Analysis of network delay measurements : Data mining methods for completion and segmentation

Sanaa Ghandi

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
MINES-TÉLÉCOM ATLANTIQUE BRETAGNE
PAYS DE LA LOIRE – IMT ATLANTIQUE

ÉCOLE DOCTORALE N° 648
Sciences pour l'Ingénieur et le Numérique
Spécialité : *Télécommunications*

Par

Sanaa GHANDI

Analysis of network delay measurements

Data mining methods for completion and segmentation

Thèse présentée et soutenue à IMT Atlantique, campus de Brest, le 15 décembre 2023
Unité de recherche : Lab-STICC
Thèse N° : 2023IMTA0382

Rapporteurs avant soutenance :

M. Jean-Louis ROUGIER	Professeur à Télécom Paris
M. Yezekael HAYEL	Professeur à Université d'Avignon

Composition du Jury :

Président :	M. Tijani CHAHED	Professeur à Télécom Sud Paris, Evry
Examinateurs :	M. Jean-Louis ROUGIER	Professeur à Télécom Paris
	M. Yezekael HAYEL	Professeur à Université d'Avignon
	M. Kevin VERMEULEN	Chargé de Recherches au CNRS LAAS CNRS, Toulouse
Dir. de thèse :	Mme Sandrine VATON	Professeure à IMT Atlantique, Brest
Co-dir. de thèse :	M. Thierry CHONAVEL	Professeur à IMT Atlantique, Brest
Co-encadrant :	M. Alexandre REIFFERS-MASSON	Maître de conférences à IMT Atlantique, Brest

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisors, Sandrine Vaton, Thierry Chonavel, and Alexandre Reiffers-Masson. I feel incredibly fortunate to have had your guidance throughout my PhD journey. Despite challenging times, your presence and professional support were invaluable. Your encouragement allowed me to explore various subjects, participate in diverse activities, and conduct the experiments that form the foundation of this document. I am deeply grateful for your support in pursuing research topics I am passionate about.

I would also like to thank Brest Metropole and IMT Atlantique for funding this thesis and providing an enjoyable work environment.

I extend my sincere thanks to the jury members Tijani CHAHED, Jean-Louis ROUGIER, Yezekael HAYEL and Kevin VERMEULEN for their dedication in reading my thesis and the insightful feedback and stimulating discussions during the defense.

Special thanks to Armelle and Coraline for the lightness they bring to the department and for diligently managing all my missions and travel arrangements throughout my PhD.

I am also profoundly grateful to my colleagues and friends, Robin, Cherhazad, Navdeep, Maxence, Amine, Mary, François-Maël, Maxine, Ihab, and Gaël. Your support, laughter, and endless conversations have been a constant source of motivation and joy.

Lastly, I want to thank my family for their constant presence, belief, and trust in me. To my mother, this thesis is dedicated to you for your continuous support, even from afar.

Thank you all.

Résumé en Francais

Cette thèse explore l'analyse des données dans le cadre de la supervision d'Internet, mettant un accent particulier sur l'examen des mesures de délais réseau. Ces mesures sont cruciales car elles révèlent des informations essentielles sur la qualité de service, la connectivité, et les potentielles défaillances au sein du réseau. Un avantage notable est la capacité d'acquérir des mesures de délais à travers l'Internet sans nécessiter le statut d'opérateur réseau, grâce à l'utilisation de sondes permettant d'effectuer des mesures de bout en bout sans accès direct aux équipements réseau. Un exemple pertinent est l'utilisation de plateformes de mesure Internet publiques, telles que RIPE Atlas, exploitées dans cette thèse.

L'intérêt pour ces données réside dans leur utilité pour évaluer et améliorer la qualité de service, ainsi que dans leur capacité à faciliter l'analyse des incidents réseau. Bien que l'analyse puisse être réalisée manuellement par des experts, la volumétrie des données impliquées rend rapidement le processus laborieux, surtout lorsqu'il s'agit d'incidents majeurs pouvant requérir plusieurs jours d'analyse. D'où l'intérêt marqué pour l'automatisation du traitement de ces données.

Cependant, l'exploitation de ces mesures de délai n'est pas exempte de défis. Parmi ceux-ci figure le problème des données manquantes, qui peut survenir à la suite de défaillances des sondes, de congestion du réseau, ou de pertes de paquets. Ce phénomène, particulièrement prononcé durant les incidents réseau, nécessite une attention particulière, tout comme la stratégie de sous-échantillonnage adoptée pour prévenir la surcharge du réseau, qui peut également entraîner un manque de données. Un autre défi est la non-stationnarité des données, avec des délais Internet restant généralement stables sur quelques heures mais pouvant subir des changements brusques, souvent dus à des modifications de routage.

Ainsi, cette thèse se penche sur la résolution du problème des données manquantes et sur la segmentation multivariée des délais.

Contributions

Les travaux portant sur l'aspect de la complétion des données manquantes dans les délais réseau ont abouti à deux contributions significatives, chacune conduisant à une publication dans des conférences internationales. La première s'est concentrée sur les méthodes basées sur la factorisation matricielle, explorant l'application des algorithmes de factorisation matricielle positive pour reconstruire les matrices de délais incomplètes. La seconde contribution se base sur les méthodes de filtrage collaboratif neuronal (NCF), inspirées des systèmes de recommandation.

— **Non-negative matrix factorization for network delay matrix completion.** Sanaa Ghandi, Alexandre Reiffers-Masson, Sandrine Vaton and Thierry Chonavel. In : NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium. IEEE, 2022. p. 1-6.

— **Neural collaborative filtering for network delay matrix completion.** Sanaa Ghandi, Alexandre Reiffers-Masson, Sandrine Vaton and Thierry Chonavel. In : 2022 18th International Conference on Network and Service Management (CNSM). IEEE, 2022. p. 359-363.

La contribution finale de cette thèse concerne la segmentation des délais dans les réseaux. Pour commencer, les séquences de délais ont été regroupées grâce au regroupement hiérarchique. Ensuite, leur segmentation a été réalisée en utilisant une méthode combinant le regroupement hiérarchique et l'algorithme de Viterbi, dans le but d'optimiser la précision et l'efficacité du processus. Cette approche a été validée par deux publications, une présentée à une conférence nationale et l'autre à une conférence internationale.

— **Lightweight Network Delay Segmentation Based on Smoothed Hierarchical Clustering.** Sanaa Ghandi, Alexandre Reiffers-Masson, Sandrine Vaton and Thierry Chonavel. In : 2023 IFIP Networking Conference (IFIP Networking). IEEE, 2023. p. 1-9.

— **Segmentation à faible coût des délais réseau basée sur le regroupement hiérarchique.** Sanaa Ghandi, Alexandre Reiffers-Masson, Sandrine Vaton and Thierry Chonavel. In : GRETSI 2023.

Cette thèse est structurée en cinq chapitres. Le premier chapitre offre une introduction globale, tandis que le second met en évidence la complexité et l'importance de la surveillance d'un réseau Internet. Le troisième chapitre est dédié à la collecte des délais, tant

synthétiques que réels. Les quatrième et cinquième chapitres abordent, respectivement, les processus de complétion et de segmentation de ces délais. Une exploration approfondie de chaque chapitre sera présentée dans les sections suivantes, concluant par un résumé des travaux réalisés ainsi que des perspectives d'avenir.

La complexité d'Internet

L'Internet se distingue par une complexité inhérente à son architecture étendue et à sa capacité à relier des milliards d'usagers via des milliers de systèmes autonomes interconnectés. Outre les aspects techniques et technologiques, l'Internet a également un impact économique, politique et réglementaire significatif. Cette complexité de connexions et de systèmes impose une surveillance attentive afin d'assurer un fonctionnement fluide et une compréhension approfondie.

Néanmoins, superviser l'Internet est parsemé de défis. L'automatisation des processus de supervision, la gestion des erreurs et l'efficacité du stockage des données en font partie. Ces obstacles soulignent la complexité du contrôle du fonctionnement de l'Internet, dont tous les aspects, des performances à la sécurité, doivent être gérés méticuleusement. Il s'agit de naviguer dans d'énormes quantités de données, de garantir l'exactitude des informations recueillies et de traiter rapidement toute anomalie ou tout problème afin de prévenir ou d'atténuer les perturbations. Cela nécessite non seulement des solutions technologiques sophistiquées, mais aussi une approche stratégique.

L'importance des mesures dans ce processus de supervision est primordiale. Dans cette étude, nous privilégions l'utilisation de mesures actives par opposition aux mesures passives, pour des raisons de sécurité et de disponibilité. Des plateformes comme RIPE Atlas constituent un outil inestimable, permettant d'avoir une vue d'ensemble sur l'Internet et de mieux comprendre son fonctionnement afin de l'améliorer si nécessaire. Grâce à des mesures actives, nous pouvons tester de manière proactive les performances et la fiabilité du réseau, offrant ainsi des informations essentielles pour la maintenance et l'amélioration continues de l'infrastructure Internet. Cette approche permet une stratégie plus dynamique et réactive dans la supervision et l'amélioration de la performance et de la résilience de l'Internet.

Mesures synthétiques et réelles des délais

Le troisième chapitre de la thèse se concentre sur les méthodes de collecte de données de délai, en utilisant à la fois un environnement synthétique, via un simulateur de délai

spécialement conçu, et un contexte réel, par le biais de plateformes de mesure et de supervision d'Internet telles que RIPE Atlas. La première contribution de ce chapitre réside dans le développement d'un simulateur de délais réseau. Le simulateur de délai est élaboré pour reproduire fidèlement les caractéristiques des délais réels dans un système autonome, offrant une solution simplifiée mais efficace pour tester divers algorithmes dans un cadre maîtrisé. En effet, le simulateur de délais permet de régler plusieurs facteurs, y compris la topologie sous-jacente, la distribution des délais et la longueur des séries temporelles de délais générées.

Le simulateur permet la modélisation précise des délais de bout en bout à travers des graphes représentant des systèmes autonomes, en générant du trafic suivant un modèle gravitaire, et en simulant des variations brusques qui peuvent être dues à des incidents ou à des changements de routage et des congestions. Cette démarche vise à reproduire la stabilité temporelle et la corrélation spatiale observées dans les réseaux actuels, constituant ainsi une base solide pour l'évaluation et l'optimisation des algorithmes.

La seconde contribution de ce chapitre est l'acquisition de données réelles en exploitant la plateforme de surveillance Internet RIPE Atlas. Cette démarche a permis d'accéder à un large éventail de mesures par deux moyens principaux : le téléchargement de mesures publiques existantes et la mise en œuvre de campagnes de mesure personnalisées. Nous avons priorisé l'utilisation de deux types de mesures disponibles sur la plateforme : les temps de trajet aller-retour (RTT) et les mesures traceroute, permettant de constituer plusieurs ensembles de données.

Dans le cadre de notre campagne de mesure, deux ensembles de données spécifiques ont été créés. Le premier se basait sur l'utilisation d'ancres aléatoires sélectionnées dans RIPE Atlas, tandis que le second ciblait des sondes situées au sein des systèmes autonomes exploités par Google. En outre, nous avons enrichi notre collection de données en extrayant des informations à partir de mesures existantes liées à deux incidents significatifs survenus sur les points d'échange Internet AMS-IX et DE-CIX. Pour ce faire, des traceroutes ont été réalisés et les données de délai associées aux chemins empruntant ces IXP spécifiques ont été filtrées et analysées.

L'objectif de cette collecte de données était de capturer la diversité et la variabilité des délais observés sur l'ensemble du réseau Internet, offrant ainsi une vue d'ensemble de sa performance. Les données relatives aux périodes de crise, notamment durant les incidents affectant les IXP majeurs, ont été particulièrement précieuses pour évaluer l'impact de telles perturbations sur les délais. Cette stratégie globale de collecte de données a considé-

rablement enrichi la thèse, en fournissant des scénarios pratiques pour valider les modèles théoriques et les simulations développées.

Complétion des délais

Le quatrième chapitre de cette thèse aborde le défi de la mesure et de la prédiction précise des délais au sein des réseaux, un enjeu de plus en plus crucial dans le contexte de la surveillance des performances réseau, notamment avec l'émergence d'applications exigeant une grande réactivité. L'obtention de mesures de délai précises est essentielle pour améliorer la qualité de service (QoS) du réseau et enrichir l'expérience utilisateur dans divers domaines tels que le streaming vidéo en temps réel ou le trading financier. Cependant, la collecte systématique et complète de données sur les délais représente un défi majeur. Cette tâche peut s'avérer coûteuse et entraîner une congestion du réseau due au trafic supplémentaire généré par ces activités de mesure. De plus, le nombre de mesures nécessaires pour analyser exhaustivement un réseau augmente de manière quadratique avec sa taille, ce qui conduit souvent à limiter le nombre de mesures, entraînant ainsi des données potentiellement incomplètes. En outre, divers facteurs techniques tels que les pannes de réseau, la perte de paquets ou la défaillance de sondes peuvent également contribuer à l'absence de mesures, entraînant ainsi l'omission de données. En estimant les délais manquants à partir d'une observation partielle des mesures de délai d'un réseau, la complétion de la matrice des délais s'apparente à la tomographie de réseau. Cette dernière estime les caractéristiques internes d'un réseau en se basant sur des mesures de bout en bout. Les deux approches exploitent des modèles mathématiques pour inférer les données manquantes, permettant ainsi d'obtenir une vision plus complète des performances du réseau malgré les lacunes dans les données directement mesurées.

Pour surmonter ces défis, le chapitre explore la possibilité de considérer la complétion des délais manquants comme un problème de complétion de matrice, en exploitant les corrélations spatiales et temporelles inhérentes aux délais dans les réseaux. La corrélation spatiale se réfère aux similarités de motifs dans les délais lorsqu'on considère des itinéraires avec des segments ou des nœuds communs, tandis que la corrélation temporelle dénote la stabilité des délais dans le temps pour une paire de nœuds particulière. Ces corrélations suggèrent que les mesures manquantes peuvent être efficacement inférées à partir des données disponibles, réduisant ainsi le besoin d'une surveillance en temps réel exhaustive.

Une structure de données a été choisie pour les mesures de délai, définissant une structure de matrice de délai qui encapsule les délais entre les nœuds au fil du temps. Cette

structure bénéficie des corrélations spatiales et temporelles mentionnées précédemment, contribuant ainsi à ses caractéristiques de rang faible, une propriété clé qui facilite les méthodes de complétion de matrice efficaces pour prédire les délais manquants.

Le problème de complétion de matrice qui consiste à prédire ou à estimer les valeurs manquantes dans une matrice à partir des valeurs connues et observées dans cette même matrice est souvent abordé en utilisant des techniques d'apprentissage automatique, telles que la factorisation de matrice, les méthodes de régression, les réseaux de neurones, etc. Ces techniques exploitent les relations entre les différentes entrées pour prédire les valeurs manquantes de manière efficace.

Dans le cadre de cette thèse, deux approches ont été explorées pour la prédiction des délais réseau non observés : la factorisation de matrice non négative (NMF) et le filtrage collaboratif neuronal (NCF), ouvrant ainsi de nouvelles perspectives dans ce domaine.

La NMF est une méthode qui consiste à approximer une matrice donnée comme le produit de deux matrices de rang faible, avec des entrées non négatives. Cette technique est particulièrement adaptée à la nature des délais réseau, qui sont intrinsèquement non négatifs. En appliquant la NMF à la complétion des matrices de délais réseau, des considérations supplémentaires sur la stabilité temporelle et les corrélations spatiales des délais ont été intégrées. Deux variantes de la NMF ont été étudiées : une méthode de gradient projeté alterné et une méthode utilisant le gradient optimal de Nesterov. Ces deux méthodes visent à minimiser les erreurs de reconstruction pour les valeurs observées tout en garantissant que les estimations des délais restent non négatives.

Nous avons pu tester ces méthodes dans un environnement contrôlé en utilisant notre générateur de délai synthétique et sur des données réelles avec des délais de Ripe Atlas. Les deux approches sont simples, faciles à mettre en œuvre et montrent une grande précision sur la tâche de complétion lorsqu'elles sont appliquées à des données synthétiques. Cependant, les expériences ont souligné la différence de vitesse entre les deux algorithmes. Le gradient projeté alterné converge plus lentement que le NeNMF. Par conséquent, la scalabilité du NeNMF a été exploitée en l'appliquant à des ensembles de données réels. La complétion donnée par cet algorithme a montré une grande précision avec un petit nombre d'itérations.

La deuxième approche, le filtrage collaboratif neuronal (NCF), largement utilisé dans les systèmes de recommandation, a été appliquée pour prédire les délais réseau non observés en identifiant les interactions latentes dans les données des délais réseau. Deux architectures ont été examinées : la factorisation matricielle généralisée (GMF) et le per-

ceptron multicouche (MLP). Le GMF agit essentiellement comme une forme de factorisation matricielle, simplifiant ainsi les relations complexes des délais réseau en un modèle plus simple. En revanche, l'architecture MLP introduit de la non-linéarité, améliorant ainsi sa capacité à capturer les relations plus complexes et subtiles présentes dans les données. Grâce à ces deux approches, le modèle NCF offre un cadre complet pour la compréhension des délais réseau, en passant des interactions de base décrites par le modèle GMF aux dynamiques plus nuancées révélées par le modèle MLP.

Testées sur des données synthétiques et réelles, les approches NCF nous permettent d'atteindre un taux d'erreur très faible pour la tâche de complétion de matrices sur les deux ensembles de données. Par ailleurs, l'impact de plusieurs paramètres tels que l'optimiseur, le taux d'apprentissage ou le nombre d'époques sur la qualité de la reconstruction a été étudié de manière approfondie. Cela a permis de définir un environnement d'entraînement optimal pour les modèles NCF. En outre, les résultats de la comparaison entre les algorithmes NMF révèlent que NCF surpasse NeNMF sur des données synthétiques, mais cette dynamique s'inverse lorsqu'appliquée aux données de Ripe Atlas. Cette inversion pourrait être attribuée à un nombre insuffisant d'itérations et à l'absence de régularisation par morceaux. Par ailleurs, il a été constaté que NCF est plus efficace en termes de calcul que NeNMF. Finalement, l'ajout d'un terme de régularisation n'a pas amélioré la qualité de la complétion de l'ensemble de données synthétique.

Segmentation des délais

Le cinquième chapitre souligne l'importance des mesures de délai dans la supervision des réseaux et propose une nouvelle approche pour automatiser la segmentation de ces délais en utilisant le regroupement hiérarchique et l'algorithme de Viterbi. En effet, les métriques de délai Internet de bout en bout, telles que le délai aller-retour (RTT), sont essentielles pour évaluer la performance et garantir la qualité de service.

Ces mesures de délai peuvent révéler une variété de motifs, certains étant récurrents dans le temps ce qui reflète la présence de dépendances temporelles dans les séries de délai. Ces schémas récurrents offrent souvent des indications sur l'état du réseau et peuvent également signaler la présence d'anomalies. De plus, il n'est pas rare que des motifs identiques apparaissent de manière synchrone dans plusieurs séries temporelles de délai, en particulier lorsque les chemins réseau associés partagent des segments de route communs. Cette synchronisation des motifs entre les séries temporelles correspond à la corrélation spatiale des délais, particulièrement observable entre les chemins passant par le même point

d'échange Internet (IXP). Dans ces cas, les délais peuvent souvent illustrer des changements synchronisés, mettant en évidence l'importance de comprendre et de segmenter ces données de manière appropriée.

Historiquement, la segmentation des délais nécessitait une intervention humaine. Cependant, avec l'afflux énorme de données issues de sources telles que Ripe Atlas et les serveurs des fournisseurs d'accès à Internet, la segmentation manuelle devient non seulement laborieuse mais aussi susceptible d'erreurs, rendant cette approche peu viable. Ce contexte accentue la nécessité de disposer d'outils capables de traiter ces données de manière efficace, et met en évidence l'importance cruciale d'automatiser le processus de segmentation pour les délais réseaux.

Dans la revue de la littérature, diverses tentatives de modélisation des délais Internet et de caractérisation des changements sont mentionnées, y compris l'utilisation de l'apprentissage profond et d'autres méthodes statistiques. Toutefois, ces approches se heurtent souvent à des exigences computationnelles élevées et à un manque de prise en compte des dépendances spatiales parmi les séries de RTT. Pour combler ces lacunes, le chapitre propose une méthode novatrice pour la segmentation multivariée des délais réseau. Dans cette approche, le regroupement hiérarchique est utilisé pour gérer la corrélation spatiale, en commençant par considérer chaque point de données comme un cluster individuel, puis en fusionnant progressivement les clusters les plus proches pour former des clusters plus grands, jusqu'à obtenir un seul cluster global, créant ainsi une hiérarchie de clusters. L'algorithme de Viterbi est ensuite employé pour prendre en compte la dépendance temporelle dans les données.

La méthodologie présentée dans ce chapitre commence par une application du regroupement hiérarchique sur la matrice de corrélation de Pearson des séries temporelles de temps de réponse (RTT), ciblant la détection de clusters de séries de délai basée sur leur corrélation spatiale. Cette étape initiale vise à regrouper les séries temporelles présentant des similitudes dans leurs motifs de retard. Par la suite, une approche de segmentation est adoptée pour ces séries au sein de chaque groupe, combinant une nouvelle fois le regroupement hiérarchique avec un processus de lissage utilisant l'algorithme de Viterbi. Cette démarche est conçue pour affiner la segmentation en tenant compte des dépendances temporelles des données, améliorant ainsi la gestion des séries de délai avec des variances et des motifs diversifiés. L'efficacité de cette méthode a été validée sur des données réelles, montrant des améliorations notables en termes de précision et d'efficacité du traitement par rapport aux techniques existantes.

La contribution principale de cette approche réside dans son efficacité à révéler les interdépendances spatiales et temporelles au sein des données de délai réseau, tout en réalisant des gains significatifs en réduisant à la fois le coût de calcul et le temps d'exécution. La méthode exploite stratégiquement deux cycles de regroupement hiérarchique, couplés à l'algorithme de Viterbi, pour distiller ces relations complexes entre les séries. Le premier cycle de regroupement hiérarchique filtre les séries temporelles par similarité de motifs, où le type de liaison choisi—complète pour une homogénéité accrue sans compression significative, ou de Ward pour une classification en larges ensembles—affecte directement la cohésion des motifs au sein des groupes. La présence notable de corrélations spatiales, illustrée par des sources ou destinations communes au sein des clusters, enrichit cette analyse. Le second cycle de regroupement, suivi de l'application de l'algorithme de Viterbi, entame le processus de segmentation proprement dit, mettant en avant l'efficacité du lissage de Viterbi pour affiner la segmentation. Cette technique démontre des performances comparables à celles du modèle HDP-HMM, mais se distingue par une réduction considérable de la charge de calcul, par un facteur de neuf, par rapport à HDP-HMM, soulignant ainsi l'efficience remarquable de l'approche proposée.

Conclusions et travaux futurs

La thèse s'est focalisée sur la surveillance des réseaux, notamment sur les mesures de délai Internet de bout en bout. Un simulateur de délai réseau a été développé afin de produire des données de délai fiables, et des ensembles de données réelles provenant de plateformes telles que Ripe Atlas ont été explorés. La thèse a aussi traité le problème des valeurs manquantes dans les délais et a proposé des solutions utilisant des approches de complétion de matrice et des réseaux de neurones. De plus, une approche de segmentation multivariée pour les séries temporelles de délai a été introduite.

Pour les travaux futurs, la thèse envisage d'explorer des termes de régularisation alternatifs afin d'améliorer la précision du modèle et d'appliquer ces méthodes à différents ensembles de données. Elle suggère également le développement d'algorithmes de complétion en ligne et de détection d'anomalies pour gérer efficacement la surveillance des réseaux en temps réel. Enfin, des stratégies d'échantillonnage intelligentes pour les mesures réseau sont proposées afin de trouver un équilibre entre l'efficacité des ressources et l'intégrité des données.

Table of Contents

List of figures

Chapter 1

Introduction

1.1 Context and problems addressed in the thesis

This thesis falls within the context of the analysis of Internet monitoring data. It focuses on monitoring the Internet network principally through the lens of network delay measurements. Actually, network delays can provide resourceful information regarding the quality of service, connectivity, and network failures to cite a few. Delay measurements can be obtained on an Internet scale without being a network operator yourself. Indeed, it is possible to deploy probes allowing end-to-end measurements without having access to network equipment. This is for example the case of public access Internet measurement platforms such as RIPE Atlas [97]. In the framework of this thesis, we have used such type of data.

These data are widely used to assess and help improve the quality of service and to analyze and help perform root-cause analysis of network incidents. To some extent, this analysis can be done by experts in a non-automated way. However, given the quantity of data to analyze, this task quickly becomes tedious. Analyzing a major network incident can take days of work. It is therefore desirable to automate the processing as much as possible.

Despite the availability of delay measurements, some problems need to be addressed in order to automate their processing. One of them is the missing data problem, which can be due to probe failures or congestion and packet loss. The rate of missing observations can sometimes be very high, especially during network incidents. Such a phenomenon is observed on monitoring platforms. Missing data can also result from an under-sampling measurement strategy designed to avoid network overloading. Another problem is the nonstationarity of data. Indeed, Internet delays, are often stationary on a timescale of a few hours. However, abrupt changes occur from time to time. Typically such changes can be caused by routing changes.

In the framework of this thesis, we have particularly studied the problem of missing

data and of segmentation of multivariate piecewise stationary data. We focus on the completion and segmentation of network delays.

1.2 Contributions of the thesis

To address the problem of missing observations we propose to take advantage of dependencies among and inside delay time series. This led us to estimate missing delays via a matrix completion strategy. We first considered positive matrix factorization algorithms based on standard minimization techniques such as alternated gradient descent. This has led to one publication which is a communication in a conference.

[37] **Non-negative matrix factorization for network delay matrix completion.** Sanaa Ghandi, Alexandre Reiffers-Masson, Sandrine Vaton and Thierry Chonavel. In : NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium. IEEE, 2022. p. 1-6.

Alternatively, we investigated deep learning approaches inspired by recommender systems for the completion of time delay matrices. This has led to another publication which is a communication in a conference.

[36] **Neural collaborative filtering for network delay matrix completion.** Sanaa Ghandi, Alexandre Reiffers-Masson, Sandrine Vaton and Thierry Chonavel. In : 2022 18th International Conference on Network and Service Management (CNSM). IEEE, 2022. p. 359-363.

Finally, our last contribution is related to the segmentation of network delays. Given the frequent presence of numerous time delay sequences that necessitate analysis, it is advantageous to cluster them into groups prior to time-based segmentation. We have used the hierarchical clustering data-mining approach to cluster time series. In a second step, we have proposed to associate hierarchical clustering with Hidden Markov Models (HMMs) for the joint segmentation of clustered time series. We have studied different variants of the method to enhance its accuracy and speed. Finally, this has led to two communications, one at a national conference and one at an international conference.

— [35] **Lightweight Network Delay Segmentation Based on Smoothed Hierarchical Clustering.** Sanaa Ghandi, Alexandre Reiffers-Masson, Sandrine Vaton and Thierry Chonavel. In : 2023 IFIP Networking Conference (IFIP Networking). IEEE, 2023. p. 1-9.

— [38] **Segmentation à faible coût des délais réseau basée sur le regroupement hiérarchique.** Sanaa Ghandi, Alexandre Reiffers-Masson, Sandrine Vaton and Thierry Chonavel. In : GRETSI 2023.

1.3 Organization of the document

The organization of the thesis is as follows. The second chapter shows the importance of network monitoring and presents the most common challenges that can be encountered in this framework. Furthermore, it underlines the importance of network delay measurements with a focus on round trip times, the measurements used in this thesis. Finally, it defines the research objectives, which are the completion and the segmentation of the round trip time series, and provides an overview of the existing methods.

The third chapter tackles all the aspects related to the datasets used in this work. It covers the generation of synthetic datasets, the collection and scheduling of delay and routing measurements on the Ripe Atlas platform, and the selection of significant measurements to monitor specific Internet infrastructures such as Internet eXchange Points (IXPs). It also provides a small tutorial showing how to launch measurement campaigns in Ripe Atlas.

The fourth chapter presents the different models used to treat the matrix delay completion problem and validates them on both synthetic and real-world datasets. It suggests two approaches: the first one is based on non-negative matrix factorization (NMF) and presents three NMF algorithms. These algorithms are the alternative gradient descent, its regularized version, and an algorithm based on the Nesterov NMF (NeNMF). The other approach considered for the completion problem is neural collaborative filtering (NCF) and we provide two different neural network architectures. This chapter is based on the references [37] and [36].

The fifth chapter covers the segmentation problem and gives a holistic approach validated on real-world data that integrates routing measurement for a deeper understanding

of the network. The approach is based on hierarchical clustering followed by Viterbi smoothing treatment. This chapter is based on the references [35] and [38].

Finally, we conclude this thesis with a chapter that presents the general conclusions of this work and its different perspectives.

Chapter 2

Analysis of Internet delay measurements: some challenges

2.1 On the complexity of Internet monitoring

For most people, it is unimaginable to think of a day without the Internet, as it has become an essential part of our lives. From airports to healthcare, communication to shopping, business to entertainment, the Internet is present in every sphere. In most cases, it performs flawlessly, which justifies its resilient aspect. Nevertheless, there are instances when the Internet proves to be fragile. A recent illustration of this fragility is the Internet outage experienced by Hawaiian Airlines, leading to over 250 flight delays and 50 cancellations [44]. Another critical scenario arises in the healthcare sector, where the heavy reliance on connected devices and services, coupled with digitalization, amplifies the consequences of Internet malfunctions [99]. Beyond quantifiable impacts, businesses can suffer severe damage to service reputation due to network disconnections, as clients expect uninterrupted availability. Prominent examples of this are social media platforms like Facebook, which lost over 100 million dollars of revenue, 5% of its stock value, and watched the fortune of its CEO drop by 4 billion dollars because of a 6-hour outage [77]. This incident is not isolated, as similar events can affect other content providers and companies such as Netflix, YouTube, or Amazon [33]. All these reasons give both customers and companies enough incentive to make sure that their network works continuously. But doing so is more complex than it seems.

On the complexity of Internet network Being able to manage correctly the Internet is difficult due to three main challenges: volume and quality of service requirements, its intertwined nature, and the presence of multiple stakeholders.

We can identify four recent causes for the large *volume* of traffic on the Internet. (1) A high number of devices are connected to the Internet since 66% of the world population is online. (2) International bandwidth usage has almost doubled between 2020 and 2022

reaching a total of 120 Tbits/s, a consequence of advancements in technology and the rise of bandwidth-intensive applications such as 4K streaming and online gaming [51]. (3) The COVID-19 crisis contributed as well to this amplification that increased traffic demand due to the lockdown effect and the remote working trends afterward [29]. (4) The IoT holds now 15.14 billion connected devices and aiming for more than 30 billion by 2030 can only contribute to additional network complexity [27]. Having a large volume of traffic is already hard to manage because the network infrastructure has to be adapted. Moreover, the different requirements of each application make the problem even more complex and force the need to design efficient algorithms (load balancing, routing, etc...) to ensure a good quality of service [87, 19, 115].

To understand the Internet, it would be nice to have a simple model or simple representation of its topology. Such a model could help design algorithms for resource allocation or to identify precisely the location of a failure. However, the Internet is not a simple network of networks. It comprises various infrastructures such as Autonomous Systems (AS) and Internet Exchange Points (IXP) connected between them [40]. These infrastructures are often intertwined, which makes it hard to create precise models where nodes and links are properly delimited [39]. As a consequence, determining the appropriate course of action in the event of an outage or incident is not a straightforward task. One solution for this problem is to create a specific model for every use case [26]. For instance, the Internet can be viewed as an AS graph, but different layers have different topologies based on each layer's constraints [98].

Despite not being covered in this work, it is important to address the graph representation of the Internet and some of its challenges. The Internet has often been referred to as a graph, and different models have been proposed to understand its structure. One of the major characteristics of modeling the Internet as a graph is its "robust yet fragile" nature. In fact, while central nodes on the Internet provide good connectivity and make it robust against node failure, peripheral nodes are vulnerable to attacks that can reach these central nodes, making the Internet fragile [26]. The Internet can also be seen as a hub network, but this is not always the case since it is based only on power law degree [1]. In a hub network, a few highly connected nodes (hubs) play a crucial role in facilitating connections between other nodes. These hubs act as central points for information exchange. And the power law degree distribution results in a scale-free network, where a few nodes have a disproportionately large number of connections while the majority have relatively few. However, the backbone of the Internet is sparse, and the highest degree

nodes can be found in the periphery of the network, which is made of a non-central set of nodes that are linked to the core, to address high-performance issues. These peripheral nodes often operate as content delivery servers or serve as regional hubs that aggregate traffic from smaller networks and provide a connection point to the core of the Internet. To create a more accurate model of the Internet, its function and constraints, such as the cost of deployment, performance, and bandwidth, must be taken into account. The difference between peripheral and backbone network structures is also an important factor. Some recent work aims to incorporate these factors and combine them with the graph perspective of the Internet by suggesting graph neural network approaches [9]. Internet modeling must also take into account the real constraints of the Internet to create an accurate representation.

Furthermore, the Internet's complexity extends beyond its technical aspects. It is intertwined with political, economic, and social factors, all of which influence its functioning and development. Multiple stakeholders, including countries' regulatory bodies, users, service providers, content providers, and economic entities, in addition to the current market competition, and the different technological aspects such as infrastructure, technology, and devices, play an essential role in shaping the Internet [98]. For instance, Internet service providers (ISPs) have the responsibility of troubleshooting problems that arise for their customers. They must monitor and analyze clients' consumption for billing purposes and to ensure compliance with their Service Level Agreements (SLAs). ISPs manage billing procedures and dedicate teams to address customer issues effectively. Another example is policy regulators and governing bodies who need to remain informed about the activities and trends on the Internet. They require insights into the Internet's operations, developments, and potential issues to enact appropriate regulations and policies. Each of these stakeholders requires a different view of the network and presents different monitoring requirements.

To achieve their respective goals, all these stakeholders need tools and methods to assess or improve the quality of Internet connections. This is precisely where the field of Internet monitoring comes into play. It is essential to consider the Internet from the performance lens by taking into account the use of measurements and their purpose.

Monitoring and the network management cycle Before specializing in monitoring the Internet, we will first describe briefly the network management cycle and the importance of network monitoring in communication networks. Network monitoring is the process of observing and analyzing the performance, availability, and overall health

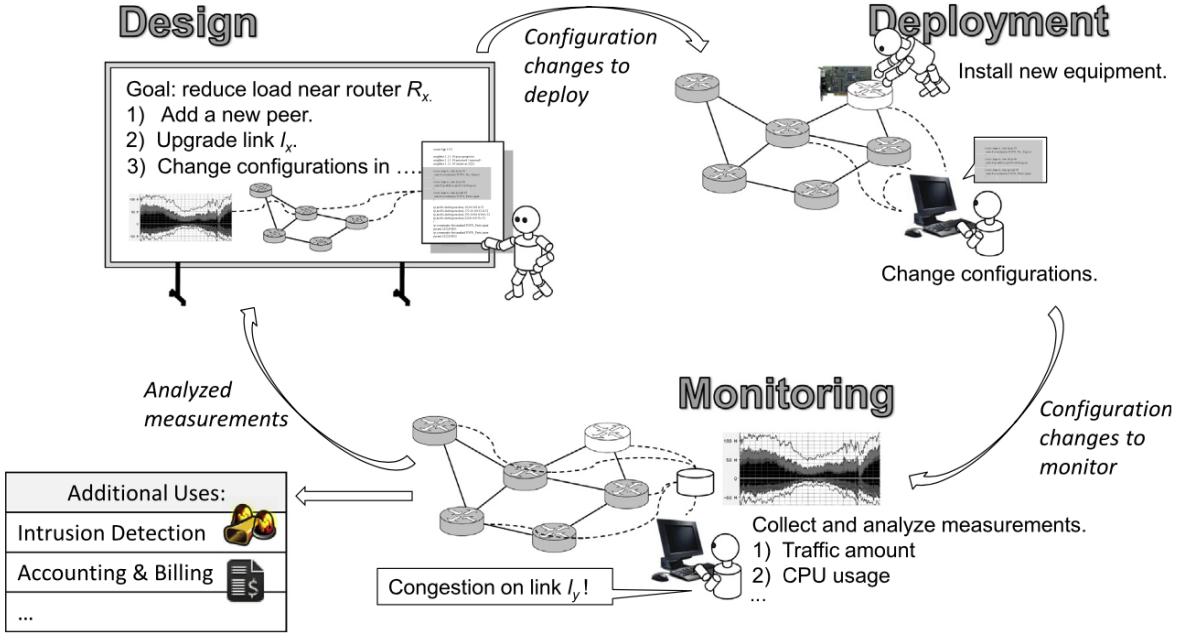
of a computer network. This practice involves the use of various tools and technologies to gather data about the network's traffic, devices, and infrastructure. It plays a crucial role in network management operations, allowing for early identification of trends and patterns in network traffic and devices. It helps network operators understand the current state of the network and helps to find new configurations to improve the state of the network. However, network operators face challenges in monitoring and troubleshooting, leading to disruptions and increased costs. Indeed, the complexity of monitoring tasks can often be a challenge when it comes to supervising a communication network. Additionally, these operations are error-prone and rely heavily on human intervention.

It has been identified by network operators and companies that monitoring and troubleshooting networks are important. For instance, after experiencing multiple outages lasting over three hours, Amazon has allocated a growing portion of its IT budget to network monitoring instead of investing in value-adding services and equipment [118].

Part of the difficulty in monitoring is due to its interaction with the rest of the network management cycle. Indeed, it is an essential part of the network management cycle: design, deployment, and monitoring (illustrated in Figure 2.1). Network operators can manage their network using measurements and configurations. Measurements provide insights into the network's current behavior, while configurations define the desired behavior of network devices. Another challenge to keep in mind while doing network monitoring is the 'intrusiveness' of the measurement tool, which can impact network performance, as it adds extra traffic and potentially disrupts the network flow [89]. This intrusiveness issue is particularly relevant in the context of delay measurements. In fact, certain measurement platforms limit the number of measurements conducted each day to prevent overloading the overall network traffic.

Let us start by illustrating a network management cycle for load reduction near a router. Such cycle can be summarized as follows (see Figure 2.1): the monitoring operations collect and analyze measurements to understand the network's behavior, influencing design operations to make necessary changes to configurations and infrastructure. These changes aim to meet specific network requirements, such as traffic distribution, QoS or business constraints. The planned changes are then deployed in the network, while monitoring ensures that the network operates as intended. The cycle continues as deviations between the current and intended behavior are identified through monitoring, prompting configuration redesign to restore the desired behavior. This cycle of operations repeats as requirements change and new issues arise.

Figure 2.1 – Network management cycle for load reduction [62].



Notice that, in the previous example, the interactions between components require additional work to progress efficiently. A standard language capable of describing policies for various aspects of network operations is necessary as well as implementing a configuration checker to reduce errors. Finally, visualization tools can help operators understand clearly the configuration to implement and its potential impact.

On the other hand, the monitoring step can be classified into five layers: collection, representation, report, analysis, and presentation.

1. The *collection layer* focuses on collecting accurate measurement data without disrupting network operations.
2. The *representation layer* determines the formats for storing and communicating measurements.
3. The *report layer* ensures timely and efficient transfer of data from collection devices to management stations.
4. The *analysis layer* analyzes measurements and extracts high-level interpretations, which can influence configuration settings.
5. The *presentation layer* presents analysis results in user-friendly formats, aiding net-

work operators in prioritizing issues.

Towards a more efficient monitoring accuracy After analyzing previous works on the five layers of monitoring, a natural tradeoff can be identified. Highly accurate monitoring goes hand in hand with a more frequent measurement scheme, a wider range of metrics, and more complicated treatment tools. This enhancement of precision comes at a cost. It often requires more data storage and computing power for efficient data analysis. In fact, collecting more data more frequently, for better failure identification will increase the CPU load. Another example is the deployment of new monitoring devices which increases the configuration effort. A way of dealing with this is to act on the collection level. In fact, controlling how to sample by only acquiring packets from the flow of interest and avoiding redundancy can be helpful to increase both the accuracy and efficiency of the monitoring. This is the case of cSamp, [105] a network-wide flow monitoring system that controls packet sampling to enhance accuracy and efficiency. It employs high quantiles for precise delay monitoring to effectively capture changes. Another approach includes raising the intelligence of collection devices so that some of the analysis operations are computed directly within these devices. An example of this is the Highly Available Monitoring Services Architecture (HAMSA) [12]. It allows a dynamically enhanced monitoring functionality and enables its decentralization efficiently. Finally, other monitoring policies focus on flexibility by collecting diverse data types and allowing programmable measurements [72].

Performance metrics To ensure accurate monitoring, it is crucial to select the most suitable performance metrics that can assess the reliability of the network. The most used metrics include:

1. *Network connectivity* is monitored across all network layers and is crucial as it guarantees end-to-end communication between nodes.
2. *Network delay*, measured through end-to-end pings, is widely used to assess network path performance, especially for small data transmissions. It is the cumulative sum of all individual delays within the network.
3. *Packet loss rate*, measured through SNMP packet statistics. It is measured as a percentage of packets lost with respect to packets sent. For ISPs, it is a vital metric to minimize as it affects their marketing. Packet loss, a common network problem, often occurs due to router queues reaching their capacity, leading to packets being dropped.

4. *Available bandwidth*, which directly affects data transmission speed, is essential for highly demanding services such as video streaming but is less popular than the delay metric due to its high measurement overhead. As a substitute, the link load metric, which is easier to measure, is often used to capture available bandwidth information.

Another key factor to consider is the *capacity of a path* which is the minimum capacity among all its constituent links.

Internet measurements: active monitoring and the challenges As researchers explore the Internet through active measurements, the scope of activities is predominantly shaped by the privacy, legal, and proprietary constraints that render passive measurements, which involve collecting data from ongoing traffic without injecting additional probes, inaccessible. As a result, the focus is on utilizing publicly available platforms for conducting active Internet measurements, where specific investigations can be scheduled and executed.

Active measurements, unlike passive ones, involve generating additional traffic to conduct network monitoring. While they do not require any storage and cause no privacy issues, they can potentially disrupt the network by adding extra traffic. Additionally, they demand more resources, both in terms of the network capacity used for the test traffic and the processing power needed to generate and analyze this traffic. However, the benefits of active measurements, such as their ability to provide real-time data, targeted testing, and not requiring access to all devices on the network, justify their use.

Despite the advantages of active measurements, navigating publicly available platforms devoted to this task presents significant challenges. Common issues include missing data due to non-responsive nodes, network connectivity disruptions, and limitations set by service providers. Furthermore, scheduling measurement campaigns requires careful consideration to prevent an overload of monitoring traffic, which could negatively impact network performance or breach ethical guidelines.

In the sphere of active monitoring, another challenging aspect is the development of the analysis layer on monitoring platforms. This requires devising and exploring diverse methods adaptable to network data. We predominantly use delay data in this process due to its accessibility and a direct reflection of the network's quality of service (QoS). The latter is a metric of service performance overall. Augmenting this data with routing measurements enables us to account for the underlying network topology, thereby devising efficient scheduling schemes for monitoring specific Internet infrastructures and uncovering correlations within the network topology. An example of this, developed in Chapter 3,

covers a measurement scheme used to monitor specific Internet exchange points of the Internet.

Internet and Measurement Platforms Ripe Atlas [97], M-Lab [116], Caida [13] and other Internet measurement platforms offer different views of the Internet through topology discovery and performance measurement. They are crucial in determining Internet performance. While these platforms measure various metrics, their accuracy is not always reliable, and they generate overhead [6]. The process of conducting measurements requires specific probes, techniques, and hardware. Furthermore, these platforms face challenges such as limited resources and the need to motivate users to participate in measurements.

At the core, performance metrics for networks can be measured at two levels: link-level or end-to-end level. For network operators, Simple Network Management Protocol (SNMP), which is a standard protocol used for managing devices on IP networks, guarantees the accessibility of link-level measurements such as link packet, loss rate, and link load. However, it's challenging to infer end-user performance solely based on this information. It demands synchronization of measurements on all links along an end-to-end path, which is technically challenging. Moreover, combining performance information from multiple links to infer the overall path performance can be unclear. On the other hand, end-to-end monitoring provides a more realistic representation of a user's data transmission experience. It doesn't require access to network internal information, thus making it an attractive option for both ISP network operators and end users. However, end-to-end monitoring also has its challenges, such as the difficulty of designing techniques to measure end-to-end performance. Currently, the most common end-to-end monitoring techniques are ping and traceroute, which can measure delay, connectivity, and route information.

Measurement Strategy and Tools Our research predominantly utilized the Ripe Atlas measurement platform [97], a global, open, distributed Internet measurement platform. Ripe Atlas consists of thousands of measurement devices, called probes and anchors, which measure Internet connectivity in real time. This platform is operated by the RIPE NCC, an organization supporting the infrastructure of the Internet through technical coordination in Europe, the Middle East, and parts of Central Asia. More details regarding this platform are developed in Chapter 3.

These probes and anchors, hosted by volunteers worldwide, provide an ongoing understanding of the state of the Internet in real time. They generate invaluable data about

network latency, reachability, routing, and other key performance indicators. Network operators, developers, and researchers widely use this data to improve network resilience, monitor service level agreements, verify reachability, and other applications.

2.2 Internet end-to-end delays

In this thesis, we will focus on Round-Trip Time (RTT) measurements which is one possible measurement to monitor network delays. RTTs capture the time needed for a data packet to travel from a sender to a receiver and then back to the sender. There are many reasons behind the monitoring of RTTs. In the context of applications requiring low delay such as gaming or emergencies, making sure that RTTs don't exceed a certain level is crucial. Another aspect is the impact of RTTs on bandwidth provisioned by higher-level transport protocols [122]. Finally, RTTs can be useful to indicate paths having congestion, once the values exceed the minimum RTT [76]. The minimum RTT is viewed as the constant sum of propagation and transmission delays, measured on networks with low traffic.

One limitation of the RTTs metric when compared to the one-way delay metric, is the fact that the Internet paths are rarely symmetric [46]. Estimating one-way delay from RTTs, for example, by dividing RTTs by two, is therefore unfeasible due to factors like different network loads or QoS provisions leading to path asymmetry.

It is not possible to use RTTs properly without the knowledge of network topology and more specifically information regarding the paths taken by the packets. In fact, RTT provides a global idea about the overall route, but it is hard to extract any information regarding specific links. For this reason, we need to use Internet routing measurements, particularly traceroutes. The traceroute utility provides a detailed view of the path a packet takes from its origin to its destination and the time taken at each hop. This sequence of hops forms the RTT. Although having access to all hops might be difficult, by pinpointing most hops in the packet's journey, traceroute enables network engineers to identify potential points of delay or failure in the network [30]. This information is particularly relevant to understanding the source of delays and designing a more efficient routing.

Working with RTTs comes with challenges like incomplete data, which can happen due to equipment failure, congestion, measurement policies [5], or privacy issues as many routers can block ICMP packets for security reasons [25]. Another one is the automation of

RTT treatment. In fact, despite the simplicity of figuring out change points and anomalies in RTT time series, their large data volume makes this task error-prone and fastidious for most humans.

RTT representations Delays can be treated as *time series*. In this approach, each time series represents the delay between a specific origin and destination pair over time. By capturing the temporal variations in delays for each pair, one can analyze patterns, trends, and anomalies in the time series data.

In addition, to account for dependence among network delay time series, they can be organized into a matrix format. In this representation, the rows correspond to pairs of origin and destination nodes, while the columns represent the instances of measurements. Each entry in the matrix corresponds to the delay between the specific origin and destination at the given time of measurement. This matrix representation provides a comprehensive overview of delays between different node pairs at different time points, allowing for analysis and computations based on the matrix structure.

When observed over a long period, typically spanning a few days, network delays tend to exhibit stability over time with minimal variation. However, abrupt changes can occur due to network congestion, equipment failures, or routing changes. The stability in network delays during normal operation implies that the delay matrix, organized as described previously, possesses a low-rank property. This low-rank property arises from the fact that the columns of the delay matrix change very little over time.

RTT challenges: Completion and segmentation

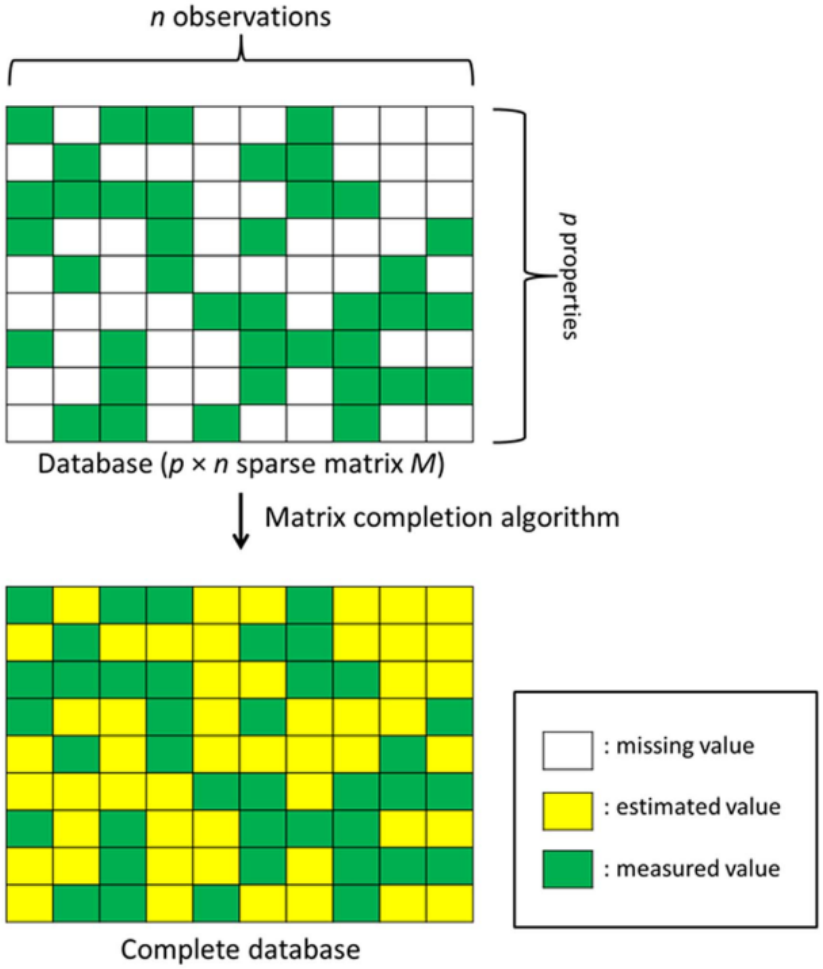
During this Ph.D. we have focused on *RTT completion* and *RTT segmentation*

— *Completion* is a task that consists of inferring missing values from an incomplete set of observations. It helps to palliate network problems that can arise and result in missing values. It can also contribute to programming smarter measurement schemes that rely on fewer probes in the network.

— *Segmentation* is a technique used to dissect the RTT signal into smaller parts, which helps to identify patterns and relations between different sections of the network. Through this analysis, network engineers can improve network performance, identify network bottlenecks and congestion, and troubleshoot problems. RTT segmentation, therefore, plays a crucial role in network monitoring, optimization, traffic analysis, and capacity planning.

2.3 Estimation of missing data in RTT measurements

Figure 2.2 – The matrix completion problem [74].



Related works

In large-scale distributed networks, explicit end-to-end delay measurements between all network pairs become impractical due to computational and network traffic overheads. Delay prediction techniques are thus used as an alternative to these direct measurements, considerably reducing these overheads. These techniques employ matrix completion methods, an essential task involving filling in missing entries of a partially observed matrix. Applications are found in numerous fields such as recommendation systems, image processing, and bioinformatics.

Missing values in network delay have captured the interest of previous work in the networking community. [107] For instance, investigates the missing delays within the Ripe Atlas platform and tries to evaluate the extent of probes' disconnectivity contributing to this issue. Missing delays can be a challenge during the analysis phase. Some works on modeling and segmenting delays based on hidden Markov models (HMM) [80] needed to address this problem using different methods. These methods include assigning elevated values to missing data, treating them as outliers, or substituting missing values in each delay time series with the last known value. Consequently, the considered models for analyzing delays, are forced to adapt to accommodate the missing values. This emphasizes the importance of being able to recover missing delays accurately. Matrix completion methods include Singular Value Thresholding (SVT) [49] which is an optimization technique that reduces the rank of a matrix by applying a thresholding operation to its singular values, Nuclear Norm Minimization [128] that encourages low-rank solutions by minimizing the sum of the singular values of a matrix, Alternating Least Squares (ALS) [43] an iterative optimization algorithm, and Gradient Descent a general optimization technique that aims to minimize a cost function by iteratively adjusting the matrix elements based on the gradient of the cost function. Modern techniques also incorporate Bayesian Probabilistic Matrix Factorization (BPMF) [65] which provides a way for capturing uncertainty in the completion process. Autoencoders [111], a product of deep learning advancements, can learn a compact representation of the input matrix, allowing for efficient and accurate completion of missing entries. The latter is particularly efficient when dealing with large data. While these methods are efficient computationally, BPMF can be more resource-consuming and the efficiency of deep learning models often can vary according to the complexity of the data. On the other hand, BPMF and autoencoders can provide more flexibility than the rest of the mentioned models. The selection of the appropriate method is based on the specific characteristics of the problem at hand, such as the amount and pattern of missing data, and the underlying data distribution.

All these methods can be extended and adapted to particular problems using regularization techniques. Indeed it helps to manage the complexity of models and prevent overfitting. The regularization is used to also add prior knowledge on the data by promoting desirable properties like low-rank representations or sparsity. Techniques include L1 Regularization (Lasso Regression) [112], L2 Regularization (Ridge Regression) [54], and Elastic Net, a blend of both L1 and L2 penalties [57, 92]. L1 regularization favors sparse solutions while the L2 enhances stability. Elastic Net combines both and finds a

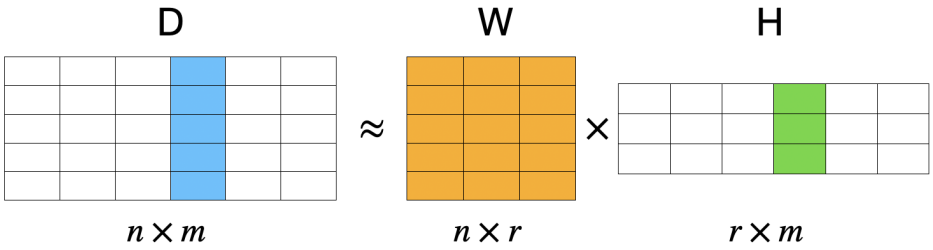
balance between sparsity and stability. Finally, these techniques enhance the robustness, generalization, and practicality of matrix completion models.

Regarding deep learning models, many techniques can be used to customize the model such as Dropout [109], Early Stopping [130], and Batch Normalization [100]. Dropout randomly ignores a subset of layer outputs during training, Early Stopping terminates training before the model's performance on a validation set deteriorates, and Batch Normalization normalizes neurons' activations on a mini-batch basis. Weight Decay, equivalent to L2 regularization, adds a penalty term to the loss function to discourage the model from learning overly complex patterns.

In summary, the interplay between matrix completion, and regularization techniques is crucial in addressing delay prediction in large-scale networks. These combined methods not only deal with network overheads but also tackle missing delays due to network issues such as probe disconnectivity, congestion, or packet loss contributing significantly to efficient network management and optimization.

NMF for RTT completion

Figure 2.3 – The non-negative matrix factorization.



The first method that we have used in this thesis to perform RTT completion is the Non-negative Matrix Factorization (NMF). NMF algorithms are commonly used in multivariate analysis and linear algebra where a matrix D with non-negative entries and dimensions $(n \times m)$ is factorized into (usually) two matrices W and H having respectively the dimensions $(n \times r)$ and $(r \times m)$, with lower rank r where $r \leq n, m$, where $D \approx WH$. Additionally, W and H must have non-negative elements. The matrix W is generally interpreted as the basis matrix, where its columns (the basis vectors) capture the main patterns of the original matrix D learned during the factorization. The matrix H is often referred to as the coefficient matrix, its rows indicate how much each basis vector contributes to each column of the original matrix D. This non-negativity makes the resulting matrices easier to interpret and the factors physically meaningful. This decomposition is

illustrated in Figure 2.3.

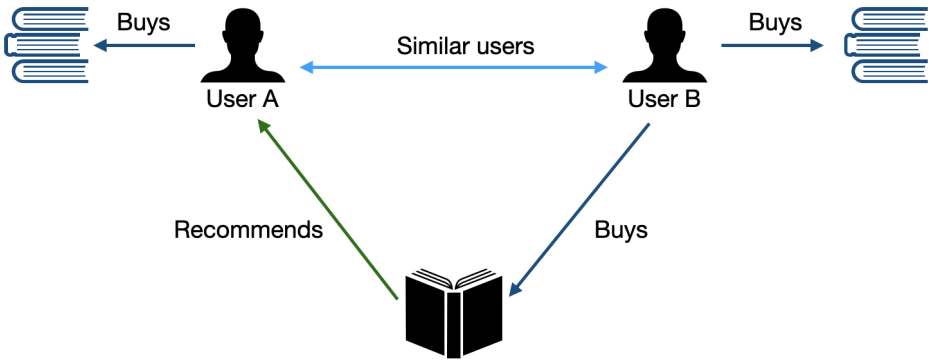
The basic idea is that the observed delays (the known elements of the matrix) can be represented as a product of two lower-rank matrices, one representing the "features" of the source/destination nodes and the other representing the "features" of the measurement instants. Once these feature matrices are learned, they can be used to estimate the unobserved time delay.

One key advantage of NMF here is that it produces interpretable results. Because the factorization is constrained to non-negative elements, the resulting factors can often be interpreted in a meaningful way, such as the intrinsic latency of each network node or the inherent delay of each link in the network [34]. In addition, using other completion methods such as singular value decomposition (SVD) for example, can lead to a completion containing negative values if the number of missing values is important. An example of this result is provided in Chapter 4. In the context of network delays that can only have positive values, methods imposing the nonnegativity constraints such as NMF are indeed important to consider. In this work, we considered two regularizations for the NMF that were able to incorporate the temporal correlation among the RTT time series in order to capture the piecewise-like behavior of network delays.

Moreover, NMF can also handle the high-dimensional and sparse nature of the data often encountered in network monitoring. By reducing the dimensionality of the data, NMF can help alleviate issues with computational complexity and data sparsity, making it easier to analyze and visualize the data.

Neural collaborative filtering (NCF) for RTT completion

Figure 2.4 – The collaborative filtering principle.



Related works

Neural Collaborative Filtering (NCF) [45] is an approach to recommendation systems that utilizes the flexibility and non-linearity of neural networks to learn user-item interactions. It's a powerful technique that can effectively capture complex user-item relationships and improve recommendation performance.

It is inspired by collaborative filtering, which is a popular technique used in recommendation systems and information filtering to provide personalized suggestions or recommendations to users. As shown in Figure 2.4, it relies on the idea that users who have interacted with similar items in the past are likely to have similar preferences and interests in the future. In other words, it collaboratively uses the behavior and preferences of a group of users to make recommendations to an individual user.

The rationale behind using NCF for delay completion in network analysis can be considered analogous to the way neural collaborative filtering is used for recommendation systems. Let's consider network nodes as "users" and measurement instants as "items". Now, instead of predicting user preference for items, we predict time delays between network nodes at different measurement instants.

In many network scenarios, you might not have complete information about time delays between every pair of nodes in the network, similar to how in a recommendation system, you don't know every user's preference for every item. The goal of NCF in this context is to predict the unknown time delays.

NCF leverages the power of neural networks to learn a better function for predicting these delays. Conventional collaborative filtering methods [101, 129] often depend on linear models, which may fail to capture complex patterns. Neural networks, however, can model intricate non-linear interactions.

In summary, NCF is used for delay completion because of its ability to handle sparse data, model complex interactions, and fill in unknown values effectively. This can be particularly useful in scenarios such as network traffic analysis, where fully understanding the delay times between different nodes can aid in optimizing network performance.

The use of NCF has some advantages, such as the possibility to represent much more complex patterns and not being very tied to the low-rank matrix hypothesis [96]. On the other side, the NMF can be more efficient in capturing data schemes when these present simple patterns.

2.4 Segmentation of RTT time series

Related works Delay segmentation refers to the process of dividing a network delay time series into distinct segments or clusters based on similarities or patterns in the delay values.

Time series segmentation is a crucial technique used in data analysis and pattern recognition, particularly when handling sequential data such as network delays. By breaking down a complex time series into smaller segments with distinct and interpretable characteristics, one can uncover meaningful patterns, trends, and anomalies that can contribute significantly to network performance optimization.

One common method is the Sliding Window approach [127]. This technique segments the time series by moving a fixed-size window along the data. It enables the detection of local patterns or trends, making it a suitable choice for identifying anomalies or variations in network delays. However, the key to its effectiveness lies in the appropriate choice of window size, which depends on the specific characteristics of the data. It operates by anchoring the left point of a potential segment at the first data point of a time series, then approximating the data to the right with segments of increasing lengths. When the regimen changes it detects the presence of a new segment.

The Bottom-up [22] and Top-down [10] methods provide another perspective to segmentation. The Bottom-up approach starts by considering each data point as an individual segment and progressively merges the segments based on a predefined similarity criterion. The Top-down method, in contrast, starts with the entire time series as a single segment and iteratively splits it into smaller parts based on a predefined criterion such as the fitting error or the Euclidean distance.The Bottom-up approach offers the advantage of capturing fine-grained patterns and providing a hierarchical view of the data but can be computationally intensive and sensitive to initial conditions. In contrast, the Top-down method efficiently identifies significant changes and is guided by domain knowledge, yet it may overlook fine-grained details and involve subjective criteria for splitting. Both these methods offer a hierarchical view of the data, which can be valuable when dealing with multi-scale phenomena often encountered in network delays.

The Pott's Model [117] is a statistical method that allows for the detection of change points in the time series, points where the statistical properties of the data change. This model can be particularly useful for network delay segmentation as it can identify shifts in delay patterns, offering insights into network performance and potential issues.

Clustering-based techniques offer yet another approach to time series segmentation. By treating each segment as an instance of a particular cluster, these methods allow the identification of common patterns across different segments of the data. These common patterns can then be used to better understand the behavior of network delays.

These classical models can be used for segmentation. Indeed, when a drift is observed in the estimated parameters, we could segment at the moment of change. Interestingly, the principles underlying these segmentation techniques resonate with those used in time series analysis and forecasting. For instance, methods like autoregressive model (AR), moving average model (MA), and autoregressive moving average model (ARMA) models, which consider past observations and errors to estimate future values, echo the principle of examining previous data points in the sliding window or hierarchical approaches. Autoregressive integrated moving average model (ARIMA) and seasonal autoregressive integrated moving average model (SARIMA) models, which account for trends and seasonality, align to identify change points and patterns in segmentation methods like Pott's Model or Clustering-based techniques. Additionally, State Space Models and the Kalman Filter, which make predictions based on hidden states, could be compared to determine the structure of network delays using segmentation.

Further, vector autoregression (VAR) and generalized autoregressive conditional heteroskedasticity (GARCH) models, which address the interdependency of variables and time-varying volatility respectively, can be useful in the context of network delays where interdependencies and variability are often present. Finally, tools like Prophet and recurrent neural networks (RNNs) or long short-term memory (LSTM), designed to handle strong seasonality and sequence prediction, underscore the essence of segmentation, which is to identify recurring patterns and predict future network behavior based on these detected patterns.

Using hierarchical clustering and Viterbi for delay segmentation Hierarchical clustering offers a systematic and intuitive approach to achieving this segmentation. The main advantage of hierarchical clustering is its ability to create a hierarchical structure or tree-like representation called a dendrogram. The dendrogram illustrates the clustering process, showing how individual delay points or segments merge at different levels of similarity. This hierarchical structure provides a visual depiction of the relationships and similarities between delay segments.

In the context of delay segmentation, hierarchical clustering can help identify different patterns or groups of delay behavior. By grouping similar delay values, it becomes

possible to identify periods of high delay, low delay, or distinct delay patterns within the time series. The hierarchical clustering, however, didn't incorporate the dimension related to temporal stability. One way to solve this problem was adding a post-treatment step using the Viterbi algorithm to smooth the results. The regularization to get more coherent and smooth segments was introduced using the transition probability matrix of the Viterbi algorithm. It favors the behavior of staying within the same state if the emission probabilities are close and changing the state in case of real distinct emission laws.

Chapter 3

Measurements and other datasets

In this chapter, we will present the different datasets used in this thesis and the purpose and specificities of each one of them. The first part is dedicated to a synthetic network delay simulator, that serves as a controlled ground truth environment, and the second part is dedicated to real-world datasets coming from an Internet measurement platform called RIPE Atlas.

3.1 Synthetic datasets

The necessity of synthetic dataset First of all, synthetic datasets provide a valuable source of ground truth. In fact, whether it is related to the completion or the segmentation problem, it can be challenging to find real-world datasets with ground truth values. Indeed, we can never know the value of missing measurements occurring in real scenarios. In addition, labeling the segments and the change points by hand is often a tedious task. That's why having datasets with ground truth is very important as they provide a realistic view of the algorithm's performance and help tune and enhance them.

Second, using synthetic data we have total control over the complexity and the nature of the patterns of the delay time series. We can change the distributions of the generated time series, and we can also decide on the level of spatial correlation within the dataset by controlling the rate of route segments shared by the different paths. One might note, that by distribution we mainly mean the overall behavior of the time series: the number of abrupt changes, the level of stationarity and the level of noise. In fact, the model described later in this section enables us to guarantee certain characteristics found in RTTs without exact control over the mathematical distribution of the time series. Additionally, we control the length of the time series providing us for instance with a time series long enough to capture some of the periodical characteristics of delays.

Finally, having total control over the network topology can enable us to test a large set of real-world network topologies and evaluate the impact of routing changes or routing

incidents. In this context, we opted for two topologies: GEANT and Renater. Additionally, it allows us to control the routing, as we can either opt for the shortest route strategy or for another routing scenario. This provides the possibility to implement easily routing changes, or congestion by changing weights on the route segments. It can also serve to represent failure or attack scenarios by removing one or more route segments.

Reproducing RTT characteristics There are some characteristics in actual network delay measurements that we want to reproduce which are the temporal stability and the spatial correlation. As it can be noticed in Figure 4.1, in many RTT series the delay is stable over several hours and changes occur abruptly. The statistical distribution of the delay during the stable periods displays a baseline and some random variations above this baseline. Indeed, the delay in a network has various components [83]. One component corresponds to signal propagation and packet processing at router interfaces, which accounts for the baseline delay. This baseline remains constant due to fixed geographical distances between the source and destination, as well as consistent processing times. Another component relates to queuing effects, responsible for introducing randomness into the delay. This randomness is dependent on the network's traffic load.

Additionally, when routing changes occur it often induces abrupt changes in network delays. Moreover, network protocols like OSPF (Open Shortest Path First) and BGP (Border Gateway Protocol) take some time to converge after a routing change. During this period, packets might be sent to less optimal and efficient routes leading to an increase in delays. Regarding the spatial characteristic, the delays between different (Origin, Destination) pairs of nodes are sometimes correlated with each other. Indeed, part of the paths between origins and destinations are shared, which creates spatial correlation.

3.1.1 The network delay simulator

Different tools for network emulation and simulation exist such as Mininet [56] and ns-3 [86]. Using these tools is beneficial in scenarios that require complex and feature-rich network simulations. However, using them for simple simulations might not be optimal considering their complexity, resource-intensive nature, and the time required to get familiar with the tool and its setup. We decided for this reason to build our own simulator since the main goal of the simulator is obtaining end-to-end delays that replicate realistic delay characteristics. As explained below, many simplifying assumptions are made to reduce the model complexity, making sure at the same time that the desired delay requirements are met.

The autonomous system model For our delay simulator we consider the case of a single Autonomous System (AS) which is modeled as a graph, the nodes of which are the routers. Some nodes can be the origin or destination of the traffic. In fact, an AS is a collection of IP networks and routers governed by a single entity and has a unique number assigned to it referred to as the autonomous system number (ASN). We opted for the AS model, as the internal network management and routing policy are autonomous and controlled by the AS entity. We consider that the entering traffic crosses the autonomous system (AS) by going from the entry border routers (sources) to the exit border routers (destinations). In this model, we will be interested in unidirectional full-mesh traffic. The unidirectional assumption is made in order to reduce the complexity of the model, however, this can be reflected in some real-world cases where the traffic is indeed unidirectional such as browsing and video streaming services where the traffic is mainly sent from the server to the user. So we consider that the entering traffic will be sent from all the sources to all the destinations. For simplicity reasons, we order the pairs $(source, destination)$ following the lexicographic order on the indexes of the sources and destinations.

The traffic generation We consider slotted time, where at the beginning of each time slot $k \in \mathbb{N}$, $T(k)$ denotes the traffic entering and leaving the autonomous system. The total traffic demand $T(k)$ may not be constant but may be a quasi-periodic term that takes into account daily variability. We assume that for every k:

$$T(k) = f(k) + \epsilon_k, \tag{3.1}$$

where:

— $f(k)$ is a deterministic function capturing the average traffic at the instant k,

— $\epsilon(k)$ is a zero mean normal random variable with finite variance.

After defining the total traffic volume present at each instant k in the network, we need to define the amount of traffic present on each path. In fact, the Origin-Destination (OD) traffic matrix represents the traffic demand (in bytes) between each origin node and each destination node. To generate this traffic matrix, we use a simple gravity model [132].

The gravity model Let us recall the gravity model. It assumes that the traffic between node i and node j is proportional to the product of two terms, one of them representing the proportion of traffic entering through origin node i, and the other representing the

proportion of traffic which exits through the destination node j. So, $T_{(i,j)} = T \times y_i^{(o)} \times y_j^{(d)}$ where $T_{(i,j)}$ is the traffic between nodes i and j and T is the total traffic demand. $y_i^{(o)}$ and $y_j^{(d)}$ are proportions so that $\sum_i y_i^{(o)} = \sum_j y_j^{(d)} = 1$ and $T = \sum_{i,j} T_{(i,j)}$.

In our approach, the time is slotted so that $T_{(i,j)}(k)$ (respectively $T(k)$) represent the traffic between nodes i and j (respectively, the total traffic demand) in a time window k (lasting a few minutes). Similarly, the previously defined proportions depend on k: $y_i^{(o)}(k)$ and $y_j^{(d)}(k)$.

The traffic per link Once the OD traffic matrix has been generated, we can deduce the traffic volumes on the links of the network. If one considers a particular link l then the traffic $T^l(k)$ through link l is the aggregation of all the origin-destination demands which routes go through link l.

So, $T^l(k) = \sum_{\{(i,j), l \subset P(i,j)\}} T_{(i,j)}(k)$ where $P(i,j)$ is the path from i to j. In our simulator, we have assumed that routes follow the shortest path, which is the most realistic scenario in normal network conditions. Other models to distribute the traffic exist such as the Frank-Wolfe algorithm [123] or the Dial's algorithm [114] which are two traffic assignment algorithms used to determine the traffic flow in the network considering factors like link capacity, and congestion and travel time. More recently, algorithms based on machine learning and reinforcement learning such as [113, 28] are used.

The need for abrupt changes Second, our goal is to simulate realistic time series, taking into account the spatial correlation but also the temporal correlation of the delays. In particular, we want to simulate abrupt changes in the statistical distribution of delays, as shown in Figure 4.1. In the case of an AS, such abrupt changes can be due to modifications of the external routing, which leads to changing the entry or exit points of part of the traffic [119]. Eventually, these changes in addition to the convergence time needed by the routers within the AS lead to changes in the corresponding network delays. So, in our simulator, the terms $y_i^{(o)}(k)$ and $y_j^{(d)}(k)$ of the gravity model are not constant.

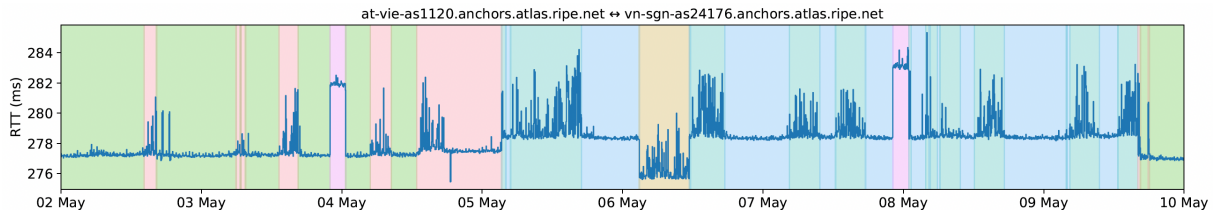


Figure 3.1 – RTT between two anchors on RIPE Atlas over 1 week [79].

They remain constant for a while, and then, according to Markovian dynamics, a change of state occurs. As the state of the Markov chain changes, the values of the proportions $y_i^{(o)}(k)$ and $y_j^{(d)}(k)$ change, while respecting the normalization constraints.

Abrupt changes generation The model we choose behind the abrupt changes is as follows: when no change occurs in the state of the network, the proportions of traffic corresponding to each origin and destination $y_i^{(o)}(k)$ and $y_j^{(d)}(k)$, as described in paragraph "Gravity model", do no change. When a change occurs, part of the sources or the destination witnesses a change share. For this purpose, we redistribute the shares between those selected nodes and keep the shares of the rest of the nodes untouched. The process of modifying the traffic proportion is in fact, for a given source or destination, the distribution shares evolve in time according to the following equation:

— If the state doesn't change: $y_i^{(o)}(k+1) = y_i^{(o)}(k)$

— If the state changes, we vary the traffic proportion on a set of selected origins :

$$y_i^{(o)}(k+1) = \left(I\{i \in N_{\mathcal{O}}(k)\} \frac{X_i^{(o)}}{\sum_{j \in N_{\mathcal{O}}(k)} X_j^{(o)}} (1 - \sum_{j \notin N_{\mathcal{O}}(k)} y_j^{(o)}(k)) + (I\{i \notin N_{\mathcal{O}}(k)\} y_i^{(o)}(k)) \right)$$

with $\mathcal{O}$ is the set of origins,denotes the indicator function. We also have:

— $X_i^{(o)} \sim \exp(I\{S_{k+1} = 1\}\lambda_1 + I\{S_{k+1} = 2\}\lambda_2)$, with $\lambda_1, \lambda_2 > 0$. S_k is the state of the Markov-chain at instant k

— $N_{\mathcal{O}}(k)$ is the set of nodes among the sources and $N_{\mathcal{D}}(k)$ is the set among the destinations that will be affected by the new traffic distribution. In order to determine these sets, we draw uniformly a number n_s from 1 to $|\mathcal{S}|$ and n_d from 1 to $|\mathcal{D}|$ respectively. Then we choose n_o nodes among $\mathcal{O}$ and n_d among $\mathcal{D}$ to modify their weights' distribution at the instant k using X_i.

The update of $y_i^{(d)}(k)$ follows the same process as the one $y_i^{(o)}(k)$.

This formulation was motivated by two factors. First, the overall stability of the network was modeled by a more likely state of the Markov chain that induces abrupt changes but forces the network to remain in this regimen for a long period of time. Then the fact that the traffic distribution was operated due to external factors such as a change in the routing, which explains the introduction of some randomness through X_i and through the construction of $N(k)$.

We will now prove that the proposed model follows a simple gravity model.

Lemma The proposed traffic distribution model is equivalent to a simple gravity model, and we have

$$T_{(o_i,d_j)}(k) = T(k) \times y_{o_i}(k) \times y_{d_j}(k) = T_i^{(o),in}(k) \times \frac{T_j^{(d),out}(k)}{\sum_{d_n \in \mathcal{D}} T_{d_n}^{out}(k)},$$

with $T_i^{(o),in}(k)$ referring to the total traffic entering the AS from source s_i and $T_j^{(d),out}(k)$ the total traffic leaving it through destination d_j.

Proof We will use an induction argument. The initialization is coming from the fact that the traffic conservation assumption ensures that $\sum_{o_i \in \mathcal{O}} y_i^{(o)}(0) = \sum_{d_j \in \mathcal{D}} y_j^{(d)}(0) = 1$. Let us assume that $\sum_{s_i \in \mathcal{O}} y_i^{(o)}(k) = \sum_{d_j \in \mathcal{D}} y_j^{(d)}(k) = 1$ holds.

— If $S_k = S_{k+1}$, then: $y_i^{(o)}(k) = y_i^{(o)}(k+1)$ and $y_j^{(d)}(k) = y_j^{(d)}(k+1)$ for each i and j. Hence, the equality is true for $k+1$.

— If $S_k \neq S_{k+1}$, then for nodes in $\mathcal{D}$: $\sum_{d_j \in \mathcal{D}} y_j^{(d)}(k+1)$
$= \sum_{i \in N_{\mathcal{D}}(k)} y_i(k+1) + \sum_{i \notin N_{\mathcal{D}}(k)} y_i(k+1)$
$= \sum_{i \in N_{\mathcal{D}}(k)} \frac{X_i}{\sum_{j \in N_{\mathcal{D}}(k)} X_j}(1 - \sum_{j \notin N_{\mathcal{D}}(k)} y_j(k)) + \sum_{i \notin N_{\mathcal{D}}(k)} y_i(k)$
$= 1.$

The same applies to nodes in $\mathcal{O}$. According to the traffic distribution model (3), we assume that at each instant k:

$$\begin{aligned} T_i^{(o),in}(k) &= T(k) \times y_i^{(o)}(k), \\ T_j^{(d),out}(k) &= T(k) \times y_j^{(d)}(k). \end{aligned} \tag{3.2}$$

with $T(k)$ following the formula defined earlier in the section "The traffic generation". Therefore, we can write

$$\begin{aligned} T_{(o_i,d_j)}(k) &= T(k) y_i^{(o)}(k) y_j^{(d)}(k) = T(k) y_i^{(o)}(k) \times T(k) y_j^{(d)}(k) \times \frac{1}{T(k)} \\ &= T_i^{(o),in}(k) \times \frac{T_j^{(d),out}(k)}{T(k) \times \sum_{d_n \in \mathcal{D}} y_{d_n}(k)} \\ &= T_i^{(o),in}(k) \times \frac{T_j^{(d),out}}{\sum_{d_n \in \mathcal{D}} T(k) \times y_n^{(d)}(k)} \\ &= T_i^{(o),in}(k) \times \frac{T_j^{(d),out}(k)}{\sum_{d_n \in \mathcal{D}} T_n^{(d),out}(k)}. \end{aligned}$$

The last line is the simple gravity formula, and this concludes our proof.

The delay generation The delay from node i to node j is then obtained as a sum of delays along the path $P(i,j)$. A delay is associated with each of the links using an $M/M/1$ queue model. In an $M/M/1$ model, the average delay of a link l with capacity C_l and offered traffic $T^l(k)$ is equal to $\frac{1}{\mu}\frac{1}{1-\rho}$. $\frac{1}{\mu}$ is the average service time, that is to say, the packet size divided by the bandwidth of the router interface, and does not depend on the offered traffic. $\rho = \frac{T^l(k)}{C_l}$ is the load and approximates the fraction of time a server is busy (and $\rho < 1$ since we assume that the queue is stable). We make the following assumptions:

— All the links are identical and have the same capacity C.

— There is no congestion in the network. This means that there is no packet loss due to traffic bottlenecks for instance.

— We consider that the load doesn't reach the link capacity which means $\rho = \lambda/\mu < 1$, where $1/\lambda$ is the average interarrival time of packets and $1/\mu$ is the average service time.

Then, in the simulator the delay $D_{ij}(k)$ from node i to node j is modeled by an M/M/1 and given by

$$D_{ij}(k) = \sum_{l, l \subset P(i,j)} \frac{\text{Packet Size}}{C_l} \frac{1}{1 - \frac{T^l(k)}{C_l}} \tag{3.3}$$

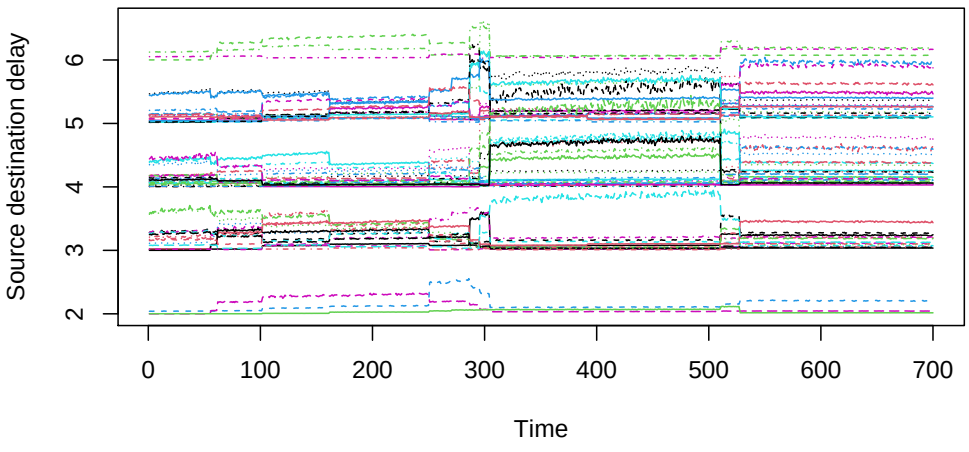


Figure 3.2 – Simulated delay time series.

3.1.2 The simulation setup

As an example of simulation, we used the model described previously with a network of 30 nodes: 6 sources, 6 destinations and the rest of the nodes served as transition routers. This provides 64 OD pairs. In order to define the corresponding paths, we chose the shortest path on the graph for each pair. For this we use the Dijkstra algorithm, as it is classically used for shortest path network routings and is more computationally efficient than the Bellman-Ford algorithm [71] We sample RTTs for OD pairs at 800 successive instants for our delay time series. In order to induce abrupt changes in the delay time series, we followed the two-state Markov model described earlier. In this example, we use the following transition matrix for our two-state Markov chain:

$$\begin{pmatrix} 0.99 & 0.01 \\ 0.01 & 0.99 \end{pmatrix}.$$

Such transition probability choice will force a piecewise-like behavior since the delay remains stationary for a long time until the occurrence of the next state change.

The goal, however, is to be able to try different network topologies and to simulate delays on networks that are big enough to reflect real-world scenarios. For this purpose, we simulate delays on two real-world topologies: Abilene and Renater. Abilene Network was a high-performance backbone US network created by the Internet2 community in the late 1990s. It is composed of 12 nodes and 15 physical links [60]. Renater [61] on the other hand, is made of 45 nodes and 54 links.

Figure. 3.2 displays some delay time series obtained from the simulator. It can be observed that the signals present abrupt changes, some of which are synchronized between several (Origin, Destination) pairs, with a baseline delay value over stable periods. For the simulations, we considered a network of 150 nodes with 8 origins and 8 destinations of traffic ($n = 64$ (Origin, Destination) pairs).

More details regarding the implementation of the simulator are found in the corresponding github repository [84].

3.2 Real-world datasets

In this section, we analyze delay measurements from the RIPE Atlas platform. RIPE Atlas [97] is a global and public measurement platform held by RIPE NCC, that provides

data on Internet network availability. It contains multiple types of Internet measurements, including delays, traceroutes, DNS, SSL, HTTP, and NTP. One can either get access to the results of past and running measurements or launch his own measurement campaign. A detailed tutorial to launch a measurement campaign is provided in the appendix. In order to provide wide geographical coverage, Ripe Atlas has more than 9000 probes and 800 anchors distributed as shown in Figure 3.3. Ripe probes are small hardware devices able to perform network measurements, Ripe anchors on the other hand are an evolved version of the probes able to conduct measurements with a higher stability and capacity and provide valuable information about Internet connectivity on a regional level. Measurements conducted on both types of devices are aggregated and made publicly available. Ripe Atlas uses these measurements for Internet monitoring purposes but also provides users with public access to many tools like IXP Jedi to find if measurements cross an IXP, or RipeIPmap to find an IP geolocalisation. In this work, we have used Ripe datasets for both the completion and the segmentation problem, but for each research topic, we opted for specific datasets with convenient properties. For the completion problem, we wanted

Figure 3.3 – Ripe Atlas probes distribution over the world. In green are the connected probes, in yellow are the disconnected ones, and in grey are the ones that have been abandoned [97].

first to get an idea about the real rate of missing data in real Internet scenarios such as the one encountered in the Ripe Atlas platform [107] as well as how these missing values are distributed in time and in space, then we needed to evaluate the accuracy of our models

in inferring the missing values. For the first criterion, we needed to run measurements on randomly chosen probes so that we could get a realistic view regarding their availability and response, for the second criterion we needed robust and highly stable available probes so that we could have a complete ground truth to evaluate our results.

Regarding the segmentation problem, first, we wanted to target time series with varying patterns over time, second, we want to explore the spatial correlation between delay time series, finally, we want these datasets to be as complete as possible. For this purpose, we consider for time series of pairs passing through the same Internet infrastructure to reflect the spatial dependency. In fact, we assume that paths sharing significant route segments will display similar changes in performance if these segments are touched by an event or an outage. In this context, we are interested in monitoring an IXP. For this reason, when a period of time contains big Internet incidents like outages, pairs passing through the same damaged IXP display synchronous varying patterns. Finally, to assure the stability of delay measurements we opted for the anchor mesh of Ripe Atlas.

In particular, in Chapter 5, to highlight the potential use of segmentation to analyze incidents over the Internet, we chose two datasets of delays measured between Ripe anchors. The first dataset contains pairs of anchors passing through DE-CIX over the period of an incident in 2018 and the second contains pairs passing through AMS-IX over the period of an incident in 2015.

3.2.1 How to conduct measurements on Ripe Atlas ?

In order to conduct measurements on the Ripe Atlas platform, several steps need to be followed:

1. Create an account in the platform and log in
2. Get Ripe Atlas credits. To get credits, a person can host a probe, get a credit transfer, or ask the Ripe moderators for a token.
3. In the measurements section, we specify that we want to create a measurement
4. Then, we select the measurement type, the frequency, and the destination see Figure 3.4. If more than one target or destination is needed, we simply add a new measurement. Since there is one destination per measurement, many measurements are possible within a single campaign.
5. Later we specify the sources of the measurement. These can be selected randomly, picked from a given map as in Figure 3.10, or specified according to predefined

selection. In this case, we provide probe identifiers.

6. Then, we should specify the period of the measurements as seen in Figure 3.6.
7. Finally, the campaign can be launched.
8. Once the measurements are running or ended, we can fetch them by going to measurements and then selecting the section that indicates 'mine'.

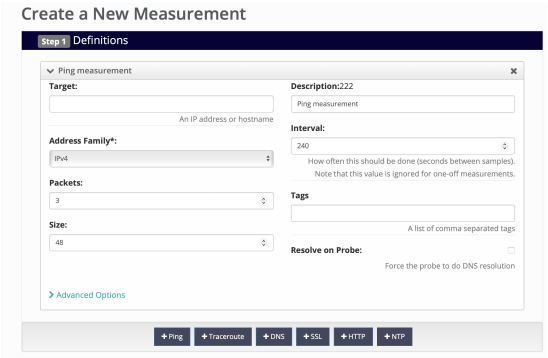


Figure 3.4 – Defining the measurement type and the target [97].

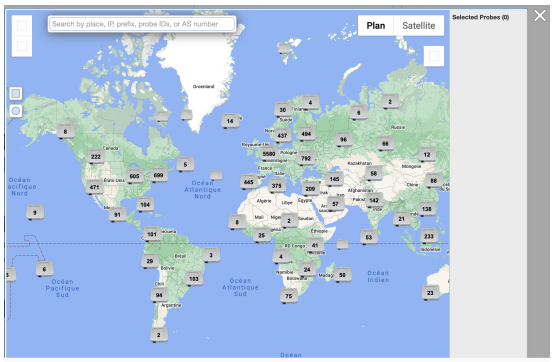


Figure 3.5 – Selecting source probes from a map [97].

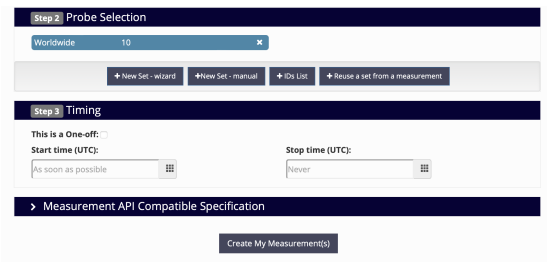


Figure 3.6 – Providing sources and the campaign duration [97].

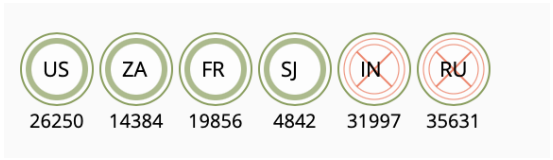


Figure 3.7 – Few probes participating in the campaign with 2 non-responding (red) [97].

3.2.2 Completion datasets

Randomly chosen anchors Our dataset corresponds to delay measurements conducted every four minutes between a set of anchors chosen randomly around the world. In this section, the measurement campaign covers a period of one week starting from the 11th of January 2022. These delays are collected using three ICMP pings, and the minimum value is saved for each timestamp. The minimum value reflects the optimal

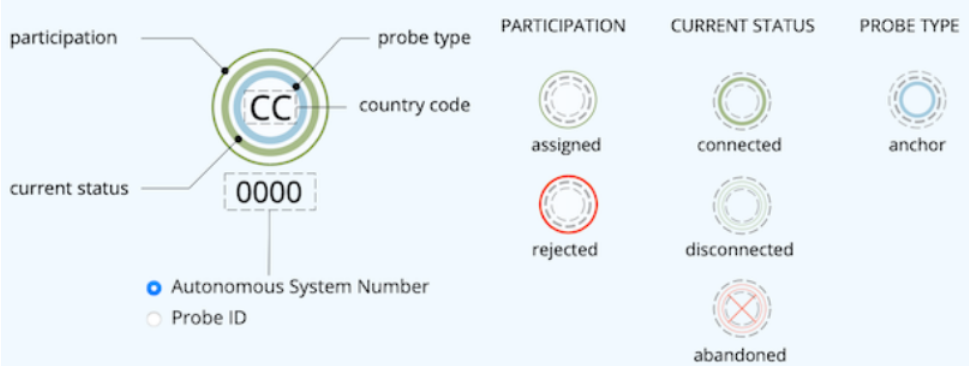


Figure 3.8 – Legend key of a Ripe Atlas probe [97].

performance of the network and is the value encountered in normal circumstances since one of the three pings can get higher values due to lost packets for example. This provides a dataset of 720 RTT series of 2520 time slots each. Figure 3.7 contains 6 of the participating probes, and we can see that 2 of them, located in India and Russia, were unavailable for this measurement.

In this dataset, the overall missing delays represent 25% of the measurements. We can observe on the heatmap Figure 3.10 that missing values are not distributed uniformly. Patterns vary from one (Origin, Destination) pair to another. Missing measurements may be due to a dysfunction of the origin or destination anchor, or in a device located on the taken route and such problems can be temporary or permanent. Only (Origin, Destination) pairs with a rate of missing data lower than 80% were kept for the experiments. Recovering missing delays is done based on the information and redundancy of captured delays. For this reason, we aim to get rid of the bias that can be induced through present outliers. These outliers can happen due to devices, probes or anchors, dysfunction, or packet loss for instance. In order to remove the outliers, we do the following: for each (Origin, Destination) time serie, we have considered as outliers the points that are above (resp. below) $\mu + 2\sigma^2$ (resp. $\mu - 2\sigma^2$) where μ is the mean of the known delay values and σ^2 is their variance These values were removed and considered as missing delays. After applying this treatment, the size of the conserved matrix D is 572×2520 and the proportion of missing values is 18% (i.e. $S_{ij} = 0$).

Groundtruth dataset The dataset in Figure 3.9 comes from a measurement campaign on Ripe Atlas. Unlike in the work [37], we wanted to have a ground truth for all the entries of our delay matrix. To this end, we searched for anchors belonging to some

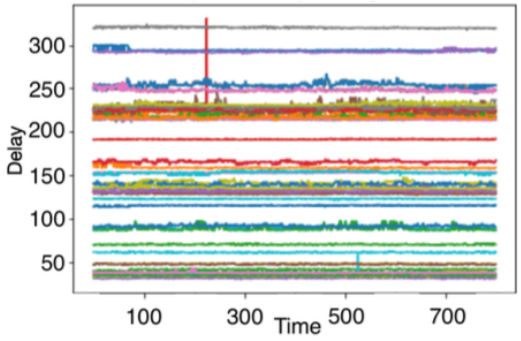


Figure 3.9 – Time series from the ground truth dataset.

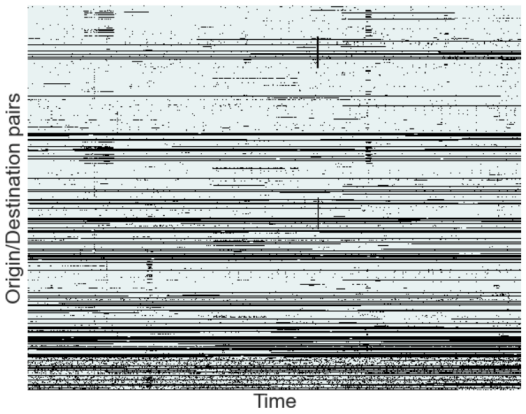


Figure 3.10 – Heatmap showing the missing measurements (in black) in Ripe data.

Google ASs in the Ripe Atlas database using Google autonomous system number (ASN). Then we selected 14 probes among them belonging to different regions around the world. The selected probes and their location are given in Figure 3.11 using the country code, with Figure 3.8 explaining each component of the legend. They are prone to be more stable and have a higher probability of being connected and responding to requests. The rationale behind this assumption is that Google equipment would have more chances to be maintained and fixed in case of connectivity or responsiveness problems. This was actually the case since we did not have missing delays on this campaign.The dataset contains 50 RTT time series of length 800 each corresponding to 22 hours of measurements. The RTT measurements are given in ms.

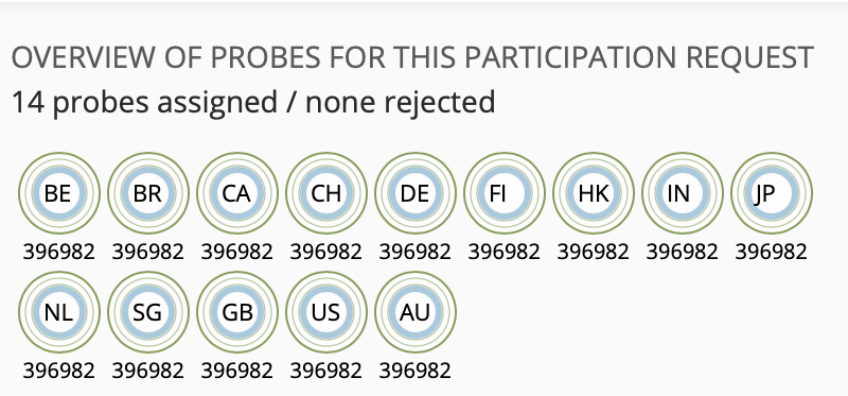


Figure 3.11 – The probes selected for the Google dataset [97].

3.2.3 Segmentation datasets

In order to have a spatially correlated time series, we wanted to make sure that our origin-destination (OD) paths have a shared route segment. For this purpose, we decided to filter ODs that pass through a given IXP and we define below the used strategy. A reason for choosing to filter through an IXP, is its importance in the overall traffic and the possibility of having a strong correlation, especially in case of global Internet incidents.

Filtering path going through an IXP In order to get delays between pairs passing through a specific IXP, we need to make use of traceroute measurements. Which provides a map of how data on the internet travels from its source to its destination. In fact, we run traceroute measurements between all Ripe anchor pairs for 24 hours, then we investigate every IP in these measurements. If an IP belongs to the desired IXP, by being part of its netmask we keep the pair of anchors. We run the traceroute campaign for 24 hours so that we don't miss a given pair which can happen for several reasons: routing changes, non-responding devices, hidden IP addresses etc. Once the pairs of anchors are selected, we then run our delay campaign or fetch existing measurements between them for the desired period. This selection and filtering process was done using Python.

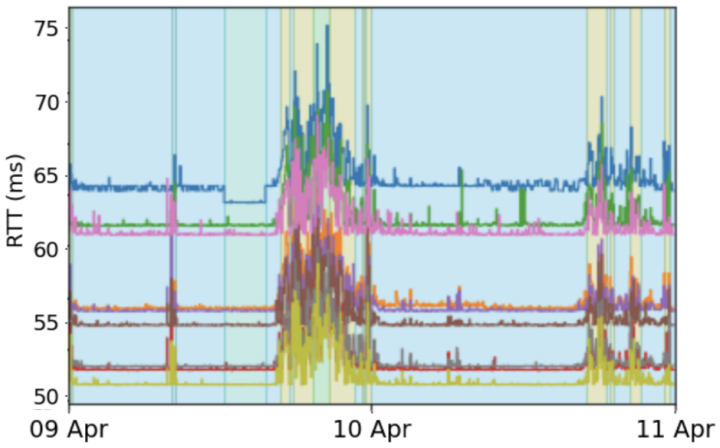


Figure 3.12 – 10 Delay time series passing through DE-CIX from April 9th to April 10th 2018.

RTT traces passing through DE-CIX Between April 9th and April 10th 2018, DE-CIX Frankfurt experienced a disruption of its network connectivity to route servers, resulting in rerouted traffic and an interruption of traffic. This was evidenced by an analysis of the rates of BGP updates, which are messages to advertise routing information

such as path attributes and prefixes or withdraws previously advertised routes, received by route collectors located at DE-CIX, which dropped close to zero between 19:43 and 23:28 on the 9th of April, and between 02:02 and 03:51 on the 10th of April, as reported in [22]. This dataset contains 38k pairs passing through DE-CIX. Each of these time series has 720 timeslots which correspond to two days of measurements, the 9th and 10th of April 2018. Figure 3.12 represents 10 delay time series taken from this dataset. We can observe the delay disruption during the incident interval

RTT traces passing through AMS-IX On May 13th, 2015, AMS-IX experienced a seven-minute, two-second partial outage due to a switch interface generating looped traffic on the peering LAN. This resulted in some peers at the exchange losing their BGP session, as reported by [23]. The outage lasted between 10:22:12 and 10:29:14 UTC before the switch interface was disconnected. This dataset contains 26k pairs passing through AMS-IX. Each has 360 timeslots corresponding to the 13th of May 2015.

LINX incident A more recent incident happened to the LINX, an IXP in London. The outage took place between the 23rd and the 25th of March 2021. On March 23rd, LINX LON1 experienced two brief service degradations due to technical issues. A significant traffic loss occurred during diagnosis. On March 24, the LON1 IXP continued to operate stably, and a maintenance window was set for intrusive testing. The LON1 issue was ultimately resolved by March 25, allowing the provisioning of new connections [64].

3.3 Conclusion

This chapter describes the different datasets used in this thesis. It underscores the importance of having both synthetic and real-world datasets. The simulator enables us to have a solid ground truth and a controlled environment, whereas real delays provide valuable insights about scenarios encountered practically in network monitoring. In the next chapters, we test and evaluate different methods and algorithms on these datasets to address two problems: the completion and segmentation of network delay. Looking ahead, it is essential to further expand our dataset collection, especially by incorporating real-world incident data from various Internet Exchange Points (IXPs). Additionally, we should explore different network topologies and simulate a wide array of incidents within our simulator to enhance the comprehensiveness of our study.

Chapter 4

Network delay completion

Contributions: Developed two highly accurate methods for network delay completion:

— The first is based on non-negative matrix factorization.

— The second is based on neural collaborative filtering.

These approaches have yielded remarkable accuracy rates of more than 98%.

4.1 Problem: introduction and motivation

4.1.1 The problem of missing delays within the Internet

In this section, we first underline the importance of monitoring network delays at a fine granularity. Then we show that being able to do so is often challenging due to multiple factors such as device problems and restrictions on measurement policies. Finally, we demonstrate that treating the issue as a matrix completion problem enables us to have a good estimate of the unobserved delays.

The importance of delay measurements in networking

Internet delay measurements have wide applications and are crucial for proper network monitoring. With the rise of time-sensitive applications such as gaming, live video streaming, financial trading, or videoconferencing, delays are generally crucial to ensure the right functioning of these applications and to guarantee a good experience for the end user. On a general level, delays are continuously used to help network administrators measure network QoS and indicate the network state so they can be able to pinpoint potential congestion or high latency areas that require optimization. Moreover, in distributed services knowledge of delays helps to improve responsiveness in content distribution networks

by choosing the most suitable communication peer and enhance load balancing for overlay routing by sending packets to less congested routes. For all these reasons, full knowledge of network delays sounds attractive. However, conducting on-demand measurements regularly on the network is costly and time-consuming. Actually, for a full-mesh network of n nodes, the number of all possible delays is n^2 which grows quadratically with the network size and can constitute a serious limitation as this latter grows [55]. In fact, injecting many probes in the network for the sole purpose of monitoring can disturb the overall traffic leading to a bias in the measured performances. Moreover, congestion, packet loss, network outage, or device issues are also contributing factors to missing measurements. Consequently, whether it is due to restrictions imposed by measurement policies or incidents caused by network anomalies, being able to predict missing delays from limited available measurements is necessary.

The spatial and temporal correlations of network delay measurements

The network delay prediction problem is well-posed due to the high correlation that can be found within network delays. Indeed, this correlation is twofold: first, delay measurements between different origins and destinations are spatially correlated with each other. Second, when a single delay is observed during a sufficiently long period, it displays a strong temporal correlation. Regarding the spatial correlation, delays having the same source or the same destination, show some similar pattern, especially in the case of a device deterioration scenario. Similarly, measurements going through paths that share some route segments will possibly display close patterns. This can be especially the case if the shared segments contain important Internet infrastructures such as autonomous systems (AS) or Internet exchange points (IXP) or if these segments witness specific network behaviors such as incidents or outages [80]. Spatial correlation can also happen in the case of congestion where the congested neighboring routers get affected by each other [32]. For the temporal correlation, if one considers a particular pair of nodes, the delay is generally stable over long periods of time, resulting in temporal stability. It is explained by the fact that a determinant aspect of the delay values is the geographical distance which remains stable over time. Besides, when observed over a long period, some periodic behaviors can be observed [75] due to periodic traffic changes, congestion, and route changes [88]. An example of this is the diurnal property of the Internet, linked to users' consumption and behavior [4]. Having these correlations, both spatial and temporal [75], makes it easier and possible to approximate missing measurements from existing ones,

which can help reduce the number of conducted measurements and overcome network hazardous events.

4.1.2 The datasets architecture

We consider that time is slotted with time slots of $\sim$ 5 minutes, a common measurement frequency used on Internet monitoring platforms such as Ripe Atlas. An RTT matrix over a network of N nodes at a given instant t is a $N \times N$ matrix $X(t)$ where $X_{ij}(t)$ is the delay between the origin node i and the destination node j at instant t. The delay matrix has many characteristics, including positive entries and low effective rank. The low effective rank property in this case results from spatial correlation. Indeed, some nodes, such as nodes in the same AS, often share route segments and therefore exhibit similar performance. Moreover, successive delays between the same pair of origin and destination are correlated through time since the delay is stable over long periods of time on a particular path. For instance, Figure. 4.1 displays the time series of Round Trip Times (RTT) between two anchors on RIPE Atlas [97] during one week. We can observe that the delay is stable for hours and abrupt changes in the statistical distribution of the delay occur due to routing changes (see for example [80]). In order to take advantage of this property, we will construct a matrix D where the t-th column is given by $D_t = vec(X(t))$ and $vec(.)$ is the operator that reshapes a matrix into a column vector. Therefore, a row of D represents successive RTTs for a (Origin, Destination) pair, and a given column represents an instant. The matrix D is of size $n \times m$ where n is the number of node pairs, and m is the number of time slots. Spatial correlation among origin-destination pairs and the temporal stability of the rows of this matrix contributes to the low-rank property. This property constitutes a sufficient condition for a proper matrix factorization [18].

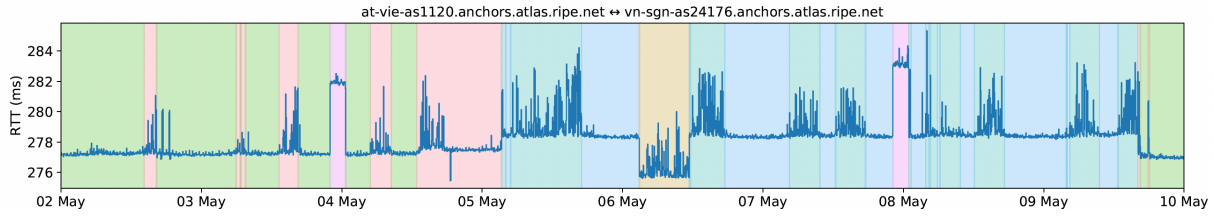


Figure 4.1 – RTT between two anchors on RIPE Atlas over 1 week [79].

4.1.3 The matrix completion problem

Given the low-rank property of the measurements, one natural way to use these correlations is to formulate our delay inference problem as a matrix completion problem [14]. Often when having noisy data, accurately determining the true rank of a matrix can be challenging. To ascertain whether the matrix exhibits a low-rank property, a common approach is to examine its singular values. When the singular values decrease rapidly, this characteristic strongly indicates the presence of a low-rank structure within the matrix. In fact, when the singular values decline significantly as we move along the diagonal of the diagonal matrix in the singular value decomposition (SVD), it means that only a few singular vectors are responsible for most of the data's variation. This phenomenon indicates that the matrix can be well-approximated by a lower-rank version. The matrix completion problem is defined as follows: we consider a given matrix M of dimension $m \times n$ and we assume that M is partially observed. The goal of the matrix completion is to estimate a new low-rank matrix that approximates M on the observed values and recovers its unobserved entries. The objective of this estimation is to minimize the reconstruction error, generally calculated between the observed entries of M and the entries with the same index of the new estimates matrix $\tilde{M}$

4.2 State of the art and suggested models

4.2.1 Existing models

Network delay estimation has first been addressed in the context of network coordinate systems using either Euclidean embedded models [21, 85] or matrix factorization [70, 63, 133]. Euclidean embedded approaches are embedding network nodes in a low-dimensional space. In this space, a good approximation for the delay between nodes is assumed to be equal to the Euclidean distance. Such approaches decrease the number of end-to-end delays needed for the estimation of distances. However, their major drawbacks are that they are limited by geometrical constraints (triangle inequality, symmetry), and they are not efficient in the presence of complex routing policies [121].

To overcome this problem, matrix factorization approaches gained a lot of interest. These methods are often based on the low-rank approximation of the delay matrix. Many models use nuclear norm for rank minimization and others use non-negative matrix factorization (NMF) exploiting the positivity of delays [69, 16]. However, these works applied

NMF to network coordinates rather than delays and presented algorithms with multiplicative updates which are different than the one proposed in this work. Matrix and tensor completions have also been recently used to estimate the traffic matrix [42, 124, 125] and to infer top-k Elephant flows [126].

The estimation of network delays with completion approaches has been studied in multiple recent works [120, 73, 50]. An adaptive completion algorithm has been proposed in [120] to estimate network delays. Moreover, in [73], authors have studied an efficient probing strategy of (Origin, Destination) pairs to improve the performance of completion. Finally, the effect of graph-Laplacian regularization on the performance has been studied in [50]. Such regularization is used to add more non-linear interactions between network delays.

In recent years, matrix completion has gained a lot of popularity with applications ranging from recommender systems, and image and audio reconstruction, to networking and genomics [93]. Until recently, the matrix completion problem was resolved using matrix factorization approaches [15]. Lately, other non-linear methods have appeared to solve this problem, such as neural networks. These approaches can be more interesting in terms of computational cost and their ability to capture the complexity of high dimensional data [45].

4.2.2 Suggested models

We suggest two approaches: the first one is based on non-negative matrix factorization and suggests 3 variations of the method, and the second approach is inspired by neural collaborative filtering and is based on two neural network architectures. The proposed models introduce innovation to the field of network delay estimation by departing from traditional approaches such as matrix factorization and Euclidean embedded models. With a focus on non-negative matrix factorization (NMF) and neural network architectures, these models offer variations in NMF and a shift towards computational efficiency, making them adaptable to diverse network scenarios and capable of handling high-dimensional data. Additionally, it extends the applicability of matrix completion techniques and collaborative filtering, previously employed in fields like recommender systems and genomics, into the domain of network delay estimation. In the non-negative factorization problem, we define the network delay prediction problem as a piecewise constant non-negative matrix factorization to incorporate expert knowledge into the completion problem [104]. Firstly, the non-negative constraints on the coefficients of the factorized matrices is consistent

with the fact that network delays cannot be negative. Secondly, we are looking for a piecewise constant factorization to encode the temporal correlation.

The second approach addresses the delay matrix completion problem from a neural network perspective by using a neural collaborative filtering (NCF) approach [45]. Collaborative filtering [59] is a widely used technique in recommender systems. It gathers data from other users in order to identify similarities between them. This helps the recommender system to predict missing users' preferences. Neural collaborative filtering, in particular, makes use of the flexibility and complexity of neural networks to accomplish this recommendation task.

In the following, we provide more details regarding both models.

4.3 Non-negative matrix factorization (NMF)

4.3.1 Problem formulation

Non-negative matrix factorization (NMF) consists of approximating the matrix D as the product of two low-rank matrices $W \in \mathbb{R}_+^{n\times r}$ and $H \in \mathbb{R}_+^{r\times m}$ with $r \leq min(n, m)$. To derive this approximation, we solve the following constrained optimization problem: $\min_{W,H} \|D - WH\|_F$ subject to $W \in \mathbb{R}_+^{n\times r}$ and $H \in \mathbb{R}_+^{r\times m}$ where $\|.\|_F$ is the Frobenius matrix norm.

Let $D_{\cdot j} = (D_{ij})_{1\leq i\leq n}$ denotes the column vector of delays at time j. Then $D_{\cdot j} \simeq \sum_{k=1}^{r} W_{\cdot k} H_{kj}$ with $W_{\cdot k}$ the k^{th} column of matrix W. The columns of W are the basis vectors of the decomposition of the matrix delays, while the lines of H indicate how each basis vector contributes to the delay at time j.

In real-world applications, network performance is often only partially monitored (in time or space). This fact implies that in practice, only part of the elements of D are observed. Moreover, it is possible that some devices are not replying to the probes, and therefore, are producing extra missing values in D.

The idea behind matrix completion is to approach D solely on the basis of available measurements. We introduce a $n \times m$ binary matrix S, named the sampling matrix, where $S_{ij} = 1$ if D_{ij} is known and $S_{ij} = 0$ otherwise. The optimization criterion then becomes: $\min_{W,H} \|S \circ (D - WH)\|_F$ subject to $W \in \mathbb{R}_+^{n\times r}$ and $H \in \mathbb{R}_+^{r\times m}$, where $\circ$ is the Hadamard product (term by term product of matrices).

In order to take advantage of the time stability observed among the column of D, we

add an additive regularization term $\beta \sum_{i=1}^{r} \sum_{j=2}^{n} |H_{ij} - H_{i(j-1)}|$ to $\|S \circ (D - WH)\|_F{}^2$. This permits to favor solutions for which delays are more stable over time. β is a weighting hyperparameter that controls temporal smoothness. We arrive at the optimization problem:

$$\min_{W,H} C(W,H) \text{ s.t. } W \geq 0, H \geq 0, \tag{4.1}$$

where the optimization criterion can be split into two terms:

$$\begin{array}{rll} & C(W,H) & = F(W,H) + L(H), \\ \text{with} & F(W,H) & = \|S \circ (D - WH)\|_F{}^2, \\ \text{and} & L(H) & = \beta \sum_{i=1}^{r} \sum_{j=2}^{n} |H_{ij} - H_{i,j-1}|. \end{array} \tag{4.2}$$

4.3.2 Solution of NMF factorization

In the next sections, we consider two algorithms that can be used to solve the optimization problem (4.1). The first one is an alternating projected gradient algorithm [104]. And the second one involves Nesterov's optimal gradient method [41].

4.3.3 Alternating projected gradient

The alternating projected gradient algorithm is an iterative algorithm where estimates of W and H are updated sequentially. Each iteration k can be decomposed into two steps: i) a steepest descent method is used to update W with $H = H^{k-1}$ fixed and with a projection over the set $W \geq 0$ leading to the update W^k, and ii) a steepest descent method is used to update H with $W = W^k$ fixed and with a projection over $H \geq 0$:

$$\begin{array}{l} W^{k+1} = [W^k - \alpha \nabla_W F(W^k, H^k)]_+ \\ H_{ij}^{k+1} = [H_{ij}^k - \alpha [\nabla_H F(W^{k+1}, H^k)]_{ij} + \frac{\partial L(H)}{\partial H_{ij}}]_+, \forall (i,j) \end{array} \tag{4.3}$$

With these equalities indicating the matrix equality entrywise, $[x]_+ = max\{0, x\}$ and $\alpha \geq 0$ is a step-size. The mathematical closed expression of the first order derivatives $\nabla_W F(W,H)$, $\nabla_H F(W,H)$ and $\frac{\partial L(H)}{\partial H_{ij}}$ are:

$$\begin{array}{l} \nabla_W F(W,H) = S \circ ((WH - D)H^\top) \\ \nabla_H F(W,H) = W^\top (S \circ (WH - D)) \\ \frac{\partial L(H)}{\partial H_{ij}} = \beta(\text{sign}(H_{ij} - H_{i(j+1)}) - \text{sign}(H_{i(j-1)} - H_{ij})) \end{array} \tag{4.4}$$

4.3.4 NeNMF: Nesterov gradient

The NeNMF applies the Nesterov optimal gradient. The NeNMF overcomes NMF solver's limitations such as numerical instability, slow convergence, and theoretical convergence problems. In addition, it achieves the optimal convergence rate $\mathcal{O}(\frac{1}{k^2})$ by using the Nesterov accelerated gradient [82] when focusing on a convex problem. The NMF is not a convex optimization problem, still, it is known that NeNMF is usually faster than standard gradient descent even if no theoretical proof exists for non-convex problems. The NeNMF consists of updating sequentially W^k and H^k at each outer iteration k by running an inner loop of Nesterov accelerated gradient method to approximately minimize the objective function with respect to W with H^k fixed (and vice versa). In fact, in order to estimate H^k, Nesterov gradient method constructs two sequences $\{H_t\}$ and $\{Y_t\}$ and updates them at each iteration round t :

$$\begin{aligned} &H_t = \min_{H\geq 0}\{\phi(Y_t, H) = F(W^k, Y_t) + \langle \nabla_H F(W^k, Y_t), H - Y_t\rangle + \tfrac{L_c}{2}\|H - Y_t\|_F{}^2\} \\ &Y_{t+1} = H_t + \tfrac{\alpha_t - 1}{\alpha_{t+1}}(H_t - H_{t+1}), \end{aligned} \tag{4.5}$$

where $H \mapsto \phi(Y_t, H)$ is a quadratic majorant function of $H \mapsto F(W^k, H)$ at Y_t [106]. $\langle\cdot,\cdot\rangle$ is the matrix inner product, $L_c = \|W^\top W\|_2$ is a Lipschitz constant of the gradient of the objective function $H \mapsto F(W^k, H)$ and Y_t is a linear combination of the two last approximate solutions, i.e. H_{t-1} and H_t. The coefficient α_t is updated at each iteration according to the formula $\alpha_{t+1} = (1 + \sqrt{4(\alpha_t)^2 + 1})/2$.

When solving the first-order optimality conditions (KKT conditions) for the convex optimization problem (4.5), the previous equations can be rewritten as:

$$\begin{aligned} &\text{Step 1:} \quad H_t = [Y_t - \tfrac{1}{L_c}\nabla_H F(W^\top, T_k)]_+, \\ &\text{Step 2:} \quad Y_{t+1} = H_t + \tfrac{\alpha_t - 1}{\alpha_{t+1}}(H_t - H_{t+1}). \end{aligned} \tag{4.6}$$

This algorithm doesn't use the penalization $L(H)$, and therefore, in the rest of the chapter, when we refer to the NeNMF algorithm, we implicitly assume that $\beta = 0$ (i.e. $C(W, H) = F(W, H)$).

4.4 Performance evaluation of NMF

To start our study, we work on synthetic data. It provides a ground truth that will be used to compare the results of the completion algorithms. Then we test the method

on a real-world dataset from Ripe Atlas. This shows that NMF can adapt to real missing delay patterns.

4.4.1 Matrix completion results on synthetic datasets

The matrix D is obtained by vectorizing the (Origin, Destination) delays and by concatenating these vectors of delays at successive periods. The low-rank property can be justified through the fast decrease of its ordered singular values. In fact, the 5 greatest of the 64 singular values are 409.13, 10.12, 5.45, 3.65 and 2.45. So a small value of r is sufficient to capture most of the energy of the matrix D.

To evaluate the performance of matrix factorization algorithms we consider the convergence stress [66]. It measures the quality of reconstruction of the missing entries of D_{ij} (i.e. such that $S_{ij} = 0$). It is defined as:

$$P^k_{\text{stress}} = \sqrt{\frac{\sum_{i,j}(1 - S_{ij})(D_{ij} - [W^k H^k]_{ij})^2}{\sum_{i,j}(1 - S_{ij})(D_{ij})^2}}, \tag{4.7}$$

where k is the iteration number in the matrix factorization algorithm.

Impact of the hyperparameter β Figure 4.2 represents the evolution of the convergence stress of the alternating projected gradient algorithm as a function of the number of iterations. As a first study, we consider the first 100 and 10000 iterations and two different values $\beta = 0$ and $\beta = 0.4$. The sampling rate is 0.7 that is to say that 70% of the delay values D_{ij} are known (i.e. $S_{ij} = 1$) and 30% of the delays are supposed to be unknown. These graphs aim to give an idea of the algorithm behavior with and without the regularization term. A more detailed study of the impact of β is provided taking into account a larger set of values.

We can notice that the stress decreases rapidly in the first iterations. But one can also notice oscillations of the stress value over the long term when $\beta \neq 0$. These oscillations are due to the choice of an $L1$ norm in the penalty term $L(H)$. Indeed the subgradient $\frac{\partial L(H)}{\partial H_{ij}}$ takes only three values $(2\beta, 0, -2\beta)$ and there are some abrupt changes in the value during iterations. Another algorithm to tackle the oscillations due to the piecewise regularization would be to do alternate minimization, instead of alternating gradient descent, using standard solvers. However, this approach could have two major drawbacks. One is that the algorithm can get stuck in bad local minimum; and also because we alternate between solving two optimization problems, the time for the algorithm to converge will be huge.

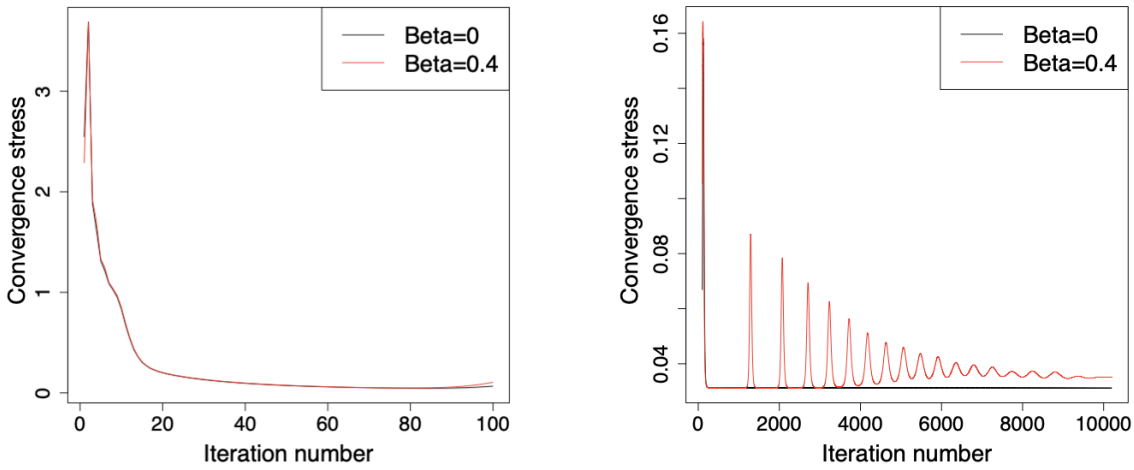


Figure 4.2 – Evolution of the convergence stress over the first 100 and 10000 iterations of the alternating projected gradient algorithm.

$L2$ norm would as well probably lead to a smoother evolution of the stress. Table 4.1 gives the influence of β on the stress value for the alternating projected gradient (APG) with a 30% missing data and after 150.000 iterations. Choosing $\beta \in [10^{-2}, 5.10^{-2}]$ yields a stress smaller than 2%.

Table 4.1 – Impact of β on the stress.

β	0.001	0.01	0.05	0.1	0.3	0.5	0.7
Stress	0.017	0.015	0.015	0.019	0.018	0.017	0.019

Impact of the rank r We are also interested in the impact of rank r on the quality of the reconstruction when $\beta = 0$ and when the percentage of missing data is 30%. Table 4.2 gives the stress values of the APG once the algorithm has converged (after 150.000 iterations) and of the NeNMF (after 10.000 iterations). It is natural to see that stress improves with a higher rank, as this allows for a greater number of basis vectors, thereby enhancing the model's capability to capture more complex patterns within the data. However, a higher rank may also result in longer convergence times due to the larger size of the matrices involved.

Impact of the sampling rate We have also assessed the influence of the sampling rate. For values ranging from 50% to 95% of delays known (i.e. $S_{ij} = 1$), Table 4.3 gives the value of the stress after 150000 iterations of APG, 10000 iterations of the NeNMF method and for $\beta = 0$.

Table 4.2 – Impact of rank r on the stress.

Rank r	1	2	3	4	6	8	10
Stress APG	0.031	0.031	0.014	0.011	0.007	0.005	0.003
Stress NeNMF	0.031	0.021	0.016	0.013	0.009	0.007	0.007

Table 4.3 – Impact of sampling rate on the stress.

Sampling rate	50%	60%	70%	80%	90%	95%
Stress APG	0.020	0.015	0.015	0.014	0.014	0.014
Stress: NeNMF	0.014	0.013	0.013	0.013	0.012	0.012

Evaluation of the execution time Finally, it is interesting to observe the acceleration provided by Nesterov's method in the optimization of the criterion $F(W, H)$. In Figure 4.3, the value of the stress over the first 100 iterations of the projected alternating gradient and of the NeNMF algorithm is represented. While the projected alternating gradient exhibits slow convergence and numerical instabilities, the NeNMF algorithm converges within fewer iterations. We also evaluated the execution time of the alternating gradient method over 100.000 iterations and the NeNMF method over 1000 iterations. We used a $2,6$ GHz Intel Core i7 processor with a 32Go 2667 MHz DDR4 memory. For the alternating gradient, the execution time was 16 minutes and for the NeNMF it was 472s. Such a difference in convergence time confirms the acceleration property of the NeNMF. When considering real-world data in the following section, this execution time difference leads us to opt for the NeNMF algorithm.

Importance of the positivity constraints In Figure 4.4, we see the reconstruction of a 80% observed synthetic dataset using both SVD and NMF. We can notice that the reconstructed delays are positive using both methods. However, we can see that the reconstruction using SVD displays more noise and variability, whereas NMF provides more accurate results corresponding to the ground truth dataset.

In Figure 4.5, we observe the reconstruction of a synthetic dataset with a 50% observation rate using both SVD and NMF techniques. The SVD reconstruction exhibits negative delay values, demonstrating the significance of the positivity constraint when addressing delay completion. As shown earlier, when the observation rate decreases, the reconstruction tends to be more challenging and often provides less accurate results. Nevertheless, the NMF reconstruction in Figure 4.5 still corresponds to the baseline of the original delays, even though it introduces more variance in one of the delay time series.

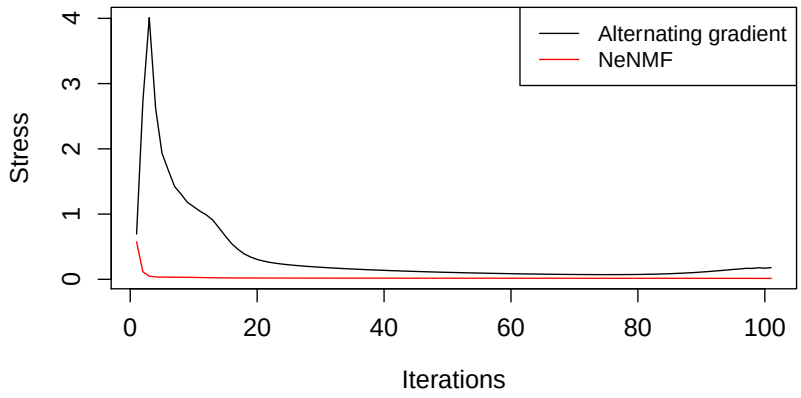


Figure 4.3 – Stress evolution over the first 100 iterations.

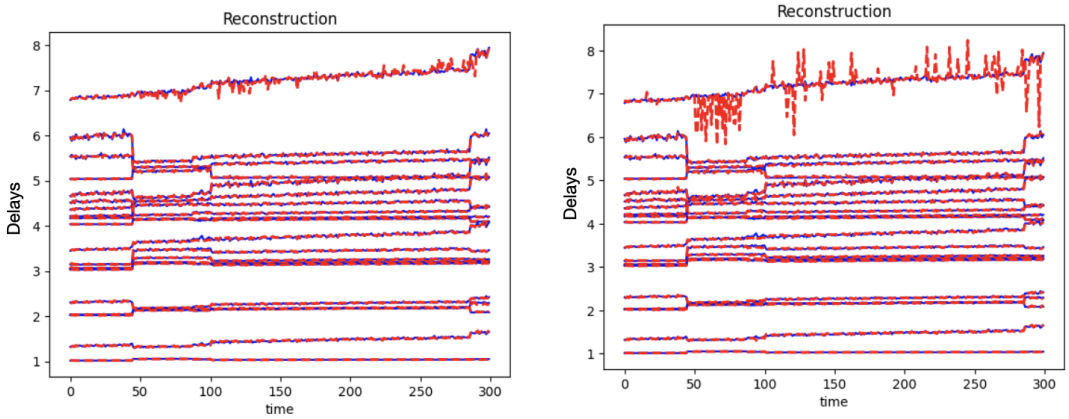


Figure 4.4 – Reconstruction of an 80% observed synthetic dataset using NMF(left) and SVD (right). The delays are in blue and the reconstruction is in red.

4.4.2 Matrix completion results on RIPE Atlas RTT dataset

The measurement campaign covers a period of one week. Delays were collected using three ICMP pings, and the minimum value was saved for each timestamp. This provides a dataset of 720 RTT series with 2520 time slots each.

In the dataset, missing delays account for 25% of measurements, and their distribution varies for each (Origin, Destination) pair. Only (Origin, Destination) pairs with missing data rates below 80% were retained for experiments. As mentioned in Chapter 3, outliers

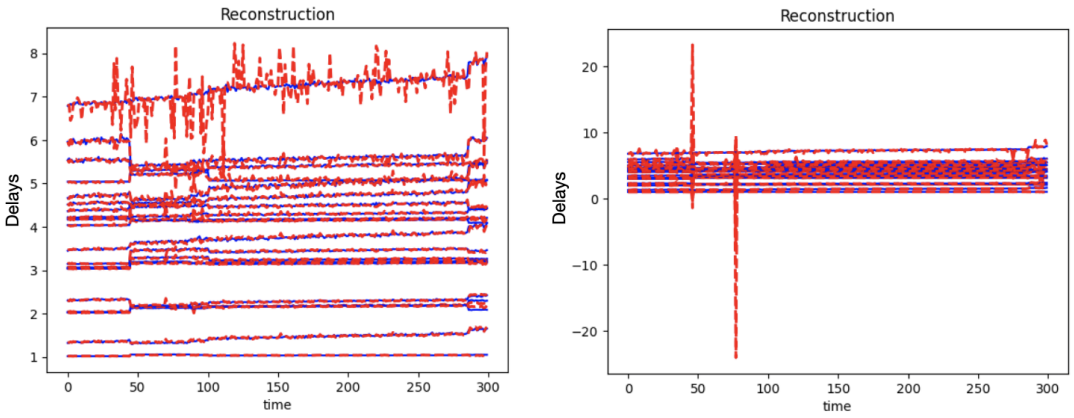


Figure 4.5 – Reconstruction of an 50% observed synthetic dataset using NMF(left) and SVD (right). The delays are in blue and the reconstruction in red.

have been removed to avoid bias during the reconstruction. This treatment reduced the size of the retained matrix D to 572×2520, with a proportion of missing values at 18%.

According to the previous results on the synthetic data, we restrict our interest to the NeNMF on the Ripe dataset as it appears to be faster than the alternating projection gradient. Since we lack ground truth for the non-measured values of this dataset, we use the following error criterion to evaluate the factorization quality:

$$P^k_{\text{error}} = \sqrt{\frac{\sum_{i,j} S_{ij}(D_{ij} - [W^k H^k]_{ij})^2}{\sum_{i,j} S_{ij}(D_{ij})^2}}, \tag{4.8}$$

where k is the iteration number. The algorithm converges within hundreds of iterations, and we observe that the error decreases with the rank. We fix the number of iterations to 1000 and the rank to 100. In this experiment, the reconstruction error on the observed values is 2%. This low error rate is highlighted by Figure 4.6 which displays 5 different RTT series completed using the NeNMF. We can notice that the completed segments capture the overall baseline of the original RTT series.

Relevance of the piecewise constant assumption

In Figure 4.7, we can observe the application of L1 trend filtering [58] to two delay time series derived from a real-world dataset. When the time series exhibits minimal variance, L1 trend filtering effectively functions as a segmentation tool, highlighting the significance of assuming piecewise behavior in delay time series. However, in the second

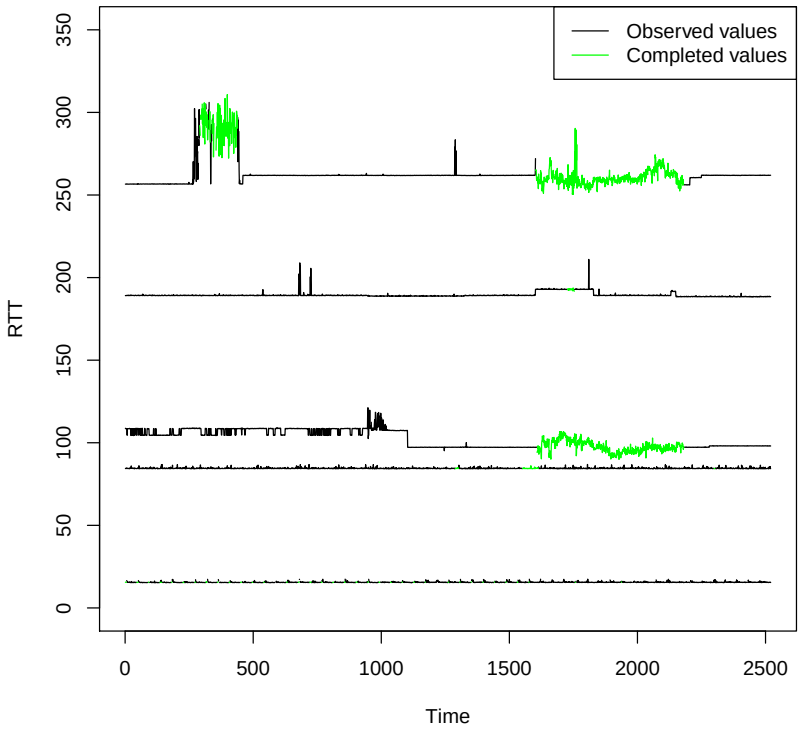


Figure 4.6 – RTT completion using NeNMF.

plot of Figure 4.7, the time series presents substantial variance in two specific sections. In these regions, L1 trend filtering struggles to identify a single continuous segment, which justifies our exploration of alternative methods, including clustering in the upcoming chapter, to address this segmentation challenge.

4.4.3 Conclusion and takeaways

We addressed the problem of inferring missing delays in the matrix of OD RTTs within a matrix completion approach. This was conceivable thanks to the stability of delays throughout time which is a contributing factor to the low-rank property of the delay matrix. We used two Non-negative matrix factorization algorithms: the alternating projected gradient and the NeNMF. We were able to test these methods in a controlled environment by using a synthetic delay generator and on real-world data with delays from Ripe Atlas. The two approaches are simple, easy to implement, and show great accuracy on the completion task when applied to synthetic data. The experiments, however,

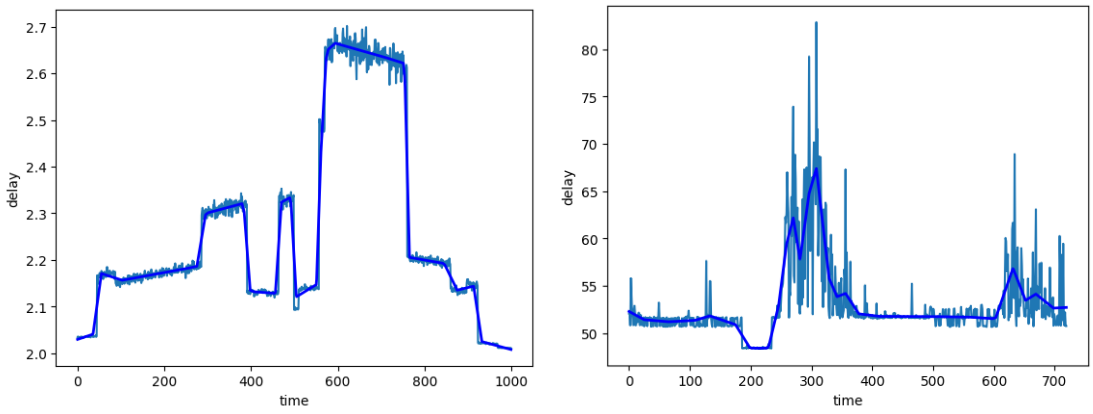


Figure 4.7 – Application of the L1 trend filtering to two real-world delay time series.

pointed out the speed difference between the two algorithms. The alternating projected gradient converges slower than the NeNMF. Hence, the scalability of the NeNMF was exploited by applying it to real-world datasets. The completion given by this algorithm has shown great accuracy within a small number of iterations.

4.5 Neural collaborative filtering (NCF)

This section addresses the delay matrix completion problem from a neural network perspective by using the neural collaborative filtering (NCF) approach [45]. Collaborative filtering [59] is a widely used technique in recommender systems. It gathers data from other users in order to identify similarities between them. This helps the recommender system to predict missing users' preferences. In this section, we would like to investigate whether NCF can be an interesting alternative to NeNMF in terms of speed and accuracy.

Neural collaborative filtering, in particular, makes use of the flexibility and complexity of neural networks to accomplish this recommendation task.

We compare the performance obtained with other techniques based on non-negative matrix factorization (NMF) [37]. We base our comparison on two datasets. The first is generated by our delay simulator of an autonomous system with variations linked to external routing changes. The second one consists of real RTT measurements on the Internet obtained from the RIPE Atlas platform [97].

As before, we assume that there are some *unobserved entries* in the matrix $\boldsymbol{D}$ that correspond to missing measurements. Let us introduce $\mathbb{S}$ the set of index pairs (i, j)

such that $\boldsymbol{D}_{ij}$ is *observed*, and $\overline{\mathbb{S}}$ the set of indices of *unobserved* delays. Each index (i, j) of an entry of the delay matrix can be represented by two one-hot encoded vectors. For example, we associate to the entry $\boldsymbol{D}_{ij}$ with $(i, j) \in \mathbb{S}$ the vectors $\boldsymbol{p}_i \in \{0, 1\}^n$ and $\boldsymbol{t}_j \in \{0, 1\}^m$ where:

$$\boldsymbol{p}_i(k) = 1_{\{k=i\}} \quad \text{and} \quad \boldsymbol{t}_j(l) = 1_{\{l=j\}}, \tag{4.9}$$

where 1 is the indicator function. An example of this encoding is given in Fig. 4.8. We

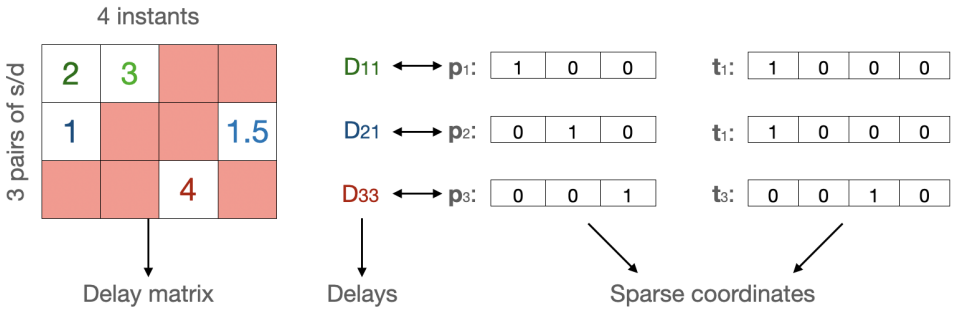


Figure 4.8 – Sparse representation of the delays coordinates, pink squares represent missing delays.

are now going to present two different Neural Collaborative Filtering architectures that can be used to solve this matrix completion problem. The Generalized Matrix Factorization (GMF) generalizes the standard matrix factorization approach and the multi-layer perceptron (MLP), a standard deep learning architecture [45].

Each architecture is trained on a given number of epochs over the set $\mathbb{S}$ of observed delays in order to minimize a loss, defined as a measure of discrepancy between the predicted and the observed delays. In order to do so, the neural network uses an optimizer that updates each parameter during the backpropagation with a given learning rate. The impact of all these parameters on the quality of the reconstruction will be studied in the evaluation section. On the other hand, the test phase is performed on $\overline{\mathbb{S}}$. To be more precise, we act as if the values were not known, and we check the quality of the prediction by evaluating the loss between the actual and the predicted values.

4.5.1 Generalized Matrix Factorization (GMF)

Following [45], the architecture of the GMF algorithm is composed of three main components.

1. For each $\boldsymbol{D}_{ij}$ to be predicted, the *input layer* consists of sparse entry vectors $\boldsymbol{p}_i$ and $\boldsymbol{t}_j$.

2. The *embedding layer* is a fully connected layer projecting a sparse vector to a dense one. Generally speaking, an embedded layer is designed to handle categorical input data and aims to learn a low-dimensional, dense representation (embedding) for each category or entity in the input space with a smaller dimension, say k. Its dimension k is considered as a parameter of the model. This layer is seen as a latent feature extractor.
3. The *neural collaborative filtering block* is a multi-layered neural architecture that connects the output of the embedding layer to the predicted delays. Its goal is to minimize the loss between the estimated delay and the target value.

In this model, each layer output serves as an input for the next one. This architecture is described in Figure 4.9, with the size indicating the dimension of the input, with k, r and 1 corresponding respectively to the dimension of the embedding, the FC_1 and the FC_{out} layers, with FC denoting a fully connected layer. The ReLu is the activation function used between the two FCs. We can formulate the GMF model as:

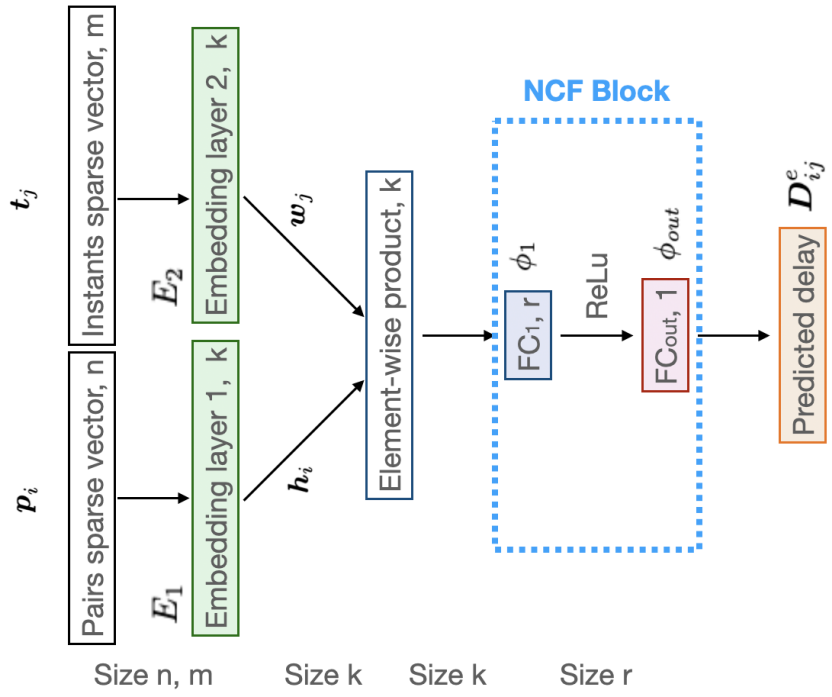


Figure 4.9 – Generalized Matrix Factorization (GMF) model.

$$\boldsymbol{D}^e_{ij}(\boldsymbol{\Xi_1}, \boldsymbol{\Xi_2}, \boldsymbol{\Theta}_1, \boldsymbol{\Theta}_{out}) = \phi_{out}(\phi_1(E_1(\boldsymbol{p}_i, \boldsymbol{\Xi_1}) \circ E_2(\boldsymbol{t}_j, \boldsymbol{\Xi_2}), \boldsymbol{\Theta}_1), \boldsymbol{\Theta}_{out}), \tag{4.10}$$

where $\boldsymbol{D}^e_{ij}$ is the estimated delay corresponding to the entry (i, j), ϕ_{out} and ϕ_1 are respectively the mapping functions of the output layer FC_{out} and the first neural collaborative filtering layer FC_1. $\circ$ denotes the element-wise Hadamard product of vectors and E_1 and E_2 refer to the functions of the embedding layers. Finally, $\boldsymbol{\Theta}_1$, $\boldsymbol{\Theta}_{out}$, $\boldsymbol{\Xi_1}$ and $\boldsymbol{\Xi_2}$ correspond to trainable parameters. For simplicity reasons, $\boldsymbol{D}^e_{ij}(\boldsymbol{\Xi_1}, \boldsymbol{\Xi_2}, \boldsymbol{\Theta}_1, \boldsymbol{\Theta}_{out})$ is denoted by $\boldsymbol{D}^e_{ij}(\boldsymbol{\Xi}, \boldsymbol{\Theta})$.

Such neural network is called the Generalized Matrix Factorization (GMF) since we can find the classical matrix factorization if ϕ_1 is a product by the all one vector and ϕ_{out} is the identity function. Indeed, let us denote $\boldsymbol{h}_i = E_1(\boldsymbol{p}_i, \boldsymbol{\Xi_1}) \in \mathbb{R}^k$ and $\boldsymbol{w}_j = E_2(\boldsymbol{t}_j, \boldsymbol{\Xi_2}) \in \mathbb{R}^k$ the dense vectors resulting from the embedding layers. The classical reduced rank matrix factorization model $\boldsymbol{D} \approx \boldsymbol{H}^\top \boldsymbol{W}$ with $\boldsymbol{H} = [\boldsymbol{h}_1 \cdots \boldsymbol{h}_n] \in \mathbb{R}^{n \times k}$ and $\boldsymbol{W} = [\boldsymbol{w}_1 \cdots \boldsymbol{w}_m] \in \mathbb{R}^{k \times m}$ estimates $\boldsymbol{D}^e_{ij}(\boldsymbol{\Xi}, \boldsymbol{\Theta})$ by: $\boldsymbol{D}^e_{ij}(\boldsymbol{\Xi}, \boldsymbol{\Theta}) = \boldsymbol{h}_i^\top \boldsymbol{w}_j$. When $\phi_{out} = I_d$ and $\phi_1 = \mathbf{1}$ with I_d the identity function and $\mathbf{1}$ a vector of ones of length k, and there is no intermediate ReLu:

$$\boldsymbol{D}^e_{ij}(\boldsymbol{\Xi}, \boldsymbol{\Theta}) = \phi_{out}(\phi_1(\boldsymbol{h}_i \circ \boldsymbol{w}_j, \boldsymbol{\Theta}_1), \boldsymbol{\Theta}_{out}) = I_d(\mathbf{1}^\top(\boldsymbol{h}_i \circ \boldsymbol{w}_j)) = \mathbf{1}^\top(\boldsymbol{h}_i \circ \boldsymbol{w}_j) = \boldsymbol{h}_i^\top \boldsymbol{w}_j. \quad (4.11)$$

4.5.2 Multi layer perceptron (MLP)

The MLP model also takes as an input the sparse vectors $\boldsymbol{p}_i$ and $\boldsymbol{t}_j$ followed by an embedding layer. The embedded vectors $\boldsymbol{h}_i$ and $\boldsymbol{w}_j$ are then concatenated and supplied to a towered multi-layered architecture as observed in Figure 4.10. This multi-layer architecture introduces more flexibility and non-linearity to the model. We denote $\phi_i(\boldsymbol{x}, \boldsymbol{\Theta}_i)$ the mapping function of each hidden layer i and $\boldsymbol{\Theta}_i$ its trainable parameter. For the layers of this model, we choose the ReLu as an activation function.

4.5.3 Loss and optimizers

For both architectures, we consider the mean squared error as the loss function:

$$L(\boldsymbol{\Xi}, \boldsymbol{\Theta}) = \frac{1}{|\mathbb{S}|} \sum_{(i,j) \in \mathbb{S}} (\boldsymbol{D}_{ij} - \boldsymbol{D}^e_{ij}(\boldsymbol{\Xi}, \boldsymbol{\Theta}))^2, \quad (4.12)$$

where $|\mathbb{S}|$ is the cardinal of $\mathbb{S}$ and $\boldsymbol{\Xi}, \boldsymbol{\Theta}$ denoting the trainable parameters of the network. In the evaluation section, we use a batch size of 1 and we test different optimizers. We opted for a batch of size 1, since with piecewise regularization, we were unsure how to

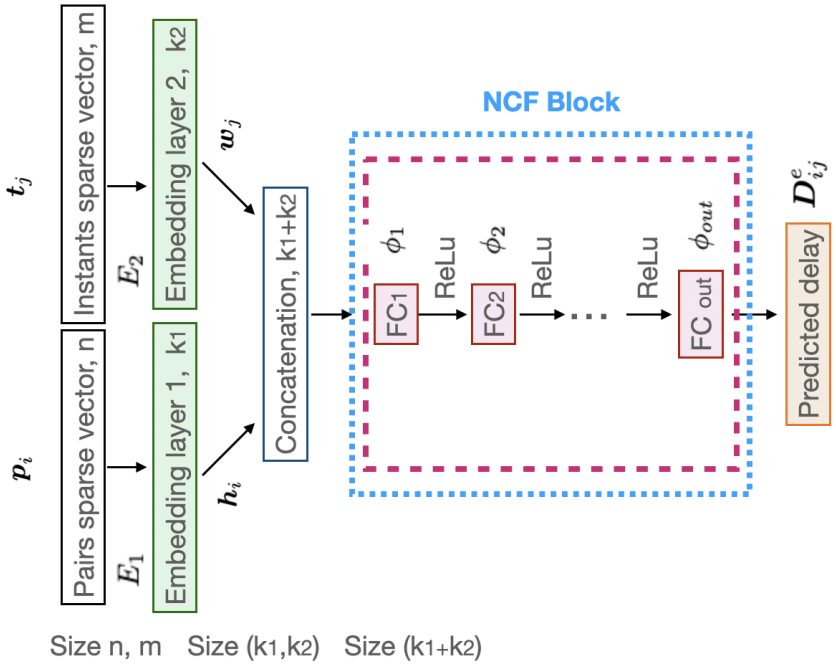


Figure 4.10 – Multi Layer Perceptron (MLP) model.

compute the gradient with batch processing. Moreover, this provides the possibility of fine-grained updates. With a batch size of 1, the model's parameters are updated after processing each individual data point. This can provide fine-grained control over the learning process and might be beneficial when the data has specific patterns that need to be captured.

4.6 Performance evaluation of NCF

We evaluate the NCF introduced in the previous section on two different datasets. The first one is a synthetic dataset generated by an AS simulator. The second dataset is a real-world dataset from a measurement campaign that we have conducted on the Ripe Atlas platform [97]. This campaign involves Ripe anchors located around the world and the measurements are conducted at a frequency of 4 minutes. More details about these datasets are provided in Chapter 3.

To evaluate the completion performance of the two NCF architectures, we consider the convergence stress [66] as a performance measure. It measures the quality of the reconstruction on the missing values $\boldsymbol{D}_{ij}$ with $(i,j) \in \overline{\mathbb{S}}$:

$$Stress = \sqrt{\frac{\sum_{(i,j)\in\overline{\mathbb{S}}}(\boldsymbol{D}_{ij} - \boldsymbol{D}^{e}_{ij}(\boldsymbol{\Xi}, \boldsymbol{\Theta}))^2}{\sum_{(i,j)\in\overline{\mathbb{S}}}(\boldsymbol{D}_{ij})^2}} \tag{4.13}$$

We use the stress to compare the performance of NCF approaches to the ones obtained with the Non-negative Matrix Factorization (NMF) ones. To this end, we consider two NMF algorithms: the alternated projected gradient (APG) [103] and the NeNMF [41] algorithm that uses a Nesterov gradient.

4.6.1 Matrix completion results on synthetic datasets

Our simulated dataset consists of a delay matrix of $n = 20$ *source-destination* pairs over $m = 400$ instants.

Impact of the embedding layer dimension We start our study by varying the embedding layer dimension k in the model architectures. As we can see in Table 4.6.1 the stress decreases with the embedding layer dimension. This is due to the fact that a larger embedding layer captures more latent features and is more adapted to complex high-dimensional data. We set the embedding dimension to 35 for the MLP approach since it reaches its minimum stress at this value, whereas for the GMF the embedding dimension will be fixed at 30.

Table 4.4 – Impact of the embedding dimension on the stress.

Emb dim k	5	10	15	20	25	30	35
Stress GMF	0.0183	0.0188	0.0134	0.0133	0.012	**0.0103**	0.0106
Stress MLP	0.0222	0.0135	0.0154	0.0113	0.0144	0.0108	**0.0085**

Impact of the number of epochs We have also assessed the influence of the number of epochs on the completion accuracy. We can notice in Figure 4.11 that the performance is enhanced when we increase the training phase for both GMF and MLP. We can see that stress starts to become stable between 1.5e5 and 2e5 epochs. We fix the number of epochs to 1.5e5 for the rest of the experiences for both architectures.

Impact of the optimizer and the learning rate We are also interested in the impact of the type of optimizer. We have tried three different optimizers: the stochastic gradient descent (SGD), the Adam optimizer, which is a method for stochastic optimization, widely used in training deep learning models, and the AdamW which corresponds to the Adam optimizer with weight decay. Table shows that the Adam optimizer is optimal for this study for both models. Moreover, we analyzed the influence of the learning rate by trying

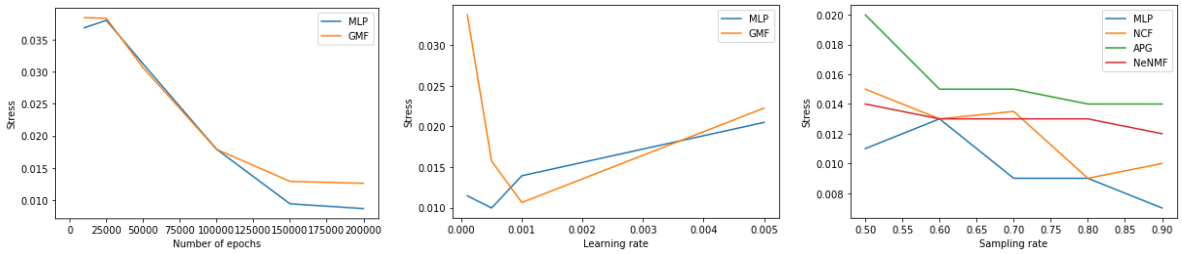


Figure 4.11 – The impact of the number of epochs, of the learning rate and of the sampling rate on the stress.

different values. Figure 4.11 indicates that the learning rates of 1e−3 and 5e−4 minimize the stress respectively for the GMF and MLP models. Hence, the learning rate will be fixed for each architecture accordingly.

Comparison of the execution times The execution time plays a crucial role in the real-world deployment of completion methods and should take part in the evaluation process. By using a 2,6 GHz Intel Core i7 processor and a 32Go 2667 MHz DDR4 memory the MLP and GMF were executed respectively within 335 and 222 seconds each. On the other hand, NeNMF takes 472s to converge whereas APG has a much longer execution time of 16 minutes.

Testing piecewise regularization term on NCF As previously highlighted in the NMF section, we can add a regularization term to the loss in order to incorporate more information about the temporal stability of the RTTs observed in the data. Let us denote by $L_\beta(\boldsymbol{\Xi}, \boldsymbol{\Theta})$ the regularized loss function, where β is the regularization parameter. Omitting the dependencies with respect to $(\boldsymbol{\Xi}, \boldsymbol{\Theta})$ for simplicity reasons, L_β writes:

$$\begin{aligned} L_\beta = \; & \tfrac{1}{|\mathbb{S}|} \textstyle\sum_{(i,j)\in\mathbb{S}} (\boldsymbol{D}_{ij} - \boldsymbol{D}^e_{ij})^2 + \beta \textstyle\sum_i \Big(|\boldsymbol{D}^e_{i1} - \boldsymbol{D}^e_{i2}| + |\boldsymbol{D}^e_{i(T-1)} - \boldsymbol{D}^e_{iT}| \\ & + \textstyle\sum_{t=2}^{T-1} |\boldsymbol{D}^e_{it} - \boldsymbol{D}^e_{i(t+1)}| + |\boldsymbol{D}^e_{it} - \boldsymbol{D}^e_{i(t-1)}| \Big), \end{aligned} \tag{4.14}$$

We observe in Table 4.6.1 that the regularization does not have a positive impact on stress evolution. Therefore, we will set $\beta = 0$.

Impact of the sampling rate and NMF/NCF comparison When a signal is partially observed, we denote the sampling rate, the proportion of the observed values with respect to the total number of entries of the matrix. In Figure 4.11, we investigate its impact on the stress for NCF and NMF methods. Despite the fact that all the methods display a stress less than 2%, we can notice that the MLP clearly outperforms the other methods with a stress smaller than 1% for sampling rates higher than 70%. On the other hand, the

performance of the GMF tends to approach the NeNMF results on lower sampling rates but surpasses it on higher ones. APG displays however worse results than other methods.

Table 4.5 – Impact of the optimizer on the stress.

Optimizer	SGD	Adam	AdamW
Stress GMF	0.0194	**0.0112**	0.0210
Stress MLP	0.0344	**0.0089**	0.0102

Table 4.6 – Impact of β on the stress.

β	0	0.01	0.1	0.3	0.5
Stress GMF	**0.0104**	0.0130	0.0201	0.0230	0.0260

Figures 4.12 and 4.13 show the reconstruction of two delay matrices sampled respectively at 70% and 50% using GMF and MLP. We can notice that the reconstruction captures the baseline of each time series. Besides, the completion corresponding to a higher sampling rate is less noisy and shows more stability.

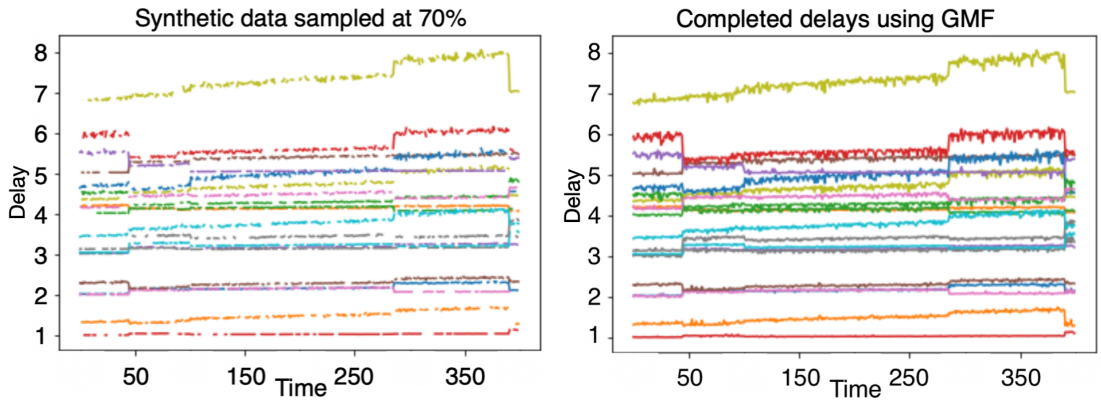


Figure 4.12 – Reconstruction for a simulated dataset using GMF with 70% observed data.

4.6.2 Matrix completion results on RIPE Atlas RTT dataset

This dataset comes from a measurement campaign on Ripe Atlas as mentioned in Chapter 3. We wanted to have a ground truth for all the entries of our delay matrix and this was actually the case since we did not have missing delays on this campaign.

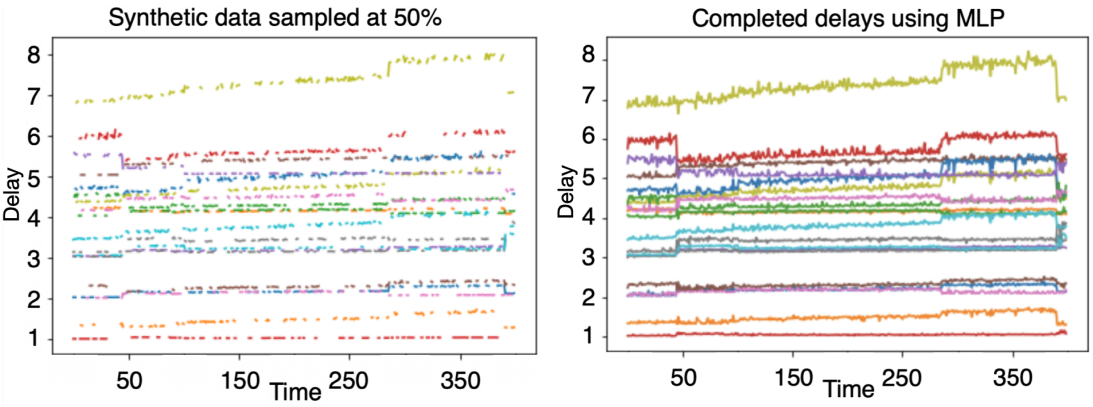


Figure 4.13 – Reconstruction for a simulated dataset using MLP with 50% observed data.

The dataset contains 50 RTT time series of length 800 each corresponding to 22 hours of measurements.

Impact of the embedding layer dimension First, we explore the impact of the embedding layer on the stress in Table 4.7. Results were obtained by running 1.5e5 epochs. The stress remains overall constant for both architectures, but we can see that GMF and MLP achieve their minimum stress at 20. For this reason, we fixed this dimension to 20.

Table 4.7 – Impact of the embedding dimension on the stress for 1.5e5 epochs.

Emb dim	5	10	15	20	25	30
Stress GMF	0.0118	0.0115	0.0115	**0.0113**	0.0115	0.0118
Stress MLP	0.0121	0.0134	0.0132	**0.0119**	0.0142	0.0131

Impact of the number of epochs Table 4.11 shows that MLP reaches its minimum stress at 1.5e5 epochs, whereas GMF stress continues to decrease if more epochs are considered. Due to time execution considerations, we set the number of epochs to 1.5e5 for both architectures.

Table 4.8 – Impact of the number of epochs on the stress.

Number of epochs	5e4	1e5	1.5e5	2e5	2.5e5
Stress GMF	0.0182	0.0117	0.0119	0.0116	**0.0115**
Stress MLP	0.0123	0.0120	**0.0118**	0.0119	0.0142

Impact of the optimizer and the learning rate We investigate the impact of the learning rate in Table 4.10 . One can see that 1e−4 is the best learning rate for both architectures. We fix the learning rate at this value. Table 4.9 shows the stress after 1.5e5 epochs with

a learning rate of 1e−4 for different optimizers. The Adam optimizer is the most suitable for both the GMF and the MLP architectures.

Table 4.9 – Impact of the optimizer on the stress.

Optimizer	SGD	Adam	AdamW
Stress GMF	0.4433	**0.0113**	0.0116
Stress MLP	0.4408	**0.0117**	0.0118

Table 4.10 – Impact of the learning rate on the stress.

Learning rate	0.005	0.001	0.0005	0.0001
Stress GMF	0.0192	0.0173	0.0144	**0.0118**
Stress MLP	0.0168	0.0151	0.0144	**0.0117**

Impact of the sampling rate and NMF/NCF comparison We can see in Table 4.11 that the stress decreases with the sampling rate for all the architectures. Moreover, we can observe that the MLP is better than the GMF and that the overall stress remains smaller than 2%. However, we see that the NeNMF outperforms the NCF approaches when applied to the real-world dataset, which is in line with the findings of the comparative studies [2, 95]. Such a result can be explained by the time stability of the real-data matrix, and its sparsity considering the low rank, combined with its high dimensions. These conditions can be in favor of a simple matrix product rather than a more complex model that needs to learn many additional parameters.

Table 4.11 – Impact of the sampling rate on the stress.

Sampling rate	0.50	0.6	0.7	0.8	0.9
Stress GMF	0.0198	0.0186	0.0166	0.0166	0.0158
Stress MLP	0.0126	0.0170	0.0125	0.0121	0.0119
Stress NeNMF	0.0182	0.0117	0.0119	0.0116	**0.0115**

By comparing Figure 4.14 and Figure 4.15, one can notice that both reconstructions correspond to the original time series, but we can clearly see that at equal sampling rate, the GMF reconstruction is noisier than the MLP one.

4.6.3 Conclusion and takeaways

NCF approaches enable us to achieve a very low stress rate for matrix completion task, both on the simulated and real-world datasets. In this chapter, we have studied

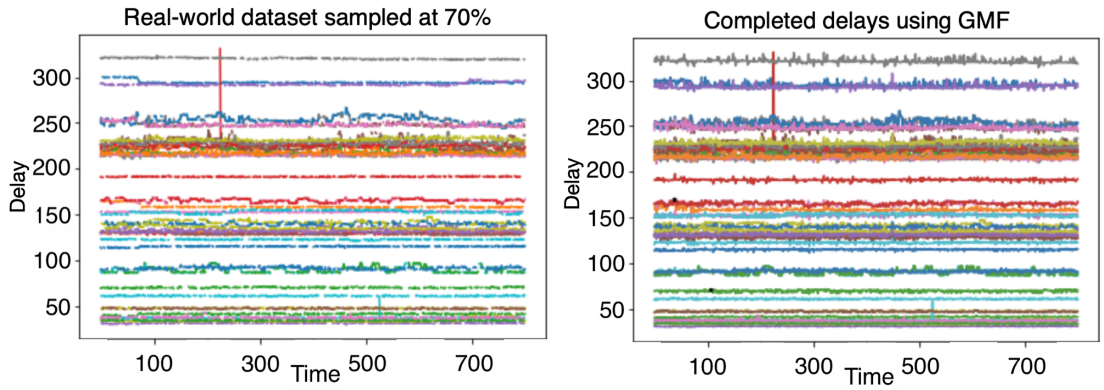


Figure 4.14 – Reconstruction of an 70% observed real-world dataset using GMF.

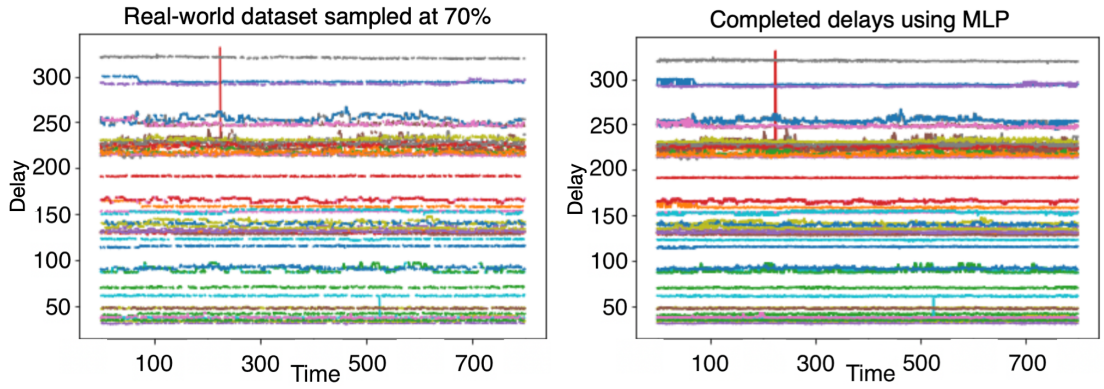


Figure 4.15 – Reconstruction of an 70% observed real-world dataset using MLP.

extensively the impact of multiple parameters such as the optimizer, the learning rate, or the number of epochs on the reconstruction quality. This allowed us to set an optimal training environment for the NCF models. The comparison with the NMF algorithms showed that NCF outperforms NeNMF on synthetic data, whereas this tendency is reversed when applied to Ripe Atlas data. One of the possible reasons is that we didn't use enough iterations, and we did not use piecewise regularization. Moreover, we have shown that NCF is more computationally efficient than the NeNMF. Finally, the addition of a regularization term didn't improve the completion quality of the synthetic dataset.

Chapter 5

Network delay segmentation

Contributions:

— Time series clustering based on hierarchical clustering and Pearson correlation matrix.

— Multivariate segmentation of time series within the same cluster based on the hierarchical clustering and Viterbi algorithm.

— Demonstration of the efficiency of the method for delays with various patterns.

— Computational efficiency regarding an existing state-of-the-art technique.

Delay measurements are important for network monitoring. With Internet monitoring platforms providing an unparalleled amount of data, it becomes necessary to automate their treatment. In particular, segmenting these delays permits supervising infrastructures and analyzing possible incidents. This chapter explores the use of hierarchical clustering for the segmentation of multivariate network delays. The proposed method offers a computationally efficient way to identify the spatial correlations among network delays and to jointly segment time series within the same cluster. A post-treatment step is introduced that involves the Viterbi algorithm to smooth segmentation and handle the temporal dependency more effectively. This global method is evaluated on the real-world datasets introduced earlier, demonstrating its suitability for managing delays with varying variance and changing patterns. The proposed approach provides an efficient and cost-effective method for automated delay characterization.

5.1 Problem: introduction and motivation

As discussed in Chapter 2, the importance of Internet end-to-end delay metrics, such as the round trip time (RTT), is well established, as it can be used to assess performance,

the status of a specific path, the quality of user experience, and real-time applications. Typically, delay measurements exhibit similar patterns when the routes followed contain shared segments. In [79], it was observed that delays between pairs passing through the same damaged Internet Exchange Point (IXP), presented synchronized pattern changes. An example of spatially correlated paths can be seen in Figure 5.1, which displays a group of Internet delay time series that exhibit similar behaviors. This similarity can be attributed to the underlying network routing.

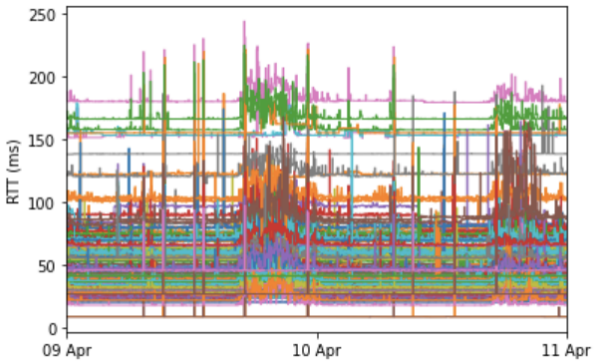


Figure 5.1 – Time series with correlated change patterns.

In the networking community, Internet delay segmentation and characterization have been widely studied as an intriguing problem. Traditionally, the segmentation of delays has been left to human experts. Nonetheless, with the abundance of data available on platforms such as Ripe Atlas and Caida, as well as on private ISP servers, relying on human analysts to segment delays is both expensive and challenging. Therefore, there is a need for an automated layer and the creation of appropriate tools and algorithms to meet this increasing demand.

Many works have aimed to model Internet delays in time [131, 78] in order to gain a better understanding of this metric and more insight into network performance and functioning [23]. Furthermore, Shao et al. [108] studied RTT segmentation in order to evaluate the impact of the routing changes at different levels (AS, IXP) on RTT changes, and Mouchet et al. [79] used HDP-HMM to detect IXP outages based on RTT changes frequency. Other characterization studies of the delays using different distributions have been conducted [131], [102], [47], as well as approaches based on deep learning [91].

Despite the high accuracy of RTT series segmentation achieved in several of the cited works, the state of the art presents two primary issues. Firstly, the suggested methods are computationally intensive. Secondly, many studies only take into account temporal

stability, without taking full advantage of spatial dependencies among RTT series. Many works got interested in the problem of the segmentation of multivariate time series [17], [24], [48], however, to the best of our knowledge, this paper proposes the first multivariate segmentation of delay networks based on joint segmentation of multiple delay time series.

In this chapter, we explore the use of hierarchical clustering for the segmentation of multivariate network delays. To capture the spatial correlation in the dataset, we first apply hierarchical clustering on the Pearson correlation matrix of the RTT time series [7]. Subsequently, when the clusters are identified, we jointly segment the time series within each cluster using hierarchical clustering [11]. This clustering alone, however, does not incorporate the temporal stability property of these delays. Therefore, we propose a post-treatment step by exploiting the Viterbi algorithm [31], which is able to smooth the resulting clustering and handle the temporal dependency more effectively. Our global method is then evaluated on the two real-world datasets considered in this thesis, demonstrating its suitability for managing delays with varying variance and changing patterns.

In the following, we first describe the methodology and an illustrative example, starting with the description of the hierarchical clustering, and then explaining the clustering of time series. Later we present the two-step multivariate segmentation using the Viterbi smoothed hierarchical clustering. Finally, we evaluate the methodology, on the clustering of time series and then on the segmentation by comparing it to the most efficient state-of-the-art method both in terms of accuracy and execution time.

5.1.1 Principles of hierarchical clustering

Hierarchical clustering is a popular agglomerative clustering technique used to group variables based on similarity. This approach considers that each element is its own cluster initially. Then, clusters are iteratively merged into larger clusters. At each step, the two closest clusters are merged into one new parent cluster. This process is repeated until a single global cluster remains after N-1 iterations, with N the number of variables [67]. This iterative clustering process can be represented in the form of a dendrogram, that is a tree structure plot.

The similarity between clusters is measured using a linkage method, such as single (sgl), complete (cpl), average (avg), centroid (ctr), or Ward (wrd) linkage [81]. Lets consider two clusters: C_1 and C_2. Single linkage considers the closest elements of the clusters: $d_{\text{single}}(C_1, C_2) = \min_{\mathbf{x}_1 \in C_1, \mathbf{x}_2 \in C_2} \|\mathbf{x}_1 - \mathbf{x}_2\|$, complete and average linkage considers the furthest elements and the average distance between all elements, respectively. Their formu-

las are the following: $d_{\text{complete}}(C_1, C_2) = \max_{\mathbf{x}_1 \in C_1, \mathbf{x}_2 \in C_2} \|\mathbf{x}_1 - \mathbf{x}_2\|$ and $d_{\text{average}}(C_1, C_2) = \frac{1}{|C_1| \cdot |C_2|} \sum_{\mathbf{x}_1 \in C_1} \sum_{\mathbf{x}_2 \in C_2} \|\mathbf{x}_1 - \mathbf{x}_2\|$. Centroid linkage is based on the distance between the respective cluster centroids and is calculated by $d_{\text{centroid}}(C_1, C_2) = \|\mathbf{m}_1 - \mathbf{m}_2\|$, where m_1 and m_2 are the centroids of clusters C_1 and C_2, respectively. Ward linkage minimizes the variance of the merged clusters, which is calculated by the error sum of squares defined for a given cluster C as $ess(C) = \sum_{x \in C}(x - \frac{1}{|C|} \sum_{y \in C} y)^2$, with $|C|$ being the cardinal of the cluster C. The Ward linkage between two clusters C_1 and C_2 is then given by $d_{\text{Ward}}(C_1, C_2) = ess(C_1 \cup C_2) - (ess(C_1) + ess(C_2)$.

Table 5.1 – Pros and cons of each linkage method.

Pros and *Cons*	Sgl	Cpl	Avg	Ctr	Wrd
Handle noise between the clusters		x	x	x	
Less susceptible to noise and outliers		x			x
Handle non-elliptical shapes	x				
Capture clusters of different sizes	x				
Biased towards globular clusters		x	x	x	x
Sensitive to noise and outliers	x				
Sensitive to noise between the clusters	x				
Tend to break large clusters		x			

We can see in Table 5.1 that highlights the pros and cons (in italic) of different linkage strategies [110] that ward and complete linkages are the only ones that can handle noise and outliers. The complete however is not good for handling stationarity since it tends to break large clusters. For these reasons, we consider using the complete linkage for time series clustering and the Ward linkage for the segmentation.

5.1.2 Time series clustering

In this research, the hierarchical model is employed to efficiently cluster multivariate features. It is first used to make a profit from the spatial correlation of Internet delays by detecting groups of time series that display similar patterns. For this purpose, we introduce the following notation. We consider slotted time and suppose that a network with n available pairs of source/destination is observed during T timeslots. We observe the delay measurements between each source and destination of these pairs at different timeslots. D denotes the n-variate time series of length T, $D_{i:}$ the row i of the matrix D which represent the $i-th$ observed time serie of delays and $D_{:j}$ the column j of the matrix D which represent the vector of measured delays at instant j.

To shed light on the spatial correlation among time series, a Pearson correlation matrix R is computed between time series of the matrix D. In fact, we define R as follows: $R = \frac{1}{N-1}\left(\frac{D-\bar{D}^T}{\sigma_D}\right)\left(\frac{D-\bar{D}^T}{\sigma_D}\right)^T$.

Where :

— R is the matrix of Pearson correlation coefficients between rows in matrix D.

— N is the number of rows (time series) in D.

— $\bar{D}$ is a column vector of means for each row in D.

— σ_D is a column vector of standard deviations for each row in D.

— D is the matrix of time series, where each row represents a time series.

A hierarchical clustering is then performed on the values of the correlation matrix. Finally, the rows and columns of the Pearson correlation matrix are reordered according to the resulting cluster labels. In this new matrix, diagonal blocks are formed and each block represents groups of time series with higher correlation.

In order to estimate the number of clusters within a given dataset, we must determine the dendrogram cut-off. As seen in Figure 5.2, the cut-off threshold set at 0.7 results in three clusters, and the one at 1 yields two clusters. A lower threshold leads to a higher number of clusters, while a higher threshold corresponds to fewer clusters.

To set this threshold we would like to minimize the number of clusters while maximizing the correlation of time series within each cluster. For this purpose, we define the following criterion: For a partition P of the dataset, we define the metric: $h(P) = \sum_{c \in P} \frac{1}{|c|} \sum_{s_k, s_l \in c} |corr(s_k, s_l)|$, with $corr(x, y)$ being the Pearson correlation between x and y.

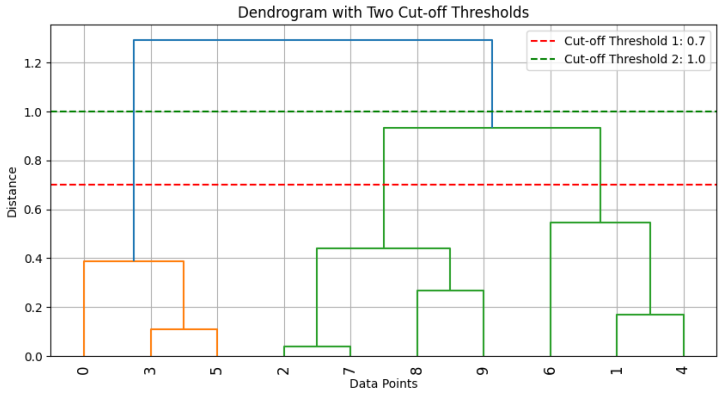


Figure 5.2 – Example of a dendrogram with two cut-off thresholds generating respectively 2 and 3 clusters.

An example of this process is given in Figure 5.3 that shows the correlation matrix of 200 time series before and after the hierarchical clustering. We can notice that in the second figure, blocks of highly correlated pairs are formed and are easier to identify.

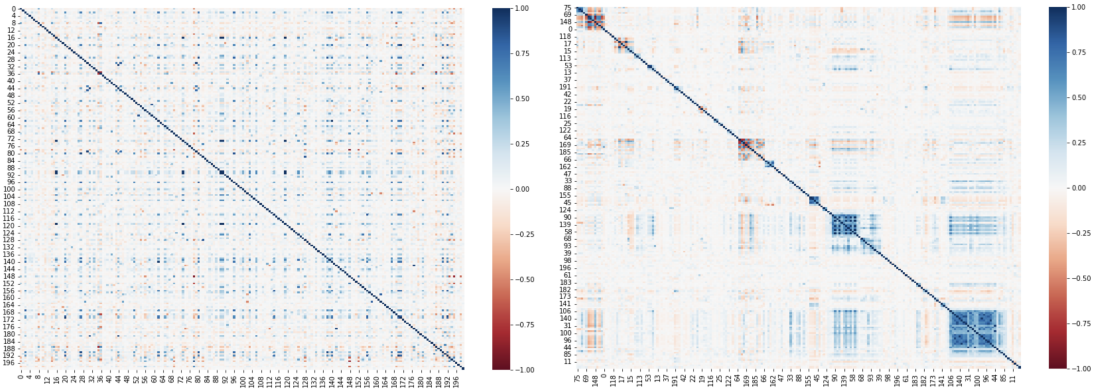


Figure 5.3 – Heatmap of the Pearson correlation matrix before (left) and after (right) hierarchical clustering. The x and y axis correspond to the delay time series of the dataset.

In order to have a closer look at the formed clusters, consider for instance Figures 5.1 and 5.4 that show the time series within the last two formed clusters at the bottom right of the Pearson matrix after clustering. It is easy to notice that within each cluster, the time series show synchronous and similar patterns, whereas the pattern nature and temporal distribution change from one cluster to another. This property demonstrates that the change points are highly synchronized between correlated pairs. This motivates the intracluster segmentation that we introduce the following.

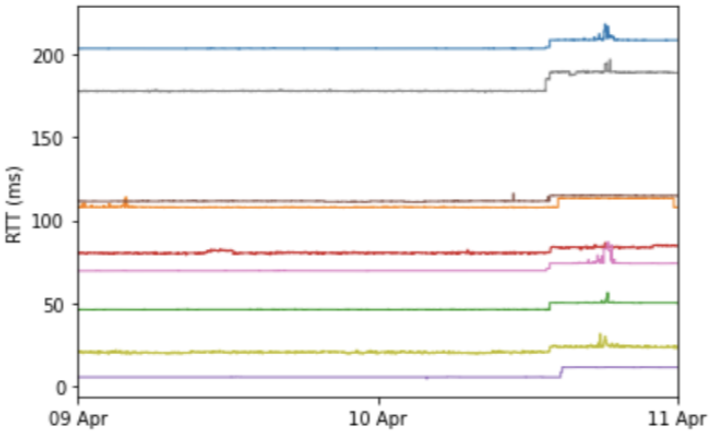


Figure 5.4 – Example of ten time series found in the same cluster.

5.1.3 Viterbi smoothing post-treatment

Let us denote by D^c the submatrix of D that contains the observed time series within the identified cluster c. Once the clusters of similar time series are identified, we are interested in segmenting jointly the delay time series within each cluster. In this second step, we first apply the hierarchical clustering on the columns of the matrix D^c. Figure 5.5 shows the result of this clustering. Despite the effectiveness of capturing the spatial correlation, hierarchical clustering imperfectly incorporates the temporal stability inside times series. To address this problem, we propose a Viterbi algorithm as a post-treatment. To this end, we suggest capturing the temporal correlation by modeling the multivariate delays present in D^c for each identified cluster c using a multivariate hidden Markov model (HMM) where the column of (D^c) is the vector observation at time t. This model considers that the observed delay vectors $(D_i^c)_{i\leq T}$ are generated by some hidden states $(s_k)_{k\leq m}$. In fact, the Markovian assumption regarding Internet delay changes is commonly used in the literature [79], [131]. The segmentation problem can hence be reformulated as follows: the segment labels are defined by the hidden states and the goal is to infer them based on the observed delay vectors. To this end, we use a Viterbi algorithm which is a dynamic programming algorithm for finding the most likely sequence of hidden states, called the Viterbi path [68], that results in the sequence of observed events. It is commonly used in the context of Hidden Markov Models (HMMs). In simple terms, it calculates the most likely sequence of events that have led to the observed sequence.

We use the output of the hierarchical clustering segmentation on each matrix D^c to roughly identify the number of hidden states and estimate the emission probability of

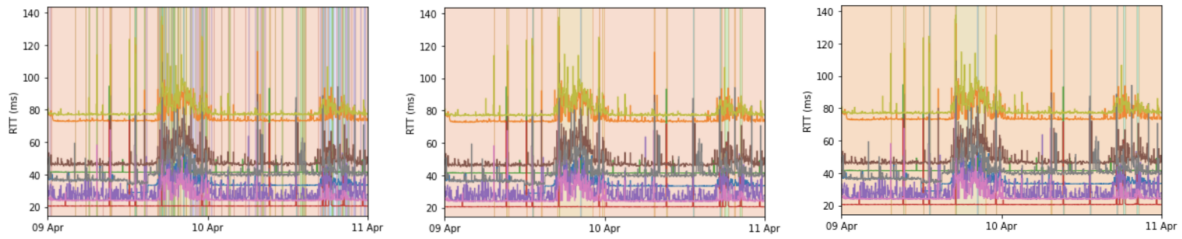


Figure 5.5 – Segmentation of the series in a cluster using hierarchical clustering, after one Viterbi smoothing and after two Viterbi smoothings.

each state. First, we separate the data points into different clusters based on the labels provided by the hierarchical clustering. Then we use observations identified within a cluster to initialize the distribution of emissions using either a density kernel method or a histogram method. Moreover, for the sake of simplicity, we assume prior knowledge of the transition probability matrix which stresses the fact that two successive delay vectors have more chance to be generated by the same hidden state. For this reason, we put ourselves in the context of transition probabilities that are small compared to the probability of remaining in the same state. For the sake of simplicity, we define the transition probability $m \times m$ matrix P_t as follows: $P_t(i,j) = \epsilon/(m-1)$ if $i \neq j$ and $P_t(i,i) = 1-\epsilon$, with m the number of hidden states. Choosing a small value of ϵ enforces state persistence. Finally, we assume that the network states are initially equally distributed.

However, this joint treatment of several time series is sensitive to the size of the cluster. Indeed, when the cluster contains many time series, the influence of the transition probability term tends to become negligible compared to the emission probability term resulting in weak smoothing. In order to overcome this imbalance, we introduce the following alternative regularization that depends on the cluster size, say N: the transmission probability is set to $P_t(i,j) = \epsilon^N/(m-1)$ if $i \neq j$ and $P_t(i,i) = 1-\epsilon^N$. This version of smoothing will be called regularized Viterbi smoothing.

If we consider the result of the hierarchical clustering in Figure 5.5 and apply the Viterbi algorithm, we get the segmentation result in Figure 5.5. We can see that the segments are smoothed and more stability is incorporated. To further improve these results we update the estimates of emission probabilities from this new segmentation and apply again the Viterbi algorithm. The final result is provided in Figure 5.5 which displays a slightly more smoothed segmentation than Figure 5.2.4.

In order to quantify the smoothing effect and measure the quality of the segmentation, we have chosen to use the adjusted Rand index. Indeed, the Rand index is a measure of

similarity between two partitions of the same set and it represents the proportion of pairs of points that are grouped in the same way in the two partitions [94]. The adjusted Rand index is an enhancement of the Rand index as it adjusts it to account for agreement that might occur by random chance. It is mainly used to assess the quality of a clustering when the ground truth is provided. And it ranges from -1 to 1, where 1 indicates perfect agreement, 0 indicates random agreement and negative values suggest less agreement than random. This index will allow us to decide the best variant of the method to choose in terms of the Viterbi version and the number of smoothings to use. It detailed formula is as follows: $\text{ARI} = \frac{\text{NAP}-\text{ENAP}}{\frac{1}{2}(\text{TNP}-\text{ENAP})+\frac{1}{2}(\text{TNP}-\text{ENDP})}$.

Where:

— NAP is the number of agreeing pairs, which is the number of pairs with the same cluster in both predicted and ground truth clustering. It represents how many data point pairs are correctly placed in the same cluster by both the predicted and ground truth clusterings.

— ENAP is the expected number of agreeing pairs, which is what you would expect to happen if the data were randomly assigned to clusters. It's essentially the level of agreement that could occur purely due to random chance.

— ENDP is the expected number of disagreeing pairs, which is the counterpart to the expected agreement. It's the number of data point pairs that would be placed in different clusters by random chance.

— TNP is the total number of pairs.

5.2 Performance evaluation

5.2.1 Datasets

We will test our methods on different real-world datasets. The datasets are collected from Ripe Atlas. They represent delays between pairs passing through specific IXPs, DE-CIX, and AMS-IX respectively. For these datasets, we use as ground truth approximation for the segmentation an HDP-HMM method [80] that was validated and adapted by Ripe Atlas in an API [3]. The different datasets are described more in detail in Chapter 3. We denote them as follows: Real dataset 1 (RD1), RTT traces passing through DE-CIX, this dataset contains 38k pairs passing through DE-CIX. Each of these time series has 720 timeslots which correspond to two days of measurements, the 9th and 10th of April 2018.

And Real dataset 2 (RD2), RTT traces passing through AMS-IX, this dataset contains 26k pairs passing through AMS-IX. Each has 360 timeslots corresponding to the 13th of May 2015. This dataset will be denoted RD2.

5.2.2 Baseline method: HDP-HMM

To our knowledge, HDP-HMM (Hierarchical Dirichlet Process Hidden Markov Models) is state-of-the-art for efficient segmentation of HMMs with an unknown number of states and unknown emission distributions that are modeled as Gaussian mixtures (with an unknown number of components).

We compare the successive treatments proposed with HDP-HMM segmentation: the hierarchical clustering alone (HC), HC with one Viterbi smoothing (V1), HC with two Viterbi smoothings (V2) and finally HC with one regularized Viterbi smoothing (V1 REG) and HC with two regularized Viterbi smoothings (V2 REG). V2 version of Viterbi smoothing introduces a re-estimation of the emission densities after the first smoothing.

5.2.3 Delay time series clustering

Hierarchical clustering is employed to group time series with similar and homogeneous patterns. The optimal clustering is obtained by determining a threshold for cutting off the dendrogram. A low threshold leads to simple clusters with highly similar patterns, yet a large number of clusters; thus, failing to fully capitalize on the similarity and dimensionality reduction. A high threshold, on the other hand, lightly compresses data, yielding a low number of clusters, but also presenting the risk of combining time series with distinct patterns within the same group, making it difficult to achieve an accurate multivariate segmentation. To maximize the correlation within each cluster, a threshold is chosen that lies between these two extremes. For the following, compression refers to the reduction in the number of clusters. It comes into play to simplify the clustering structure and it involves merging or combining some of the initially identified clusters to reduce their number. This can make the interpretation and analysis of the data more manageable and can help in summarizing the key patterns or groups in a more concise way.

In this section, we are going to explore the impact of the different linkage methods on the clustering quality. We focus in particular on the comparison between Ward and complete linkage along with defining the optimal dendrogram cut-off threshold.

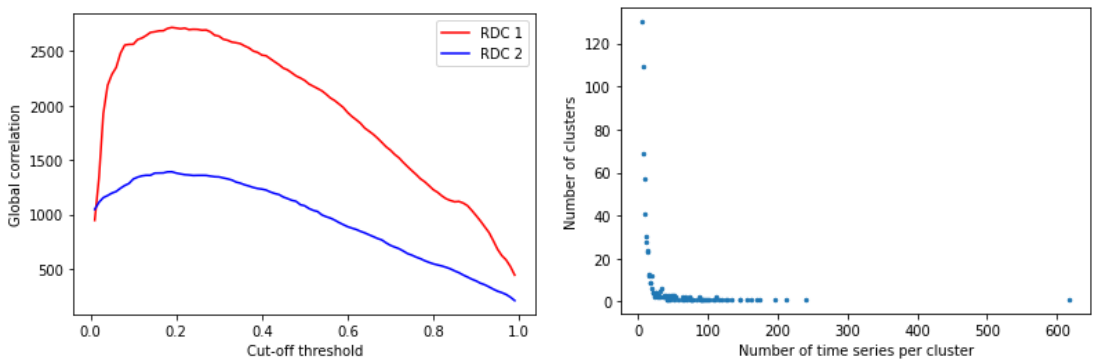


Figure 5.6 – Evolution of the global correlation with the dendrogram cut-off threshold and the number of clusters having more than 5 time series for RD1.

We can see in Figure 5.6 that for RD1 the optimal cut-off threshold is 0.2. This clustering used the complete linkage method. When we cut the dendrogram at this value, we obtain the distribution given in Table 5.2 and Figure 5.6. We can clearly notice that a large part of the data remained unclustered. For a dataset of nearly 38k time series, 16200 time series remained ungrouped. One way to deal with this is to raise the cut-off threshold. Nonetheless, the unclustered time series frequently consists of a considerable amount of noise that doesn't correspond to any existing pattern within the clustered time series. Given that the primary objective of this clustering step is to create clusters of time series sharing similar patterns of change, which can subsequently be segmented in a multivariate manner, we have chosen to maintain the threshold at its optimal value. This being said the rest of the data were clustered in a homogeneous way, with the majority of clusters containing between 2 and 50 time series.

Table 5.2 – Number of clusters having 5 time series or less.

Nb of time series	1	2	3	4	5
Nb of clusters for RD1	16200	1393	544	316	180
Nb of clusters for RD2	5006	699	321	230	119

We should note that a maximum number of time series per cluster is 617 as can be seen in Figure 5.7. We can see that the clustering detects the pertinent common patterns despite the presence of noise and outliers. Figure 5.7 shows on the other hand a cluster of time series with high stability. 5.7

Figures 5.8 and 5.8 present two clusters with very similar patterns. We should notice that despite the similarity displayed between these clusters, the complete linkage sepa-

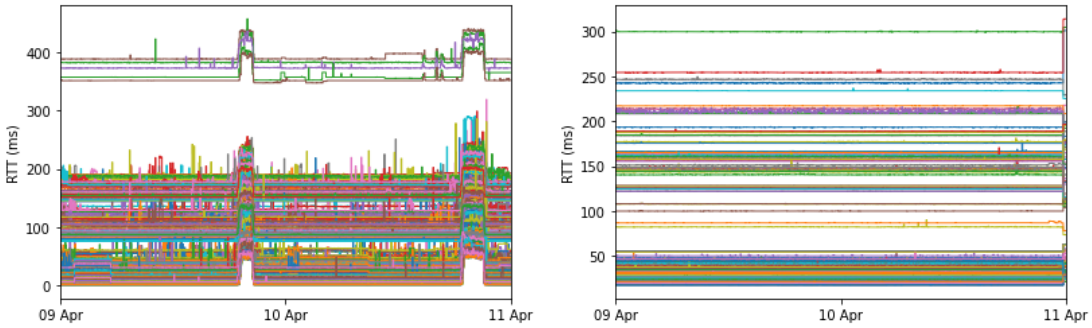


Figure 5.7 – Two clusters with respectively 617 and 162 time series from RD1 using complete linkage.

rated them while they could have been grouped into one larger cluster. This result goes along with one of the cons of complete linkage described in Table 5.1 which is its tendency to break large clusters.

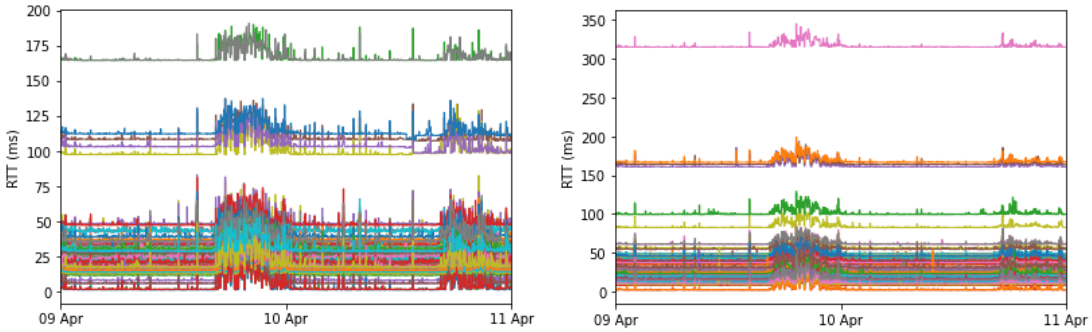


Figure 5.8 – Two clusters with respectively 115 and 108 time series from RD1 using complete linkage.

In Figure 5.9 we can see the dendrogram of the Ward linkage. On the x-axis we see the number of time series per cluster, knowing that each vertical line refers to an independent cluster. On the y-axis, we can see the distance between these clusters using the Ward linkage. The clustering remains the same for cut-off thresholds ranging from 0 to 8. For thresholds starting from 8, the process of agglomeration starts again. In order to have a clustering from this stable zone, the cut-off value was decided to be 8, leading to 30 clusters. This choice achieves a good trade-off between the number of clusters and the degree of their compression.

One difference that we can outline compared to the complete linkage, is the fact that the number of time series per cluster is higher on average. Moreover, in Figures 5.10, we

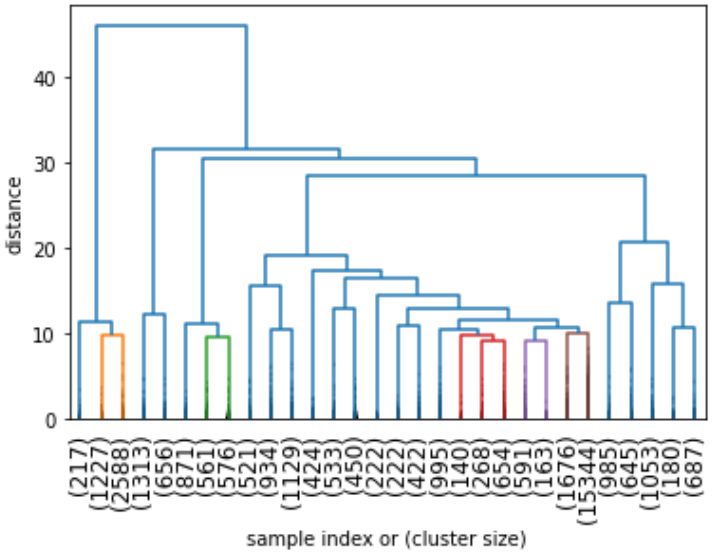


Figure 5.9 – The dendrogram of the hierarchical clustering of the delay time series using Ward linkage.

can see that the pattern that was split using the complete linkage in Figures 5.8 is totally grouped into one cluster of 1227 time series using the Ward method.

In Figures 5.12 and 5.11, we can notice that one drawback of the compression power of Ward is the fact that it will group together time series having a similar baseline, but different patterns. This is especially noticeable when the time series exhibit repeated peaks. As depicted in Figure 5.11, Ward clustering might combine time series with similar baseline trends, even if their peak patterns differ. This behavior is referred to as a globular bias, as mentioned in Table 5.1, where Ward tends to create clusters with a common baseline but potentially varying patterns. The complete linkage, on the contrary, didn't group these time series even though they have a common baseline behavior since they don't display a similar pattern change.

A possible approach to take advantage of the pros of both methods is to first cluster the data using Ward linkage. First, apply Ward linkage to the data. Once the homogeneous clusters are identified, these clusters can be further processed. The remaining data, which was initially grouped by Ward, and consists of heterogeneous clusters, can be considered as a new dataset. This new dataset can then be subjected to complete linkage clustering for a finer grouping, aiming to create more homogeneous clusters that consider both baseline and pattern changes.

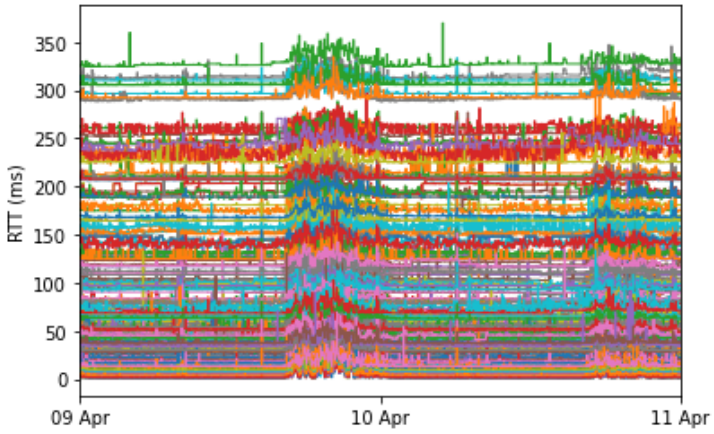


Figure 5.10 – Cluster containing 1227 time series in RD1 obtained using Ward linkage.

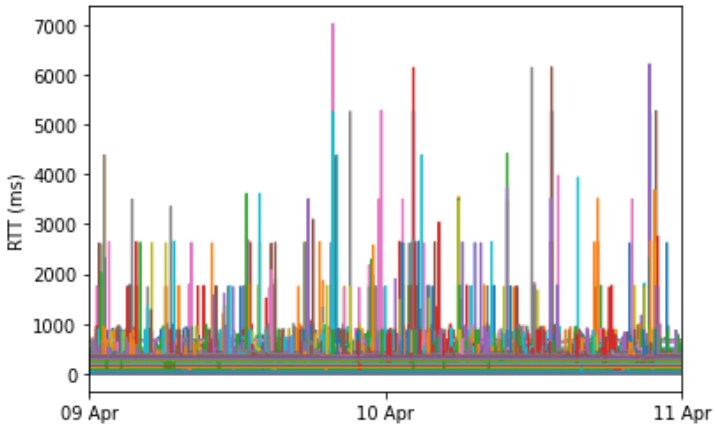


Figure 5.11 – Cluster with 500 time series from RD1 using complete linkage.

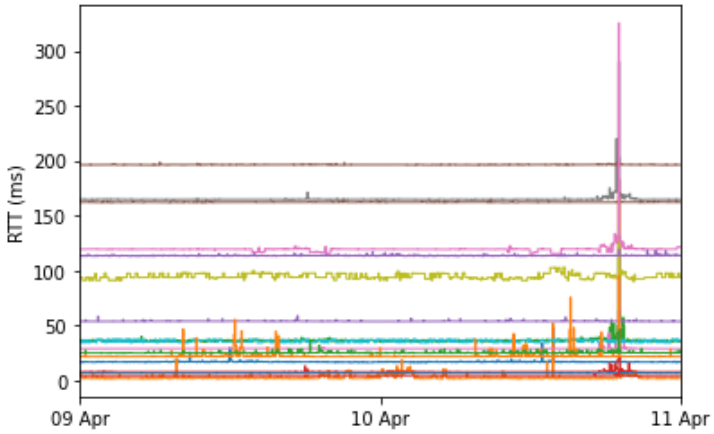


Figure 5.12 – Cluster with 150 time series from RD1 using complete linkage.

After the analysis of the RD1, the same results were observed on RD2. The cut-off threshold was set as well to 0.2 for the complete linkage and to 12 for the Ward method. The complete linkage provided a quite similar distribution of the number of time series per cluster of RD1. Furthermore, this clustering helps to explore the various patterns of delays along the global dataset as the number and type of changes differ from one cluster to another.

The spatial correlation within the clusters An analysis was conducted on each cluster to investigate possible spatial correlations. Specifically, for the RD2 dataset, we selected clusters with more than 10 time series, which represents a set with 3200 clusters. As shown in Figure 5.13, a high number of clusters has series with only a small number of sources or destinations. This suggests that spatial correlations can often be related to measurements with a common source or destination and potentially share a few common links.

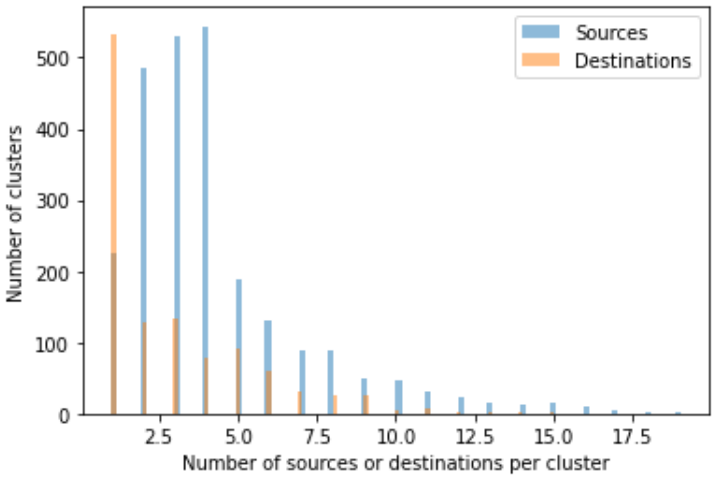


Figure 5.13 – Repartition of the number of common OD pairs per cluster.

5.2.4 Viterbi smoothing

It's important to clarify that the color-coded segments in the following figures represent different states within each segmentation. However, these colors are not consistent from one figure to another. In this section, we evaluate the impact of some parameters on the smoothing, considering the stationarity of the time series. First, we show the impact of ϵ on the transition between states.

For values of ϵ ranging from 10^{-5} to 10^{-2}, the distribution state doesn't change. An example of this can be seen in Figures 5.14, where we can clearly notice that there is little difference between the state distribution for $\epsilon = 10^{-2}$ on the left and $\epsilon = 10^{-5}$ on the right. The state transitions remain consistent which indicates the robustness of the model regarding the choice of ϵ.

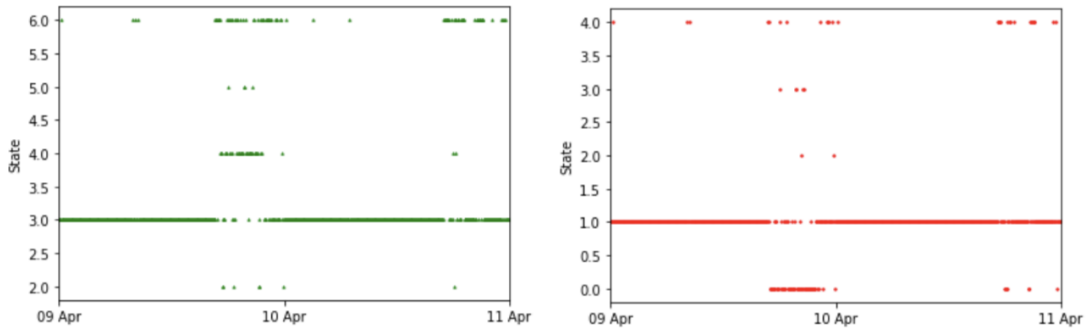


Figure 5.14 – States after the first Viterbi smoothing for $\epsilon = 10^{-2}$ and for $\epsilon = 10^{-5}$.

For the following experiences, we fix ϵ to 0.001 for the first smoothing of Viterbi and to 0.0001 for the second smoothing, since we consider that the states are more stable after the smoothing leading to smaller transition probabilities between states.

We will now evaluate the segmentation quality and the impact of the number of time series in the cluster on both the quality of segmentation and the execution time. For this purpose, we compare the segmentation of each cluster with the segmentation of a representative of that cluster using both our method and the method based on the HDP-HMM.

Comparison with the HDP-HMM method: execution time We measured the processing time of the series per cluster for the method based on the histograms. We can see that a hierarchical plus Viterbi approach offers processing gain by a factor of 9 compared to HDP-HMM, as shown in Table 5.3. The values given in the table are mean values for ten experiments and the time series are 720 timeslots long. Similar results were observed for the RD2 dataset.

Table 5.3 – Comparison of the execution time in seconds.

Method	Hierarchical + Viterbi	HDP-HMM
Execution time in seconds	0.28	2.6

Effect of the smoothing on the number of states In Figure 5.15 we show a summary of

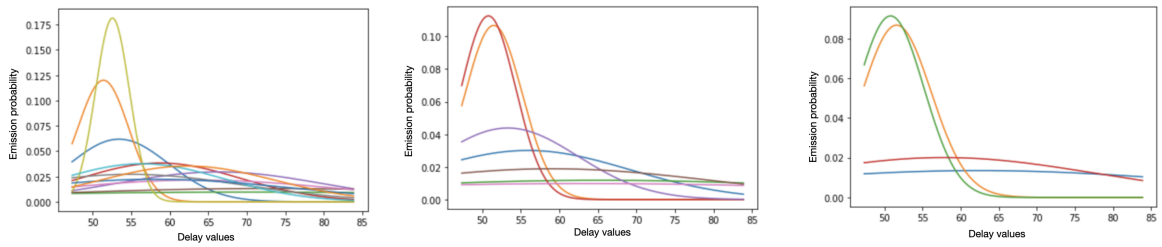


Figure 5.15 – The emission probabilities for each state after hierarchical clustering (12 states), first Viterbi smoothing (7 states) second Viterbi smoothing (4 states) for a time serie from RD1.(From left to right)

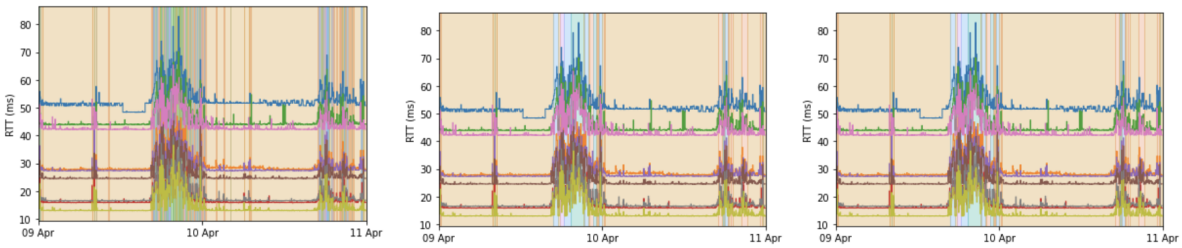


Figure 5.16 – Segmentation result after hierarchical clustering, one and two Viterbi smoothings. We respectively get 12, 7 and 4 states. The cluster contains 10 time series from RD1.(From left to right)

multivariate emission probabilities before and after the Viterbi smoothing. This summary is obtained by gathering data in all dimensions and computing the compounding kernel density estimator. We can notice that the number of states diminishes throughout the process passing from 12 to 4 states. This illustrates the smoothing effect that happens since the model adopted for the transition probabilities favors stability within one state instead of transitions. This enables us to get rid of states with low probability and replace them with more dominant ones in the segment.

Effect of the smoothing on the segmentation Moreover, the impact of Viterbi smoothing is also very clear in terms of segmentation quality. In fact, in Figure 5.16, we can notice

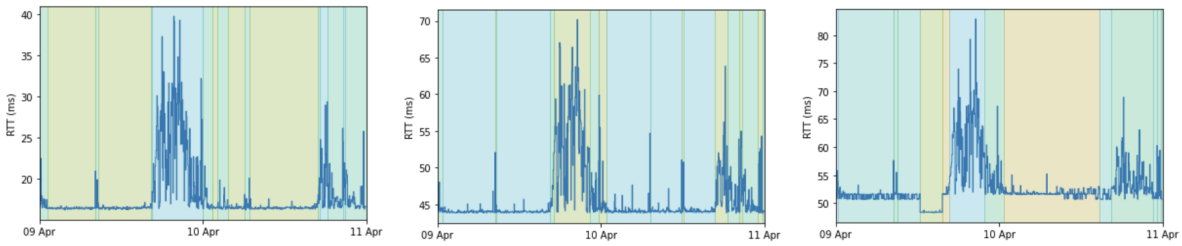


Figure 5.17 – Segmentation of three time series from the cluster in Fig. 5.16 using HDP-HMM. 3 states are found for each time series.

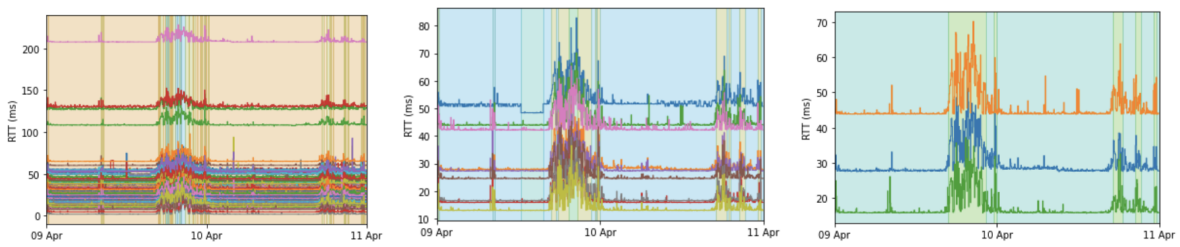


Figure 5.18 – Impact of the number of time series on the segmentation quality, cluster with 130, the full cluster, 10 and 3 time series.

that the segments become more homogeneous. The cluster represented in Figure 5.16 presents roughly two major states: one that is stable and one that has a high variance. At the end of the process in Figure 5.16 on the right, we can see that the stable state (in green) is well segmented, and the unstable state on the other hand is separated into two segments. This segmentation makes sense since the delay distribution is homogeneous within the blue segments while presenting higher variance in the yellow one.

Global patterns VS local patterns When segmenting jointly a high number of series, we can notice the fact that local patterns that are specific for each time serie tend to disappear from the segmentation result of the whole cluster. For instance, many local peaks first detected with hierarchical clustering in Figure 5.16 on the left were merged to a more global state (in green) in Figure 5.16 on the right. This phenomenon can also be noticed in univariate segmentation with the HDP-HMM method. In the series of the first two clusters in Figure 5.17, one can observe that small peaks, relative to the overall variation of the time series, are not segmented independently but are instead included in larger segments. However, when considering isolated peaks or distinctive patterns, these are still identifiable by both our approach and the HDP-HMM and can be detected within separate segments as is the case in Figures 5.17 for the second time serie and in 5.16 for the last time serie.

Effect of the number of time series In Figure 5.18 we highlight the impact of the number of time series inside the cluster on the segmentation quality. In fact, as we can see in Figure 5.17, despite the similarity between these three time series, the HDP-HMM provides a different segmentation for each one. As we can see the highly variable pattern in the middle of these time series has been treated differently: in the first serie it was considered as a whole segment, in the second it was divided into three segments, and in the last serie to two segments. This can mean that each time serie has its own specificities that can be related to the path taken or to other network properties.

When segmenting jointly a high number of time series as is the case in Figure 5.18 on the left, the smoothing effect isn't as efficient. We can see that for a smaller number of time series, 10 and 3 in the last two clusters of Figure 5.18, the segmentation is better smoothed and is closer to the one given by the HDP-HMM. As the number of states tends to increase, maybe a smoothing of transition probabilities stronger than ϵ^N would better mitigate the influence of the number of series.

Effect of the Viterbi regularization on clusters with high cardinality In order to assess the quality of the segmentation in regards to the number of time series within a specific cluster, we employed the adjusted Rand Index as a criterion. This index is computed by comparing the multivariate segmentation of the cluster with the univariate segmentation generated by HDP-HMM for each individual series. Subsequently, an average score was computed to provide insight into the segmentation quality across the entire cluster. We compared the four variants of our approach, and how this average adjusted Rand index evolves in response to the increasing number of time series within the segmented cluster.

It has been shown that the contribution of the post-processing steps always improves the quality of segmentation compared to hierarchical clustering alone. It has also been observed that the Viterbi regularized version with one run shows the best results. Therefore, when treating jointly a high number of time series we choose the regularized version with only one pass of Viterbi as the segmentation strategy.

5.3 Conclusion and takeaways

In this chapter, a multivariate segmentation approach for network delays has been developed. It takes advantage of the spatial correlation between pairs of a network and their temporal stability. The method is based on two successive applications of hierarchical clustering and on the Viterbi algorithm to account for space and time dependencies among series. The hierarchical clustering among series for time series enabled us to group a large number of series having similar patterns. We have seen that the choice of the linkage method has an impact on the homogeneity of the patterns within the same cluster. Complete linkage tends to give more homogeneous clusters but does not have a big compression power, whereas Ward linkage gives clusters with a large number of elements. The spatial correlation within clusters has also been evidenced. Most clusters show the presence of common sources or destinations. After clustering the time series, the process of segmentation begins by applying again the hierarchical clustering followed by the

Viterbi algorithm. We have shown the impact of Viterbi smoothing on the enhancement of the segmentation quality. Its results are similar to the ones given by HDP-HMM, while we have shown that the computational burden is reduced by a factor of 9 compared to HDP-HMM.

Chapter 6

Conclusion and perspectives

6.1 Conclusion

During this thesis, we have been interested in problems of network monitoring. More specifically, we considered problems related to Internet end-to-end delay measurements.

The first aspect of the thesis was about measuring Internet delays and being able to generate delays via network simulation. For this purpose, we developed a network delay simulator that replicates delays within an autonomous system. This was important to obtain a reliable ground truth and to control several factors, including the underlying topology, the delay distribution, and the length of the generated delay time series.

The second one was about the acquisition of real-world datasets. This has been done through exploring the Internet monitoring platform Ripe Atlas and getting to know all the possible options to access measurements: download publicly available measurements or conduct our measurement campaigns. We mainly used two types of measurements available on the platform: round trip times and traceroutes. Several datasets have been recorded.

We generated two datasets by launching our measurement campaign. The first dataset used random anchors in Ripe Atlas, and the second one used probes belonging to Google's autonomous systems. Two other datasets were acquired through existing measurements corresponding to two major Internet incidents touching AMS-IX and DE-CIX. To fetch these delays, we conducted traceroute measurements and filtered the delays associated with paths containing the given IXP in their traceroute results.

The second step after acquiring network delay datasets is their analysis. Within the scope of this thesis, we recognized a critical issue when dealing with Internet delays which is the presence of missing delay values. Missing delays can occur for different reasons, such as device malfunctions, security restrictions, or limitations on the number of allowed measurements. This lead us to the second contribution of this thesis which involves proposing two solutions to infer and recover missing delays via matrix completion and

neural network approaches.

The first model based on non-negative matrix factorization, uses the redundancy and the stability that can be found in round trip time delays. When observed over a long period, from a few hours to several days, delays display an important stationarity. This property makes the network delay matrix low-rank. Two algorithms were considered: the alternated projected gradient and the NeNMF. These two algorithms were tested on synthetic datasets generated by our delay simulator to have the ground truth of the missing delays. It has shown great accuracy in the completion task. We have noticed, however, a significant difference in terms of execution time between the two algorithms. Indeed, the NeNMF was faster than the alternated projected gradient. As a result, we opted for the exclusive utilization of NeNMF during the evaluation phase on the Ripe Atlas dataset. The resulting completion was highly accurate after only a small number of iterations.

The second model used to address the completion task is neural collaborative filtering (NCF). It is based on two neural network architectures: the generalized matrix factorization (GMF) or the multi-layer perceptron (MLP). Both architectures were tested on synthetic and real-world datasets and have shown very low stress. To find the optimal framework for this approach, we have tested the impact of many parameters on these two suggested models such as the optimizer, the learning rate as well as the number of epochs needed. Another important result was the comparison between the NeNMF and the NCF approaches. NCF approach was better than the NeNMF when applied to synthetic datasets but this tendency was reversed for real-world datasets.

The final contribution of this thesis is the proposition of a multivariate segmentation approach for network delay time series. This approach used both the temporal and the spatial dependencies that can be found in delays. It uses hierarchical clustering and the Viterbi algorithm. The initial phase is based on the hierarchical clustering of delay time series. The goal is to form clusters of time series that exhibit similar and synchronous patterns and changes. Subsequently, the time series within the same cluster are concurrently segmented via hierarchical clustering, resulting in the initial clustering of measurement instants. To enhance the temporal consistency of this initial clustering and ensure its smoothness, we applied the Viterbi algorithm as a post-treatment. The efficacy of Viterbi smoothing in improving the quality of segmentation was evidenced through the comparison with the state-of-the-art segmentation method based on HDP-HMM modeling. Finally, we established that our approach significantly reduces computational costs,

compared to the HDP-HMM method.

6.2 Perspectives

During this work, we attempted to incorporate the temporal dependencies within network delays using specific regularization terms to induce a piecewise structure. To further enhance the accuracy of our models, we consider exploring alternative regularization terms in the future. It will reflect the knowledge we have developed regarding the behavior and dynamics of Internet delays. It will be interesting as well, to apply the elaborated methods to different datasets, including different topologies, use cases, and recent incidents.

An aspect of importance that falls in the continuity of our work is the real-time treatment of delays. Network anomalies or incidents can have a significant impact on performance, security, and user experience. Detecting these events as they occur is vital for quick mitigation and issue resolution and can be a tool to raise alerts to network operators to help them investigate the ongoing issue. Similarly, being able to recover missing values simultaneously with their occurrence, can help speed the treatment of the delays and facilitate subsequent analysis.

For this reason, it will be interesting to conceive online completion and anomaly detection algorithms. Regarding the online completion, we can get inspired by the two following works. [8] suggests a non-negative passive-aggressive (NN-PA) [20], a family of online algorithms for non-negative matrix factorization (NMF). The second work [52], presents online collaborative filtering using iterative user clustering (OCTAL). For online anomaly detection, [90] presents an unsupervised multivariate anomaly detection in time series. This approach ensures that the choice of kernels in the machine learning model is not fixed but is dynamically adjusted based on the inherent structure and patterns identified in the unsupervised training data through hierarchical clustering.

Another interesting aspect to be addressed is the development of efficient sampling strategies for network measurements. The goal is to acquire data that brings critical information while reducing the overall volume of measurements to be collected. These strategies aim to reach a balance between resource efficiency and maintaining the accuracy and integrity of the monitoring process. Recent work addressed this for the traffic completion problem [53]. This online completion is based on rank estimation of the considered sliding window, followed by an adaptive sampling scheduling algorithm to acquire the next data points.

REFERENCES

[1] Réka Albert, Hawoong Jeong, and Albert-László Barabási. « Error and attack tolerance of complex networks ». In: *nature* 406.6794 (2000), pp. 378–382.

[2] Vito Walter Anelli et al. « Reenvisioning the comparison between Neural Collaborative Filtering and Matrix Factorization ». In: *Fifteenth ACM Conference on Recommender Systems*. 2021, pp. 521–529.

[3] *API Atlas*. URL: `https://github.com/maxmouchet/atlas-trends-demo` (visited on 01/01/2022).

[4] Åke Arvidsson et al. « Analysis of user demand patterns and locality for YouTube traffic ». In: *Proceedings of the 2013 25th International Teletraffic Congress (ITC)*. 2013, pp. 1–9. DOI: 10.1109/ITC.2013.6662935.

[5] Vaibhav Bajpai, Steffie Jacob Eravuchira, and Jürgen Schönwälder. « Lessons learned from using the ripe atlas platform for measurement research ». In: *ACM SIGCOMM Computer Communication Review* 45.3 (2015), pp. 35–42.

[6] Vaibhav Bajpai and Jürgen Schönwälder. « A survey on internet performance measurement platforms and related standardization efforts ». In: *IEEE Communications Surveys & Tutorials* 17.3 (2015), pp. 1313–1341.

[7] Michael R Berthold and Frank Höppner. « On clustering time series using euclidean distance and pearson correlation ». In: *arXiv preprint arXiv:1601.02213* (2016).

[8] Mathieu Blondel, Yotaro Kubo, and Ueda Naonori. « Online passive-aggressive algorithms for non-negative matrix factorization and completion ». In: *Artificial intelligence and Statistics*. PMLR. 2014, pp. 96–104.

[9] Loıck Bonniot, Christoph Neumann, and François Taıani. « Towards Internet-Scale Convolutional Root-Cause Analysis with DIAGNET ». In: *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE. 2021, pp. 746–755.

[10] Eran Borenstein and Shimon Ullman. « Class-specific, top-down segmentation ». In: *Computer Vision—ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 28–31, 2002 Proceedings, Part II 7*. Springer. 2002, pp. 109–122.

[11] Athman Bouguettaya et al. « Efficient agglomerative hierarchical clustering ». In: *Expert Systems with Applications* 42.5 (2015), pp. 2785–2797.

[12] David Breitgand et al. « Facilitating efficient and reliable monitoring through HAMSA ». In: *Integrated Network Management VIII: Managing It All* (2003), pp. 263–276.

[13] *CAIDA*. URL: `https://www.caida.org` (visited on 09/01/2023).

[14] Emmanuel J Candes and Yaniv Plan. « Matrix completion with noise ». In: *Proceedings of the IEEE* 98.6 (2010), pp. 925–936.

[15] Emmanuel J Candès and Benjamin Recht. « Exact matrix completion via convex optimization ». In: *Foundations of Computational mathematics* 9.6 (2009), pp. 717–772.

[16] Lixiang Chai et al. « Network coordinate system using non-negative matrix factorization based on KL divergence ». In: *2017 19th International Conference on Advanced Communication Technology (ICACT)*. 2017, pp. 193–198. DOI: `10.23919/ICACT.2017.7890082`.

[17] Faicel Chamroukhi et al. « Joint segmentation of multivariate time series with hidden process regression for human activity recognition ». In: *Neurocomputing* 120 (2013), pp. 633–644.

[18] Yudong Chen et al. « Completing any low-rank matrix, provably ». In: *The Journal of Machine Learning Research* 16.1 (2015), pp. 2999–3034.

[19] C. Cicconetti et al. « Quality of service support in IEEE 802.16 networks ». In: *IEEE Network* 20.2 (2006), pp. 50–55. DOI: `10.1109/MNET.2006.1607896`.

[20] Koby Crammer et al. « Online passive aggressive algorithms ». In: (2006).

[21] Frank Dabek et al. « Vivaldi: A decentralized network coordinate system ». In: *ACM SIGCOMM Computer Communication Review* 34.4 (2004), pp. 15–26.

[22] Rahul Deb Das and Stephan Winter. « Automated urban travel interpretation: A bottom-up approach for trajectory segmentation ». In: *Sensors* 16.11 (2016), p. 1962.

[23] Lily Davisson et al. « Reassessing the constancy of end-to-end internet latency ». In: *IFIP Network Traffic Measurement and Analysis Conference*. 2021.

[24] Nicolas Dobigeon, Jean-Yves Tourneret, and Jeffrey D Scargle. « Joint segmentation of multivariate astronomical time series: Bayesian sampling with a hierarchical model ». In: *IEEE Transactions on Signal Processing* 55.2 (2007), pp. 414–423.

[25] Ziqian Dong, Santhanakrishnan Anand, and Rajarathnam Chandramouli. « Estimation of missing RTTs in computer networks: Matrix completion vs compressed sensing ». In: *Computer networks* 55.15 (2011), pp. 3364–3375.

[26] John C Doyle et al. « The "robust yet fragile" nature of the Internet ». In: *Proceedings of the National Academy of Sciences* 102.41 (2005), pp. 14497–14502.

[27] Fabio Duarte. *Number of IoT Devices (2023)*. URL: `https://explodingtopics.com/blog/number-of-iot-devices#` (visited on 05/01/2023).

[28] Wenbo Fan et al. « Deep Learning-Based Dynamic Traffic Assignment With Incomplete Origin–Destination Data ». In: *Transportation Research Record* 2677.3 (2023), pp. 1340–1356.

[29] Anja Feldmann et al. « A year in lockdown: how the waves of COVID-19 impact internet traffic ». In: *Communications of the ACM* 64.7 (2021), pp. 101–108.

[30] Romain Fontugne et al. « Pinpointing delay and forwarding anomalies using large-scale traceroute measurements ». In: *Proceedings of the 2017 Internet Measurement Conference*. 2017, pp. 15–28.

[31] G David Forney. « The viterbi algorithm ». In: *Proceedings of the IEEE* 61.3 (1973), pp. 268–278.

[32] Kensuke Fukuda, Hideki Takayasu, and Misako Takayasu. « Spatial and temporal behavior of congestion in Internet traffic ». In: *Fractals* 7.01 (1999), pp. 23–31.

[33] Maurice Gagnaire et al. « Downtime statistics of current cloud solutions ». In: *International Working Group on Cloud Computing Resiliency, Tech. Rep* (2012).

[34] Ascensión Gallardo Antolın and Jimmy D Ludeña Choez. « NMF-based temporal feature integration for acoustic event classification ». In: (2013).

[35] Sanaa Ghandi et al. « Lightweight Network Delay Segmentation Based on Smoothed Hierarchical Clustering ». In: *2023 IFIP Networking Conference (IFIP Networking)*. IEEE. 2023, pp. 1–9.

[36] Sanaa Ghandi et al. « Neural collaborative filtering for network delay matrix completion ». In: *2022 18th International Conference on Network and Service Management (CNSM)*. IEEE. 2022, pp. 359–363.

[37] Sanaa Ghandi et al. « Non-negative Matrix Factorization For Network Delay Matrix Completion ». In: *7th IFIP/IEEE International Workshop on Analytics for Network and Service Management*. 2022.

[38] Sanaa Ghandi et al. « Segmentation à faible coût des délais réseau basée sur le regroupement hiérarchique ». In: *GRETSI 2023*. 2023.

[39] Enrico Gregori, Luciano Lenzini, and Chiara Orsini. « k-Dense communities in the Internet AS-level topology graph ». In: *Computer Networks* 57.1 (2013), pp. 213–227.

[40] Enrico Gregori et al. « The impact of IXPs on the AS-level topology structure of the Internet ». In: *Computer Communications* 34.1 (2011), pp. 68–82.

[41] Naiyang Guan et al. « NeNMF: An Optimal Gradient Method for Nonnegative Matrix Factorization ». In: *IEEE Transactions on Signal Processing* 60.6 (2012), pp. 2882–2898. DOI: 10.1109/TSP.2012.2190406.

[42] Gonca Gürsun and Mark Crovella. « On traffic matrix completion in the internet ». In: *Proceedings of the 2012 Internet Measurement Conference*. 2012, pp. 399–412.

[43] Trevor Hastie et al. « Matrix completion and low-rank SVD via fast alternating least squares ». In: *The Journal of Machine Learning Research* 16.1 (2015), pp. 3367–3402.

[44] *Hawaiian Airlines grapples with mounting flight delays after temporary internet outage*. URL: https://www.hawaiinewsnow.com/2023/05/12/dozens-hawaiian-airlines-flights-delayed-after-temporary-internet-outage/ (visited on 09/01/2023).

[45] Xiangnan He et al. « Neural collaborative filtering ». In: *Proceedings of the 26th international conference on world wide web*. 2017, pp. 173–182.

[46] Ana Hernandez and Eduardo Magana. « One-way delay measurement and characterization ». In: *International Conference on Networking and Services (ICNS'07)*. IEEE. 2007, pp. 114–114.

[47] José-Alberto Hernández and Iain W Phillips. « Weibull mixture model to characterise end-to-end Internet delay at coarse time-scales ». In: *IEE Proceedings-Communications* 153.2 (2006), pp. 295–304.

[48] Jun Hu et al. « Visibility graph-based segmentation of multivariate time series data and its application ». In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 33.9 (2023).

[49] Yao Hu et al. « Accelerated singular value thresholding for matrix completion ». In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2012, pp. 298–306.

[50] Yaying Hu et al. « Network Latency Estimation with Graph-Laplacian Regularization Tensor Completion ». In: *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE. 2020, pp. 1–6.

[51] ITU. *International Telecommunication Union (ITU)*. URL: `https://www.itu.int/itu-d/reports/statistics/facts-figures-2022/index/` (visited on 05/01/2023).

[52] Prateek Jain and Soumyabrata Pal. « Online low rank matrix completion ». In: *arXiv preprint arXiv:2209.03997* (2022).

[53] Kai Jin et al. « Low cost online network traffic measurement with subspace-based matrix completion ». In: *IEEE Transactions on Network Science and Engineering* 10.1 (2022), pp. 53–67.

[54] Xiaobo Jin et al. « Sparse matrix factorization with L2, 1 norm for matrix completion ». In: *Pattern Recognition* 127 (2022), p. 108655.

[55] Sebastian Kaune et al. « Modelling the internet delay space based on geographical locations ». In: *2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing*. IEEE. 2009, pp. 301–310.

[56] Karamjeet Kaur, Japinder Singh, and Navtej Singh Ghumman. « Mininet as software defined networking testing platform ». In: *International conference on communication, computing & systems (ICCCS)*. 2014, pp. 139–42.

[57] Eunwoo Kim, Minsik Lee, and Songhwai Oh. « Elastic-net regularization of singular values for robust subspace learning ». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 915–923.

[58] Seung-Jean Kim et al. « \ell_1 trend filtering ». In: *SIAM review* 51.2 (2009), pp. 339–360.

[59] Yehuda Koren, Steffen Rendle, and Robert Bell. « Advances in collaborative filtering ». In: *Recommender systems handbook* (2022), pp. 91–142.

[60] Arie MCA Koster and Manuel Kutschka. « Network design under demand uncertainties: A case study on the abilene and GEANT network data ». In: *Photonic Networks, 12. ITG Symposium*. VDE. 2011, pp. 1–8.

[61] *LE RÉSEAU RENATER*. URL: `https://www.renater.fr/reseau/national-et-international/le-reseau-renater/` (visited on 09/01/2023).

[62] Sihyung Lee, Kyriaki Levanti, and Hyong S Kim. « Network monitoring: Present and future ». In: *Computer Networks* 65 (2014), pp. 84–98.

[63] Yongjun Liao et al. « DMFSGD: A decentralized matrix factorization algorithm for network distance prediction ». In: *IEEE/ACM Transactions on Networking* 21.5 (2012), pp. 1511–1524.

[64] *LINX Incidents Log*. URL: `https://www.linx.net/incidents-log/` (visited on 01/01/2022).

[65] Juntao Liu, Caihua Wu, and Wenyu Liu. « Bayesian probabilistic matrix factorization with social relations and item contents for recommendation ». In: *Decision Support Systems* 55.3 (2013), pp. 838–850.

[66] Shuo Liu and Qiaoling Wang. « Poster: online adaptive sampling for network delay measurement via matrix completion ». In: *2019 IFIP Networking Conference (IFIP Networking)*. IEEE. 2019, pp. 1–2.

[67] Xiao Liu et al. « A correlation-matrix-based hierarchical clustering method for functional connectivity analysis ». In: *Journal of neuroscience methods* 211.1 (2012), pp. 94–102.

[68] H-L Lou. « Implementing the Viterbi algorithm ». In: *IEEE Signal processing magazine* 12.5 (1995), pp. 42–52.

[69] Yun Mao and Lawrence K Saul. « Modeling distances in large-scale networks by matrix factorization ». In: *Proceedings of the 4th ACM SIGCOMM conference on Internet Measurement*. 2004, pp. 278–287.

[70] Yun Mao, Lawrence K Saul, and Jonathan M Smith. « Ides: An internet distance estimation service for large networks ». In: *IEEE Journal on Selected Areas in Communications* 24.12 (2006), pp. 2273–2284.

[72] P de Meer et al. « Programmable agents for flexible QoS management in IP networks ». In: *IEEE Journal on Selected Areas in Communications* 18.2 (2000), pp. 256–267.

[73] Wei Meng and Laichun Li. « Matrix completion based adaptive sampling for measuring network delay with online support ». In: *KSII Transactions on Internet and Information Systems (TIIS)* 14.7 (2020), pp. 3057–3075.

[74] Samuel Mercier et al. « Estimation of missing values in a food property database by matrix completion using PCA-based approaches ». In: *Chemometrics and Intelligent Laboratory Systems* 166 (2017), pp. 37–48. ISSN: 0169-7439. DOI: https://doi.org/10.1016/j.chemolab.2017.04.008. URL: https://www.sciencedirect.com/science/article/pii/S0169743916302404.

[75] Ilka Miloucheva, Ulrich Hofmann, and PA Aranda Gutiérrez. « Spatio-temporal QoS pattern analysis in large scale internet environment ». In: *Interactive Multimedia on Next Generation Networks: First International Workshop on Multimedia Interactive Protocols and Systems, MIPS 2003, Naples, Italy, November 18-21, 2003. Proceedings 1*. Springer. 2003, pp. 282–293.

[76] Radhika Mittal et al. « TIMELY: RTT-based congestion control for the datacenter ». In: *ACM SIGCOMM Computer Communication Review* 45.4 (2015), pp. 537–550.

[77] Chris Morris. *Facebook's outage cost the company nearly $100 million in revenue.* URL: https://fortune.com/2021/10/04/facebook-outage-cost-revenue-instagram-whatsapp-not-working-stock/# (visited on 05/01/2023).

[78] Maxime Mouchet, Sandrine Vaton, and Thierry Chonavel. « Poster Abstract: A flexible infinite HMM model for accurate characterization and segmentation of RTT timeseries ». In: *IEEE INFOCOM 2019 - IEEE Conference on Computer*

Communications Workshops (INFOCOM WKSHPS). 2019, pp. 1055–1056. DOI: 10.1109/INFCOMW.2019.8845296.

[79] Maxime Mouchet, Sandrine Vaton, and Thierry Chonavel. « Statistical characterization of round-trip times with nonparametric hidden markov models ». In: *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE. 2019, pp. 43–48.

[80] Maxime Mouchet et al. « Large-scale characterization and segmentation of Internet path delays with infinite HMMs ». In: *IEEE Access* 8 (2020), pp. 16771–16784.

[81] Fionn Murtagh and Pedro Contreras. « Algorithms for hierarchical clustering: an overview ». In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2.1 (2012), pp. 86–97.

[82] Y Nesterov. « A method of solving a convex programming problem with convergence rate *mathcal*{*O*}(1/*k*{2}) ». In: *Sov. Math. Dokl.* Vol. 27.

[83] *Network delay*. URL: https://en.wikipedia.org/wiki/Network_delay#:~:text=Queuing%20delay%20%E2%80%93%20time%20the%20packet,to%20propagate%20through%20the%20media (visited on 05/01/2023).

[84] *Network delay simulator*. URL: https://github.com/Ghandisanaa/network_datasets (visited on 01/01/2022).

[85] TS Eugene Ng and Hui Zhang. « Predicting Internet network distance with coordinates-based approaches ». In: *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*. Vol. 1. IEEE. 2002, pp. 170–179.

[86] *ns-3 Network Simulator*. URL: https://www.nsnam.org (visited on 05/01/2023).

[87] Saravana Murthy Palanisamy et al. « Preserving privacy and quality of service in complex event processing through event reordering ». In: *Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems*. 2018, pp. 40–51.

[88] Konstantina Papagiannaki et al. « Analysis of measured single-hop delay from an operational backbone network ». In: *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*. Vol. 2. IEEE. 2002, pp. 535–544.

[89] Attila Pásztor and Darryl Veitch. « High precision active probing for Internet measurement ». In: *Proceedings of INET'01*. 2001.

[90] Xinggan Peng et al. « An extreme learning machine for unsupervised online anomaly detection in multivariate time series ». In: *Neurocomputing* 501 (2022), pp. 596–608.

[91] Daniel Perdices, Jorge E López de Vergara, and Javier Ramos. « Deep-FDA: Using functional data analysis and neural networks to characterize network services time series ». In: *IEEE Transactions on Network and Service Management* 18.1 (2021), pp. 986–999.

[92] Andy Ramlatchan et al. « A survey of matrix completion methods for recommendation systems ». In: *Big Data Mining and Analytics* 1.4 (2018), pp. 308–323.

[93] Andy Ramlatchan et al. « A survey of matrix completion methods for recommendation systems ». In: *Big Data Mining and Analytics* 1.4 (2018), pp. 308–323. DOI: 10.26599/BDMA.2018.9020008.

[94] *Rand index*. URL: https://en.wikipedia.org/wiki/Rand%5C_index (visited on 01/01/2022).

[95] Steffen Rendle et al. « Neural Collaborative Filtering vs. Matrix Factorization Revisited ». In: *CoRR* abs/2005.09683 (2020). arXiv: 2005.09683. URL: https://arxiv.org/abs/2005.09683.

[96] Steffen Rendle et al. « Neural collaborative filtering vs. matrix factorization revisited ». In: *Proceedings of the 14th ACM Conference on Recommender Systems*. 2020, pp. 240–248.

[97] *RIPE Atlas*. URL: https://atlas.ripe.net/m (visited on 01/01/2022).

[98] Matthew Roughan et al. « 10 lessons from 10 years of measuring and modeling the internet's autonomous systems ». In: *IEEE Journal on Selected Areas in Communications* 29.9 (2011), pp. 1810–1821.

[99] Heath Russell. *How Internet Outages Disrupt Healthcare*. URL: https://www.thousandeyes.com/blog/internet-outages-disrupt-healthcare (visited on 05/01/2023).

[100] Shibani Santurkar et al. « How does batch normalization help optimization? » In: *Advances in neural information processing systems* 31 (2018).

[101] Badrul Sarwar et al. « Item-based collaborative filtering recommendation algorithms ». In: *Proceedings of the 10th international conference on World Wide Web*. 2001, pp. 285–295.

[102] Yasuhiro Sato et al. « Using mixed distribution for modeling end-to-end delay characteristics ». In: *Proceedings of the 8th Asia-Pacific Network and Management Symposium (APNOMS 2005), Japan, Okinawa*. 2005.

[103] N. Seichepine et al. « Piecewise constant nonnegative matrix factorization ». In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2014.

[104] Nicolas Seichepine et al. « Piecewise constant nonnegative matrix factorization ». In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2014, pp. 6721–6725.

[105] Vyas Sekar et al. « cSamp: A system for network-wide flow monitoring ». In: (2008).

[106] D Seung and L Lee. « Algorithms for non-negative matrix factorization ». In: *Advances in neural information processing systems* 13 (2001), pp. 556–562.

[107] Wenqin Shao et al. *Missing measurements on RIPE Atlas*. 2017. arXiv: 1701.00938 [cs.NI].

[108] Wenqin Shao et al. « One-to-one matching of RTT and path changes ». In: *2017 29th International Teletraffic Congress (ITC 29)*. Vol. 1. IEEE. 2017, pp. 196–204.

[109] Mohit Sharma and George Karypis. « Adaptive matrix completion for the users and the items in tail ». In: *The World Wide Web Conference*. 2019, pp. 3223–3229.

[110] Shweta Sharma, Neha Batra, et al. « Comparative study of single linkage, complete linkage, and ward method of agglomerative clustering ». In: *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*. IEEE. 2019, pp. 568–573.

[111] Wei Shen et al. « Inductive matrix completion using graph autoencoder ». In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 1609–1618.

[112] Qiquan Shi, Haiping Lu, and Yiu-Ming Cheung. « Rank-one matrix completion with automatic rank estimation via L1-norm regularization ». In: *IEEE transactions on neural networks and learning systems* 29.10 (2017), pp. 4744–4757.

[113] Zhenyu Shou et al. « Multi-agent reinforcement learning for Markov routing games: A new modeling paradigm for dynamic traffic assignment ». In: *Transportation Research Part C: Emerging Technologies* 137 (2022), p. 103560.

[114] Bing-Feng Si et al. « An improved Dial's algorithm for logit-based traffic assignment within a directed acyclic network ». In: *Transportation planning and technology* 33.2 (2010), pp. 123–137.

[115] Qingyang Song and Abbas Jamalipour. « An adaptive quality-of-service network selection mechanism for heterogeneous mobile networks ». In: *Wireless Communications and Mobile Computing* 5.6 (2005), pp. 697–708.

[116] *Speed Test by Measurement Lab*. URL: `https://speed.measurementlab.net/fr/#/` (visited on 09/01/2023).

[117] Martin Storath et al. « Joint image reconstruction and segmentation using the Potts model ». In: *Inverse Problems* 31.2 (2015), p. 025003.

[118] *Summary of the Amazon EC2 and Amazon RDS Service Disruption in the US East Region*. URL: `https://aws.amazon.com/fr/message/65648/` (visited on 09/01/2023).

[119] Renata Teixeira and Jennifer Rexford. « A measurement framework for pin-pointing routing changes ». In: *Proceedings of the ACM SIGCOMM workshop on Network troubleshooting: research, theory and operations practice meet malfunctioning reality*. 2004, pp. 313–318.

[120] Ruchi Tripathi and Ketan Rajawat. « Adaptive Network Latency Prediction From Noisy Measurements ». In: *IEEE Transactions on Network and Service Management* 18.1 (2021), pp. 807–821.

[121] Guohui Wang, Bo Zhang, and TS Eugene Ng. « Towards network triangle inequality violation aware distributed systems ». In: *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. 2007, pp. 175–188.

[122] Ren Wang et al. « TCP startup performance in large bandwidth networks ». In: *IEEE INFOCOM 2004*. Vol. 2. IEEE. 2004, pp. 796–805.

[123] Andrés Weintraub, Carmen Ortiz, and Jaime González. « Accelerating convergence of the Frank-Wolfe algorithm ». In: *Transportation Research Part B: Methodological* 19.2 (1985), pp. 113–122.

[124] Kun Xie et al. « Accurate recovery of Internet traffic data under dynamic measurements ». In: *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE. 2017, pp. 1–9.

[125] Kun Xie et al. « Accurate recovery of Internet traffic data: A tensor completion approach ». In: *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE. 2016, pp. 1–9.

[126] Kun Xie et al. « Efficiently inferring top-k elephant flows based on discrete tensor completion ». In: *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE. 2019, pp. 2170–2178.

[127] Yahyia Ben Yahmed et al. « ADAPTIVE SLIDING WINDOW ALGORITHM FOR WEATHER DATA SEGMENTATION. » In: *Journal of Theoretical & Applied Information Technology* 80.2 (2015).

[128] Mengyun Yang, Yaohang Li, and Jianxin Wang. « Feature and nuclear norm minimization for matrix completion ». In: *IEEE Transactions on Knowledge and Data Engineering* 34.5 (2020), pp. 2190–2199.

[129] Hanwang Zhang et al. « Discrete collaborative filtering ». In: *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 2016, pp. 325–334.

[130] Tong Zhang and Bin Yu. « Boosting with early stopping: Convergence and consistency ». In: (2005).

[131] Wei Zhang and Jingsha He. « Modeling end-to-end delay using pareto distribution ». In: *Second International Conference on Internet Monitoring and Protection (ICIMP 2007)*. IEEE. 2007, pp. 21–21.

[132] Yin Zhang et al. « Fast accurate computation of large-scale IP traffic matrices from link loads ». In: *ACM SIGMETRICS Performance Evaluation Review* 31.1 (2003), pp. 206–217.

[133] Rui Zhu et al. « Network latency estimation for personal devices: A matrix completion approach ». In: *IEEE/ACM Transactions on Networking* 25.2 (2016), pp. 724–737.

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Analyse des mesures de délai réseau. Méthodes de data mining pour la complétion et la segmentation.

Mot clés : Délais de bout en bout, Mesures d'Internet, Complétion, Segmentation multivariée, Regroupement hiérarchique

Résumé : La croissance exponentielle d'Internet nécessite une supervision régulière des métriques réseau. Cette thèse se concentre sur les délais aller-retour et la possibilité de résoudre les problèmes de données manquantes et de segmentation multivariée. La première contribution comprend l'orchestration de campagnes de mesure des délais, ainsi que le développement d'un simulateur qui génère des traces de délais de bout en bout. La deuxième contribution de cette thèse est l'introduction de deux méthodes de complétion de données manquantes. La première méthode repose sur la factorisation de matrices non négatives et la seconde utilise le filtrage collaboratif neuronal. Testées sur des données synthétiques et réelles, ces méthodes démontrent leur efficacité et précision. La troisième contribution de cette thèse porte sur la segmentation multivariée des délais. Cette approche repose sur le regroupement hiérarchique et se déroule en deux étapes. Dans un premier temps, il s'agit de regrouper les séries de délais afin d'obtenir des séries présentant des variations similaires et synchrones. Ensuite, on segmente de manière conjointe les séries groupées. On utilise le regroupement hiérarchique suivi d'un post-traitement à l'aide de l'algorithme de Viterbi qui vise à lisser le résultat de la segmentation. Cette méthode a été testée sur des traces de délais réels et les résultats indiquent que cette méthode se rapproche de l'état de l'art en matière de segmentation tout en réduisant de manière significative la rapidité et les coûts de calcul.

Title: Analysis of network delay measurements. Data mining methods for completion and segmentation.

Keywords: End-to-end delays, Internet measurements, Completion, Multivariate segmentation, Hierarchical clustering

Abstract: The exponential growth of the Internet requires regular monitoring of network metrics. This thesis focuses on round-trip delays and the possibility of addressing the problems of missing data and multivariate segmentation. The first contribution includes the orchestration of delay measurement campaigns, as well as the development of a simulator that generates end-to-end delay traces. The second contribution of this thesis is the introduction of two missing data completion methods. The first is based on non-negative matrix factorization, while the second uses collaborative neural filtering. Tested on synthetic and real data, these methods demonstrate their efficiency and accuracy. The third contribution of this thesis involves multivariate delay segmentation. This approach is based on hierarchical clustering and is implemented in two stages. Firstly, the delay time series are grouped to obtain, within the same group, series with similar and synchronous variations and trends. Next, the multivariate segmentation step collectively and jointly segments the series within each group. This step uses hierarchical clustering followed by post-processing using the Viterbi algorithm to smooth the segmentation result. This method was tested on real delay traces from two major events affecting two Internet Exchange Points (IXPs). The results show that this method approximates the state-of-the-art in segmentation, while significantly reducing computing speed and costs.