, et d'autre part d'évaluer le potentiel des GNN pour la détection d'anomalies sur des graphes. Pour cela, ce travail s'appuie sur un modèle issu de la littérature. Le modèle Graph Auto-encoder (GAE) [91] est entraîné de manière non-supervisée afin un produire un plongement d'un graphe d'entrée et à utiliser ce plongement pour reconstruire la matrice d'adjacence du graphe. Ainsi, dans ce travail, ce modèle est entraîné sur un ensemble de graphes qui reflètent le comportement normal de BGP puis l'évolution de l'erreur de reconstruction lors d'une anomalie BGP est évaluée.

6.1.1 Jeu de données

Ce travail a été effectué avant l'élaboration de l'outil BML. Cependant, c'est ce travail préliminaire qui a permis d'identifier un certains nombre de problématiques pour la construction de jeux de données BGP qui ont amené à la conception de cet outil.

Le jeu données est décrit dans le tableau 6.1 et est constitué d'un ensemble de graphes BGP (voir section 3.3.4). Pour extraire ces graphes des données BGP ont été collectées à partir de tous les collecteurs du projet Ripe RIS [41]. Chaque noeud des graphes extraits a un poids qui correspond au nombre d'adresses IP couvertes par tous les préfixes appartenant à l'AS. Ainsi, un AS possédant deux préfixes $\backslash 24$ aura un poids $p = 2 \times 2^{(32-24)} =$

512. De même, chaque lien possède un poids qui correspond au nombre d'IP couvertes par tous les préfixes routés sur ce lien. Le jeu de données est constitué de 372 graphes qui sont répartis dans deux sous ensembles :

L'ensemble d'apprentissage : l'ensemble d'apprentissage contient uniquement des graphes sans anomalie et qui reflète donc le fonctionnement normal de BGP. Les graphes sont extraits entre 03h00 UTC et 04h00 UTC le 25 de chaque mois de 2017 sauf les mois de février, avril, août et octobre. Durant cette période d'intérêt d'une heure, un graphe est extrait toutes les 2 minutes. Le premier graphe est construit avec les mises à jour BGP collectées pendant les 10 heures précédant le début de la période d'intérêt et le graphe suivant est construit en ajoutant les mises à jour BGP collectées les 2 minutes suivantes. Pour chaque période d'intérêt 31 graphes sont extraits et l'ensemble d'apprentissage contient donc $31 \times 8 = 248$ graphes.

L'ensemble de test : l'ensemble de test est construit de la même manière que le l'ensemble d'apprentissage mais pour les 4 autres mois de 2017 (février, avril, août et octobre). L'ensemble de test contient ainsi $31 \times 4 = 124$ graphes. L'ensemble de test inclut une anomalie car un incident BGP de grande échelle [1] s'est produit le 25 août 2017 à 03h22 UTC. Cet incident était dû à une mauvaise configuration de l'AS 15169 qui appartient à Google.

Début	Fin	Année	Jour	Mois	Apprentissage	Test	Anomalie
03h00 UTC	04h00 UTC	2017	25	Janvier	✓		
				Février		✓	
				Mars	✓		
				Avril		✓	
				Mai	✓		
				Juin	✓		
				Juillet	✓		
				Août		✓	✓
				Septembre	✓		
				Octobre		✓	
				Novembre	✓		
				Décembre	✓		

TABLE 6.1 – Description du jeu de données

6.1.2 Le modèle Graph Auto-Encoder

Le modèle GNN est une variation du Graph Auto-Encoder (GAE) proposé par Kipf et al. dans [91] et est implémenté en utilisant la bibliothèque *PyTorch Geometric* [116]. L'architecture du modèle est présentée sur la figure 6.1. La partie encodeur du modèle a été modifiée pour utiliser 4 couches de GCN [77] et pour tirer profit des poids des liens. Chaque couche de GCN produit un vecteur à 32 dimensions pour chaque noeud du graphe d'entrée et la fonction d'activation *ReLu* a été remplacée par la fonction *Sigmoïde* en raison d'un problème d'optimisation pendant l'apprentissage. La dernière couche GCN produit

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

un plongement à 16 dimensions pour chaque noeud du graphe d'entrée. On obtient alors une matrice Z de dimension $N \times 16$ où N est le nombre de noeuds du graphe. La partie décodeur n'a pas été modifiée et utilise le produit scalaire des plongements pour reconstruire la matrice d'adjacence du graphe :

$$\hat{A} = \sigma(ZZ^T) \tag{6.1}$$

où σ est la fonction d'activation *sigmoïde*.

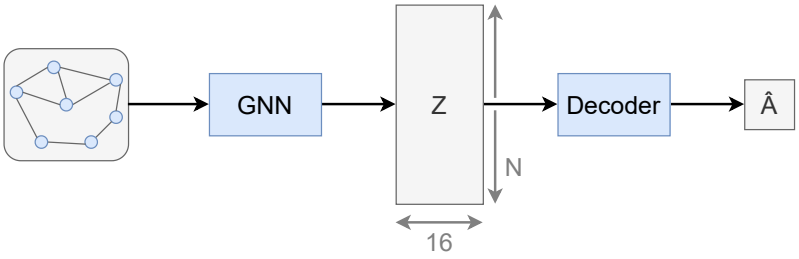


FIGURE 6.1 – Architecture du modèle GAE

Afin d'évaluer la qualité de la reconstruction de la matrice d'adjacence, l'erreur de reconstruction est calculée en utilisant la perte d'entropie croisée binaire ou *binary cross entropy loss* :

$$BCELoss = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} -(A_{ij} * \log(\hat{A}_{ij}) + (1 - A_{ij}) * \log(1 - \hat{A}_{ij})) \tag{6.2}$$

Cependant, étant donné qu'il existe beaucoup plus de liens négatifs (absence de lien entre deux noeuds) que de liens positifs (lien existant) cette métrique donne plus d'importance aux erreurs commises sur les liens négatifs. Les liens négatifs sont donc sous-échantillonnés afin de balancer cette métrique.

6.1.3 Résultats

Le modèle a été entraîné en utilisant l'optimiseur Adam [117] pendant 200 *epochs* avec un taux d'apprentissage de $0,01$. Le modèle est entrainé de manière à minimiser l'erreur de reconstruction (voir équation 6.2). De plus, à chaque *epoch*, l'erreur de reconstruction est calculée pour tous les graphes de l'ensemble de test. La figure 6.2 montre la distribution de l'erreur de reconstruction pour plusieurs *epochs*.

Quelque soit le nombre d'*epoch*, on observe que la distribution de l'erreur de reconstruction avec ou sans anomalie n'est pas significativement différente. Cela peut s'expliquer par le fait que cette métrique est calculée pour l'ensemble des liens du graphe. L'impact de l'anomalie est donc imperceptible.

L'impacte d'une anomalie de manière locale est étudiée en se concentrant sur l'erreur de reconstruction calculée uniquement pour les liens du noeud correspondant à l'AS 15169 (Google). La figure 6.3 montre la distribution de cette métrique pour plusieurs *epochs*.

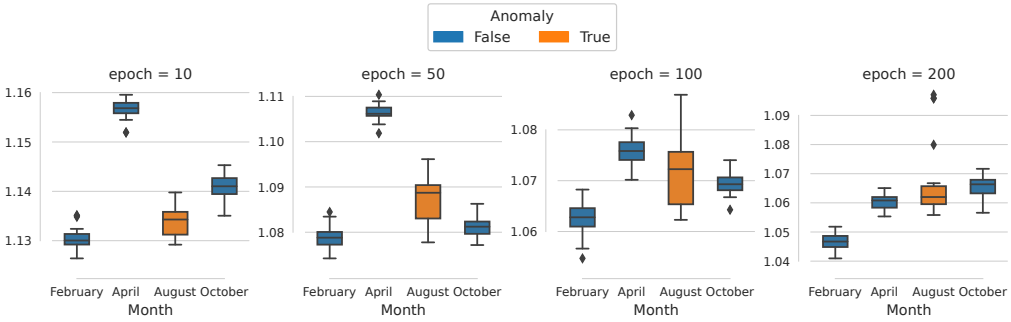


FIGURE 6.2 – Erreur de reconstruction globale

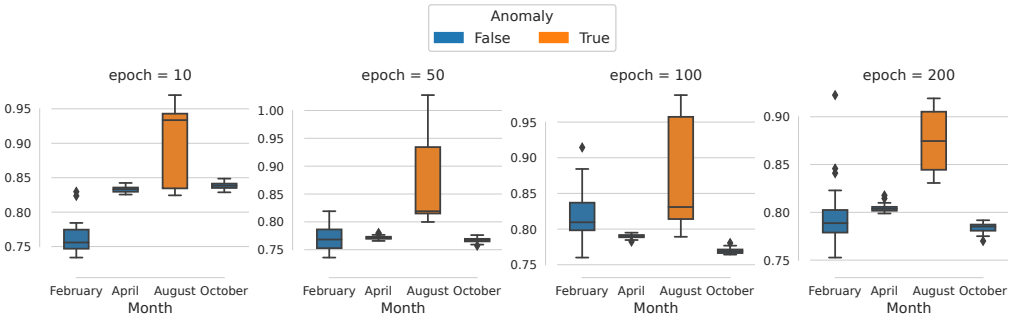


FIGURE 6.3 – Erreur de reconstruction pour l'AS 15169 (Google)

On observe que même avec un faible nombre d'*epochs* (10), la distribution de l'erreur de reconstruction pour l'AS 15169 (Google) est significativement impactée par l'anomalie. Plus, particulièrement, l'écart interquartile est bien plus important durant une anomalie. Ceci pouvant être expliqué par le fait que la période d'intérêt commence avant l'anomalie. Il existe alors une grande variation entre l'erreur de reconstruction avant et après l'anomalie.

6.1.4 Discussion

Les résultats de ce travail préliminaire démontrent que l'utilisation des GNN pour la détection d'anomalies dans des graphes BGP est envisageable. Cependant, une approche locale c'est à dire focalisé sur un AS ou un voisinage semble être à privilégier. En effet, la métrique utilisée (voir équation 6.2) ne permet pas d'identifier l'anomalie étudiée au niveau du graphe dans sa globalité. Cependant, en se concentrant sur le noeud concerné par l'anomalie, l'impact de l'anomalie est bien visible.

6.2 Le modèle BGNN

Les résultats et observations des chapitres précédents ainsi que sur le travail préliminaire de la section précédente, amènent à l'élaboration du modèle BGNN. Ce modèle a pour objectif la détection d'une anomalie BGP dans une séquence de graphes BGP. BGNN fonctionne de manière localisée et permet de détecter si un AS en particuliers est impliqué dans une anomalie. Pour cela, le modèle ne nécessite aucune étape d'extraction d'attributs, mais exploite directement une séquence de graphes BGP.

6.2.1 Construction d'un jeu de données

Le jeu de données comprend 14 enregistrements dont 7 enregistrements positifs et 7 d'enregistrements négatifs. Les enregistrements positifs sont extraits pendant l'occurrence d'une anomalie de grande échelle. Les enregistrements négatifs sont arbitrairement collectés 24h avant les enregistrements positifs. Il n'y a pas d'anomalies connues pendant les enregistrements négatifs. Pour chaque enregistrement, 1 heure de données BGP est collecté durant laquelle le graphe BGP est extrait toutes les deux minutes. Il en résulte pour chaque enregistrement dans le jeu de données, une séquence de 30 graphes BGP $G = G_1, ..., G_{30}$. Le reste de cette section détaille les événements inclus dans le jeu de données, le processus de collecte des données et d'extraction du graphe BGP.

Évènements d'anomalies BGP

Les événements d'anomalie BGP inclus dans le jeu de données s'étendent de 2004 à 2021. Premièrement, 4 événements plus anciens sont inclus (TTNet, IndoSat, TM et AWS) pour leur utilisation dans des recherches antérieures [4]. Ces évènements ont également été utilisés au chapitre 4. Deuxièmement, des événements plus récents ont également été ajoutés et reflètent la topologie moderne de BGP. Le tableau 6.2 résume tous les événements inclus dans le jeu de données. Pour chaque événement, l'AS qui est à l'origine de l'anomalie est également identifié.

Anomaly	Date	AS Number
TTNet	Dec. 24, 2004 (9 :20 UTC)	9121
IndoSat	April 2, 2014 (18 :25 UTC)	4761
TM	June 12, 2015 (8 :43 UTC)	4788
AWS	April 22, 2016 (17 :10 UTC)	200759
Google	August 25, 2017 (3 :22 UTC)	15169
ChinaTelecom	June 6, 2019 (9 :57 UTC)	21217
India	April 16, 2021 (13 :48 UTC)	55410

TABLE 6.2 – Événements d'anomalie BGP inclus dans le jeu de données

Collecte de données

Pour la collecte de données et l'extraction du graphe BGP, BML (voir chapitre 3) a été utilisé. Pour tous les enregistrements positifs, les données ont été collectées une demi-heure avant et une demi-heure après le début estimé de l'événement. Pour les enregistrements négatifs, les données ont été collectées un jour avant chaque événement pendant 1 heure. Par conséquent, 1 heure de données BGP est utilisé pour chaque enregistrement. Comme aux chapitres précédents (voir chapitre 4 et 5), les données sont collectées à partir des collecteurs `rrc04` et `rrc05` situés à Genève et à Vienne et une période d'amorçage (voir section 3.2) 10 heures à été utilisée.

Extraction du graphe BGP

Pour tous les enregistrements du jeu de données, BML a été utilisé pour extraire un instantané du graphe BGP toutes les 2 minutes. Les enregistrements d'une heure donnent alors des séquences de 30 graphes. Cependant, contrairement aux chapitres précédents, BML a été configuré pour extraire un graphe BGP pondéré (voir section 3.3.4). Chaque noeud du graphe est pondéré par le nombre de préfixes provenant de l'AS. Ainsi, si un AS annonce être l'AS d'origine de p préfixes alors le noeud correspond aura un poids p.

6.2.2 Le modèle GNN

L'entrée du modèle est une séquence de graphes BGP $G = G_1, ..., G_{30}$ à partir de laquelle il vise à prédire $Y = 1$ si une anomalie est détectée et $Y = 0$ sinon. Cependant, les algorithmes d'apprentissage automatique et les réseaux de neurones classiques sont conçus pour des données tabulaires et ne peuvent donc pas exploiter toutes les informations contenues dans une représentation sous forme de graphe. Récemment, le domaine des *Graph Neural Networks* (GNN) a émergé pour surmonter ce problème en proposant des réseaux de neurones conçus pour prendre en entrée un graphe (voir section 2.5.1).

Architecture

L'architecture du modèle est décrite sur la figure 6.4. Premièrement, chaque graphe G_i est donné en entrée à un réseau GNN composé de k couches où chaque couche contient 8 blocs GNN. L'impact de l'hyperparamètre k sera discuté dans la section 6.2.3. Le bloc GNN est un Graph Convolutional Networks [77]. Pour chaque graphe G_i, un plongement de dimension 1×8 est produit pour chaque noeud (8 étant le nombre de blocs GNN par couche). Cependant, seul le plongement du noeud qui est à l'origine de l'anomalie (voir table 6.2) est conservé. À partir de la séquence de graphes, cette opération résulte en une matrice de plongements des noeuds de dimension 30×8. Cette matrice est ensuite aplatie pour obtenir un vecteur de dimension 240×1. Enfin, un perceptron multicouche ou *multilayer perceptron* (MLP) est utilisé pour produire la sortie Y sur la base du vecteur 240×1.

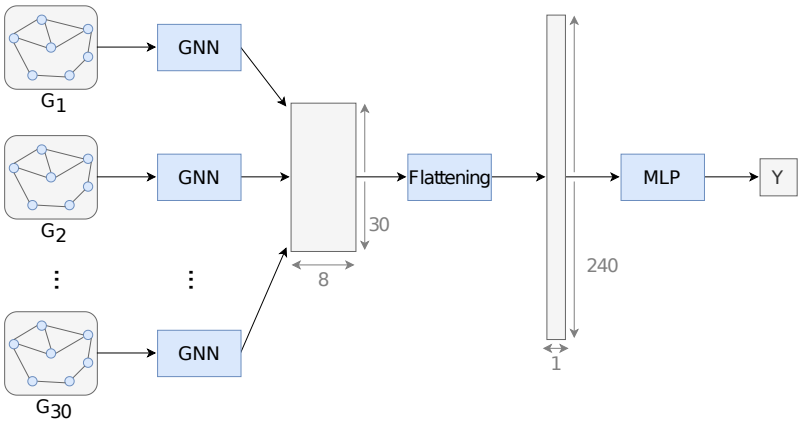


FIGURE 6.4 – Architecture du modèle

Pour chacune des expériences, le modèle a été entraîné pendant 50 *epochs* avec un taux d'apprentissage de $0,001$ en utilisant l'optimiseur Adam [117]. L'initialisation par défaut des couches GNN a été modifiée par l'initialisation normale de Kaiming [118] car une meilleure convergence a été observée en utilisant cette technique.

Métriques

L'*accuracy* et le *F1 score* sont utilisés pour évaluer les performances du modèle. Ces métriques ont été introduites dans la section 5.3.2. Cependant, la sortie d'un réseau de neurones n'étant pas une valeur binaire mais une valeur comprise dans la plage $[0, 1]$, un seuil doit être appliqué pour obtenir la classe finale. Ce seuil a donc un impact sur les métriques mentionnées ci-dessus. Par exemple, un seuil bas permettra de classer la plupart des enregistrements comme positifs, ce qui se traduira par une valeur de *recall* élevée mais de *precision* faible. L'aire sous la courbe ou *area under the curve* (AUC), qui est une métrique dérivée de la courbe ROC (*receiver operating characteristic*), évalue la performance d'un modèle à différents seuils. L'AUC représente la mesure de la séparabilité des classes de sortie où une valeur de 1 indique une séparabilité parfaite et une valeur de $0,5$ ou moins indique aucune séparabilité.

6.2.3 Résultat

Les expériences ont été implémentées à l'aide des bibliothèques *PyTorch Geometric* [116] et *scikit-learn*. Les implémentations sont disponibles en ligne [2]. Les performances du modèle sont évaluées à l'aide d'un schéma de validation croisée *leave-one-out* (LOOCV) où chaque événement est utilisé de manière itérative pour tester le modèle et les autres sont utilisés pour l'entraînement. Lorsqu'un événement est utilisé pour le test, l'enregistrement positif et l'enregistrement négatif associés à cet événement sont utilisés comme ensemble de test. Le modèle produit alors deux sorties. Lorsque les 7 événements ont été utilisés pour tester le modèle, un vecteur de sortie de dimension 14 est alors construit en

2. https ://github.com/KevinHoarau/BGNN

concaténant toutes les sorties. Enfin, le vecteur de sortie est utilisé pour calculer les valeurs d'*accuracy*, de *F1 score* et d'AUC. La convergence d'un réseau de neurones pouvant varier en fonction de l'initialisation de ses poids internes, il est important d'évaluer la stabilité du modèle sur plusieurs exécutions. Le modèle est entraîné plusieurs fois (au moins 30 fois avec des graines différentes) pour calculer et réduire l'écart-type des mesures de performance.

GNN layers	**Accuracy**		**F1 score**		**AUC**	
	Mean	Std dev	Mean	Std dev	Mean	Std dev
1	0.89	0.05	0.88	0.05	0.86	0.03
2	0.90	0.03	0.89	0.04	0.87	0.05
4	**0.96**	**0.05**	**0.96**	**0.05**	**0.99**	**0.03**
8	0.91	0.08	0.91	0.05	0.91	0.08
16	0.78	0.11	0.72	0.20	0.79	0.09

TABLE 6.3 – Résultats sur l'ensemble de test

Le tableau 6.3 montre les performances du modèle pour plusieurs valeurs du nombre k de couches GNN variant de 1 à 16. Il est visible que la meilleure performance est obtenue en utilisant 4 de couches GNN avec une valeur d'*accuracy* de $0,96$, de *F1 score* de $0,96$ et d'AUC de $0,99$. Ces résultats démontrent donc bien l'intérêt de modèles basés sur des GNN pour la détection d'anomalies BGP.

Il convient de rappeler que le nombre de couches GNN correspond au nombre de sauts pendant le processus de passage de messages (voir section 2.5.1). D'une part, l'hypothèse peut être émise qu'avec moins de 4 sauts, il n'y a pas assez de données agrégées dans le voisinage d'un noeud. D'autre part, les valeurs supérieures à 4 ajoutent du bruit au processus. Avec le meilleur modèle, il est à noter que l'écart-type des différentes métriques est inférieur à $0,06$ sur plusieurs exécutions, ce qui montre la stabilité du modèle.

Visualisation des plongements

Une fois le modèle entraîné, sa partie GNN peut être utilisée pour produire des plongements des noeuds d'un graphe d'entrée. Pour chaque noeud, ce plongement est un vecteur dans un espace de dimension 8. Cette section étudie comment les plongements successifs d'un noeud évoluent avec et sans anomalie sur ce noeud. Une approche courante pour analyser les plongements consiste à utiliser une technique de réduction de dimension qui projette des vecteurs d'un espace de grande dimension vers un espace de dimension inférieure. L'analyse en composantes principales (ACP) ou *principal component analysis* est largement utilisée à cette fin en ne conservant que les n premières composantes principales ou *principal components* (PC) des données. Ici, on ne garde que le premier PC pour obtenir une projection 1d des plongements. Les résultats sont une série temporelle représentant l'évolution du plongement d'un noeud.

La figure 6.5 montre la projection 1d des événements où les lignes bleues correspondent aux 30 plongements du noeud dans l'enregistrement négatif (un jour avant

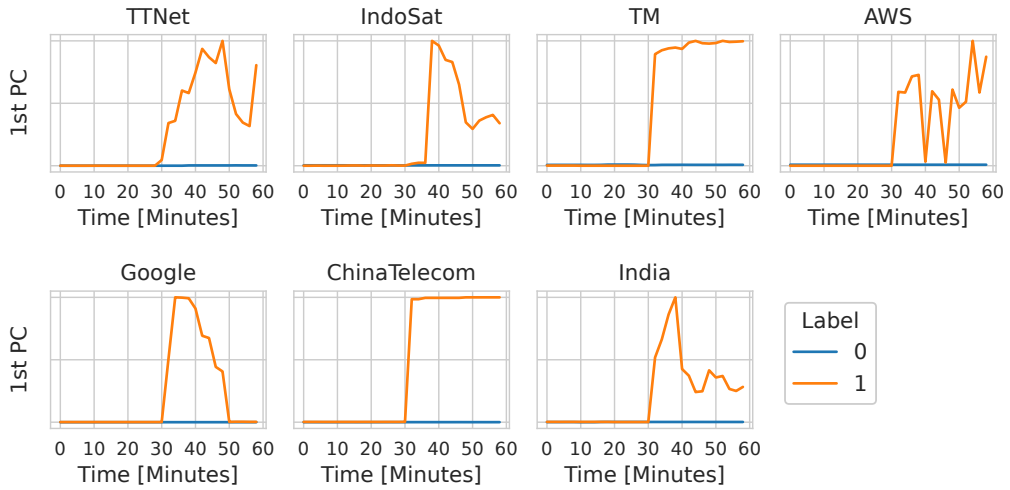


FIGURE 6.5 – Plongement d'un AS avec et sans anomalie

l'anomalie) et les lignes oranges aux 30 plongements du noeud dans l'enregistrement positif (pendant l'anomalie). Pour tous les événements, il est possible d'observer que les plongements changent radicalement lorsqu'une anomalie se produit sur un noeud. Il est également visible qu'avant l'anomalie, les plongements de l'enregistrement positif sont proches des plongements de l'enregistrement négatif. Ainsi, les plongements ne semblent pas varier significativement dans un intervalle de 24 heures. En conclusion, ces résultats tendent à montrer la stabilité du plongement d'un noeud dans des circonstances normales et sa perturbation pendant une anomalie. C'est un comportement souhaité, car ce signal peut être exploité par le classificateur subséquent.

Séparabilité

Le processus de classification est pris en charge par la partie MLP du modèle. En entrée, ce classificateur reçoit un vecteur de dimension 240 correspondant à l'aplatissement de la séquence des plongements d'un noeud. Pour chaque enregistrement dans le jeu de données, ce vecteur dans un espace de dimension 240 est créé par la partie GNN du modèle. Cette section analyse la séparation entre les vecteurs correspondant aux enregistrements négatifs et les vecteurs correspondant aux enregistrements positifs dans cet espace de dimension 240 . Comme dans la section précédente, une ACP est utilisée pour projeter ces vecteurs de haute dimension dans un espace 2d en ne gardant que les 2 premières composantes principales.

La figure 6.6 montre la projection 2d des vecteurs de plongements où les points bleus correspondent aux enregistrements négatifs et les points orange aux enregistrements positifs. Ces vecteurs sont obtenus à l'aide du modèle entraîné en utilisant tous les événements sauf *India* qui est utilisée comme ensemble de test. Il est visible qu'une ligne qui sépare presque tous les enregistrements négatifs des enregistrements positifs peut être tracée. Ainsi, les deux classes semblent être séparables même dans cette projection 2d.

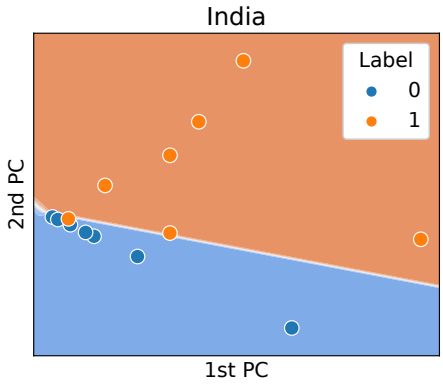


FIGURE 6.6 – Frontière de décision du classifieur

Pour avoir une intuition sur la forme de la frontière de décision capturée par le classifieur, il est possible de définir une grille de points dans l'espace 2d et appliquer le classifieur à ces points pour connaître sa sortie. En sélectionnant un grand nombre de points, une carte de la frontière de décision du modèle peut être établie. Cependant, le classificateur ne peut pas prendre directement en entrée un point issu de cet espace 2d, il faut donc projeter ce point dans l'espace original à 240 dimensions. Cette approche de visualisation de la frontière de décision d'un modèle dans un espace de grande dimension est un sujet de recherche à part entière et sort du cadre de ce travail. Cependant, il est possible d'obtenir une cartographie approximative de la frontière de décision du modèle en tirant parti de l'opération inverse de l'ACP pour projeter un point 2d dans l'espace original. Cette approche a été utilisée pour coloriser l'arrière-plan de la figure 6.6. La ligne blanche correspond à la frontière de décision entre les deux classes apprises par le classificateur. Dans cet exemple, le classificateur classe correctement 13 enregistrements sur 14.

Temps de détection

Une des caractéristiques critiques d'un système de détection d'anomalies est son temps de réponse, surtout s'il doit fonctionner en temps réel. Cependant, peu de travaux dans la littérature abordent cette question en fournissant des informations sur cette caractéristique de leur modèle. Afin de combler cette lacune, le temps de réponse est défini comme l'intervalle entre l'heure de début connue d'une anomalie et le moment où cette anomalie peut être détectée.

Le temps de réponse du modèle est évalué en notant t_0 le temps de début connu d'une anomalie et en utilisant les données collectées dans l'intervalle $[t_0 - 30, t_0 + 2]$ pour entraîner et tester le modèle. Cette fenêtre temporelle est progressivement agrandie de 2 minutes. Ainsi, l'exécution suivante se fait sur l'intervalle $[t_0 - 30, t_0 + 4]$ et la dernière sur l'intervalle $[t_0 - 30, t_0 + 30]$ qui correspond au modèle initial.

La figure 6.7 montre le résultat de cette évaluation. Il est visible d'après ces résultats que 4 minutes c'est-à-dire une séquence de 2 graphes après l'anomalie sont suffisantes pour atteindre des valeurs d'*accuracy*, de *F1 score* et d'AUC supérieurs à $0,85$. Ces valeurs

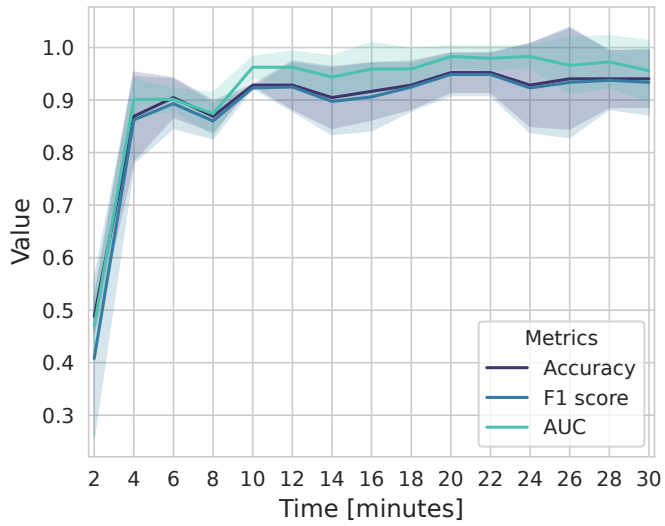


FIGURE 6.7 – Performance du modèle en fonction du temps après l'anomalie

atteignent $0,90$ après 6 minutes (c'est-à-dire une séquence de 3 graphes) et sont autour de $0,96$ après 18 minutes (c'est-à-dire une séquence de 9 graphes). Ces résultats sont donc prometteurs pour une détection d'anomalies BGP en temps réel.

6.3 Conclusion

Ce chapitre a tout d'abord présenté un travail préliminaire qui a permis de valider l'approche consistant à adopter une solution localisée et basée sur un GNN pour la détection d'anomalies BGP. BGNN qui, à notre connaissance, est la première proposition d'un GNN pour la détection d'anomalies BGP a ensuite été présenté et évalué. Ce modèle consiste en un GNN à k couches à partir duquel le plongement d'un noeud est extrait et classé comme normal ou anormal par un MLP.

L'impact du nombre de couches k a d'abord été évalué c'est-à-dire le nombre de sauts pendant le processus de passage de messages (voir section 2.5.1). Il a été montré que la précision augmente avec k et qu'elle atteint un maximum de 96% pour $k = 4$ et diminue pour les valeurs supérieures. Cela suggère que les interactions locales transportent suffisamment d'informations pour détecter une anomalie, ce qui réduit la complexité du modèle. La première composante principale du plongement d'un noeud a ensuite été étudiée et a montré une forte variation quelques minutes après le début attendu de l'anomalie. L'analyse de l'espace de plongement en utilisant une projection 2d a révélé une frontière de décision simple. Enfin, l'impact du temps de réponse de BGNN sur ses performances a été évalué. Un temps de détection de 6 minutes est alors suffisant pour obtenir un *accuracy* de 90% et 18 minutes sont suffisantes pour obtenir un *accuracy* de 96%.

En conclusion, ce travail a démontré que même un modèle relativement simple basé sur un GNN est capable de fournir des performances intéressantes. Cela ouvre donc la voie à l'approfondissement de cette approche. Notamment, une perspective majeure est la détection des anomalies BGP de plus petites échelles étant donné que les approches actuelles basées des attributs statistiques ne permettent pas de détecter ce type d'anomalies et que l'utilisation des graphes semble être une piste à privilégier (voir chapitre 4).

Chapitre 7

Conclusion et perspectives

Sommaire

Ce dernier chapitre résume les principales contributions de cette thèse à savoir : l'élaboration d'un outil de construction de jeu de données BGP, l'étude de la pertinence des attributs issus des graphes BGP pour la détection d'anomalies, l'exploitation de la dimension temporelle des données BGP et la proposition d'un modèle de *Graph Neural Networks* (GNN) ne nécessitant pas d'étape préalable d'extraction d'attributs. Ensuite, quelques perspectives de travaux futurs sont présentées telles que l'identification du noeud à l'origine d'une attaque, la prédiction de l'impact d'une anomalie sur un noeud ou de nouvelles évaluations des modèles proposés.

7.1 Conclusion

Cette thèse s'intéresse aux méthodes d'analyse de données pour la détection d'anomalies dans les données issues des réseaux, de leurs protocoles et applications. Elle vise plus particulièrement à déterminer s'il est pertinent de considérer la représentation de ces données sous leur forme naturelle basée sur les graphes plutôt que sous forme de données tabulaires comme cela est plus souvent réalisé. Le *Border Gateway Protolcol* (BGP) en raison du volume et la dimension importante des graphes qu'il génère est cadre applicatif retenu dans cette thèse. Dans ce contexte, une méthodologie incrémentale a été adoptée afin d'évaluer tout d'abord la plus-value des représentations sous forme de graphes avec des techniques d'apprentissage automatique classiques puis en explorant des solutions basées sur les avancées récentes dans le domaine de l'apprentissage profond.

Avant d'attaquer la problématique majeure de cette thèse, un travail préalable a été nécessaire pour apporter une solution à l'absence de jeux de données de référence dans le domaine. Un outil nommé BML a été élaboré afin d'automatiser la collecte, l'agrégation, le stockage et la transformation des données BGP en vue de l'application d'algorithmes d'apprentissage automatiques. Des transformations issues de la littérature ont été implémentées dans l'outil telles que l'extraction d'attributs statistiques, l'extraction du graphe BGP et d'attributs sur ce graphe. Afin d'encourager son adoption par la communauté scientifique, le fonctionnement de BML peut être ajusté pour correspondre à des besoins spécifiques (*e.g.* points de collectes utilisés, ajout de fonctions de transformation, etc.). L'outil a par ailleurs servi de base pour les autres contributions de cette thèse ainsi que pour les travaux d'autres équipes [109].

La première contribution de cette thèse se penche sur l'évaluation de la pertinence des attributs issus des graphes BGP pour la détection d'anomalie. Ce type d'attributs est mis en perspectives avec les attributs statistiques qui sont généralement adoptés dans la littérature. Cette comparaison est faite tout d'abord à l'aide d'outils statistiques puis avec des algorithmes d'apprentissage automatique classique et permet de conclure sur l'intérêt des attributs issus des graphes BGP en fonction du type d'anomalies considéré. Ainsi, ils permettent d'améliorer de 18% l'*accuracy* pour des anomalies de petite échelle tout en offrant des performances conformes à l'état de l'art pour des anomalies de grande échelle.

Dans une seconde contribution, un modèle plus avancé est proposé afin d'intégrer la composante temporelle dans l'algorithme d'apprentissage automatique. Il s'agit d'un réseau de neurones récurrent (RNN) étant capable de consommer une série temporelle d'attributs issus d'une séquence de graphe BGP. Sur un jeu de données comprenant des anomalies BGP de petite échelle il apparaît que les performances de ce modèle sont impactées par la perte d'information provenant de l'étape d'extraction d'attributs.

Dans la dernière contribution de cette thèse, une solution est proposée afin de se libérer de l'étape d'extraction d'attributs à partir des graphes BGP. Pour cela, des modèles basés sur les avancés récentes dans le domaine des *Graph Neural Networks* (GNN) sont introduits. Ce type d'architecture permet de construire des réseaux de neurones capables

d'ingérer des graphes en entrée. Tout d'abord, un travail préliminaire a permis de valider la possibilité d'appliquer ce type de modèles à des graphes aussi massifs que ceux issus de BGP. De plus, sur un évènement de grande échelle, il est apparu que ce type de modèle est en mesure de capturer l'impact d'une anomalie sur un AS. Par la suite, un modèle exploitant cette observation a été construit. Le modèle BGNN est composé d'un GNN permettant d'ingérer en entrée une séquence de graphes BGP et de produire en sortie une séquence de plongement pour chaque AS. Un classifieur consomme ensuite cette séquence pour détecter une anomalie au niveau d'un AS spécifiquement. Ce modèle est capable de détecter une anomalie de grande échelle (sur l'AS à l'origine de l'anomalie) 6 minutes après son occurrence avec une valeur d'*accuracy* de 90%. Cette valeur atteint 96% après 18 minutes.

7.2 Perspectives

Pour conclure ce manuscrit, ci-dessous sont présentées quelques perspectives de travaux futurs.

7.2.1 Identification des AS malicieux

La dernière contribution de cette thèse montre qu'en se basant sur un GNN il est possible de détecter une anomalie avec une granularité fine. Pour les anomalies considérées dans ce travail, seul l'AS à l'origine de l'évènement est impliqué dans l'anomalie. Cependant, dans le contexte d'une attaque comme cela peut être le cas lors d'un détournement de préfixe, deux AS sont impliqués, l'attaquant et la victime. Sur ce type d'anomalie, une perspective est d'évaluer la capacité d'un GNN à identifier l'AS malicieux lors d'une attaque. Après la détection d'une attaque, un tel GNN pourrait, comme pour BGNN, être appliqué à la séquence des graphes BGP correspondantes afin de construire une séquence de plongements pour chaque noeud du graphe. Cette séquence serait alors consommée par un classifieur pour classer chaque noeud comme attaquant, victime ou non impliqué. Pour entraîner un tel modèle, il est nécessaire de disposer d'un jeu de données d'anomalies pour lesquels les AS attaquants et victimes sont identifiés comme cela est le cas pour les détournements de préfixes.

7.2.2 Prédiction de l'impact d'une anomalie

De manière prospective, une question intéressante est de savoir si une anomalie survient sur un AS donnée quel va être son impact à l'échelle du réseau BGP. Un modèle répondant à cette question donnerait pour une anomalie sur un AS donnée une mesure de l'impact de cette dernière pour chaque noeud du graphe. Par exemple, dans le cadre d'un détournement de préfixe, un tel modèle pourrait prédire une valeur binaire déterminant si la route détournée sera adoptée par le noeud. Cependant, en pratique il est impossible de collecter des informations depuis tous les AS du réseau BGP afin de construire un jeu de données permettant d'entraîner un modèle de ce type. Avec un modèle tel que le Graph Auto-Encodeur (*c.f.* section 6.1.2) il est toutefois possible de quantifier l'impact d'une anomalie sur un noeud à partir du changement qu'elle engendre sur le plongement du noeud. Il devient possible de construire un jeu de données d'anomalies BGP pour lesquelles chaque AS dispose d'une valeur de l'impact de cette dernière. Un GNN prenant en entrée une séquence de graphes pour lequel un AS est identifié comme l'origine de l'anomalie pourrait être entraîné pour prédire la valeur de l'impact pour tous les autres noeuds.

7.2.3 Évaluation sur un plus grand jeu de données

L'entraînement des modèles d'apprentissage automatique pour la détection des anomalies BGP est pénalisé par la taille réduite des jeux de données utilisés. La disponibilité d'un jeu de données de grande dimension permettrait l'entraînement de modèles

plus complexes tout en rendant l'évaluation plus robuste. La section 3.4.2 présente la construction d'un tel jeu de données d'anomalies BGP. Ce jeu de données est composé de 150 détournements d'origines, 150 fuites de routes et 150 défaillances matérielles. Un tel jeu de données de grande taille pourrait devenir une référence permettant la comparaison des travaux dans le domaine. Un travail en cours consiste alors à appliquer les modèles proposés dans cette thèse sur ce jeu de données et à le rendre publiquement disponible. Les résultats préliminaires sur les attributs issus des graphes BGP montrent une séparabilité plus importante par rapport aux attributs statistiques avec un coefficient de silhouette au minimum 10 fois plus important pour les trois types d'anomalies.

7.2.4 Évaluation de BGNN sur des anomalies de petites échelles

Une extension directe du travail présenté dans la dernière contribution de cette thèse est l'application du modèle BGNN à des anomalies de plus petite échelle. Ce travail est d'autant plus pertinent qu'il est apparu une amélioration significative des performances sur ce type d'anomalies lors de l'utilisation d'attributs issus des graphes BGP. La raison pour laquelle cela n'a pas été abordé dans cette thèse est que l'entraînement d'un modèle sur ce type d'anomalie est plus complexe car elles sont bien plus subtiles. Pour aborder ce problème, une solution envisageable est d'augmenter le nombre d'anomalies présent dans le jeu de donnée afin que le modèle puisse capturer les motifs spécifiques à ce type d'anomalies. Il est également possible de choisir plus soigneusement le bloc GNN utilisé à la place du GCN [77]. Il est également possible, d'enrichir la représentation du graphe BGP en ajoutant des informations au niveau des noeuds tels que le nombre de routes annoncées par l'AS ou le nombre de préfixes lui appartenant.

7.2.5 Classification des anomalies

Cette thèse se concentre sur la détection des anomalies BGP et présente les performances de plusieurs d'algorithme d'apprentissage automatique pour différent type d'anomalies. Cependant, dans le monde réel toutes ces anomalies existent. Ainsi, une fois une anomalie détectée, il est intéressant de pouvoir déterminer son type afin par exemple de lui affecter une valeur de criticité. Une perspective serait donc d'évaluer la capacité des modèles proposés à classifier une anomalie préalablement détectée. Les jeux de données et modèles présentés dans cette thèse peuvent être utilisés pour cette évaluation en considérant un problème de classification multiclasse.

Liste des publications

[1] K. HOARAU, P. U. TOURNOUX et T. RAZAFINDRALAMBO, « BML : an efficient and versatile tool for BGP dataset collection, » *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, p. 1-6, 2021.

[2] K. HOARAU, P.-U. TOURNOUX et T. RAZAFINDRALAMBO, « Suitability of Graph Representation for BGP Anomaly Detection, » *2021 IEEE 46th Conference on Local Computer Networks (LCN 2021)*, 2021.

[3] K. HOARAU, P. U. TOURNOUX et T. RAZAFINDRALAMBO, « Detecting forged AS paths from BGP graph features using Recurrent Neural Networks, » *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, IEEE, p. 735-736, 2022.

[4] K. HOARAU, P. TOURNOUX et T. RAZAFINDRALAMBO, « BML : l'outil pour l'analyse de données BGP, » *CORES 2022–7ème Rencontres Francophones sur la Conception de Protocoles, l'Évaluation de Performance et l'Expérimentation des Réseaux de Communication*, 2022.

[5] K. HOARAU, P. U. TOURNOUX et T. RAZAFINDRALAMBO, « BGNN : Detection of BGP Anomalies Using Graph Neural Networks, » *IEEE Symposium on Computers and Communications (ISCC)*, 2022.

Bibliographie

[1] A. KRIZHEVSKY, I. SUTSKEVER et G. E. HINTON, « Imagenet classification with deep convolutional neural networks, » *Advances in neural information processing systems*, t. 25, 2012.

[2] T. MIKOLOV, K. CHEN, G. CORRADO et J. DEAN, « Efficient estimation of word representations in vector space, » *arXiv preprint arXiv :1301.3781*, 2013.

[3] M. M. BRONSTEIN, J. BRUNA, Y. LECUN, A. SZLAM et P. VANDERGHEYNST, « Geometric deep learning : going beyond euclidean data, » *IEEE Signal Processing Magazine*, t. 34, p. 18-42, 2017.

[4] O. R. SANCHEZ, S. FERLIN, C. PELSSER et R. BUSH, « Comparing Machine Learning Algorithms for BGP Anomaly Detection using Graph Features, » *Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks*, ACM, p. 35-41, 2019.

[5] D. F. RUEDA, E. CALLE et J. L. MARZO, « Robustness comparison of 15 real telecommunication networks : Structural and centrality measurements, » *Journal of Network and Systems Management*, t. 25, p. 269-289, 2017.

[6] W. L. HAMILTON, R. YING et J. LESKOVEC, « Representation learning on graphs : Methods and applications, » *arXiv preprint arXiv :1709.05584*, 2017.

[7] B. PEROZZI, R. AL-RFOU et S. SKIENA, « Deepwalk : Online learning of social representations, » *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, p. 701-710, 2014.

[8] J. YOSINSKI, J. CLUNE, Y. BENGIO et H. LIPSON, « How transferable are features in deep neural networks ? » *Advances in neural information processing systems*, t. 27, 2014.

[9] Y. REKHTER, S. HARES et T. LI, *A Border Gateway Protocol 4 (BGP-4)*, RFC 4271, 2006,
DOI : 10.17487/RFC4271.

[10] Q. VOHRA et E. CHEN, *BGP Support for Four-Octet Autonomous System (AS) Number Space*, RFC 6793, 2012,
DOI : 10.17487/RFC6793.

[11] V. FULLER et T. LI, *Classless Inter-domain Routing (CIDR) : The Internet Address Assignment and Aggregation Plan*, RFC 4632, 2006,
DOI : 10.17487/RFC4632.

[12] L. GAO, « On inferring autonomous system relationships in the Internet, » *IEEE/ACM Transactions on networking*, t. 9, p. 733-745, 2001.

[13] *Transmission Control Protocol*, RFC 793, 1981, DOI : 10.17487/RFC0793.

[14] Q. CHEN, H. CHANG, R. GOVINDAN et S. JAMIN, « The origin of power laws in Internet topologies revisited, » *Proceedings. twenty-first annual joint conference of the ieee computer and communications societies*, IEEE, p. 608-617, 2002.

[15] E. GREGORI, A. IMPROTA, L. LENZINI, L. ROSSI et L. SANI, « On the incompleteness of the AS-level graph : a novel methodology for BGP route collector placement, » *Proceedings of the 2012 Internet Measurement Conference*, p. 253-264, 2012.

[16] B. AL-MUSAWI, P. BRANCH et G. ARMITAGE, « BGP anomaly detection techniques : A survey, » *IEEE Communications Surveys & Tutorials*, t. 19, p. 377-396, 2016.

[17] X. HU et Z. M. MAO, « Accurate real-time identification of IP prefix hijacking, » *2007 IEEE Symposium on Security and Privacy (SP'07)*, IEEE, p. 3-17, 2007.

[18] H. BALLANI, P. FRANCIS et X. ZHANG, « A study of prefix hijacking and interception in the Internet, » *ACM SIGCOMM Computer Communication Review*, t. 37, p. 265-276, 2007.

[19] X. SHI, Y. XIANG, Z. WANG, X. YIN et J. WU, « Detecting prefix hijackings in the internet with argus, » *Proceedings of the 2012 Internet Measurement Conference*, p. 15-28, 2012.

[20] K. SRIRAM, D. MONTGOMERY, D. R. MCPHERSON, E. OSTERWEIL et B. DICKSON, *Problem Definition and Classification of BGP Route Leaks*, RFC 7908, 2016, DOI : 10.17487/RFC7908.

[21] X. ZHAO, D. PEI, L. WANG, D. MASSEY, A. MANKIN, S. F. WU et L. ZHANG, « An analysis of BGP multiple origin AS (MOAS) conflicts, » *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, p. 31-35, 2001.

[22] P.-A. VERVIER, O. THONNARD et M. DACIER, « Mind Your Blocks : On the Stealthiness of Malicious BGP Hijacks., » *NDSS*, 2015.

[23] C. MCARTHUR et M. GUIRGUIS, « Stealthy ip prefix hijacking : don't bite off more than you can chew, » *GLOBECOM 2009-2009 IEEE Global Telecommunications Conference*, IEEE, p. 1-6, 2009.

[24] T. BILSKI, « Disaster's impact on Internet performance–case study, » *International Conference on Computer Networks*, Springer, p. 210-217, 2009.

[25] J. H. COWIE, A. T. OGIELSKI, B PREMORE, E. A. SMITH et T. UNDERWOOD, « Impact of the 2003 blackouts on Internet communications, » *Preliminary Report, Renesys Corporation (updated March 1, 2004)*, 2003.

[26] M. LAD, X. ZHAO, B. ZHANG, D. MASSEY et L. ZHANG, « Analysis of BGP update surge during slammer worm attack, » *International Workshop on Distributed Computing*, Springer, p. 66-79, 2003.

[27] L. WANG, X. ZHAO, D. PEI, R. BUSH, D. MASSEY, A. MANKIN, S. F. WU et L. ZHANG, « Observation and analysis of BGP behavior under stress, » *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, p. 183-195, 2002.

[28] A. TOONK, *BGP leak causing Internet outages in Japan and beyond. | BGPmon*, URL : https://bgpmon.net/bgp-leak-causing-internet-outages-in-japan-and-beyond/.

[29] *YouTube Hijacking : A RIPE NCC RIS case study*, URL : https://www.ripe.net/publications/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study.

[30] A. SHAW, *Spam ? Not Spam ? Tracking a hijacked Spamhaus IP*, 2013, URL : https://greenhost.nl/blog/2013/03/21/spam-not-spam-tracking-a-hijacked-spamhaus-ip/.

[31] A. RAMACHANDRAN et N. FEAMSTER, « Understanding the network-level behavior of spammers, » *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, p. 291-302, 2006.

[32] A. DAINOTTI, *HIJACKS : Detecting and Characterizing Internet Traffic Interception based on BGP Hijacking*.

[33] J. COWIE, *The New Threat : Targeted Internet Traffic Misdirection*, URL : https://blogs.oracle.com/internetintelligence/the-new-threat%3A-targeted-internet-traffic-misdirection.

[34] K. BUTLER, T. R. FARLEY, P. MCDANIEL et J. REXFORD, « A survey of BGP security issues and solutions, » *Proceedings of the IEEE*, t. 98, p. 100-122, 2009.

[35] A. MITSEVA, A. PANCHENKO et T. ENGEL, « The state of affairs in BGP security : A survey of attacks and defenses, » *Computer Communications*, t. 124, p. 45-60, 2018.

[36] P. SERMPEZIS, V. KOTRONIS, A. DAINOTTI et X. DIMITROPOULOS, « A survey among network operators on BGP prefix hijacking, » *ACM SIGCOMM Computer Communication Review*, t. 48, p. 64-69, 2018.

[37] M. LEPINSKI et S. KENT, *An Infrastructure to Support Secure Internet Routing*, RFC 6480, 2012, DOI : 10.17487/RFC6480.

[38] *NIST RPKI Monitor*, URL : https://rpki-monitor.antd.nist.gov/.

[39] M. LEPINSKI et K. SRIRAM, *BGPsec Protocol Specification*, RFC 8205, 2017, DOI : 10.17487/RFC8205.

[40] K. SRIRAM et A. AZIMOV, « Methods for Detection and Mitigation of BGP Route Leaks, » Internet Engineering Task Force, Internet-Draft, 2021, 10 p., URL : https://datatracker.ietf.org/doc/html/draft-ietf-grow-route-leak-detection-mitigation-05.

[41] *Routing Information Service (RIS)*,
URL : https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris/routing-information-service-ris.

[42] *Routeviews – University of Oregon Route Views Project*,
URL : http://www.routeviews.org/routeviews/.

[43] L. BLUNK, C. LABOVITZ et M. KARIR, *Multi-Threaded Routing Toolkit (MRT) Routing Information Export Format*, RFC 6396, 2011,
DOI : 10.17487/RFC6396.

[44] *RIPE-NCC/bgpdump*,
URL : https://github.com/RIPE-NCC/bgpdump.

[45] H. YAN, R. OLIVEIRA, K. BURNETT, D. MATTHEWS, L. ZHANG et D. MASSEY, « BGPmon : A real-time, scalable, extensible monitoring system, » *2009 Cybersecurity Applications & Technology Conference for Homeland Security*, IEEE, p. 212-223, 2009.

[46] *Looking Glass - Hurricane Electric (AS6939)*,
URL : https://lg.he.net/.

[47] H. ALSHAMRANI et B. GHITA, « IP prefix hijack detection using BGP connectivity monitoring, » *2016 IEEE 17th International Conference on High Performance Switching and Routing (HPSR)*, IEEE, p. 35-41, 2016.

[48] M. LAD, D. MASSEY, D. PEI, Y. WU, B. ZHANG et L. ZHANG, « PHAS : A Prefix Hijack Alert System., » *USENIX Security symposium*, p. 3, 2006.

[49] P. SERMPEZIS, V. KOTRONIS, P. GIGIS, X. DIMITROPOULOS, D. CICALESE, A. KING et A. DAINOTTI, « ARTEMIS : Neutralizing BGP hijacking within a minute, » *IEEE/ACM Transactions on Networking*, t. 26, p. 2471-2486, 2018.

[50] J. SCHLAMP, R. HOLZ, Q. JACQUEMART, G. CARLE et E. W. BIERSACK, « HEAP : reliable assessment of BGP hijacking attacks, » *IEEE Journal on Selected Areas in Communications*, t. 34, p. 1849-1861, 2016.

[51] J. LI, T. EHRENKRANZ et P. ELLIOTT, « Buddyguard : A buddy system for fast and reliable detection of IP prefix anomalies, » *2012 20th IEEE International Conference on Network Protocols (ICNP)*, IEEE, p. 1-10, 2012.

[52] Z. ZHANG, Y. ZHANG, Y. C. HU, Z. M. MAO et R. BUSH, « iSPY : Detecting IP prefix hijacking on my own, » *Proceedings of the ACM SIGCOMM 2008 conference on Data Communication*, p. 327-338, 2008.

[53] C. ZHENG, L. JI, D. PEI, J. WANG et P. FRANCIS, « A light-weight distributed scheme for detecting IP prefix hijacks in real-time, » *ACM SIGCOMM Computer Communication Review*, t. 37, p. 277-288, 2007.

[54] *PlanetLab | An open platform for developing, deploying, and accessing planetary-scale services*,
URL : https://planetlab.cs.princeton.edu/.

[55] M. LUCKIE, B. HUFFAKER, A. DHAMDHERE, V. GIOTSAS et K. CLAFFY, « AS relationships, customer cones, and validation, » *Proceedings of the 2013 conference on Internet measurement conference*, p. 243-256, 2013.

[56] L. GAO et J. REXFORD, « Stable Internet routing without global coordination, » *IEEE/ACM Transactions on networking*, t. 9, p. 681-692, 2001.

[57] N. M. AL-ROUSAN et L. TRAJKOVIĆ, « Machine learning models for classification of BGP anomalies, » *2012 IEEE 13th International Conference on High Performance Switching and Routing*, IEEE, p. 103-108, 2012.

[58] M. ĆOSOVIĆ, S. OBRADOVIÓ et L. TRAJKOVIĆ, « Classifying anomalous events in BGP datasets, » *2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, IEEE, p. 1-4, 2016.

[59] M. ĆOSOVIĆ, S. OBRADOVIĆ et L. TRAJKOVIĆ, « Performance evaluation of BGP anomaly classifiers, » *2015 Third International Conference on Digital Information, Networking, and Wireless Communications (DINWC)*, IEEE, p. 115-120, 2015.

[60] N. AL-ROUSAN, S. HAERI et L. TRAJKOVIĆ, « Feature selection for classification of BGP anomalies using Bayesian models, » *2012 International Conference on Machine Learning and Cybernetics*, IEEE, p. 140-147, 2012.

[61] Q. DING, Z. LI, P. BATTA et L. TRAJKOVIĆ, « Detecting BGP anomalies using machine learning techniques, » *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, p. 003 352-003 355, 2016.

[62] Y. LI, H.-J. XING, Q. HUA, X.-Z. WANG, P. BATTA, S. HAERI et L. TRAJKOVIĆ, « Classification of BGP anomalies using decision trees and fuzzy rough sets, » *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, p. 1312-1317, 2014.

[63] M. CHENG, Q. XU, L. JIANMING, W. LIU, Q. LI et J. WANG, « MS-LSTM : A multi-scale LSTM model for BGP anomaly detection, » *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, IEEE, p. 1-6, 2016.

[64] M. CHENG, Q. LI, J. LV, W. LIU et J. WANG, « Multi-scale LSTM model for BGP anomaly classification, » *IEEE Transactions on Services Computing*, 2018.

[65] M. COSOVIC, S. OBRADOVIC et E. JUNUZ, « Deep learning for detection of BGP anomalies, » *International Work-Conference on Time Series Analysis*, Springer, p. 95-113, 2017.

[66] I. O. de URBINA CAZENAVE, E. KÖŞLÜK et M. C. GANIZ, « An anomaly detection framework for BGP, » *2011 International Symposium on Innovations in Intelligent Systems and Applications*, IEEE, p. 107-111, 2011.

[67] J. LI, D. DOU, Z. WU, S. KIM et V. AGARWAL, « An Internet routing forensics framework for discovering rules of abnormal BGP events, » *ACM SIGCOMM Computer Communication Review*, t. 35, p. 55-66, 2005.

[68] S. PAPADOPOULOS, K. MOUSTAKAS, A. DROSOU et D. TZOVARAS, « Border gateway protocol graph : detecting and visualising internet routing anomalies, » *IET Information Security*, t. 10, p. 125-133, 2016.

[69] L. SALAMATIAN, D. KAAFAR et K. SALAMATIAN, « A Geometric Approach for Real-time Monitoring of Dynamic Large Scale Graphs : AS-level graphs illustrated, » *arXiv preprint arXiv :1806.00676*, 2018.

[70] S. CHO, R. FONTUGNE, K. CHO, A. DAINOTTI et P. GILL, « BGP hijacking classification, » *2019 Network Traffic Measurement and Analysis Conference (TMA)*, IEEE, p. 25-32, 2019.

[71] R. FONTUGNE, A. SHAH et E. ABEN, « The (thin) bridges of as connectivity : Measuring dependency using as hegemony, » *International Conference on Passive and Active Network Measurement*, Springer, p. 216-227, 2018.

[72] Z. WU, S. PAN, F. CHEN, G. LONG, C. ZHANG et P. S. YU, « A comprehensive survey on graph neural networks, » *arXiv preprint arXiv :1901.00596*, 2019.

[73] P. GOYAL et E. FERRARA, « Graph embedding techniques, applications, and performance : A survey, » *Knowledge-Based Systems*, t. 151, p. 78-94, 2018.

[74] F. SCARSELLI, M. GORI, A. C. TSOI, M. HAGENBUCHNER et G. MONFARDINI, « The graph neural network model, » *IEEE Transactions on Neural Networks*, t. 20, p. 61-80, 2009.

[75] Y. LI, D. TARLOW, M. BROCKSCHMIDT et R. ZEMEL, « Gated graph sequence neural networks, » *ICLR*, 2016.

[76] J. BRUNA, W. ZAREMBA, A. SZLAM et Y. LECUN, « Spectral networks and locally connected networks on graphs, » *ICLR*, 2013.

[77] T. N. KIPF et M. WELLING, « Semi-supervised classification with graph convolutional networks, » *arXiv preprint arXiv :1609.02907*, 2016.

[78] P. W. BATTAGLIA, J. B. HAMRICK, V. BAPST, A. SANCHEZ-GONZALEZ, V. ZAMBALDI, M. MALINOWSKI, A. TACCHETTI, D. RAPOSO, A. SANTORO, R. FAULKNER et al., « Relational inductive biases, deep learning, and graph networks, » *arXiv preprint arXiv :1806.01261*, 2018.

[79] J. GILMER, S. S. SCHOENHOLZ, P. F. RILEY, O. VINYALS et G. E. DAHL, « Neural message passing for quantum chemistry, » *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, p. 1263-1272, 2017.

[80] W. L. HAMILTON, R. YING et J. LESKOVEC, « Inductive representation learning on large graphs, » *Proceedings of the 31st International Conference on Neural Information Processing Systems*, p. 1025-1035, 2017.

[81] J. CHEN, J. ZHU et L. SONG, « Stochastic training of graph convolutional networks with variance reduction, » *arXiv preprint arXiv :1710.10568*, 2017.

[82] J. CHEN, T. MA et C. XIAO, « Fastgcn : fast learning with graph convolutional networks via importance sampling, » *arXiv preprint arXiv :1801.10247*, 2018.

[83] W.-L. CHIANG, X. LIU, S. SI, Y. LI, S. BENGIO et C.-J. HSIEH, « Cluster-gcn : An efficient algorithm for training deep and large graph convolutional networks, » *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, p. 257-266, 2019.

[84] M. ZHANG, Z. CUI, M. NEUMANN et Y. CHEN, « An end-to-end deep learning architecture for graph classification, » *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[85] I. S. DHILLON, Y. GUAN et B. KULIS, « Weighted graph cuts without eigenvectors a multilevel approach, » *IEEE transactions on pattern analysis and machine intelligence*, t. 29, p. 1944-1957, 2007.

[86] J. LEE, I. LEE et J. KANG, « Self-attention graph pooling, » *International Conference on Machine Learning*, PMLR, p. 3734-3743, 2019.

[87] R. YING, J. YOU, C. MORRIS, X. REN, W. L. HAMILTON et J. LESKOVEC, « Hierarchical graph representation learning with differentiable pooling, » *arXiv preprint arXiv :1806.08804*, 2018.

[88] D. BAHDANAU, K. CHO et Y. BENGIO, « Neural machine translation by jointly learning to align and translate, » *ICLR*, 2015.

[89] P. VELIČKOVIĆ, G. CUCURULL, A. CASANOVA, A. ROMERO, P. LIO et Y. BENGIO, « Graph attention networks, » *ICLR*, 2018.

[90] J. ZHANG, X. SHI, J. XIE, H. MA, I. KING et D.-Y. YEUNG, « Gaan : Gated attention networks for learning on large and spatiotemporal graphs, » *arXiv preprint arXiv :1803.07294*, 2018.

[91] T. N. KIPF et M. WELLING, « Variational graph auto-encoders, » *arXiv preprint arXiv :1611.07308*, 2016.

[92] S. T. ROWEIS et L. K. SAUL, « Nonlinear dimensionality reduction by locally linear embedding, » *science*, t. 290, p. 2323-2326, 2000.

[93] A. GROVER et J. LESKOVEC, « node2vec : Scalable feature learning for networks, » *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, p. 855-864, 2016.

[94] H. CHEN, B. PEROZZI, Y. HU et S. SKIENA, « Harp : Hierarchical representation learning for networks, » *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[95] D. WANG, P. CUI et W. ZHU, « Structural deep network embedding, » *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, p. 1225-1234, 2016.

[96] P. GOYAL, N. KAMRA, X. HE et Y. LIU, « Dyngem : Deep embedding method for dynamic graphs, » *arXiv preprint arXiv :1805.11273*, 2018.

[97] S. MAHDAVI, S. KHOSHRAFTAR et A. AN, « Dynamic Joint Variational Graph Autoencoders, » *arXiv preprint arXiv :1910.01963*, 2019.

[98] S. MAHDAVI, S. KHOSHRAFTAR et A. AN, « dynnode2vec : Scalable dynamic network embedding, » *2018 IEEE International Conference on Big Data (Big Data)*, IEEE, p. 3762-3765, 2018.

[99] K. DING, J. LI, R. BHANUSHALI et H. LIU, « Deep anomaly detection on attributed networks, » *Proceedings of the 2019 SIAM International Conference on Data Mining*, SIAM, p. 594-602, 2019.

[100] L. ZHENG, Z. LI, J. LI, Z. LI et J. GAO, « Addgraph : anomaly detection in dynamic graph using attention-based temporal GCN, » *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, AAAI Press, p. 4419-4425, 2019.

[101] W. YU, W. CHENG, C. C. AGGARWAL, K. ZHANG, H. CHEN et W. WANG, « Netwalk : A flexible deep embedding approach for anomaly detection in dynamic networks, » *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, p. 2672-2681, 2018.

[102] K. HOARAU, P. U. TOURNOUX et T. RAZAFINDRALAMBO, « BML : an efficient and versatile tool for BGP dataset collection, » *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, p. 1-6, 2021.

[103] G. H. TONY BATES Revised by : Philip Smith, *CIDR Report*, URL : https://www.cidr-report.org/as2.0/.

[104] J. M. HERNÁNDEZ et P. VAN MIEGHEM, « Classification of graph metrics, » *Delft University of Technology : Mekelweg, The Netherlands*, p. 1-20, 2011.

[105] W. ELLENS et R. E. KOOIJ, « Graph measures and network robustness, » *arXiv preprint arXiv :1311.5064*, 2013.

[106] *NetworkX — NetworkX documentation*, URL : https://networkx.org/ (visité le 04/10/2021).

[107] K.-I. GOH, B. KAHNG et D. KIM, « Universal behavior of load distribution in scale-free networks, » *Physical review letters*, t. 87, p. 278 701, 2001.

[108] *BGPStream*, URL : https://bgpstream.com/ (visité le 17/11/2021).

[109] T. FARASAT, M. A. RATHORE, A. KHAN, S. PARK et J. W. KIM, « BGP traffic volume forecasting using LSTM framework, » *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*, p. 465-466, 2021.

[110] K. HOARAU, P.-U. TOURNOUX et T. RAZAFINDRALAMBO, « Suitability of Graph Representation for BGP Anomaly Detection, » *2021 IEEE 46th Conference on Local Computer Networks (LCN 2021)*, 2021.

[111] DYN, *Oracle Internet Intelligence Blog*, URL : https://blogs.oracle.com/internetintelligence/ (visité le 31/05/2021).

[112] R. BELLMAN, « Dynamic programming, » *Science*, t. 153, p. 34-37, 1966.

[113] K. HOARAU, P. U. TOURNOUX et T. RAZAFINDRALAMBO, « Detecting forged AS paths from BGP graph features using Recurrent Neural Networks, » *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, IEEE, p. 735-736, 2022.

[114] M. SUNDARARAJAN, A. TALY et Q. YAN, « Axiomatic attribution for deep networks, » *International Conference on Machine Learning*, PMLR, p. 3319-3328, 2017.

[115] K. HOARAU, P. U. TOURNOUX et T. RAZAFINDRALAMBO, « BGNN : Detection of BGP Anomalies Using Graph Neural Networks, » *IEEE Symposium on Computers and Communications (ISCC)*, 2022.

[116] M. FEY et J. E. LENSSEN, « Fast Graph Representation Learning with PyTorch Geometric, » *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[117] D. P. KINGMA et J. BA, « Adam : A method for stochastic optimization, » *arXiv preprint arXiv :1412.6980*, 2014.

[118] K. HE, X. ZHANG, S. REN et J. SUN, « Delving deep into rectifiers : Surpassing human-level performance on imagenet classification, » *Proceedings of the IEEE international conference on computer vision*, p. 1026-1034, 2015.

Table des figures

Liste des tableaux