Etude de reseaux complexes et de leurs proprietes pour l'optimisation de modèles de routage

Aurélien Lancin

UNIVERSITÉ DE NICE - SOPHIA ANTIPOLIS
ÉCOLE DOCTORALE DES SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE

pour obtenir le titre de

Docteur en Sciences

de l'Université de Nice - Sophia Antipolis
Mention : Informatique

Présentée et soutenue par
Aurélien LANCIN

Étude de réseaux complexes et de leurs propriétés pour l'optimisation de modèles de routage

Thèse dirigée par : **David COUDERT**
préparée dans le projet COATI, (INRIA, I3S (CNRS/UNS))
dans le cadre du projet Européen EULER

soutenue le 9 décembre 2014

Jury :

Rapporteurs :	Jean-Claude Konig	Professeur, Université de Montpellier
	Bruno Quoitin	Professeur, Université de Mons
	Laurent Viennot	Directeur de recherche, INRIA
Examinateurs :	Dimitri Papadimitriou	Ingénieur principal de recherche, Alcatel-Lucent Bell
	Guillaume Urvoy-Keller	Professeur, Université de Nice - Sophia Antipolis

Remerciements

Je tiens à remercier tout particulièrement :

— David Coudert, mon directeur de thèse, pour ses précieux conseils et qui a su m'encadrer, me soutenir et m'encourager tout au long de ma thèse ;

— Jean-Claude Konig, Bruno Quoitin et Laurent Viennot pour m'avoir fait l'honneur d'accepter d'être les rapporteurs de ma thèse. Merci tout particulièrement à Laurent Viennot pour son retour et ses remarques supplémentaires sur mon manuscrit ;

— Dimitri Papadimitriou et Guillaume Urvoy-Keller pour m'avoir fait l'honneur de participer à mon jury de thèse. Merci tout particulièrement à Dimitri, responsable du projet EULER, sans qui le financement de ma thèse aurait été possible et pour toutes ces discussions durant lesquelles j'ai beaucoup appris ;

— Guillaume Ducoffe, Frédéric Giroire, Luc Hogie et Michel Syska pour avoir pris le temps de relire ma thèse et pour leurs précieuses remarques ;

— Patricia Lachaume, toujours présente, prête à nous rendre service et à nous sortir du patatra administratif ;

— Nicolas Nisse pour ses conseils et remarques toujours pertinentes et Joanna Moulierac pour m'avoir accepté à ses côtés durant mon monitorat à l'IUT ;

— Issam, pour ces deux années de cohabitation et avec qui j'ai beaucoup appris ;

— Léonardo et Ronan pour ces années de grand n'importe quoi pendant lesquelles je me suis bien marré ;

— Toute l'équipe COATI pour l'accueil chaleureux, la gentillesse de ses membres et l'ambiance excellente dans laquelle j'ai passé ces cinq années ;

— Ma famille, qui m'a toujours soutenu dans mon parcours et Aurélie qui m'a supporté et encouragé durant toute cette dernière année.

Un grand merci à tous.

Sophia Antipolis,
le 1er décembre 2014.

Étude de réseaux complexes et de leurs propriétés pour l'optimisation de modèles de routage

Résumé : Cette thèse s'intéresse aux problématiques de routage dans les réseaux, notamment dans le graphe des systèmes autonomes (AS) d'Internet. Nous cherchons d'une part à mieux comprendre les propriétés du graphe de l'Internet qui sont utiles dans la conception de nouveaux paradigmes de routage. D'autre part, nous cherchons à évaluer par simulation les performances de ces paradigmes.

La première partie de mes travaux porte sur l'étude d'une propriété métrique, l'hyperbolicité selon Gromov, utilisée dans la conception de nouveaux paradigmes de routage. Je présente dans un premier temps une nouvelle approche pour le calcul de l'hyperbolicité d'un graphe utilisant une décomposition du graphe par les cliques-séparatrices et la notion de paires éloignées. Je propose ensuite un nouvel algorithme pour le calcul de l'hyperbolicité qui, combiné avec la méthode de décomposition par les cliques-séparatrices, permet son calcul sur des graphes composés de 58 000 sommets en quelques heures.

La deuxième partie de mes travaux porte sur le développement de DRMSim, une nouvelle plate-forme de simulation de modèles de routage dynamiques. Celle-ci permet l'évaluation des performances des schémas de routage et leur comparaison au protocole de référence, le protocole de routeur frontière, BGP. DRMSim a permis l'étude par simulation de différents schémas de routage compact sur des topologies à O(10k) nœuds. Je détaille l'architecture de DRMSim et quelques exemples d'utilisation. Puis, je présente une étude réalisée en vue de développer une version parallèle et distribuée de DRMSim dans le cadre de la simulation de BGP.

Mots clés : Internet, Algorithme, Graphe, Hyperbolicité, Décomposition de graphe, Routage, BGP, Simulation, Simulation parallèle distribuée.

Study of complex networks properties for the optimization of routing models

Abstract : This thesis considers routing issues in networks, and particularly the graph of the autonomous systems (AS) of the Internet. Firstly, we aim at better understanding the properties of the Internet that are useful in the design of new routing paradigms. Secondly, we want to evaluate by simulation the performance of these paradigms.

The first part of my work concerns the study of the Gromov hyperbolicity, a useful metric property for the design of new routing paradigms. I show how to use a decomposition of the graph by clique-separators as a pre-processing method for the computation of the hyperbolicity. Then, I propose a new algorithm to compute this property. Altogether, these methods allows us for computing the hyperbolicity of graphs up to 58 000 nodes.

The second part of my work concerns the development of DRMSim, a new Dynamic Routing Model Simulator. It **facilitates** the evaluation of the performances of various routing schemes and their comparison to the standard routing scheme of the Internet, the border router protocol BGP. Using DRMSim, we performed simulations of several compact routing schemes on topologies up to O(10k) nodes. I describe its architecture and detail some examples. Then, I present a feasibility study for the design of a parallel/distributed version of DRMSim in order to simulate BGP on larger topologies.

Keywords : Internet, Algorithm, Graph, Hyperbolicity, Graph decomposition, Routing, BGP, Simulation, Parallel Distributed Simulation.

Table des matières

Introduction **i**
I Étude de l'hyperbolicité dans les grands graphes . . . ii
II Simulation de modèles de routage dans les grands réseaux. . . . iii
III Contributions et plan détaillé . . . v
IV Conclusion . . . vii
V Publications . . . ix

I Méthodes de calcul de l'hyperbolicité selon Gromov dans de grands graphes **1**

1 Pré-traitement et décomposition de graphe pour le calcul de l'hyperbolicité **3**
1 Définitions de la δ-hyperbolicité . . . 4
2 Méthodes de pré-traitement du graphe . . . 9
2.1 Utilisation des paires éloignées pour la réduction du nombre de quadruplets à visiter . . . 9
2.2 Décomposition en composantes biconnexes . . . 10
3 Méthode de décomposition du graphe par les cliques-séparatrices . . . 12
3.1 Relations entre l'hyperbolicité du graphe et ses séparateurs . . . 12
3.2 Hyperbolicité et cliques-séparatrices . . . 13
3.3 Hyperbolicité et décomposition par les cliques-séparatrices . . . 17
3.4 Approximation de l'hyperbolicité . . . 20
4 Méthode de substitution pour le calcul exact de l'hyperbolicité . . . 20
4.1 Méthode de substitution des sommets à partir d'une $(A|B)$-clique-séparatrice . . . 20
4.2 Calcul de la valeur exacte de l'hyperbolicité . . . 21
5 Résultats expérimentaux . . . 22
5.1 Données d'expérimentation . . . 22
5.2 Résultats empiriques . . . 23
5.3 Analyse de la décomposition par les cliques-séparatrices et de la construction des substituts . . . 25
6 Conclusion . . . 27

2 Algorithme exact pour le calcul de l'hyperbolicité dans de grands graphes **31**
1 Algorithme exact pour le calcul de l'hyperbolicité . . . 32
1.1 Algorithme pour le calcul de l'hyperbolicité . . . 33
1.2 Utilisation des paires éloignées . . . 34
1.3 Algorithme parallèle pour le calcul de l'hyperbolicité . . . 34
2 Résultats expérimentaux . . . 35
2.1 Méthodes de pré-traitement des graphes . . . 36
2.2 Évaluation des performances de l'algorithme . . . 37
2.3 Décomposition et calcul de l'hyperbolicité dans les grands graphes . . . 38
2.4 Coût du pré-traitement du graphe pour le calcul de l'hyperbolicité . . . 42
2.5 Évaluation des performances de l'algorithme parallèle . . . 42
3 Heuristique pour le calcul de l'hyperbolicité . . . 44
3.1 Conception de l'heuristique . . . 44
3.2 Calculs expérimentaux de l'heuristique . . . 45

3.3 Analyse des performances de l'heuristique. 46
4 Conclusion 47

II Simulation de modèles de routage dynamiques dans de grands réseaux 49

3 Introduction au routage dans l'Internet 51
1 Routage dans les réseaux 52
1.1 Le protocole IP 52
1.2 Architecture d'un routeur IP 52
2 Routage dans l'Internet 54
2.1 L'architecture d'Internet 54
2.2 Le routage intra-domaine 56
2.3 Le routage inter-domaine 57
2.4 Le protocole de routage BGP 59
3 Problématiques du routage dans l'Internet 63
3.1 Évolution de l'Internet 63
3.2 Conception de nouveaux systèmes de routage pour l'Internet 65
4 Conclusion 69

4 Méthodes de simulation de modèles de routage dans de grands réseaux dynamiques 71
1 Introduction 72
1.1 Émulation du protocole BGP 72
1.2 Simulation de protocoles de routage 74
1.3 Motivations pour le développement de DRMSim 76
2 Principes et méthodologie pour la simulation 77
2.1 Définitions et terminologie 77
2.2 Modèle de simulation d'un système 78
2.3 Simulation à événements discrets 78
3 DRMSim, un simulateur de modèles de routage dynamiques 79
3.1 Motivations et objectifs 79
3.2 Architecture de DRMSim 80
3.3 Exemple d'exécution 89
4 Le protocole de routeur frontière dans DRMSim 91
4.1 Description du modèle 92
4.2 Transmission des messages de mise à jour BGP 93
4.3 Métriques dans BGP 93
4.4 Validation du modèle 94
4.5 Complexité de la simulation de BGP dans DRMSim 94
5 Expérimentations de modèles de routage avec DRMSim 96
5.1 Un schéma de routage compact efficace dans les graphes k-chordaux 96
5.2 Expérimentations de schémas de routage compacts universels ou dédiés aux graphes sans échelle 97
6 Conclusion 100

5 Étude de faisabilité de la distribution des simulations de BGP 103
1 Simulation discrète parallèle et distribuée 104
1.1 Coût des communications entre LP 104
1.2 Erreurs de causalités 104
1.3 Protocole de synchronisation, l'approche conservatrice 105
1.4 Paralysie du système 107

2 Distribution des simulations de BGP dans DRMSim 107
2.1 Modèles de communication 107
2.2 Algorithme de partition 110
3 Simulation des solutions A et B 112
3.1 Scénarios de simulation 112
3.2 Environnement d'exécution 113
3.3 Résultats de simulation 113
4 Conclusion 116

Conclusion et perspectives **119**

Bibliographie **123**

Annexe **131**

Table des figures

1.1 Représentation géométrique du produit de Gromov. 5
1.2 Définitions des triangles δ-fin et δ-mince. 6
(a) Triangle δ-fin. 6
(b) Triangle δ-mince. 6
1.3 La condition des 4 points. 6
1.4 Exemple de la définition de la condition des 4 points. 7
1.5 La clique et son espace métrique avec $\delta = 0$. 8
(a) $K5$. 8
(c) Propriété métrique de la clique. 8
1.6 L'arbre et l'unicité des plus courts chemins en opposition à la grille et la dispersion des plus courts chemins. 8
(a) Arbre avec $\delta = 0$. 8
(b) Grille $n \times m$ avec $\delta = n - 1$. 8
1.7 Les deux seules paires distantes (u, v) et (x, y) dans une grille. Tous les autres sommets sont inclus dans les plus courts chemins entre les paires (u, v) et (x, y). 10
1.8 L'hyperbolicité de deux composantes connexes A et B séparées par le point d'articulation x. 11
(a) $\delta = 0$. 11
(b) $\delta(a, b_1, b_2, b_3) = \delta(x, b_1, b_2, b_3)$. 11
1.9 $(A|B)$-séparateurs avec les diamètres $k = 0$, $k = 1$ et $k = 2$. 12
(a) Un point d'articulation x $(k = 0)$. 12
(b) Une $(A|B)$-clique-séparatrice X $(k = 1)$. 12
(c) Un $(A|B)$-séparateur X $(k = 2)$. 12
1.10 Décomposition du graphe par les cliques-minimales-séparatrices. 13
(a) Un graphe et ses quatre cliques-minimales-séparatrices (b, h), (b, d), (d, f) et (f, h). 13
(b) Les cinq atomes de la décomposition du graphe. 13
1.11 Un quadruplet $(a_1, a_2|b_1, b_2)$ dont les sommets sont séparés par un $(A|B)$-séparateur X. 14
1.12 Un quadruplet $(a|b_1, b_2, b_3)$ dont les sommets sont séparés par une $(A|B)$-clique-séparatrice X. 14
1.13 Substitution du sommet a par le sommet a^*. 15
(a) Un sommet a à distance m de x_1, x_3 et $m + 1$ de x_2, x_4. 15
(b) Le sommet a^* substitue le sommet a en étant connecté aux sommets de X tels que $\{x \in X : \mathbb{d}(a, x) = \mathbb{d}(a, X)\}$. 15
1.14 $\delta(a, b_1, b_2, b_3) \leq \delta(x, b_1, b_2, b_3) + 1/2$. Au centre, les plus courts chemins formant le quadruplet (x, b_1, b_2, b_3). En pointillé et au centre, ceux formant le quadruplet (a, b_1, b_2, b_3). 16
1.15 Un graphe séparé en quatre composantes connexes par une clique-séparatrice X. 17
(a) Les composantes connexes C_1, C_2, C_3 et C_4 séparées par une clique-séparatrice X. 17
(b) Les composantes connexes G_1, G_2, G_3 et G_4 tels que $G_i = G[C_i \cup X]$. 17
1.16 Un graphe 2-hyperbolique avec cinq atomes ; quatre sont 0-hyperboliques, un est 1-hyperbolique. 18

1.17 Bornes de l'hyperbolicité à partir des sommets des cliques-séparatrices. Au centre, les plus courts chemins formant le quadruplet (x_a, b, c, x_d). En pointillé et au centre, ceux formant le quadruplet (a, b, c, d). 19
(a) $\delta(a, b, c, d) \leq \delta(x_a, b, c, x_d) + 1/2$. 19
(b) Si $\delta(G) \geq \frac{3}{2}$ alors $\delta(a, b, c, d) \leq \delta(G[A] + 1$. 19
1.18 Exemple de substitution des atomes d'un graphe. 22
(a) Arbre des atomes de la décomposition par les cliques-minimales-séparatrices du graphe. 22
(b) Les substituts G_1^*, G_2^*, G_3^* et G_4^*. 22
1.19 En figure 1.19a, la distribution cumulée de la taille des composantes biconnexes. En figure 1.19b, celle des atomes dans la LBC. 28
(a) Composantes biconnexes. 28
(b) Atomes de la LBC. 28
1.20 En figure 1.20a, la distribution cumulée de la taille des cliques-séparatrices dans le LA. En figure 1.20b, celle du pourcentage de sommets séparés en fonction de la taille des cliques-séparatrices du LA. 29
(a) Cliques-séparatrices dans le LA. 29
(b) Sommets séparés du LA. 29
1.21 En figure 1.21a, la distribution cumulée du nombre de sommets simpliciaux connectés au LA. Cette distribution est normalisée par la taille de la LBC en fonction de la taille des cliques-séparatrices auxquelles les sommets simpliciaux sont connectés. En figure 1.21b, celle du nombre de sommets simpliciaux connectés au LA normalisée par la taille de la LBC. 30
(a) Sommets simpliciaux connectés au LA en fonction de la taille des cliques-séparatrices. 30
(b) Distribution des degrés des sommets simpliciaux du LA. 30

2.1 Nombre de quadruplets visités pour atteindre la borne inférieure et supérieure. Le nombre de quadruplets pour atteindre la borne inférieure est tracé de gauche à droite, celui pour atteindre la borne supérieure de droite à gauche. 41
(a) Cartes CAIDA depuis 2004. 41
(b) Cartes DIMES depuis 2010. 41
2.2 Comparaison des temps de calcul de l'hyperbolicité sur le LS (incluant le temps de calcul du LS lui-même) et celui sur la LBC et le temps estimé donné dans [FIV12]. 43
(a) Nombre de quadruplets visités dans la LBC et dans le LS. 43
(b) Temps de calcul de l'hyperbolicité et du LS. 43
2.3 Représentation du certificat de sommets (a, b, c, d) recherché par l'heuristique et maximisant l'hyperbolicité. 44

3.1 Architecture d'un routeur 53
3.2 Exemple d'interconnexion entre AS. 54
3.3 Politiques de routage entre AS clients, fournisseurs et pairs. 58
3.4 Sessions entre routeurs iBGP et eBGP et protocole IGP. 60
3.5 Architecture d'un routeur BGP 62
3.6 Évolution du nombre d'AS dans l'Internet. 64
3.7 Évolution du nombre d'entrées dans la FIB de l'AS 65000. 65
3.8 Architecture d'un système. 66
3.9 Modélisation du schéma et du protocole de routage. 68

4.1 Modèle de simulation. 81
4.2 Architecture de DRMSim 82
4.3 Modèle du système. 84

4.4 Modèle de transmission et séquence de transmission d'un message. 85
4.5 Modèle de dynamique et séquence de génération des activités exogènes. . . 86
4.6 Modèle de mesure. 87
4.7 Modèle de routage et séquence de transmission d'un message. 88
4.8 Évolution de l'étirement additif du schéma de routage compact dans les graphes k-chordaux. Figure obtenue dans [HPTM10]. 97
4.9 Résultats des simulations des schémas DCR, OMNI, HDLBR et CLUSTER obtenus à partir de DRMSim. Figures obtenues dans [Gla13]. 99
(a) Étirement multiplicatif. 99
(b) Taille des tables de routage en pourcentage des destinations stockées. 99
(c) Coût des communications. 99

5.1 L'événement E1 affecte l'événement E3 en planifiant un troisième événement E2 qui modifie la variable d'état utilisée par E3. Une exécution séquentielle des trois événements est donc nécessaire. 105
5.2 Exemple d'exécution d'un protocole de synchronisation conservateur. 106
5.3 Modèles de communication. La solution A prend en charge l'échange direct des informations entre les routeurs de différents LP. La solution B recopie les routeurs frontières et assure leur mise à jour. 108
5.4 Évolution du nombre d'entrées annoncées dans les messages de mise à jour BGP. 114

Liste des tableaux

1.1 Caractéristiques des graphes des réseaux de collaborations. 24
1.2 Données concernant les cliques-minimales-séparatrices des graphes. 26

2.1 Comparaison avec la méthode de parallélisation massive du calcul de l'hyperbolicité 37
2.2 Performances détaillées de l'algorithme 1. 38
2.3 Performances des algorithmes sur des graphes de grande taille. 39
2.4 Comparaisons des valeurs d'hyperbolicité trouvées avec la nouvelle heuristique proposée (colonne δ_h), le processus d'échantillonnage de [KNS13] (colonne δ_s) et d'une heuristique aléatoire (colonne δ_r) 46

4.1 Comparaison des simulateurs de protocoles de routage. 76
4.2 Comparaison des probabilités de convergence pour $k = 1$ en fonction de la probabilité p d'une session entre deux nœuds de la topologie. 94

5.1 Volume des communications entre les nœuds frontières des solutions A et B. 109
5.2 Coût mémoire des solutions A et B. 110
5.3 Nombre d'entrées annoncées dans les messages de mise à jour BGP. 115

Introduction

Internet, qui à l'origine était un réseau militaire et universitaire interconnectant quelques dizaines d'acteurs, est devenu le plus grand système distribué de communication au monde. Actuellement, le cœur du réseau Internet est composé d'environ 48 000 systèmes autonomes (ou AS, Autonomous System) [Bat]. Ces AS sont des entités administratives autonomes. Ce sont par exemple des fournisseurs d'accès Internet, des universités ou encore des réseaux d'acteurs privés ou gouvernementaux.

Ces deux dernières décennies ont vu s'accroître l'usage des services et le développement de nouvelles technologies utilisant le réseau Internet pour communiquer. Des millions de terminaux sont maintenant connectés en permanence, des usages de plus en plus nombreux se développent, nous rendant dépendants du bon fonctionnement du réseau.

Cette évolution rapide d'Internet et de ses usages impose de nombreuses contraintes aux opérateurs réseaux afin de garantir le bon fonctionnement du réseau. L'augmentation du nombre d'AS, l'explosion des communications haut débit à partir de lignes fixes et mobiles, l'accès aux fournisseurs de contenu et les réseaux en nuage sont quelques exemples de ces nouveaux usages.

Une des fonctions essentielles d'Internet est d'assurer le routage distribué des données entre une source et une ou plusieurs destinations. Chaque nœud de la topologie exécute une fonction de routage qui, pour chaque destination atteignable, calcule un chemin sans boucle pour transmettre les données vers cette destination. Le système de routage d'Internet est assuré par le protocole de routage frontière [RLH06] (ou BGP, Border Gateway Protocol). L'architecture de routage d'Internet et plus particulièrement BGP, montre ses limites en termes d'ingénierie de trafic et du nombre de systèmes autonomes pouvant être supportés. L'algorithme à vecteur de chemin sur lequel repose BGP est à l'origine du problème de passage à l'échelle du protocole :

— le nombre d'entrées stockées dans les tables de routage pour assurer un routage des plus courts chemins sans boucle est linéaire en le nombre d'AS ;

— de nombreux messages doivent être échangés afin de *converger*, c'est-à-dire jusqu'à ce que l'ensemble des routeurs ait une vue cohérente de la topologie.

Afin de pallier à ces limitations, ces deux dernières décennies ont vu émerger deux catégories d'études. Une première catégorie vise à améliorer le fonctionnement de BGP afin d'assurer la pérennité de l'architecture actuelle. La seconde porte sur la conception de nouveaux *schémas* de routage en vue du remplacement de BGP. Un schéma de routage est un algorithme permettant de calculer une table de routage afin d'acheminer des données entre deux nœuds du réseau. Certains schémas de routage ont été développés afin de prendre en compte certaines spécificités de la topologie d'Internet [NRS12, LABJ01, PKBV09, Gla13]. Les performances de ces nouveaux schémas doivent ensuite être évaluées et comparées à celles du protocole de référence dans Internet, BGP.

Dans cette thèse, mes travaux portent sur deux aspects distincts mais complémentaires pour l'étude des réseaux et de leurs protocoles de routage : l'étude de l'hyperbolicité dans les grands graphes d'une part, et la simulation de modèles de routage dans les grands réseaux d'autre part.

I Étude de l'hyperbolicité dans les grands graphes

De nombreuses propriétés structurelles (distribution des degrés, clustering coefficient, structure arborescente, etc) sont étudiées depuis plusieurs années dans les grands graphes modélisant les réseaux sociaux, les réseaux biologiques, les réseaux de citations ou Internet. Ces études permettent de mieux comprendre la structure de ces réseaux, de les modéliser et de proposer de nouveaux algorithmes de routage pour Internet. Plus particulièrement, certaines études récentes [AD14, Dra13] se sont intéressées à différents paramètres mesurant la structure *arborescente* présente dans les réseaux complexes. Dans ce contexte, une propriété particulièrement étudiée est celle de l'hyperbolicité selon Gromov [Gro87]. Une définition de l'hyperbolicité, nommée *condition des quatre points*, définit un graphe $G = (V, E)$ comme δ-hyperbolique si pour tout quadruplet $a, b, c, d \in V$, les deux plus grandes des sommes $S1 = d(a, b) + d(c, d)$, $S2 = d(a, c) + d(b, d)$, et $S3 = d(a, d) + d(b, c)$ diffèrent d'au plus 2δ. L'hyperbolicité de G désigne la valeur du plus petit δ vérifiant cette définition.

L'hyperbolicité d'un graphe exprime comment son espace métrique se rapproche de celui d'un arbre. La propriété d'hyperbolicité d'un graphe s'est révélée être d'une grande importance pour le routage dans les réseaux. Dans [KPK+10], les auteurs fournissent un cadre géométrique pour l'étude de la structure de réseaux complexes et mettent en lumière la géométrie hyperbolique sous-jacente de ces réseaux. Ils montrent que les réseaux à géométrie hyperbolique sont robustes et permettent une navigabilité optimale. Par exemple, il est possible de plonger avec précision la topologie de l'Internet dans un espace hyperbolique [BPK10]. Il a été montré qu'un algorithme de routage glouton utilisant les coordonnées des sommets et les distances entre sommets dans cet espace métrique a un faible *étirement* par rapport aux plus courts chemins [PKBV09]. Pour rappel, l'étirement d'un chemin est dit multiplicatif si il mesure le rapport entre la longueur du chemin calculée par l'algorithme de routage sur celle du plus court chemin. On dira qu'il est additif dans le cas de la différence entre les longueurs de ces deux chemins. Plus généralement, l'efficacité des algorithmes de routage dépend de la nature hyperbolique de l'espace métrique d'une topologie donnée [CDE+12]. La notion d'hyperbolicité a aussi été utilisée pour exprimer la latence de l'Internet comme une métrique arborescence [RMK+09], comprendre et améliorer la fiabilité et la sécurité des réseaux [NS11, JL04] ou encore mesurer la distance géodésique entre des arbres phylogénétiques pour leur classification [CH12, DHK+11].

Méthodes pour le calcul exact de l'hyperbolicité

Un algorithme de base pour le calcul de l'hyperbolicité d'un graphe G avec une complexité temporelle en $O(n^4)$ peut être facilement déduit de la définition de la condition des quatre points. Il suffit en effet de parcourir l'ensemble des quadruplets de sommets dans un graphe G et calculer la valeur de l'hyperbolicité pour chacun des quadruplets $\delta(a, b, c, d)/2$. La plus grande valeur de δ trouvée étant l'hyperbolicité de G. Trois implémentations de cet algorithme, utilisées dans des cadres différents, sont disponibles pour le calcul de l'hyperbolicité d'un graphe :

— une première implémentation en Python a été utilisée dans [CH12] pour évaluer et classifier des arbres phylogénétiques avec moins de 500 sommets en moins d'une minute, elle peut être obtenue à partir du paquet Distory [CH10, R C14].

— une seconde implémentation massivement parallèle de cet algorithme a été utilisée par [ASHM13] pour calculer l'hyperbolicité d'un graphe de 8 104 sommets à partir de 1015 unités centrales de calcul (CPU) et a requis 13 295 secondes ($\approx$ 3.5 heures) sur un SGI Altix UV 1000 équipé de processeurs Intel Nehalem EX 2.0 GHz ;

— enfin, une troisième implémentation utilisée dans [SG11] pour calculer l'hyperbolicité d'une carte CAIDA à environ vingt mille sommets a requis quatre mois de calcul.

Afin de rendre possible le calcul de l'hyperbolicité sur de tels graphes, les auteurs ont tout d'abord calculé les composantes biconnexes du graphe, puis ils ont appliqué l'algorithme de base en considérant uniquement les quadruplets formés à partir des *paires éloignées* de la plus grande composante biconnexe.

Dans ces travaux, seul l'algorithme de base en $O(n^4)$ est utilisé. Les performances des calculs dépendent uniquement de la qualité d'implémentation de l'algorithme, de la puissance des machines utilisées, de techniques de parallélisation ou de méthodes de pré-traitement. Aucun nouvel algorithme n'est proposé. Au final, seules les techniques de parallélisation massives permettent d'atteindre des graphes de plusieurs milliers de sommets en quelques heures de calcul. De nouvelles approches pour le calcul de l'hyperbolicité doivent donc être développées.

Méthodes pour approcher la valeur de l'hyperbolicité

À défaut de pouvoir proposer de meilleurs algorithmes pour le calcul exact de l'hyperbolicité sur de grands graphes, plusieurs heuristiques ont été proposées et appliquées sur les cartes CAIDA [CAI13b] ou des graphes aléatoires de Erdös-Rényi [KNS13, NST12, dMSV11, Sha11]. Un algorithme 2-approché en temps cubique est obtenu en choisissant un sommet et en évaluant l'ensemble des quadruplets l'incluant [CDE+08]. Son temps de calcul a été réduit à $O(n^{2.69})$ [FIV12]. Récemment, un algorithme $2+\varepsilon$-approché avec une complexité temporelle en $\tilde{O}(\varepsilon^{-1}n^{2.373})$ a été proposé dans [Dua14]. Dans [CD14], il est montré que reconnaître des graphes $\frac{1}{2}$-hyperboliques est équivalent à reconnaître si il existe ou non un cycle sans corde de taille 4 dans le graphe, ceci peut être décidé en temps $O(n^{3.26})$ à partir de la multiplication rapide de matrices rectangulaires [LG12].

Méthodes de pré-traitement du graphe

Afin de réduire d'avantage la complexité temporelle du calcul de l'hyperbolicité, il est possible d'appliquer des méthodes de pré-traitement et de décomposition du graphe. Certaines méthodes ont déjà été utilisées dans différents travaux pour réduire la taille du graphe. Les auteurs dans [SG11] ont prouvé que l'hyperbolicité d'un graphe G est égale à l'hyperbolicité maximale des graphes résultant de sa décomposition *modulaire* [HP10, Gal67] ou en *coupe* [Cun82, CE80]. Dans [SG11], une heuristique pour le calcul de l'hyperbolicité basée sur la notion de *paires éloignées* a aussi été proposée.

De nouvelles techniques pour le calcul exact ou approché de l'hyperbolicité doivent être développées. L'objectif étant de pouvoir, en pratique, calculer l'hyperbolicité de graphes composés de plusieurs dizaines de milliers de sommets en quelques heures. C'est l'objet des travaux qui seront présentés dans la première partie de cette thèse.

II Simulation de modèles de routage dans les grands réseaux.

Avant de pouvoir introduire de nouveaux schémas de routage ou mises à jour de BGP dans l'architecture de routage d'Internet, il est important de tester leurs performances et de vérifier s'ils se comportent comme attendu. Une étude comparative entre le protocole de routage existant, BGP, et les nouveaux schémas de routage proposés doit être menée. Les topologies sur lesquelles sont expérimentés ces modèles doivent de plus avoir des propriétés proches de celles d'Internet. Il doit aussi être possible de reproduire la dynamique du réseau lors de l'évaluation de ces schémas.

Diverses catégories de simulateurs de protocoles de routage existent. Certains simulateurs tels que NS-2 [NS-], SSFNet [SSF] ou encore J-Sim [JS] permettent l'étude de différents protocoles de routage au sein d'une même plate-forme. La simulation de ces protocoles se fait à un niveau microscopique ce qui nécessite d'importantes ressources mémoire et processeur et limite les simulations à des topologies de l'ordre d'une centaine de nœuds.

D'autres simulateurs comme BGP++ [DR04], C-BGP [QU05] ou SimBGP [Sim] sont dédiés à la simulation de BGP. BGP++ est un simulateur basé sur NS-2 et souffre des mêmes limitations en termes de ressources mémoire et processeur. C-BGP et SimBGP permettent la simulation de BGP sur des topologies du même ordre de grandeur que celle d'Internet.

Il est donc difficile d'expérimenter de nouveaux schémas de routage sur des topologies de grandes tailles et de les comparer à BGP au sein d'une même plate-forme de simulation. De plus, un *modèle* incluant uniquement les procédures et structures strictement nécessaires au fonctionnement du schéma est souvent suffisant lors des premières étapes d'évaluation d'un nouveau schéma de routage.

Le simulateur DRMSim

Afin de répondre à ce besoin, DRMSim [DRM09], un simulateur à événements discrets de modèles de routage dynamiques a été développé. L'évaluation d'un modèle permet une simulation de plus haut niveau évitant les détails d'implémentation comme la simulation de protocoles de couches inférieures (TCP, UDP, Ethernet, etc).

Différentes classes de modèles de routage sont intégrées au simulateur autorisant leur comparaison au sein d'une même plate-forme de simulation. On pourra citer les modèles de protocole à vecteur de distances RIP [Mal98] et LFR [DDFM12], à vecteur de chemins BGP [RLH06], de routage compact [Gla13, NRS09] ou encore géographique [KD14]. Plusieurs expérimentations ont pu être menées sur des topologies composées de plus de 17 000 sommets dans le cas des schémas de routage compacts.

Afin de faciliter l'évaluation des performances de ces modèles de routage, DRMSim permet la mesure automatique de la taille des tables de routage, du nombre de messages de contrôle échangé et de l'étirement des routes calculées par le modèle de routage. Ces mesures peuvent être complétées par des métriques spécifiques au modèle simulé comme par exemple le nombre de messages de mise à jour BGP pour atteindre un état de convergence.

DRMSim permet la configuration de divers scénarios de simulation. Ces scénarios peuvent être dynamiques autorisant l'ajout ou la suppression de nœuds et de liens. L'utilisateur du simulateur peut définir ses propres scénarios, par exemple le calcul du nombre d'instances MRAI (Minimal Route Advertisement Interval) avant convergence de BGP. D'autre part, un modèle de transmission des messages entre les nœuds prenant en compte la bande passante des liens et le traitement des messages par ordre d'arrivée est disponible.

Enfin, DRMSim permet aussi l'évaluation statistique des performances d'un modèle de routage. L'exécution d'un même scénario peut être répétée plusieurs fois en variant la valeur d'initialisation du générateur de nombres aléatoires du simulateur. La moyenne des mesures de chacune des simulations étant ensuite automatiquement calculée.

Distribution des simulations de BGP

La simulation de modèles de routage sur des topologies de l'ordre de 17 000 sommets est encore insuffisante. En effet, le nombre d'AS dans l'Internet est actuellement supérieur à 48 000. De plus, en considérant l'évolution croissante du nombre d'AS année après année [Bat], des topologies de l'ordre de $O(100k)$ sommets doivent pouvoir être considérées afin d'anticiper les futurs besoins de l'architecture de routage d'Internet.

La limitation de DRMSim empêchant d'atteindre des topologies de plus grande taille est principalement due aux ressources mémoire requises pour stocker les tables de routage de BGP et la liste des messages de mise à jour dans la queue à événements discrets. Bien qu'il soit toujours possible d'optimiser les structures de données et l'exécution des procédures dans le simulateur, il semble inévitable de faire évoluer DRMSim vers d'autres paradigmes de simulation pour obtenir les performances voulues. La simulation discrète et distribuée semble être une approche prometteuse.

Cependant, lors de la distribution des simulations, la liste des événements discrets et leur exécution sont réparties entre différents *processus logiques* (ou LP, Logical Process). Un LP est une unité de calcul équipée de sa propre mémoire qui communique au travers d'un réseau avec d'autres LP. L'ordre d'exécution des événements discrets doit être respecté. Un mécanisme de synchronisation entre LP est donc nécessaire. Ce mécanisme peut être la source de nombreux échanges de messages entre les LP. Dans le cas de la simulation distribuée d'un modèle de routage, le nombre de ces messages dépend du modèle. La transmission de messages entre les LP par le réseau étant bien plus lente que dans un modèle de distribution à mémoire partagée, si le volume des communications est trop important, l'exécution de la simulation peut être significativement ralentie. La question est donc de connaître le temps de simulation supplémentaire induit par le modèle de distribution du simulateur.

Ainsi, avant de procéder au développement d'une nouvelle version distribuée de DRMSim, il est important d'évaluer la faisabilité de la distribution de ces simulations dans le pire cas. Parmi les modèles disponibles dans DRMSim, BGP est celui ayant la plus grande complexité en termes de nombre de messages de contrôle échangé pour converger. C'est pourquoi je propose une étude de faisabilité de la distribution des simulations de BGP dans DRMSim.

III Contributions et plan détaillé

Cette thèse est constituée de deux parties distinctes pouvant être lues séparément. La première partie, composée des chapitres 1 et 2, porte sur de nouvelles méthodes de calcul de l'hyperbolicité dans de grands graphes. La seconde partie, composée des chapitres 3, 4 et 5, présente la simulation de modèles de routage dynamiques dans de grands réseaux.

Chapitre 1 : Pré-traitement et décomposition de graphe pour le calcul de l'hyperbolicité

Dans ce chapitre, je présente une nouvelle méthode de pré-traitement du graphe afin de faciliter le calcul de l'hyperbolicité.

Nous savons que l'hyperbolicité d'un graphe est la plus grande valeur d'hyperbolicité parmi ses composantes biconnexes. Ce résultat repose sur le fait que les chemins entre les sommets de deux composantes biconnexes distinctes passent par un même sommet appelé *point d'articulation*. Ce résultat permet de réduire significativement le temps de calcul de l'hyperbolicité d'un graphe. Pour réduire encore le temps de calcul, nous cherchons à utiliser des séparateurs de plus grand diamètre (un point d'articulation étant un séparateur du graphe de diamètre nul) pour décomposer le graphe en composantes de tailles inférieures.

Dans [Tar85], les auteurs proposent un algorithme en $O(nm)$ pour le calcul des atomes d'une décomposition par les cliques-séparatrices du graphe, c'est-à-dire des séparateurs de diamètre $k = 1$. Ainsi, dans ce chapitre, je montre comment borner l'hyperbolicité d'un graphe à partir des atomes de sa décomposition par les cliques-séparatrices et comment les modifier afin d'obtenir la valeur exacte de l'hyperbolicité. Cette méthode ne fonctionne que si l'hyperbolicité du graphe est supérieure à 1. J'évalue ensuite cette méthodologie sur divers graphes de collaboration [LKF07].

Ces travaux ont été effectués en collaboration avec N. Cohen (LRI), D. Coudert (COATI, INRIA) et G. Ducoffe (COATI, INRIA) et ont été soumis à une revue [CCDL14].

Chapitre 2 : Algorithme exact pour le calcul de l'hyperbolicité dans de grands graphes

Dans ce chapitre, je présente un nouvel algorithme efficace en pratique pour le calcul de l'hyperbolicité et une heuristique pour approcher l'hyperbolicité dans les très grands graphes.

L'idée de cet algorithme est d'identifier au plus tôt les quadruplets de plus grande hyperbolicité et de couper l'exploration des quadruplets dont l'hyperbolicité est inférieure à celle déjà trouvée. Il est possible de combiner cet algorithme avec les méthodes de décomposition en composantes biconnexes et en atomes du graphe et par le calcul des paires éloignées.

Ainsi, malgré une complexité temporelle de ce nouvel algorithme en $O(n^4)$, l'hyperbolicité de graphes composés de plus de 58 000 sommets a pu être calculée en quelques heures, ce qu'aucune autre méthode ne permettait de faire. De plus, la complexité du calcul de l'hyperbolicité est significativement réduite dans les cycles et la grille $n \times m$. Une version parallèle de l'algorithme est aussi proposée.

Au final, les performances de ce nouvel algorithme et de sa version parallèle en combinaison avec la méthode de décomposition par les cliques-séparatrices présentée en chapitre 1 ont été évaluées sur les graphes CAIDA [CAI13b], DIMES [SS05] et divers réseaux de collaboration [CML11, LKF07].

Cet algorithme a besoin de garder en mémoire la matrice des distances dans le graphe. Il a donc une complexité en mémoire en $O(n^2)$. Une telle complexité empêche le calcul de l'hyperbolicité dans les très grands graphes. Une heuristique est donc proposée pour approcher l'hyperbolicité. Celle-ci montre de bons résultats dans des graphes composés de plusieurs millions de sommets.

Ces travaux ont été effectués en collaboration avec N. Cohen (LRI) et D. Coudert. Ils ont fait l'objet d'une publication dans [CCL13] et ont été soumis à une revue [CCL12].

Chapitre 3 : Introduction au routage dans l'Internet

Ce chapitre a pour objectif de présenter les notions et la terminologie nécessaires à la compréhension du chapitre 4 et du chapitre 5.

Pour cela, je rappelle le principe du protocole IP et je présente l'architecture de routage dans l'Internet. Plus particulièrement, je détaille les différentes relations économiques entre les systèmes autonomes qui sont à l'origine du besoin des AS de définir des politiques de routage.

J'expose la problématique du routage dans l'Internet et plus particulièrement les limites de BGP. En effet, ces dernières années, le nombre d'AS dans Internet a rapidement augmenté. Le nombre d'entrées dans les tables de routage de BGP a plus que triplé ces dix dernières années. La question du passage à l'échelle de BGP est à l'origine de nombreux travaux pour la conception de nouveaux paradigmes de routage dédiés à Internet. Les performances de ces nouveaux paradigmes de routage doivent être ensuite évaluées.

Ces besoins ont motivé le développement du simulateur DRMSim.

Chapitre 4 : Méthodes de simulation de modèles de routage dans de grands réseaux dynamiques

Dans ce chapitre, je présente le simulateur de modèles de routage dynamiques DRMSim ainsi que les expérimentations menées pour l'évaluation de différents modèles de routage.

Un premier travail dans ce simulateur a été le développement d'une architecture modulaire basée sur le principe de *composants*. Un composant est un bloc logique modifiable indépendamment des autres composants du logiciel. Ainsi, il est possible de facilement ajouter ou modifier une partie du simulateur et donc de l'adapter aux besoins des utilisateurs. Chacun des éléments essentiels de cette architecture est présenté dans ce chapitre.

Un objectif de DRMSim étant la possibilité de comparer les performances de nouveaux schémas de routage à BGP, je présente son intégration au sein du simulateur et les choix d'implémentation faits afin de réduire son empreinte mémoire. De plus, un certain nombre d'études ont été menées avec le simulateur, celles-ci incluent la simulation de plusieurs schémas de routage compacts [NRS12, Gla13].

DRMSim est le fruit d'un travail collaboratif avec L. Hogie, D. Papadimitriou et I. Tahiri et a fait l'objet d'un article dans le magazine ERCIM News [LP13].

Chapitre 5 : Étude de faisabilité de la distribution des simulations de BGP

Dans ce chapitre, je présente les raisons motivant le développement d'un nouveau simulateur distribué et sa faisabilité dans le cadre des simulations de BGP.

Plusieurs contraintes liées à la distribution des simulations et plus particulièrement aux coûts des communications entre les processus logiques motivent une étude préliminaire au développement d'un nouveau simulateur distribué. L'approche conservatrice du protocole de synchronisation et comment éviter une paralysie du système sont un exemple de ces contraintes.

Dans ce chapitre, j'étudie deux modèles connus de communication entre les LP basés sur une approche conservatrice de la distribution de la liste des événements discrets. Ces deux modèles ont un impact en mémoire et sur les communications réseau différent qui dépend du modèle de routage simulé. Le modèle de routage choisi est BGP puisqu'il représente un pire cas dans la simulation de modèles de routage. Ces deux modèles de communication sont évalué en fonction de trois scénarios de simulation. Chacun de ces scénarios ayant un impact différent sur le nombre de messages de contrôle échangé entre les nœuds.

Pour chacun des deux modèles de communication, le nombre de messages échangé entre les parties est mesuré par une simulation séquentielle de BGP. Ce nombre est ensuite utilisé comme paramètre de deux programmes mixtes en nombres d'entiers (un par modèle de communication) pour le calcul d'une bipartition des topologies. Ces topologies sont composées de 1 000 à 5 000 nœuds. Ces bipartitions sont optimales et permettent le calcul d'une borne inférieure sur le volume des communications réseau entre les LP. L'estimation de ce volume permet ensuite de déduire le temps supplémentaire de calcul de la simulation distribuée de BGP dans le cas d'une bipartition de la topologie.

Ces travaux ont été effectués en collaboration avec D. Coudert (COATI, INRIA), L. Hogie (CNRS), S. Perennes (CNRS), I. Tahiri (IMB) et D. Papadimitriou (Alcatel-Lucent Bell) et ont fait l'objet d'une publication dans [CHL+12].

IV Conclusion

Cette thèse a été effectuée et financée dans le cadre du projet européen FP7 STREP EULER. L'objectif de ce projet est l'étude de nouveaux paradigmes de routage pour la conception, le développement et la validation expérimentale de schémas de routage distribués et dynamiques adaptés à Internet et à son évolution dans le futur. Ces nouveaux schémas de routage doivent répondre aux limites du protocole actuel BGP. Ce projet a donc pour vocation d'étudier les équilibres possibles entre la taille des tables de routage (qui assure le passage à l'échelle), l'étirement des routes calculées (qui assure la qualité du

routage) et le coût des communications (qui assure l'efficacité du routage et une réponse rapide en cas de mise hors service d'un nœud ou d'un lien).

L'idée principale du projet est d'utiliser les propriétés topologiques et statistiques du graphe d'Internet et les propriétés de stabilité et de convergence des politiques de routage mise en place par les opérateurs. Ces propriétés peuvent ensuite être utilisées afin de spécialiser la conception d'un schéma de routage efficace.

De nouveaux modèles et outils pour une analyse fiable et complète de la topologie d'Internet doivent être développés. Ces modèles doivent faciliter l'élaboration de scénarios réalistes pour l'expérimentation de nouveaux schémas de routage sur des topologies de l'ordre de $O(10k)$ sommets. De ces schémas de routage doivent ensuite être dérivés des prototypes de protocoles de routage qui seront validés par expérimentation à partir des installations iLAB et PlanetLab/OneLab.

Ce projet regroupe sept partenaires universitaires et industriels :

— Alcatel-Lucent Bell, Bell Labs, Belgique ;

— INRIA, équipes-projets CEPAGE (Bordeaux - Sud-Ouest), GANG (Paris - Rocquencourt) et COATI (Sophia Antipolis - Méditerranée) ;

— Université de Ghent, Department of Information Technology (INTEC) et iMinds ;

— Université Pierre Marie Curie (UPMC), Laboratoire d'Informatique de Paris 6 (LIP6) ;

— Université Catholique de Louvain, Department of Mathematical Engineering (INMA) ;

— Université de Patras, Computer Technology Institute and Press "Diophantus" ;

— Université Polytechnique de Catalogne (UPC) et université de Gérone (UdG), Advanced Broadband Communications Center, Broadband Communications and Distributed Systems.

Dans ce projet, j'ai plus particulièrement participé à l'écriture des rapports techniques suivants et aux travaux qu'ils contiennent :

— D3.1 [par11], «Graph-based topology modelling», 2011 ;

— D4.1 [DWP+11], «Performance objectives, evaluation criteria and metrics», 2011 ;

— D4.2 [DCL+12], «Experimental methodology, scenarios, and tools», 2012 ;

— D3.3 [IPD+12], «Graph analysis / mining», 2012.

J'ai aussi organisé ou aidé à l'organisation de différents ateliers dans le cadre du projet EULER :

— présentation de DRMSim lors des différentes réunions techniques du projet EULER ;

— présentation de DRMSim à «Future Internet Assembly», Aalborg, du 9 au 11 mai 2012 ;

— démonstration de DRMSim lors de la 1ère évaluation du projet EULER à Bruxelles, le 19 octobre 2012 ;

— organisation d'une session d'introduction et d'apprentissage à DRMSim, INRIA Sophia-Antipolis – Méditerranée, du 4 au 6 mars 2013 ;

— participation à l'organisation de l'atelier EULER et présentation de DRMSim à "Future Internet Assembly", Dublin, du 7 au 10 mai 2013.

V Publications

[CCDL14] Nathann Cohen, David Coudert, Guillaume Ducoffe et Aurélien Lancin : Applying clique-decomposition for computing Gromov hyperbolicity. Rapport de recherche RR-8535, INRIA, juin 2014. Soumis à une revue.

[CCL12] Nathann Cohen, David Coudert et Aurélien Lancin : Exact and approximate algorithms for computing the hyperbolicity of large-scale graphs. Rapport de recherche RR-8074, INRIA, septembre 2012. Soumis à une revue.

[CCL13] N. Cohen, D. Coudert et A. Lancin : Algorithme exact et approché pour le calcul de l'hyperbolicité d'un graphe. *In* N.Nisse et F.Rousseau et Y.Busnel, éditeur : *15èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (AlgoTel)*, pages 1–4, Pornic, France, mai 2013. Page 1-4.

[CHL+12] David Coudert, Luc Hogie, Aurélien Lancin, Dimitri Papadimitriou, Stéphane Pérennes et Issam Tahiri : Feasibility study on distributed simulations of BGP. *In PADS - 26th ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation - 2012*, Zhangjiajie, Chine, avril 2012. IEEE.

[LP13] Aurélien Lancin et Dimitri Papadimitriou : DRMSim : A Routing-Model Simulator for Large-Scale Networks. *ERCIM News*, 2013(94), 2013.

Première partie

Méthodes de calcul de l'hyperbolicité selon Gromov dans de grands graphes

Chapitre 1

Pré-traitement et décomposition de graphe pour le calcul de l'hyperbolicité

Sommaire

1 Définitions de la δ-hyperbolicité 4
2 Méthodes de pré-traitement du graphe 9
2.1 Utilisation des paires éloignées pour la réduction du nombre de quadruplets à visiter 9
2.2 Décomposition en composantes biconnexes 10
3 Méthode de décomposition du graphe par les cliques-séparatrices 12
3.1 Relations entre l'hyperbolicité du graphe et ses séparateurs 12
3.2 Hyperbolicité et cliques-séparatrices 13
3.3 Hyperbolicité et décomposition par les cliques-séparatrices 17
3.4 Approximation de l'hyperbolicité 20
4 Méthode de substitution pour le calcul exact de l'hyperbolicité . 20
4.1 Méthode de substitution des sommets à partir d'une $(A|B)$-clique-séparatrice 20
4.2 Calcul de la valeur exacte de l'hyperbolicité 21
5 Résultats expérimentaux 22
5.1 Données d'expérimentation 22
5.2 Résultats empiriques 23
5.3 Analyse de la décomposition par les cliques-séparatrices et de la construction des substituts 25
6 Conclusion 27

Dans ce chapitre, je concentre mon étude sur de nouveaux outils pour faciliter le calcul de l'hyperbolicité dans les grands graphes. Nous verrons qu'il existe plusieurs définitions équivalentes de l'hyperbolicité d'un espace métrique à un facteur multiplicatif près. Je présenterai celles du produit de Gromov, de la condition des 4 points, des triangles δ-fins et δ-minces.

Nous utiliserons la définition de la condition des 4 points pour le calcul de l'hyperbolicité dans les graphes. En effet, un algorithme simple avec une complexité temporelle en $O(n^4)$ peut facilement en être déduit.

Dans [FIV12], les auteurs ont proposé un algorithme en $O(n^{3,69})$ pour le calcul de l'hyperbolicité à partir de la définition de l'hyperbolicité par le produit de Gromov. Cette complexité est la meilleure connue à ce jour dans la littérature, cependant aucune implémentation de cet algorithme n'est disponible et son implémentation, comme de nombreux algorithmes basés sur les produits de matrices, est difficile. Ses performances en pratique ne peuvent donc pas être évaluées.

Afin de réduire la quantité de calcul à effectuer pour déterminer l'hyperbolicité d'un graphe, je propose d'utiliser une méthode de décomposition de graphes par les cliques-séparatrices. Le calcul d'une telle décomposition permet de diviser le graphe en sous-graphes de plus petite taille. Un algorithme pour le calcul de l'hyperbolicité peut ensuite être appliqué sur chacun de ces graphes. Dans ce chapitre, je présente trois contributions :

1. je montre comment il est possible de borner l'hyperbolicité d'un graphe à partir des atomes de sa décomposition par les cliques-séparatrices (section 3) ;
2. je montre comment modifier les atomes de la décomposition afin d'obtenir la valeur exacte de l'hyperbolicité du graphe si celle-ci est supérieure à 1 (section 4) ;
3. je présente une évaluation expérimentale de cette nouvelle méthodologie (section 5).

Le code de l'algorithme que j'ai développé en language Java pour le calcul de la décomposition par les cliques-séparatrices du graphe et de la modification des atomes est libre et disponible [Lana].

Je commence en section 1 par présenter quelques définitions équivalentes de l'hyperbolicité utilisées dans mes travaux et quelques résultats connus sur l'hyperbolicité. En section 2, je détaille différentes méthodes de pré-traitement du graphe afin de réduire la taille des instances pour le calcul de l'hyperbolicité. En section 3, je montre comment approcher la valeur de l'hyperbolicité à partir d'une décomposition par les cliques-séparatrices du graphe. Puis en section 4, j'étends cette nouvelle méthode pour le calcul de la valeur exacte de l'hyperbolicité. Finalement, en section 5, j'évalue ses performances et propose une analyse comportementale de l'algorithme sur divers graphes de collaboration.

1 Définitions de la δ-hyperbolicité

Rappelons tout d'abord quelques notations et définitions importantes.

Définition 1 (Espace métrique)**.** Soit X un ensemble. Une distance $\mathtt{d}$ sur l'ensemble X est une application sur $X \times X$ à valeurs dans $\mathbb{R}^+$ et qui vérifie pour tout point a, b, c dans X :

$$\begin{aligned}
&\mathtt{d}(a,b) = 0 \text{ si et seulement si } a = b,\\
&\mathtt{d}(a,b) = \mathtt{d}(b,a) \text{ (symétrie)},\\
&\mathtt{d}(a,b) \leq \mathtt{d}(a,c) + \mathtt{d}(b,c) \text{ (inégalité triangulaire)}.
\end{aligned}$$

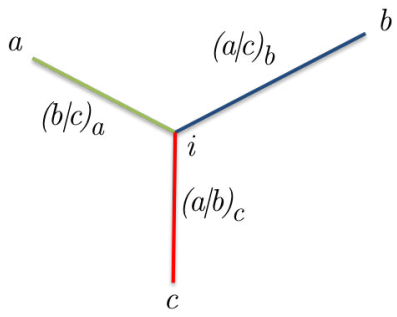
La paire $(X, \mathtt{d})$, avec X un espace muni d'une distance $\mathtt{d}$, est appelée *espace métrique*.

Une première définition de la δ-hyperbolicité d'un espace métrique se base sur la définition du *produit de Gromov* :

Définition 2 (Produit de Gromov [Gro87])**.** Soit un espace métrique $(X, \mathtt{d})$ et soit $c \in X$ appelé point de base, alors pour tout $a, b \in X$ le produit de Gromov de a et b par rapport à c est définit par :

$$(a|b)_c = \frac{1}{2}(\mathtt{d}(a,c) + \mathtt{d}(b,c) - \mathtt{d}(a,b)).$$

Admettons maintenant que $(X, \mathtt{d})$ est un espace métrique géodésique représentant un arbre, un exemple est donné par le tripode (a, b, c) en figure 1.1. Notons $[a, b]$ le segment de longueur minimum, aussi appelé *géodésique*, reliant les points a et b. On notera i l'intersection des trois segments de l'arbre. Alors $(a|b)_c$ est la distance du point c au segment $[a, b]$. De façon similaire, $(a|c)_b$ sera la distance du point b au segment $[a, c]$ et $(b|c)_a$ la distance du point a au segment $[b, c]$.



$$(a|b)_c = \frac{1}{2}(\mathbb{d}(a,c) + \mathbb{d}(b,c) - \mathbb{d}(a,b)),$$
$$(a|c)_b = \frac{1}{2}(\mathbb{d}(a,b) + \mathbb{d}(b,c) - \mathbb{d}(a,c)),$$
$$(b|c)_a = \frac{1}{2}(\mathbb{d}(a,b) + \mathbb{d}(a,c) - \mathbb{d}(b,c)).$$

FIGURE 1.1 – Représentation géométrique du produit de Gromov.

Définition 3 (δ-hyperbolicité [Gro87])**.** Soit $\delta \geq 0$, un espace métrique $(X, \mathbb{d})$ est δ-hyperbolique si pour tout $a, b, c, d \in X$:

$$(a|b)_c \geq \min\{(a|d)_c, (b|d)_c\} - \delta.$$

À partir de la définition 3, on cherchera à trouver la plus petite valeur de δ telle que l'espace métrique $(X, \mathbb{d})$ soit δ-hyperbolique.

Dans les espaces métriques *géodésiques*, c'est-à-dire ou pour tous points $x, y \in X$, x et y sont reliés par un plus court chemin de longueur $\mathbb{d}(x, y)$, on peut définir l'hyperbolicité de deux autres façons qui utilisent la notion de triangle géodésique. Ces deux définitions et celle du produit de Gromov sont équivalentes à un facteur multiplicatif près.

Définition 4 (Triangle δ-fin)**.** Soit $\delta > 0$. Considérons un triangle dans un espace métrique géodésique. On définira le δ-voisinage d'un côté du triangle comme l'ensemble des points qui en sont éloignés à distance au plus δ. Le triangle est dit *δ-fin* si chacun de ses côtés est contenu dans le δ-voisinage de l'union des deux autres côtés du triangle.

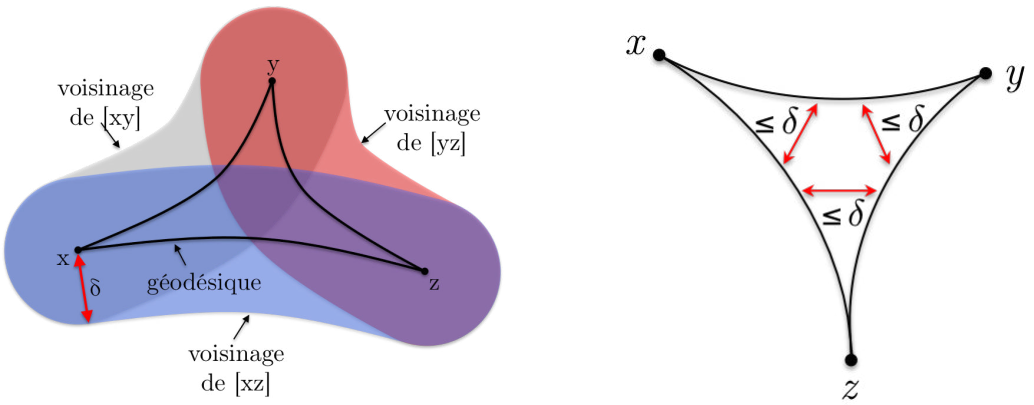
Définition 5 (Triangle δ-mince)**.** Soit $\delta > 0$. Un triangle dans un espace métrique géodésique est dit *δ-mince* si la distance en tout point d'un des côtés du triangle est d'au plus δ d'un des deux autres côtés du triangle.

La définition des triangles δ-fins est illustrée en figure 1.2a. Dans cette figure, chacune des géodésiques [xz], [xy] et [yz] est contenue dans le δ-voisinage des deux autres géodésiques. La définition des triangles δ-mince est illustrée en figure 1.2b. Dans cette figure, chacune des géodésiques est à distance au plus δ des deux autres.

Enfin, il est possible d'utiliser une quatrième définition de l'hyperbolicité proposée par Gromov appelée *condition des 4 points*, équivalente à la définition 3 (p. 5). Cette définition sera utilisée dans le chapitre 2 pour la conception d'un nouvel algorithme pour le calcul de la valeur exacte de l'hyperbolicité d'un graphe.

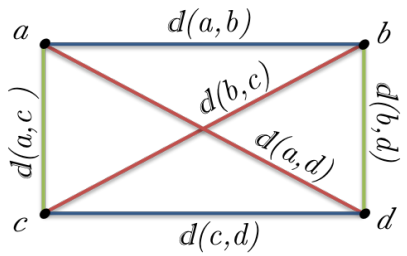
Définition 6 (Condition des 4 points [Gro87])**.** Un espace métrique $(X, \mathbb{d})$ est dit δ-hyperbolique avec $\delta \geq 0$ si pour tout quadruplet a, b, c, $d \in X$, les deux plus grandes des sommes $S_1 = \mathbb{d}(a, b) + \mathbb{d}(c, d)$, $S_2 = \mathbb{d}(a, c) + \mathbb{d}(b, d)$, et $S_3 = \mathbb{d}(a, d) + \mathbb{d}(b, c)$ diffèrent d'au plus 2δ. L'hyperbolicité de X désigne la valeur du plus petit δ vérifiant cette définition.

La figure 1.3 illustre la définition de la condition des 4 points à partir des six distances existantes dans le quadrilatère formé par les quatre points a, b, c et d. Les sommes S_1, S_2 et S_3 correspondent aux sommes des distances des côtés opposés dans le quadrilatère. L'intérêt de cette définition est montré en figure 1.4. Les deux points c et d évoluent chacun à la même vitesse de a vers b sur deux géodésiques différentes. La plus grande distance $\mathbb{d}(c, d)$ sera alors au plus de 2δ. Cette notion borne la valeur de la plus grande distance entre deux



(a) Triangle δ-fin. (b) Triangle δ-mince.

FIGURE 1.2 – Définitions des triangles δ-fin et δ-mince.



Posons :

$$S_1 = \mathrm{d}(a,b) + \mathrm{d}(c,d),$$
$$S_2 = \mathrm{d}(a,c) + \mathrm{d}(b,d),$$
$$S_3 = \mathrm{d}(a,d) + \mathrm{d}(b,c).$$

Si $S_1 \geq S_2 \geq S_3$, alors $\delta \leq \frac{S_1 - S_2}{2}$.

FIGURE 1.3 – La condition des 4 points.

géodésiques de l'espace métrique. Remarquons que si $\delta = 0$ alors les deux géodésiques se confondent, une seule géodésique relie les points a et b.

Afin de montrer l'équivalence entre la définition 3 basée sur le produit de Gromov et la définition 6 de la condition des 4 points, reformulons tout d'abord cette dernière.

Proposition 1. *Un espace métrique (X, d) est dit δ-hyperbolique avec $\delta \geq 0$ si pour tout quadruplet a, b, c, $d \in X$:*

$$\mathrm{d}(c,d) + \mathrm{d}(a,b) \leq \max\{\mathrm{d}(a,d) + \mathrm{d}(b,c), \mathrm{d}(b,d) + \mathrm{d}(a,c)\} + 2\delta.$$

Démonstration. Montrons l'équivalence entre la condition des 4 points donnée en définition 6 et la définition 3 basée sur le produit de Gromov :

$$\begin{aligned}
\mathrm{d}(c,d) + \mathrm{d}(a,b) &\leq \max\{(\mathrm{d}(a,d) + \mathrm{d}(b,c), \mathrm{d}(b,d) + \mathrm{d}(a,c)\} + 2\delta \\
\mathrm{d}(c,d) + \mathrm{d}(a,b) - (\mathrm{d}(b,c) + \mathrm{d}(a,c)) &\leq \max\{\mathrm{d}(a,d) - \mathrm{d}(a,c), \mathrm{d}(b,d) - \mathrm{d}(b,c)\} + 2\delta \\
\mathrm{d}(b,c) + \mathrm{d}(a,c) - \mathrm{d}(a,b) &\geq \mathrm{d}(c,d) + \min\{\mathrm{d}(a,c) - \mathrm{d}(a,d), \\
&\qquad \mathrm{d}(b,c) - \mathrm{d}(b,d)\} - 2\delta \\
\mathrm{d}(b,c) + \mathrm{d}(a,c) - \mathrm{d}(a,b) &\geq \min\{\mathrm{d}(a,c) + \mathrm{d}(c,d) - \mathrm{d}(a,d), \\
&\qquad \mathrm{d}(b,c) + \mathrm{d}(a,c) - \mathrm{d}(b,d)\} - 2\delta \\
(a|b)_c &\geq \min\{(a|d)_c, (b|d)_c\} - \delta
\end{aligned}$$

□

Pour montrer la réciproque de cette démonstration, et donc l'équivalence de la définition 3 vers la définition 6, il suffit de reprendre les étapes inverses de la démonstration (voir [SG11]).

La définition 6, ou la condition des quatre points, peut facilement être transposée au cas d'un graphe connexe $G(V, E)$ avec $|V| = n$ le nombre de sommets, $|E| = m$ le nombre d'arêtes et (X, d) son espace métrique équipé de la métrique d des plus courts chemins dans G. On dira qu'un graphe G est δ-hyperbolique si son espace métrique $(V(G), \mathrm{d}_G)$ est δ-hyperbolique. Nous utiliserons donc cette définition dans la suite de ce chapitre. Nous noterons aussi $\mathrm{d}(a, b)$ la distance des plus courts chemins entre les sommets a et b de G et $N(u)$ l'ensemble des voisins du sommet $u \in V$. La différence entre les deux plus grandes des trois sommes S_1, S_2, et S_3 d'un quadruplet (a, b, c, d) sera notée $\delta(a, b, c, d)$.

Un premier algorithme pour le calcul de l'hyperbolicité d'un graphe G peut facilement être dérivé de la définition 6. En effet, il suffit de calculer la valeur d'hyperbolicité de chacun des quadruplets (a, b, c, d) du graphe. Nous noterons cette valeur $\delta(a, b, c, d)/2$. L'hyperbolicité de G est donnée par $\delta = \max_{a,b,c,d \in V} \delta(a, b, c, d)/2$, c'est-à-dire la plus grande valeur de δ calculée parmi les quadruplets du graphe. Cependant, cet algorithme a une complexité temporelle en $O(n^4)$, ce qui est bien trop important pour le calcul de l'hyperbolicité de graphes composés de plusieurs milliers de sommets.

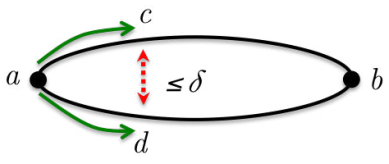
Un second algorithme pour le calcul de l'hyperbolicité d'un espace métrique proposé dans [FIV12] utilise la définition du produit de Gromov. Il se base sur l'algorithme (max, min) de multiplication des matrices proposé dans [DP09]. La complexité temporelle du calcul de l'hyperbolicité a pu être ainsi ramenée en $O(n^{3.69})$ mais aucune implémentation ou évaluation expérimentale de l'algorithme semble disponible.

Présentons maintenant quelques bornes connues de l'hyperbolicité d'un graphe. Notons $D(G)$, le diamètre d'un graphe G. Le lemme 1 nous permet de montrer que $\delta \leq D(G)/2$.

Lemme 1. *Soient $G = (V, E)$ un graphe connexe, $a, b, c, d \in V$, $S_1 = \mathrm{d}(a, b) + \mathrm{d}(c, d)$, $S_2 = \mathrm{d}(a, c) + \mathrm{d}(b, d)$, et $S_3 = \mathrm{d}(a, d) + \mathrm{d}(b, c)$. Si $S_1 \geq \max\{S_2,\ S_3\}$, alors $\delta(a, b, c, d) \leq \min(\mathrm{d}(a, b),\ \mathrm{d}(c, d))/2$.*

Démonstration. $S_2 + S_3 = (\mathrm{d}(a, c) + \mathrm{d}(b, d)) + (\mathrm{d}(a, d) + \mathrm{d}(b, c)) = (\mathrm{d}(a, c) + \mathrm{d}(b, c)) + (\mathrm{d}(a, d) + \mathrm{d}(b, d))$. L'inégalité triangulaire permet de déduire $S_2 + S_3 \geq 2 \cdot \mathrm{d}(a, b)$. Comme $S_1 \geq \max\{S_2,\ S_3\}$, nous avons $2 \cdot \delta(a, b, c, d) = S_1 - \max\{S_2,\ S_3\} \leq S_1 - (S_2 + S_3)/2 \leq S_1 - \mathrm{d}(a, b) = \mathrm{d}(c, d)$. De la même manière $2 \cdot \delta(a, b, c, d) \leq \mathrm{d}(a, b)$. □

Le lemme 1 sera particulièrement utile pour l'élaboration de nouveaux algorithmes pour le calcul de la valeur exacte de l'hyperbolicité. Cette borne supérieure sur l'hyperbolicité servira notamment en section 4.2 et lors des expérimentations du chapitre 2.



Si c et d avancent à la même vitesse, alors :

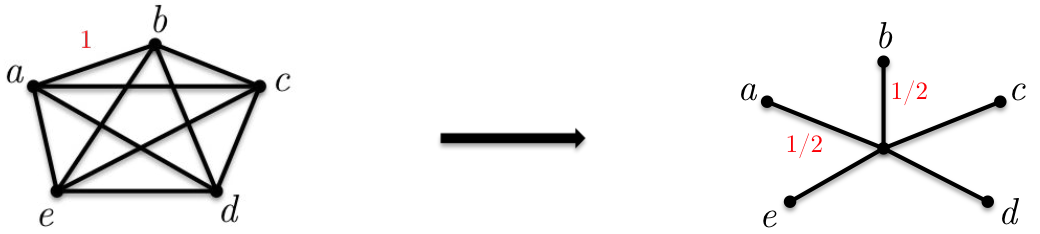
$$\mathrm{d}(a, b) = S_2 = S_3,$$
$$\mathrm{d}(a, c) = \mathrm{d}(a, d),$$
$$\mathrm{d}(b, c) = \mathrm{d}(b, d).$$

Par la condition des 4 points :

$$S_1 - S_2 = \mathrm{d}(c, d) \leq 2\delta.$$

FIGURE 1.4 – Exemple de la définition de la condition des 4 points.

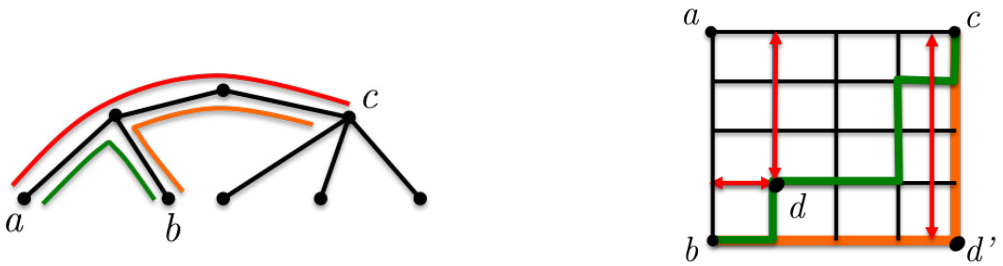
Suivant la condition des 4 points donnée en définition 6 (p. 5), l'arbre et la clique sont 0-hyperboliques (ce qui montre l'unicité des plus courts chemins comme vu dans l'exemple de la figure 1.4). En figure 1.5, on illustre la clique dont l'espace métrique peut être plongé dans un arbre tout en conservant les distances entre les sommets. La figure 1.6a montre l'unicité des chemins dans l'arbre. Que ce soit entre les paires de points (a, b), (a, c) ou (b, c), il n'existe qu'un seul et unique plus court chemin entre chacune des paires de sommets. La grille $n \times m$ avec $2 \leq n \leq m$ est $(n-1)$-hyperbolique. Cette large valeur d'hyperbolicité reflète la dispersion des plus courts chemins dans la grille par opposition à l'arbre et la clique. En effet, en considérant la grille 5×5 en figure 1.6b, le point d est à distance 1 du segment $[a, b]$. Si on cherche à élargir au maximum la distance entre le point d et les points a, b et c, alors la position la plus éloignée est celle de d' avec $\delta = n - 1 = 4$. Les cycles d'ordre $n = 4p + \varepsilon$, avec $p \geq 1$ et $\varepsilon \in \{0, 1, 2, 3\}$, sont $(p - 1/2)$-hyperboliques quand $\varepsilon = 1$, sinon p-hyperboliques [KM02]. Les graphes k-cordaux avec $k \geq 4$ sont au moins $k/4$-hyperboliques [WZ11]. Les graphes dont l'hyperbolicité est au plus un, ce qui inclut les graphes cordaux, ont été totalement caractérisés [CD14, BC03, KM02, BKM01]. Enfin, les graphes dont les composantes biconnexes sont des cliques, aussi appelés *block graphs* [BM86, How79], sont 0-hyperboliques.



(a) $K5$.

(c) Propriété métrique de la clique.

FIGURE 1.5 – La clique et son espace métrique avec $\delta = 0$.



(a) Arbre avec $\delta = 0$.

(b) Grille $n \times m$ avec $\delta = n - 1$.

FIGURE 1.6 – L'arbre et l'unicité des plus courts chemins en opposition à la grille et la dispersion des plus courts chemins.

2 Méthodes de pré-traitement du graphe

Afin de réduire la complexité temporelle des algorithmes pour le calcul de l'hyperbolicité, il est possible d'utiliser différentes méthodes visant à diminuer le nombre de quadruplets à visiter. Trois méthodes sont présentées dans ce chapitre :

1. une première méthode, présentée en section 2.1 et proposée dans [SG11, Nog09] consiste à restreindre le calcul à partir des paires éloignées dans le graphe ;
2. une seconde méthode, présentée en section 2.2, simple et connue dans la littérature, se base sur la décomposition en composantes biconnexes d'un graphe ;
3. enfin, je propose en section 3, une troisième et nouvelle méthode basée sur la décomposition du graphe en atomes.

La proposition de cette troisième méthode est la première contribution de ce chapitre. La combinaison de ces trois méthodes va nous permettre de grandement réduire les temps de calcul comme nous le verrons dans le chapitre 2.

2.1 Utilisation des paires éloignées pour la réduction du nombre de quadruplets à visiter

La notion de paires éloignées a déjà été proposée dans [SG11, Nog09] pour réduire le nombre de quadruplets à visiter pour le calcul de l'hyperbolicité. En diminuant ce nombre il est possible de réduire la complexité du calcul de l'hyperbolicité. En effet, dans le chapitre 2, je propose un nouvel algorithme pour son calcul dont la complexité temporelle dépend du nombre de quadruplets à visiter.

Définition 7 (Paire éloignée)**.** Soit $G = (V, E)$, une paire (u, v) est éloignée si pour tout $w \in V \setminus \{u, v\}$, $\mathtt{d}(w, u) + \mathtt{d}(u, v) > \mathtt{d}(w, v)$ et $\mathtt{d}(w, v) + \mathtt{d}(u, v) > \mathtt{d}(w, u)$.

La figure 1.7 illustre la définition des paires distantes dans une grille. Les sommets u, v, x, y aux quatre coins de la grille forment deux paires distantes (u, v) et (x, y) car tout autre sommet w de la grille (dans la zone rouge de la figure) est situé sur un des plus courts chemins entre ces deux paires.

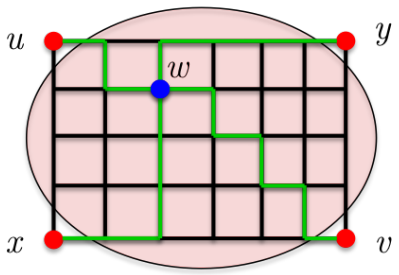
L'ensemble des paires éloignées peut être déterminé en temps $O(nm)$ dans un graphe non pondéré en utilisant un algorithme de recherche en profondeur (BFS). Il suffit d'initialiser un ensemble F avec l'ensemble de toutes les paires du graphe. Ensuite, lors de l'exécution du BFS, une paire (s, u) est supprimée de F si un voisin v de u est à distance $\mathtt{d}(s, u) + 1$ de s. À la fin de l'exécution de l'ensemble des BFS (un BFS à partir de chaque sommet $s \in V$), l'ensemble F ne contient plus que des paires éloignées. L'intérêt de l'utilisation des paires éloignées tient du lemme suivant :

Lemme 2. *Soit G un graphe connexe. Il existe deux paires éloignées (u, v) et (x, y) qui satisfont $\delta(u, v, x, y) = \delta(G)$.*

Démonstration. Soit u, v, x, y un quadruplet de G avec $S_1 = \mathtt{d}(u, v) + \mathtt{d}(x, y)$, $S_2 = \mathtt{d}(u, x) + \mathtt{d}(v, y)$ et $S_3 = \mathtt{d}(u, y) + \mathtt{d}(v, x)$. Posons $S_1 \geq S_2 \geq S_3$ tel que $\delta(G) = \delta(u, v, x, y)/2$, maximisant la valeur de S_1 dans G pour obtenir δ.

Considérons u, v une paire éloignée et x, y une paire qui n'est pas éloignée. On peut ainsi supposer sans perte de généralité qu'il existe un sommet $z \in N(x)$ qui satisfait $\mathtt{d}(z, y) = 1 + \mathtt{d}(x, y)$.

Dans un tel cas, écrivons $S'_1 = \mathtt{d}(u, v) + \mathtt{d}(z, y)$, $S'_2 = \mathtt{d}(u, z) + \mathtt{d}(v, y)$ et $S'_3 = \mathtt{d}(u, y) + \mathtt{d}(v, x)$. Par hypothèse $S_1 \geq S_2 \geq S_3$, il en résulte $S'_1 = S_1 + 1 = \max\{S'_1, S'_2, S'_3\}$. De plus, nous avons $\max\{S'_2, S'_3\} \leq \max\{S_2, S_3\} + 1$. Ainsi $\delta(u, v, x, z)/2 \geq \delta(u, v, x, y)/2$ et $S'_1 > S_1$, contredisant la maximalité de S_1. □



$$\delta(G) = \delta(u, v, x, y).$$

FIGURE 1.7 – Les deux seules paires distantes (u, v) et (x, y) dans une grille. Tous les autres sommets sont inclus dans les plus courts chemins entre les paires (u, v) et (x, y).

Considérons l'algorithme de base donné en section 1 parcourant l'ensemble des quadruplets du graphe pour le calcul de l'hyperbolicité. D'après le lemme 2, il est seulement nécessaire d'évaluer les quadruplets composés uniquement de paires éloignées. Prenons l'exemple de la grille $n \times m$ en figure 1.7. Seules les paires u, v et x, y sont éloignées. L'hyperbolicité de la grille peut donc être déterminée à partir du seul quadruplet (u, v, x, y), résultant en une complexité temporelle en $O(1)$ si les distances sont connues. Nous verrons dans le chapitre 2, à partir du nouvel algorithme proposé et du principe de paires éloignées, qu'il est possible de grandement diminuer la complexité temporelle du calcul de l'hyperbolicité pour quelques classes de graphe.

2.2 Décomposition en composantes biconnexes

Le calcul de la décomposition en composantes biconnexes d'un graphe G permet de le séparer en composantes de plus petites tailles. Une fois la décomposition calculée, un algorithme pour le calcul de l'hyperbolicité peut être appliqué indépendamment sur chacune des composantes biconnexes de G. La plus grande valeur d'hyperbolicité trouvée est l'hyperbolicité du graphe.

De façon générale, on observe que si deux composantes connexes d'un graphe G sont séparées par un *point d'articulation*, alors l'hyperbolicité des quadruplets formés par des sommets des deux composantes connexes est inférieure ou égale à celle de G. Ces quadruplets n'ont pas besoin d'être évalués, seuls ceux composés des sommets d'une même composante doivent être calculés.

Définition 8 (Point d'articulation)**.** Soit G un graphe connexe. Un sommet x est appelé *point d'articulation* si il existe deux sommets $a, b \neq x$ dans G tels que tout plus court chemin entre a et b passe par x.

Théorème 3. *Soit $C_1, \ldots, C_l$ les composantes connexes d'un graphe G séparées par des points d'articulation, alors :*

$$\delta(G) = \max_i \delta(C_i).$$

Démonstration. Supposons l'existence dans un graphe G d'un point d'articulation x séparant deux composantes connexes A et B. Seules trois formes de quadruplets de sommets sont possibles. Nous les noterons :

1. (a_1, a_2, a_3, a_4) tel que $a_1, a_2, a_3, a_4 \in A$;
2. $(a|b_1, b_2, b_3)$ tel que $a \in A$ et $b_1, b_2, b_3 \in B$ (voir figure 1.8a) ;
3. $(a_1, a_2|b_1, b_2)$ tel que $a_1, a_2 \in A$ et $b_1, b_2 \in B$ (voir figure 1.8b).

La première forme de quadruplet (a_1, a_2, a_3, a_4) ne nous intéresse pas puisqu'elle est incluse dans le calcul de l'hyperbolicité de C_i. Étudions les deux cas de figure restants :

1. Prenons le quadruplet tel que $(a|b_1, b_2, b_3)$, nous avons :

$$\begin{aligned} S_1 &= \mathtt{d}(a, b_1) + \mathtt{d}(b_2, b_3) = \mathtt{d}(a, b_1) + \mathtt{d}(b_2, x) + \mathtt{d}(x, b_3), \\ S_2 &= \mathtt{d}(a, b_2) + \mathtt{d}(b_1, b_3) = \mathtt{d}(a, b_2) + \mathtt{d}(b_1, x) + \mathtt{d}(x, b_3), \\ S_3 &= \mathtt{d}(a, b_3) + \mathtt{d}(b_1, b_2) = \mathtt{d}(a, x) + \mathtt{d}(x, b_3) + \mathtt{d}(b_1, b_2), \end{aligned}$$

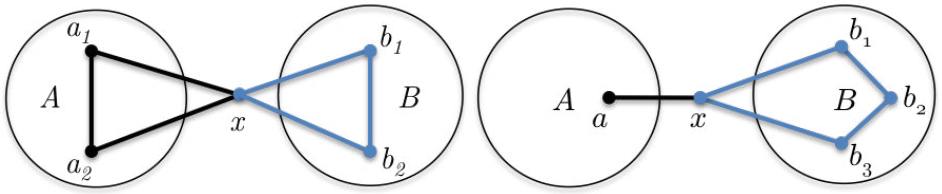
   alors $\delta(a, b_1, b_2, b_3) = \delta(x, b_1, b_2, b_3)$.

2. Prenons un quadruplet tel que $(a_1, a_2|b_1, b_2)$, par l'inégalité triangulaire nous avons :

$$S_1 = \mathtt{d}(a_1, a_2) + \mathtt{d}(b_1, b_2) \leq (\mathtt{d}(a_1, x) + \mathtt{d}(a_2, x)) + (\mathtt{d}(b_1, x) + \mathtt{d}(b_2, x)) = S_2 = S_3;$$

   alors $\delta = 0$.

□



(a) $\delta = 0$. (b) $\delta(a, b_1, b_2, b_3) = \delta(x, b_1, b_2, b_3)$.

FIGURE 1.8 – L'hyperbolicité de deux composantes connexes A et B séparées par le point d'articulation x.

Il est possible de calculer en temps linéaire [Tar72] les composantes connexes ne pouvant être séparées par un point d'articulation. Ces composantes sont appelées *composantes biconnexes*. Plus formellement :

Définition 9 (Composante biconnexe)**.** Un graphe G est dit biconnexe si, et seulement si, il n'existe aucun point d'articulation dans G.

Il est ainsi trivial d'étendre le théorème 3 à la décomposition en composantes biconnexes d'un graphe puisque les preuves sont identiques, nous obtenons :

Théorème 4. *Soit $B_1, \ldots, B_l$ les composantes biconnexes d'un graphe G alors :*

$$\delta(G) = \max_i \delta(B_i).$$

3 Méthode de décomposition du graphe par les cliques-séparatrices

Montrons maintenant comment borner l'hyperbolicité d'un graphe à partir de sa décomposition par les cliques-séparatrices. Dans un premier temps, j'explique en section 3.1 l'intérêt d'étudier les séparateurs du graphe pour le calcul de l'hyperbolicité. Je précise ensuite en section 3.2 les relations entre l'hyperbolicité d'un graphe et ses cliques-séparatrices. Puis en section 3.3, je détaille comment l'hyperbolicité d'un graphe peut être approchée à partir de sa décomposition par les cliques-séparatrices.

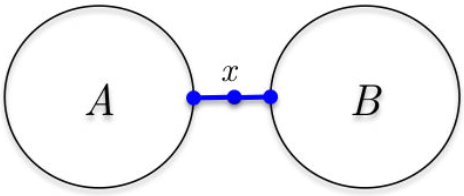
3.1 Relations entre l'hyperbolicité du graphe et ses séparateurs

Suivant le principe de la décomposition en composantes biconnexes, on s'aperçoit que l'ensemble des plus courts chemins entre les sommets de deux composantes passe par le point d'articulation qui les sépare. Cette observation peut se généraliser à tout ensemble $X \subset V$ de sommets dont la suppression déconnecte le graphe $G = (V, E)$. On appellera X un *séparateur* et k son diamètre. Deux valeurs de k nous intéressent plus particulièrement :

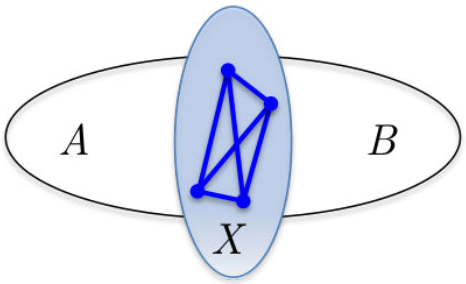
— $k = 0$, cela revient à calculer les points d'articulation de la décomposition en composantes biconnexes du graphe ;

— $k = 1$, cela revient à calculer la décomposition par les cliques-séparatrices du graphe, ce qui est faisable en temps $O(nm)$ [Tar85].

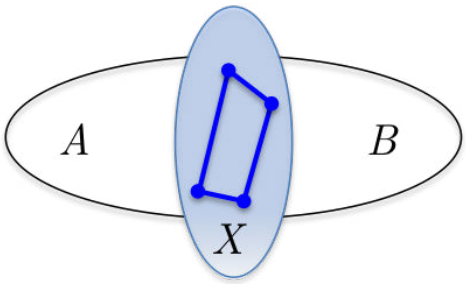
Les trois cas de séparateurs sont montrés en figure 1.9. La décomposition par les cliques-séparatrices est une méthode connue permettant d'accélérer le calcul de paramètres de graphes liés à l'hyperbolicité tel que la *longueur* d'une décomposition arborescente [DG07]. Il a aussi été montré dans [CDE+08] qu'un graphe de longueur arborescente k est k-hyperbolique. Enfin, une telle approche dans le cadre des réseaux complexes est particulièrement pertinente puisque ces réseaux sont facilement décomposables [BPS10b, DBKMS07, KPL+07].



(a) Un point d'articulation x ($k = 0$).



(b) Une $(A|B)$-clique-séparatrice X ($k = 1$).



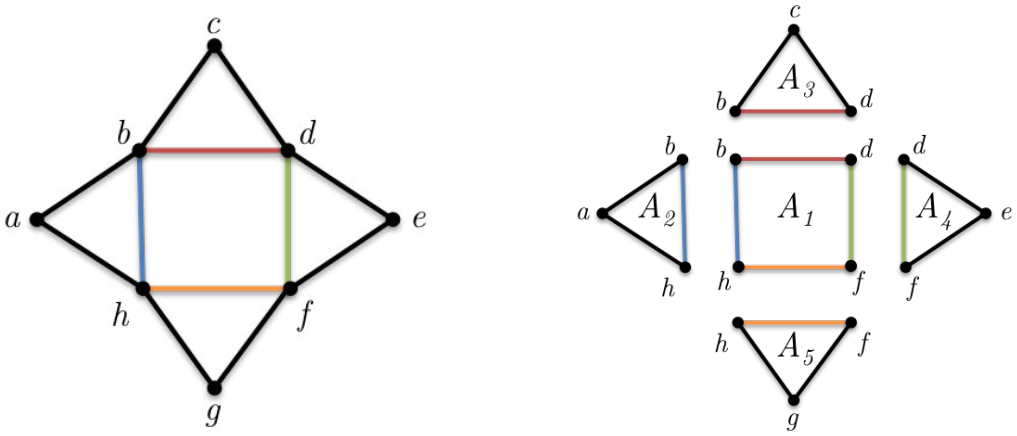
(c) Un $(A|B)$-séparateur X ($k = 2$).

FIGURE 1.9 – $(A|B)$-séparateurs avec les diamètres $k = 0$, $k = 1$ et $k = 2$.

Définissons tout d'abord ce qu'est une décomposition par les cliques-séparatrices. Notons $X \subset V$ un *séparateur* dans le graphe $G = (V, E)$. Si le sous graphe induit par X est complet,

alors on appellera X une *clique-séparatrice*. Si X est une clique-séparatrice dans $S \subseteq V$ et si S intersecte au moins deux composantes distinctes de $G \setminus X$, alors S est dit *séparable*. Nous appellerons *atome*, un ensemble de sommets non séparables. Une décomposition par les cliques-séparatrices est l'ensemble des atomes du graphe [Tar85].

Par la suite nous noterons A, B deux sous-ensembles de sommets. Nous appellerons X un (A, B)-séparateur si il déconnecte tout sommet $a \in A \setminus X$ de tout sommet $b \in B \setminus X$. En particulier, tout séparateur X contenant A ou B est un $(A|B)$-séparateur, dans un tel cas on nommera X un $(A|B)$-séparateur *trivial*. Enfin, nous appellerons X une *clique-minimale-séparatrice* si, et seulement si, $G(V - X)$ possède au moins deux composantes connexes C_1 et C_2 telles que $N(C_1) = N(C_2) = X$, avec $N(C_i)$ l'ensemble des sommets adjacents dans $G \setminus C_i$ à la composante C_i. L'ensemble des atomes d'une décomposition par les cliques-minimales-séparatrices est unique [Lei93]. Un exemple de décomposition d'un graphe en atomes est donné en figure 1.10.



(a) Un graphe et ses quatre cliques-minimales-séparatrices (b, h), (b, d), (d, f) et (f, h).

(b) Les cinq atomes de la décomposition du graphe.

FIGURE 1.10 – Décomposition du graphe par les cliques-minimales-séparatrices.

3.2 Hyperbolicité et cliques-séparatrices

Comme montré dans le théorème 4 (p. 11) pour la décomposition en composantes biconnexes, si quatre sommets sont séparés par un séparateur, seules les deux formes élémentaires d'un quadruplet $(a_1, a_2|b_1, b_2)$ et $(a|b_1, b_2, b_3)$ sont à considérer. Montrons quelles sont les bornes sur l'hyperbolicité qu'il est possible d'obtenir.

Hyperbolicité du quadruplet $(a_1, a_2|b_1, b_2)$

Étudions d'abord le quadruplet $(a_1, a_2|b_1, b_2)$ comme illustré en figure 1.11. Nous noterons $D(X)$, le diamètre du séparateur X.

Lemme 5. *Soit X un $A|B$-séparateur d'un graphe connexe G. Pour tout quadruplet $(a_1, a_2|b_1, b_2)$, $\delta(a_1, a_2, b_1, b_2) \leq D(X)$.*

Démonstration. Rappelons que nous avons les sommes suivantes :

$$S_1 = \mathrm{d}(a_1, a_2) + \mathrm{d}(b_1, b_2),$$
$$S_2 = \mathrm{d}(a_1, b_1) + \mathrm{d}(a_2, b_2),$$
$$S_3 = \mathrm{d}(a_1, b_2) + \mathrm{d}(a_2, b_1).$$

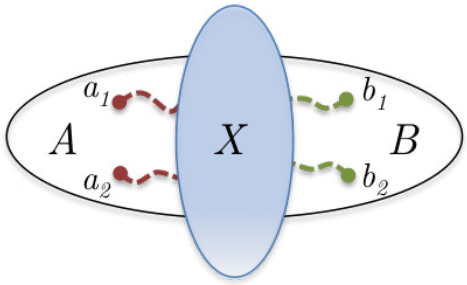


FIGURE 1.11 – Un quadruplet $(a_1, a_2|b_1, b_2)$ dont les sommets sont séparés par un $(A|B)$-séparateur X.

Sans perte de généralité posons $S_2 \geq S_3$ et $\mathbb{d}(v, X) = \min_{x \in X} \mathbb{d}(v, x)$. Nous savons aussi que pour tout $i, j \in \{1, 2\}$:

$$\begin{aligned} \mathbb{d}(a_i, b_i) &\geq \mathbb{d}(a_i, X) + \mathbb{d}(b_i, X), \\ \mathbb{d}(a_i, a_j) &\leq \mathbb{d}(a_i, X) + D(X) + \mathbb{d}(a_j, X). \end{aligned}$$

Si $S_1 \geq S_2$, nous avons :

$$\begin{aligned} S_2 &= \mathbb{d}(a_1, b_1) + \mathbb{d}(a_2, b_2) \\ &\geq \mathbb{d}(a_1, X) + \mathbb{d}(b_1, X) + \mathbb{d}(a_2, X) + \mathbb{d}(b_2, X) \\ &\geq [\mathbb{d}(a_1, X) + \mathbb{d}(a_2, X) + D(X)] + \\ &\quad [\mathbb{d}(b_1, X) + \mathbb{d}(b_2, X) + D(X)] - 2D(X) \\ &\geq S_1 - 2D(X). \end{aligned}$$

Nous obtenons donc $\delta(a_1, a_2, b_1, b_2) \leq (S_1 - S_2)/2 \leq D(X)$. De façon similaire on peut montrer que $S_3 \geq S_2 - 2D(X)$, et que lorsque $S_2 \geq S_1$ alors $S_1 \geq S_2 - 2D(X)$. Dans tous les cas, nous obtenons $\delta(a_1, a_2, b_1, b_2) \leq D(X)$. □

Corollaire 1. *Soit X une $(A|B)$-clique-séparatrice d'un graphe connexe G, pour tout quadruplet $(a_1, a_2|b_1, b_2)$ nous avons :*

$$\delta(a_1, a_2, b_1, b_2) \leq 1.$$

Hyperbolicité du quadruplet $(a|b_1, b_2, b_3)$

Étudions maintenant un quadruplet $(a|b_1, b_2, b_3)$ comme illustré en figure 1.12.

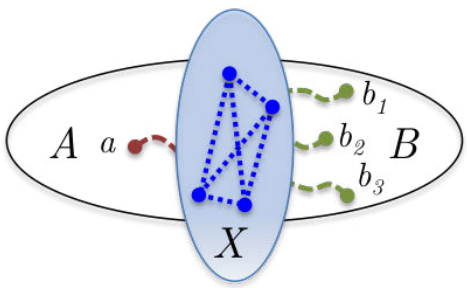


FIGURE 1.12 – Un quadruplet $(a|b_1, b_2, b_3)$ dont les sommets sont séparés par une $(A|B)$-clique-séparatrice X.

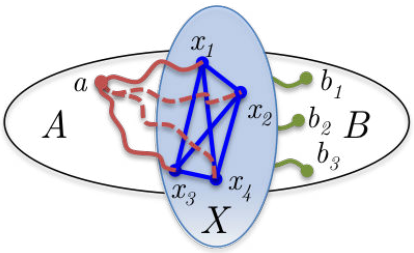
Comme X est une clique, chaque sommet $a \in A$ est à distance au moins $\mathtt{d}(u, X)$ et au plus $\mathtt{d}(u, X) + 1$ de tout sommet de X.

Lemme 6. *Soit X une $(A|B)$-clique-séparatrice d'un graphe connexe G, et soit $a \in A$. Si nous ajoutons un sommet a^* adjacent à $\{x \in X : \mathtt{d}(a, x) = \mathtt{d}(a, X)\}$, alors pour tous $b_1, b_2, b_3 \in B$ nous avons :*

$$\delta(a_1, b_1, b_2, b_3) = \delta(a^*, b_1, b_2, b_3).$$

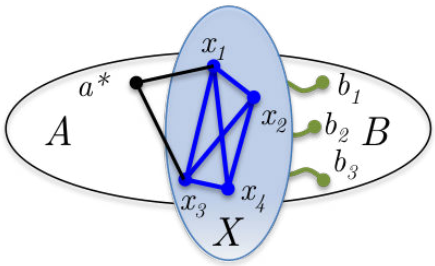
Démonstration. $\forall x \in X, \mathtt{d}(a, x) \in \{\mathtt{d}(a, X), \mathtt{d}(a, X)+1\}$ est vrai car X est une clique. Ainsi, pour tout sommet $b \in B, \mathtt{d}(a, b) = d(a^*, b) + \mathtt{d}(a, X) - 1$ avec le sommet a^* remplaçant a et dont l'hyperbolicité du quadruplet (a, b_1, b_2, b_3) ne change pas. □

Une illustration du lemme 6 est donnée en figure 1.13. Le sommet a de la figure 1.13a est à distance m des sommets x_1 et x_3 de la clique-séparatrice X. En appliquant le lemme, le sommet a est substitué par a^* comme montré en figure 1.13b. Cette méthode de substitution servira de base en section 4 pour le calcul exact de l'hyperbolicité.



Soit m une distance telle que :
$\mathtt{d}(a, x_1) = m,$
$\mathtt{d}(a, x_3) = m,$
$\mathtt{d}(a, x_2) = m + 1,$
$\mathtt{d}(a, x_4) = m + 1.$

(a) Un sommet a à distance m de x_1, x_3 et $m + 1$ de x_2, x_4.



$\delta(a, b_1, b_2, b_3) = \delta(a^*, b_1, b_2, b_3).$

(b) Le sommet a^* substitue le sommet a en étant connecté aux sommets de X tels que $\{x \in X : \mathtt{d}(a, x) = \mathtt{d}(a, X)\}$.

FIGURE 1.13 – Substitution du sommet a par le sommet a^*.

À partir d'un quadruplet de la forme $(a|b_1, b_2, b_3)$, il est aussi possible de borner son hyperbolicité en remplaçant un de ses sommets par celui d'une des cliques-séparatrices comme montré en figure 1.14.

Lemme 7. *Soit X une $(A|B)$-clique-séparatrice d'un graphe connexe G. Soit le quadruplet $(a|b_1, b_2, b_3)$, et soit $x \in X$ tel que $\mathtt{d}(a, X)$. Nous avons :*

$$\delta(a, b_1, b_2, b_3) \leq \delta(x, b_1, b_2, b_3) + 1/2.$$

Démonstration. Posons, sans perte de généralité, $S_1 \geq S_2 \geq S_3$, avec $S_1 = \mathtt{d}(a, b_1) + \mathtt{d}(b_2, b_3)$, $S_2 = \mathtt{d}(a, b_2) + \mathtt{d}(b_1, b_3)$ et $S_3 = \mathtt{d}(a, b_3) + \mathtt{d}(b_1, b_2)$.

D'après le lemme 6, chaque sommet $a \in A$ peut être remplacé par un sommet équivalent a^* et tous les sommets de A ont un voisin dans X. Dans un tel cas, pour tout sommet $b \in B$,

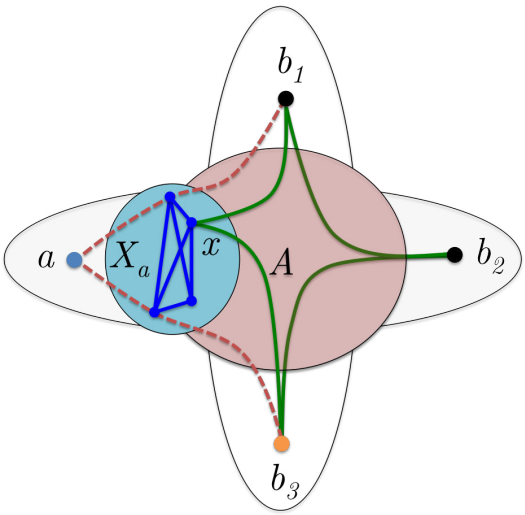


FIGURE 1.14 – $\delta(a, b_1, b_2, b_3) \leq \delta(x, b_1, b_2, b_3) + 1/2$. Au centre, les plus courts chemins formant le quadruplet (x, b_1, b_2, b_3). En pointillé et au centre, ceux formant le quadruplet (a, b_1, b_2, b_3).

$\mathbb{d}(x, b) \leq \mathbb{d}(a, b) \leq \mathbb{d}(x, b)+1$. De la même façon, pour toute somme $S_i^{'} = \mathbb{d}(x, b_i)+\mathbb{d}(b_j, b_k)$, avec $\{j, k\} = \{1, 2, 3\} \setminus \{i\}$, $S_i^{'} \leq S_i \leq S_i^{'} + 1$. Ainsi, pour tout $i \in \{2, 3\}$:

$$\begin{aligned}\delta(a, b_1, b_2, b_3) &\leq (S_1 - S_i)/2 \\ &\leq (S_1^{'} + 1 - S_i^{'})/2 \\ &\leq (S_1^{'} - S_i^{'})/2 + 1/2.\end{aligned}$$

En particulier, si $S_1^{'} \neq \max\left\{S_1^{'}, S_2^{'}, S_3^{'}\right\}$, et $S_i^{'} = \max\left\{S_2^{'}, S_3^{'}\right\}$, alors nous avons $\delta(a, b_1, b_2, b_3) \leq \frac{1}{2}$. Sinon, si $S_i^{'} = \max\left\{S_2^{'}, S_3^{'}\right\}$, nous avons $\delta(a, b_1, b_2, b_3) \leq \delta(x, b_1, b_2, b_3)+\frac{1}{2}$. □

Ensembles séparables

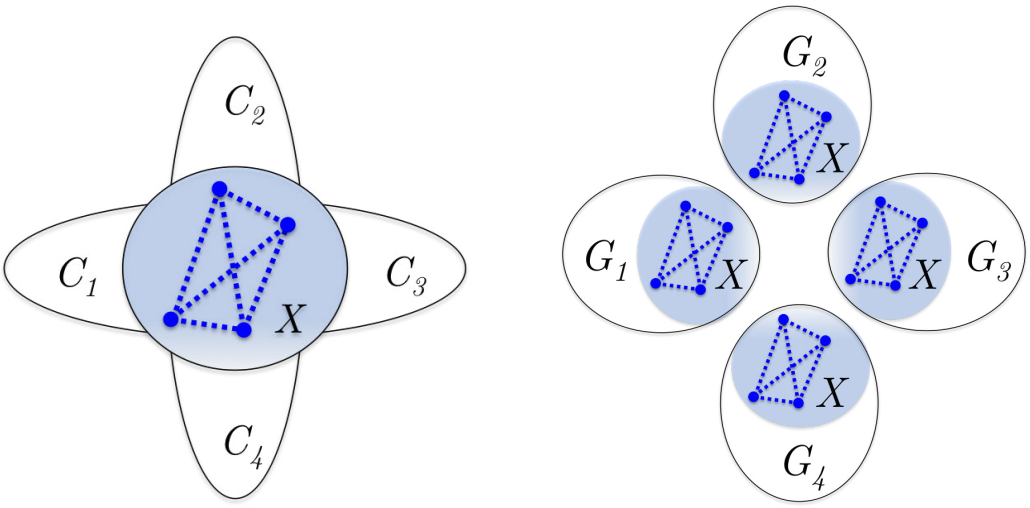
Considérons maintenant le cas ou une clique-séparatrice X ne sépare plus le graphe uniquement en deux composantes connexes mais en plusieurs nommées $C_1 \dots C_l$ comme illustré en figure 1.15. Nous noterons la composante connexe $G_i = G[C_i \cup X]$. Considérons aussi un quadruplet (a, b, c, d) de sommets et une composante connexe C_i telle que $|C_i \cap \{a, b, c, d\}|$ soit le plus grand possible. D'après les lemmes précédents trois cas se présentent :

1. $|C_i \cap \{a, b, c, d\}| = 4$, tous les sommets du quadruplet sont dans une composante C_i, alors $\delta(a, b, c, d) \leq \delta(G_i)$;
2. $|C_i \cap \{a, b, c, d\}| = 3$, posons les sommets a, b, c, d formant un quadruplet de la forme $(a|b_1, b_2, b_3)$, et posons $B = C_i \cup X$ et $A = V \setminus C_i$. À partir du lemme 7 nous obtenons $\delta(a, b, c, d) \leq \delta(G_i) + 1/2$;
3. $|C_i \cap \{a, b, c, d\}| \leq 2$, alors nous avons a, b, c, d qui forment un quadruplet $(a_1, a_2|b_1, b_2)$. Par le corollaire 1 (p. 14) nous savons que $\delta(a_1, a_2, b_1, b_2) \leq 1$.

Ce qui nous amène au théorème 8 permettant de borner l'hyperbolicité du graphe G.

Théorème 8. *Soit X une clique-séparatrice du graphe connexe G, et soient $C_1, \ldots, C_l$ les composantes connexes de $G \backslash X$. Soit la composante connexe $G_i = G[C_i \cup X]$, nous avons :*

$$\max\{\delta(G_1), \ldots \delta(G_l)\} \leq \delta(G) \leq \max\left\{\frac{1}{2}, \delta(G_1), \ldots \delta(G_l)\right\} + 1/2.$$



(a) Les composantes connexes C_1, C_2, C_3 et C_4 séparées par une clique-séparatrice X.

(b) Les composantes connexes G_1, G_2, G_3 et G_4 tels que $G_i = G[C_i \cup X]$.

FIGURE 1.15 – Un graphe séparé en quatre composantes connexes par une clique-séparatrice X.

3.3 Hyperbolicité et décomposition par les cliques-séparatrices

Voyons maintenant comment borner l'hyperbolicité d'un graphe à partir des atomes de sa décomposition par les cliques-séparatrices. Les différents cas élémentaires énoncés précédemment permettent de borner l'hyperbolicité d'un quadruplet (a, b, c, d) dont les sommets sont séparés par une clique-séparatrice. Le théorème 8 borne l'hyperbolicité d'un graphe G dont la clique-séparatrice sépare G en plusieurs composantes connexes. Nous insisterons sur le fait que les sommets de ces composantes connexes peuvent être séparés les uns des autres par plusieurs cliques-séparatrices au sein même de leur composante connexe. Il nous reste donc à étendre cette étude au cas de sommets séparés par plusieurs cliques-séparatrices.

En figure 1.16 est illustrée la séparation des quatre atomes $\{v_0, v_4, v_6\}$, $\{v_1, v_5, v_9\}$, $\{v_2, v_{10}, v_{14}\}$ et $\{v_3, v_{13}, v_{15}\}$ par un atome central. Notons que ce graphe est 2-hyperboliques alors qu'il est composé de quatre atomes 0-hyperboliques et d'un atome central 1-hyperbolique. On observe donc une différence de un entre l'hyperbolicité du graphe et celle de plus grande valeur parmi ses atomes. Rappelons que cette différence n'est que de un demi dans le cas de composantes connexes séparées par une clique-séparatrice. La question est donc de savoir si cette différence entre l'hyperbolicité du graphe et celle de plus grande valeur parmi ses atomes dépend du nombre de cliques-séparatrices séparant les sommets du quadruplet.

Deux observations permettent d'étendre nos résultats aux atomes du graphe.

1. Supposons qu'il existe un quadruplet de la forme $(a|b_1, b_2, b_3)$ dans le graphe tel que $\delta(a, b, c, d) \geq \frac{3}{2}$ avec a séparé des trois autres par un atome. Il est alors possible de

restreindre les cas d'étude uniquement aux quadruplets de la forme $(a|b_1, b_2, b_3)$.

2. Supposons que deux sommets du quadruplet (a, b, c, d) sont séparés de A par deux cliques-séparatrices distinctes. Il est alors possible d'obtenir une première borne sur l'hyperbolicité de l'atome.

Ensembles séparés par un atome

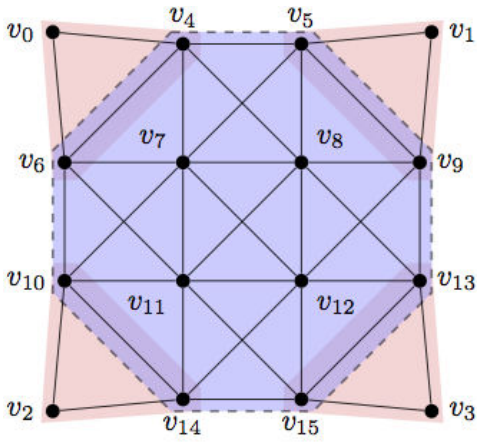


FIGURE 1.16 – Un graphe 2-hyperbolique avec cinq atomes ; quatre sont 0-hyperboliques, un est 1-hyperbolique.

Étudions le premier cas et montrons que si il existe un quadruplet dans G tel que $\delta(a, b, c, d) \geq \frac{3}{2}$, alors il peut exister un sommet $u \in \{a, b, c, d\}$ séparé des trois autres par un atome. Une telle situation peut être généralisée à des sommets séparés les uns des autres par plusieurs atomes. Plus formellement :

Lemme 9. *Soit a, b, c, d un quadruplet tel que $\delta(a, b, c, d) \geq \frac{3}{2}$ dans un graphe connexe G. Il existe un atome A tel que $\forall u \in \{a, b, c, d\} \backslash A$, il y a une clique-séparatrice $X_u \subseteq A$ séparant u de $\{a, b, c, d\} \setminus \{u\}$.*

La figure 1.16 illustre le lemme 9. Dans cette figure, tous les plus courts chemins entre les sommets du quadruplet passent par un même atome central. Chacun des sommets simpliciaux a, b, c et d est séparé des autres par une clique-séparatrice formée par son voisinage.

La preuve de ce lemme se base sur la décomposition arborescente du graphe donnée en annexe (lemme 7, p. 143). Celle-ci permet de délimiter les cas possibles de quadruplets en fonction d'un atome A bien choisi. Par exemple, si les quatre sommets du quadruplet sont inclus dans l'atome A, alors il n'y a rien à faire puisque $\{a, b, c, d\} \backslash A = \emptyset$. Au final, seuls deux cas importent.

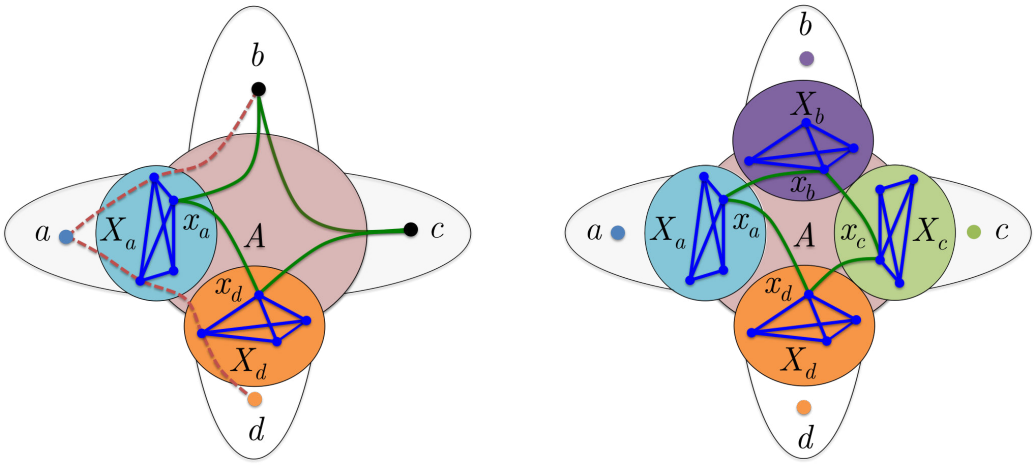
1. Une composante connexe C de $G \backslash A$ contient exactement deux éléments du quadruplet (a, b, c, d). Alors $N_G(C) \cap A$ est une clique de G séparant deux éléments de (a, b, c, d) des deux autres. Par le corollaire 1 (p. 14) ce cas est impossible puisque nous supposons que $\delta(a, b, c, d) \geq \frac{3}{2}$.
2. Donc, pour tout $u \in \{a, b, c, d\} \backslash A$, le voisinage de la composante connexe C_u de $G \backslash A$ contenant u tel que $N_G(C_u) \cap A$ est une clique qui sépare u de $\{a, b, c, d\} \setminus \{u\}$.

L'hypothèse d'un quadruplet tel que $\delta(a, b, c, d) \geq \frac{3}{2}$ va permettre de restreindre l'analyse des cas possibles aux seuls quadruplets de la forme $(a|b_1, b_2, b_3)$. De plus, le lemme 9 permet de généraliser à tout graphe les lemmes présentés par la suite, peut importe le nombre d'atomes séparant les sommets du quadruplet. Ainsi, en appliquant le lemme 7 (p. 15) successivement aux quatre sommets du quadruplet, on obtient le corollaire 2.

Corollaire 2. *Soit (a, b, c, d) un quadruplet et un graphe connexe G tel que $\delta(a, b, c, d) \geq \frac{3}{2}$. Alors il existe un atome A tel que :*

$$\delta(a, b, c, d) \leq \delta(G[A]) + 2.$$

Hyperbolicité du quadruplet (a, b, c, d) avec deux séparateurs



(a) $\delta(a, b, c, d) \leq \delta(x_a, b, c, x_d) + 1/2$. (b) Si $\delta(G) \geq \frac{3}{2}$ alors $\delta(a, b, c, d) \leq \delta(G[A] + 1$.

FIGURE 1.17 – Bornes de l'hyperbolicité à partir des sommets des cliques-séparatrices. Au centre, les plus courts chemins formant le quadruplet (x_a, b, c, x_d). En pointillé et au centre, ceux formant le quadruplet (a, b, c, d).

Étudions maintenant le deuxième cas, lorsque deux sommets du quadruplet sont séparés des autres par deux cliques-séparatrices distinctes. Dans le lemme 7 (p. 15), nous considérons le calcul de l'hyperbolicité d'un quadruplet dont un sommet a appartient à une composante connexe séparée d'un atome par une clique-séparatrice X. Dans ce cas, a peut être remplacé par un des sommets de X au prix d'une $\frac{1}{2}$-approximation de l'hyperbolicité comme montré en figure 1.14 (p. 16). Il est possible d'étendre ce résultat à un deuxième sommet du quadruplet séparé par une seconde clique-séparatrice distincte.

Lemme 10. *Soit (a, b, c, d) un quadruplet d'un graphe connexe G, et deux ensembles $X_a, X_d \subseteq V$ tels que :*

- *X_a est une $(a|b, c, d)$-clique-séparatrice ;*
- *X_d est une $(d|a, b, c)$-clique-séparatrice ;*
- *$X_a \backslash X_d$ appartient à la même composante connexe de $G \setminus X_d$ que a, b, c ;*

alors il existe $x_a \in X_a$ et $x_d \in X_d$ tel que :

$$\delta(a, b, c, d) \leq \delta(x_a, b, c, x_d) + 1/2.$$

Ce cas est illustré en figure 1.17a. La démonstration de ce lemme repose essentiellement sur la combinaison du lemme 6 (p. 15) et du lemme 7 (p. 15).

À partir du lemme 10, on s'aperçoit qu'il n'est pas nécessaire de considérer les sommets séparés par des cliques-séparatrices pour approcher l'hyperbolicité d'un atome. Il suffit de substituer un sommet du quadruplet par un des sommets de la clique-séparatrice. Considérons quatre sommets d'un quadruplet appartenant chacun à une composante connexe

différente des autres (voir figure 1.17a). Il suffit de deux applications successives du lemme pour borner l'hyperbolicité de l'atome comme illustré en figure 1.17b :

Corollaire 3. *Soit (a, b, c, d) un quadruplet et un graphe connexe G tel que $\delta(a, b, c, d) \geq \frac{3}{2}$. Alors il existe un atome tel que :*

$$\delta(a, b, c, d) \leq \delta(G[A]) + 1.$$

3.4 Approximation de l'hyperbolicité

Il est maintenant possible de généraliser le corollaire 3 à l'ensemble d'un graphe G et sa décomposition par les cliques-séparatrices.

Théorème 11. *Soit $A_1, \ldots, A_l$ les atomes d'un graphe connexe G. Alors :*

$$\max_i \delta(G[A_i]) \leq \delta(G) \leq \max_i \delta(G[A_i]) + 1.$$

On sait par isométrie que l'hyperbolicité des atomes de G ne peut être supérieure à $\delta(G)$, ce qui mène à la borne inférieure du théorème. Dans le cas ou $\delta(G) \leq 1$, alors la borne supérieure est triviale. Il reste donc le cas ou $\delta(G) \geq \frac{3}{2}$, ce qui correspond justement à la condition du lemme 9 (p. 18) et du corollaire 3 qui assure qu'il existe un atome tel que $\delta(G) \leq \delta(G[A]) + 1$.

4 Méthode de substitution pour le calcul exact de l'hyperbolicité

Nous cherchons maintenant à calculer la valeur exacte de l'hyperbolicité du graphe à partir des atomes de sa décomposition par les cliques-séparatrices. Comme le montre le théorème 11, l'hyperbolicité maximale parmi les atomes peut être légèrement inférieure à celle du graphe. Des quadruplets de plus grande hyperbolicité peuvent donc être formés entre les sommets de l'atome et ceux qui en sont séparés. À ce stade, calculer l'hyperbolicité de tels quadruplets et de ceux contenus dans les atomes pour toute la décomposition serait très couteux. Il serait plus rapide de calculer l'hyperbolicité de l'ensemble des quadruplets du graphe d'origine.

Cependant, nous savons par le lemme 6 (p. 15) qu'il est possible de conserver l'hyperbolicité d'un quadruplet de la forme $(a|b_1, b_2, b_3)$ en opérant une substitution du sommet a par un sommet a^*. Un tel sommet a^* peut remplacer plusieurs sommets a dont la distance vers les sommets les plus proches de la clique-séparatrice est identique. En substituant les sommets des composantes connexes séparées par un atome, il est possible de réduire le nombre de quadruplets à visiter pour le calcul de l'hyperbolicité de l'atome.

En section 4.1, je détaille sous quelles conditions il est possible de calculer la valeur exacte de l'hyperbolicité en se basant sur l'opération de substitution du lemme 6 (p. 15). Puis je montre en section 4.2 comment l'étendre à l'arbre des atomes de la décomposition par les cliques-minimales-séparatrices du graphe.

4.1 Méthode de substitution des sommets à partir d'une $(A|B)$-clique-séparatrice

L'opération de substitution se base sur une seule étape élémentaire décrite à partir d'une $(A|B)$-clique-séparatrice.

Soit un graphe connexe G et $V(G) = A \cup B$ tel que $X = A \cap B$ est une $(A|B)$-clique-séparatrice de G.

— Soit $G_1 = G[A]$. Pour tout $b \in B \setminus X$, considérons l'ensemble de sommets $X_b = \{x : x \in X \text{ tel que } d(b,x) = d(b,X)\}$ qui sont les plus proches de b dans X. Pour tout X_b, nous ajoutons à G_1 un sommet simplicial s_{X_b} dont le voisinage est X_b.

— Soit $G_2 = G[B]$. Pour tout $a \in A \setminus X$, nous considérons l'ensemble de sommets $X_a = \{x : x \in X \text{ tel que } d(a,x) = d(a,X)\}$ qui sont les plus proches de a dans X. Pour tout X_a, nous ajoutons à G_2 un sommet simplicial s_{X_a} dont le voisinage est X_a.

Les deux graphes résultant sont notés G_1^*, G_2^* et sont appelés des *graphes substituts* de G_1, G_2.

Théorème 12. *Soit G un graphe connexe tel que $\delta(G) \geq 1$, et A, B une couverture des sommets telle que $A \cap B = X$, avec X une $(A|B)$-clique-séparatrice de G. On définit $G_1 = G[A]$ et $G_2 = G[B]$. Alors :*

$$\delta(G) = \max\{1, \delta(G_1^*), \delta(G_2^*)\}. \tag{1.1}$$

Par le lemme 6 (p. 15), nous savons que l'hyperbolicité des quadruplets de la forme $(a|b_1, b_2, b_3)$ n'est pas modifiée par les opérations de substitution. Donc, seule l'hyperbolicité des quadruplets de la forme $(a_1, a_2|b_1, b_2)$ peut être altérée. Cependant, d'après le lemme 7 (p. 15), l'hyperbolicité de ces quadruplets est au plus 1. Il est donc possible d'ignorer ces quadruplets en ne considérant que les graphes tels que $\delta(G) \geq 1$. Le détail de la preuve du théorème 12 est donné en annexe (lemme 12, p.147).

4.2 Calcul de la valeur exacte de l'hyperbolicité

Nous pouvons désormais appliquer cette méthode de substitution à l'ensemble des cliques-séparatrices du graphe. À partir d'un atome, il suffit d'évaluer chacun des sommets des composantes connexes séparées de l'atome et les substituer comme énoncé en section 4.1. Cette opération est ensuite répétée successivement à tous les atomes de la décomposition.

Deux phases sont donc à distinguer :

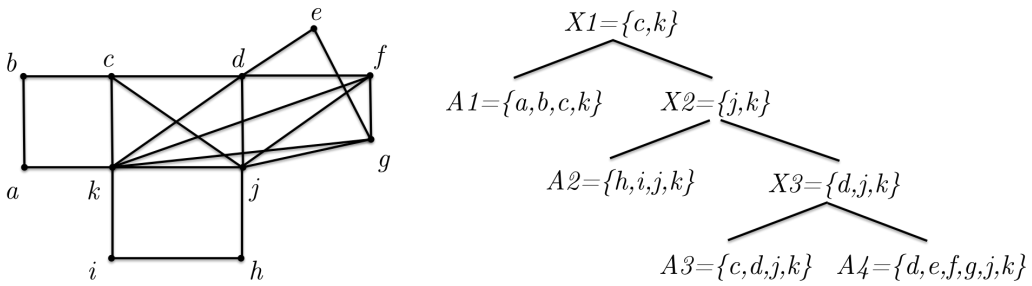
— phase 1, calcul de la décomposition par les cliques-séparatrices du graphe. Une première borne de l'hyperbolicité peut déjà être obtenue ;

— phase 2, calcul des substituts des atomes. La valeur exacte de l'hyperbolicité du graphe pourra être calculée par la suite.

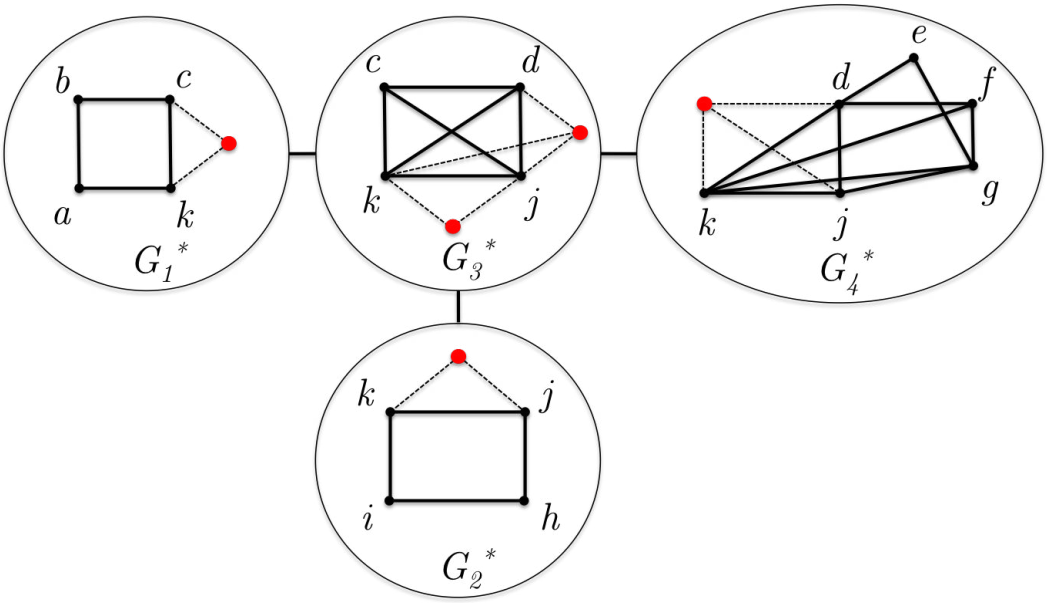
En pratique, cet algorithme se base sur l'arbre des atomes défini dans [BPS10b] et calculé à partir de la décomposition par les cliques-minimales-séparatrices du graphe. Ces deux phases sont donc combinées afin de garantir la complexité temporelle de l'algorithme en $O(nm)$ (voir algorithme en annexe (section 5.2, p.148).

Un exemple de décomposition en substituts est donné en figure 1.18. Une fois le graphe d'origine G décomposé en substituts, un algorithme quelconque pour le calcul de l'hyperbolicité peut être appliqué indépendamment sur chacun d'entre eux. Le nombre de substituts à évaluer peut être réduit au fur et à mesure du calcul de leur hyperbolicité. En effet, si H est un substitut, alors nous savons d'après le lemme 1 (p. 7) que $\delta(H) \leq D_H/2$. Ainsi, nous obtenons une borne supérieure de l'hyperbolicité du substitut H. Trions maintenant la liste des substituts par ordre de diamètre décroissant. Parcourons la liste des substituts et notons δ^* la plus grande valeur de l'hyperbolicité calculée parmi les substituts déjà évalués. Il est possible d'arrêter le calcul de l'hyperbolicité dès qu'un substitut H tel que $D_H/2 \leq \delta^*$ est rencontré.

Un algorithme pour le calcul de l'hyperbolicité peut ainsi être distribué sur chacun des substituts avec des instances exécutées en parallèle les unes des autres. La plus grande valeur δ déjà calculée est ensuite mise à jour au fur et à mesure entre les différentes instances de l'algorithme. Si $\delta \geq D_H/2$, alors le substitut H est tout simplement ignoré.



(a) Arbre des atomes de la décomposition par les cliques-minimales-séparatrices du graphe.



(b) Les substituts G_1^*, G_2^*, G_3^* et G_4^*.

FIGURE 1.18 – Exemple de substitution des atomes d'un graphe.

Notons que cette méthode de calcul est tout aussi bien applicable à une décomposition en composantes biconnexes.

5 Résultats expérimentaux

Une phase expérimentale a été menée afin d'évaluer les performances de cette nouvelle méthode de substitution des sommets et mieux comprendre quels facteurs impactent la taille finale des substituts.

5.1 Données d'expérimentation

Les substituts de cinq réseaux de collaboration ont été calculés [LKF07] :

— `ca-AstroPh`, pour le graphe de la communauté en astrophysique ;

— `ca-CondMat`, pour le graphe de la communauté en physique de la matière condensée ;

— `ca-GrQc`, pour le graphe de la communauté de la relativité générale et de la cosmologie quantique ;

— `ca-HepPh`, pour le graphe de la communauté en phénoménologie de la physique des particules ;

— `ca-HepTh`, pour le graphe de la communauté en physique théorique des hautes énergies.

Dans ces graphes, les noeuds représentent les scientifiques et les arêtes représentent les collaborations (c'est-à-dire les articles avec co-auteurs). Ces graphes ont été choisis car ils possèdent de nombreuses cliques de tailles diverses. En effet, un article avec k scientifiques en co-auteurs induit une clique de taille k dans le graphe. De plus, le nombre de co-auteurs par article varie fortement d'une communauté à une autre. Ainsi, malgré les propriétés similaires partagées par ces graphes [LKF07], nous observons des résultats variés en terme de taille des substituts.

5.2 Résultats empiriques

L'algorithme de calcul des substituts se base sur celui pour le calcul de l'arbre des atomes donné dans [BPS10a]. Pour chacun des graphes, le substitut de chaque atome de la décomposition est calculé.

Nous analyserons dans un premier temps la taille des composantes biconnexes des graphes et celle de leurs atomes. Ensuite, nous verrons comment les règles de construction des substituts impactent le nombre de sommets simpliciaux à ajouter.

Décomposition en composantes biconnexes

Les distributions cumulées, en pourcentage, du nombre de composantes biconnexes en fonction de leur nombre de sommets sont données en figure 1.19a (p. 28). La taille des composantes biconnexes est donnée en pourcentage de la taille de la LBC. Ces résultats montrent que les cinq graphes sont composés d'une seule grande composante biconnexe. Elle sera nommée LBC (pour Largest Biconnected Component). Ces LBC incluent entre 50% et 84.85% des sommets. Au mieux (`ca-GrQc`), la deuxième composante biconnexe de plus grande taille couvrent seulement 5% des sommets du graphe.

Ces composantes de petite taille peuvent être ignorées dans le cadre du calcul de l'hyperbolicité. Comme il sera montré en chapitre 2, soit la taille des composantes est inférieure à quatre sommets ou alors leur diamètre est deux fois inférieur à l'hyperbolicité de la LBC. Nous nous intéresserons donc uniquement à la LBC et à sa décomposition en atomes dans la suite des expérimentations.

Décomposition de graphe par les cliques-séparatrices

En figure 1.19b (p. 28) est donnée la distribution cumulée de la taille des atomes de la LBC. Le nombre cumulé d'atomes est donné en pourcentage du nombre total d'atomes et la taille des atomes en pourcentage du nombre total de sommets dans la LBC. Une nouvelle fois, pour chacun des graphes, nous observons un seul grand atome. Nous le nommerons LA (pour Largest Atom). Cet atome LA inclut entre 50% et 60% des sommets. Tous les autres atomes ne représentent qu'une petite fraction de l'ensemble des sommets. Au mieux (`ca-HepPh`), le deuxième atome de plus grande taille couvrent seulement 2.65% des sommets du graphes.

En conséquence et comme nous le verrons aussi dans le chapitre 2, les plus petites composantes biconnexes et les substituts des plus petits atomes de la LBC peuvent être ignorés pour le calcul de l'hyperbolicité. Seul le substitut du LA est nécessaire. Il sera nommé LS (pour Largest Substitute).

Taille des substituts

Nous savons que la taille du LS dépend de la taille initiale du LA et du nombre de sommets simpliciaux ajouté. Après calcul du substitut de chacun des LA des cinq graphes étudiés, les résultats suivants sont reportés dans le tableau 1.1 :

- le nombre de sommets n pour chacun des graphes originaux ;
- le nombre de sommets n_B de la LBC ;
- le nombre de sommets n_{LA} du LA ;
- le nombre de sommets n_{LS} du LS ;
- le pourcentage de sommets R_{LA} supprimés de la LBC pour obtenir le LA. C'est-à-dire $R_{LA} = \frac{n_B - n_A}{n_B}$;
- le pourcentage de réduction R_{LS} du LS par rapport à la LBC. C'est à dire $R_{LS} = \frac{n_B - n_{LS}}{n_B}$;
- le temps de calcul T des substituts en secondes.

On observe une réduction significative du nombre de sommets de la LBC pour obtenir le LA. Entre 37.40% et 49.22% des sommets de la LBC sont supprimés. Cependant, la réduction de taille obtenue après la construction du substitut varie selon les graphes. La méthode de substitution permet de réduire le nombre de sommet de 11.22% à 20.84% par apport à la taille initiale de la LBC.

Graphe	n	n_B	n_{LA}	n_{LS}	R_{LA}%	R_{LS}%	Coût%	T (en sec.)
`ca-CondMat`	23 133	17 234	8 751	13 643	49.22	20.84	28.39	672
`ca-GrQc`	5 242	2 651	1 386	2 107	47.72	20.52	27.20	5
`ca-HepPh`	12 008	9 025	4 925	7 170	45.43	20.55	24.88	167
`ca-AstroPh`	18 772	15 929	9 561	13 407	39.98	15.83	24.14	679
`ca-HepTh`	9 877	5 898	3 692	5 236	37.40	11.22	26.18	53

TABLE 1.1 – Caractéristiques des graphes des réseaux de collaborations.

Analyse du LA et de la construction du LS. Étudions l'impact de la taille initiale du LA et celui du nombre de sommets simpliciaux ajoutés par la méthode de substitution lors de la construction du LS. Le nombre de sommets simpliciaux en pourcentage de la taille de la LBC est donné dans la colonne `Coût` du tableau 1.1. Comparons le graphe `ca-CondMat`, qui a les meilleurs taux de réductions R_{LS} et R_{LA}, par rapport aux autres graphes. Malgré un important taux de réduction R_{LA} de 49,22%, le taux R_{LS} de `ca-CondMat` est proche de celui des graphes `ca-HepPh` et `ca-GrQc` (variant entre 20.55% et 20.84%). En proportion, un plus grand nombre de sommets simpliciaux est ajouté dans le cas de `ca-CondMat`, représentant jusqu'à 28,39% de la taille de la LBC alors qu'ils représentent seulement 24.88% dans le cas de `ca-HepPh`.

Un comportement similaire est observé entre `ca-AstoPh` et `ca-CondMat`. Même si leur taux R_{LA} varie de 9,25%, la différence du taux R_{LS} ne varie que de 5%. Cela représente 4,24% de sommets simpliciaux en moins dans `ca-AstoPh` comparativement à `ca-CondMat`. À l'opposé, la différence de 2.29% du taux R_{LA} entre `ca-HepPh` et `ca-GrQc` est compensée par 2.32% de sommets simpliciaux en moins dans `ca-HepPh`. Le taux R_{LS} de `ca-HepPh` est donc finalement plus petit que celui de `ca-GrQc`.

En comparant `ca-HepTh` et `ca-CondMat`, on remarque une différence de 11.82% entre leurs taux R_{LA}. Au final, 9,61% de sommets simpliciaux supplémentaires ont été nécessaires pour construire le LS de `ca-HepTh`.

Il est donc difficile de corréler la taille du LA et le nombre de sommets simpliciaux ajoutés pour construire le LS. L'importante réduction du nombre de sommets entre la LBC et le LA peut être au final compensée par un grand nombre de sommets simpliciaux et inversement. Il est donc nécessaire de mieux comprendre ce qui impacte la taille du LA et le nombre de sommets simpliciaux ajoutés lors de la construction du LS.

5.3 Analyse de la décomposition par les cliques-séparatrices et de la construction des substituts

Les résultats précédents étant très variés, nous analysons maintenant plus en détail les propriétés causant une telle hétérogénéité.

Décomposition par les cliques-séparatrices

Étudions la composition du LA en termes de cliques-séparatrices. Notons $\mathcal{X}_{LA} = \{X_1, \ldots, X_l\}$ les cliques-minimales-séparatrices contenues dans le LA, déconnectant les atomes dans $\mathcal{A}_{LA} = \{A_1, \ldots, A_l\}$. Des atomes autres que ceux contenus dans $\mathcal{A}_{LA}$ peuvent exister. Cependant, de tels atomes ne recouvrent pas le LA. Dit autrement, l'ensemble $\mathcal{A}_{LA}$ représente le voisinage du LA dans le graphe des atomes, comme défini dans [BPS10b].

En figure 1.20a (p. 29) est donnée la distribution cumulée de la taille des cliques-séparatrices dans le LA. Cette taille est normalisée par le nombre total de cliques-séparatrices dans le graphe. On observe de plus petites cliques-séparatrices pour les graphes `ca-HepTh` et `ca-CondMat`, avec des tailles maximales de respectivement 8 et 12. Les tailles maximales des trois autres graphes, `ca-GrQc`, `ca-AstroPh` et `ca-HepPh` étant respectivement de 42, 53 et 192. Dans le tableau 1.2 est aussi donné le ratio $R_{|\mathcal{X}_{LA}|}$ du nombre de cliques-minimales-séparatrices $|\mathcal{X}_{LA}|$ normalisé par la taille n_{LA} du LA. Ce ratio varie de 0.39 pour `ca-AstroPh` à 0.54 pour `ca-CondMat`. Au final, comparativement à `ca-AstropPh`, on observe un nombre bien plus important de cliques-séparatrices dans `ca-CondMat`, mais celles-ci sont de plus petite taille.

Analyse des atomes. Afin de compléter les mesures, étudions la répartition des sommets dans les atomes et mettons en relation la taille des cliques-séparatrices dans le LA avec la proportion de sommets qui en sont déconnectés. Les résultats suivants sont reportés dans le tableau 1.2 :

- $\alpha_1 = n_B - n_{LA}$, le nombre total de sommets séparés du LA dans la LBC ;
- $\alpha_2 = |V(\mathcal{A}_{LA} \setminus \mathcal{X}_{LA})|$, le nombre de sommets dans les atomes de $\mathcal{A}_{LA}$ qui ne sont pas dans le LA ;
- $|\mathcal{X}_{LA}|$, le nombre de cliques-minimales-séparatrices dans le LA ;
- $\Delta_1 = \frac{\alpha_1 - \alpha_2}{n_B}$, le pourcentage de sommets qui ne sont contenus ni dans le LA, ni dans aucun atome de l'ensemble $\mathcal{A}_{LA}$;
- $\Delta_2 = R_{LA} - \Delta_1$, le pourcentage de sommets des atomes de $\mathcal{A}_{LA}$ directement séparés du LA ;
- $R_{\leq 5}\%$, le pourcentage de sommets séparés par des cliques-minimales-séparatrices de taille inférieure à six sommets ;
- $R_{=2}\%$, le pourcentage de sommets séparés par des arêtes (cliques-minimales-séparatrices de taille 2).

D'après ces résultats, on en déduit que la plupart des sommets sont contenus dans le LA ou des atomes directement séparés du LA. Les autres sommets ne représentent que 2.88% à 7.03% de leur nombre total. Le pourcentage de sommets séparés du LA en fonction de la

taille des cliques-minimales-séparatrices est donné en figure 1.20b (p. 29). Les valeurs correspondantes pour $R_{\leq 5}\%$ et $R_{=2}\%$ montrent que dans tous les graphes, les cliques-minimales-séparatrices de petite taille, c'est-à-dire inférieures à cinq sommets, sont responsables d'une importante part des sommets séparés du LA. Plus particulièrement, on remarque aussi que les graphes `ca-GrQc` et `ca-HepTh` ont respectivement 7.71% et 4.91% de sommets supplémentaires séparés par une arête comparativement à `ca-CondMat`. On peut donc soupçonner les cliques-minimales-séparatrices de petite taille d'être à l'origine des différences entre la taille des substituts LS des différents graphes étudiés.

Graphe	α_1	α_2	$\lvert\mathcal{X}_{LA}\rvert$	$R_{\mathcal{X}_{LA}}$	$\Delta_1\%$	$\Delta_2\%$	$R_{\leq 5}\%$	$R_{=2}\%$
`ca-CondMat`	8 483	7 413	4 702	0.54	6.21	43.01	37.34	16.32
`ca-GrQc`	1 265	1 079	698	0.5	7.03	40.69	38.59	24.03
`ca-HepPh`	4 100	3 727	2 166	0.44	4.13	41.3	30.96	16.01
`ca-AstroPh`	6 368	5 910	3 715	0.39	2.88	37.1	23.67	9.57
`ca-HepTh`	2 206	1 942	1 506	0.41	4.47	32.93	31.11	21.23

TABLE 1.2 – Données concernant les cliques-minimales-séparatrices des graphes.

Construction des substituts

Validons maintenant l'idée que le plus grand nombre de sommets simpliciaux est connecté aux cliques-minimales-séparatrices de petite taille. En figure 1.21a (p. 30) est donnée la distribution cumulée du nombre de sommets simpliciaux connectés au LA, normalisée par la taille de la LBC, en fonction de la taille des cliques-minimales-séparatrices. Notons pour chacun des graphes, que la somme de ces valeurs correspond à celle donnée dans la colonne `Coût` du tableau 1.1. On observe pour les cliques-minimales-séparatrices de degrés deux et trois, que la proportion de sommets simpliciaux pour les graphes `ca-CondMat`, `ca-GrQc`, `ca-HepPh`, `ca-AstroPh` et `ca-HepTh` représente respectivement 65.49%, 88.63%, 76.26%, 50.16% et 88.02% du nombre total de sommets simpliciaux connectés au LA. Ces valeurs confirment l'importance des cliques-minimales-séparatrices de petite taille auxquelles une grande proportion de sommets simpliciaux sont connectés.

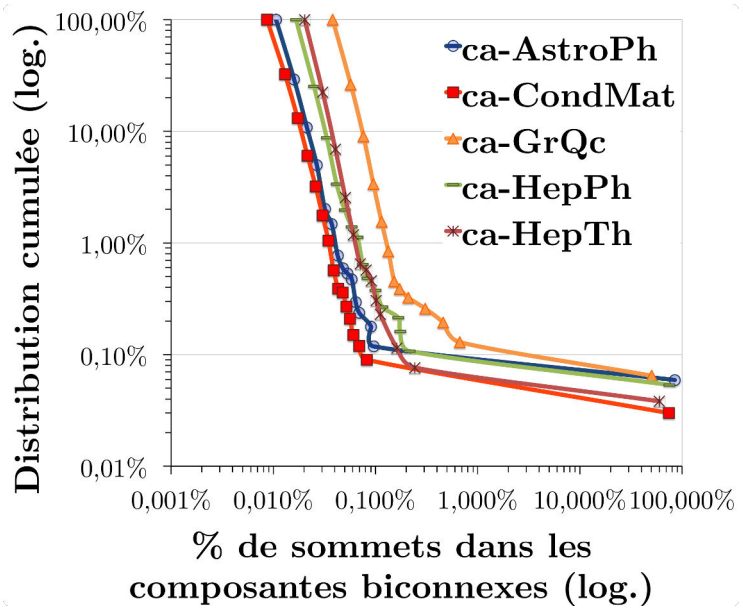
Remarquons aussi en comparant la figure 1.21a (p. 30) et la figure 1.21b (p. 30) que pratiquement tous les sommets simpliciaux sont connectés à une clique-minimale-séparatrice de même taille. Dans le pire cas (`ca-GrQc`), il n'y a pas plus de 0.75% de sommets simpliciaux dont le degré diffère et la plupart sont de degré deux. Ainsi, la proportion finale de sommets simpliciaux `Coût` donnée dans le tableau 1.1 dépend en grande partie de la distribution du degré des cliques-minimales-séparatrices.

De plus, les différents graphes étudiés ont une proportion similaire de sommets simpliciaux ajoutés lors de la construction du LS. Dans le pire cas, une différence de 4,25% est observée entre `ca-CondMat` et `ca-AstroPh`. Il est donc possible de comparer entre les graphes la distribution du degré de leurs sommets simpliciaux comme illustrée en figure 1.21b (p. 30). Considérons une clique-séparatrice de faible degré telle que sa taille est inférieure à quatre sommets. Dans le cas de `ca-AstroPh`, le nombre de sommets simpliciaux séparés par des cliques-séparatrices de petit degré est bien inférieur à celui des autres graphes. Cependant, puisque la proportion de sommets simpliciaux est semblable entre les graphes, cela signifie que `ca-AstroPh` possède bien plus de sommets simpliciaux séparés par des cliques-séparatrices de degré supérieur à trois. Le phénomène inverse est observé pour les quatre autres graphes. Plus précisément, la proportion de sommets simpliciaux séparés par des cliques-séparatrices de petit degré représentent respectivement pour les graphes `ca-CondMat`, `ca-GrQc`, `ca-HepPh`, `ca-AstroPh` et `ca-HepTh`, un pourcentage de 18.59%, 24.1%, 18.97%, 12.11% et 23.04% du nombre total de sommets simpliciaux.

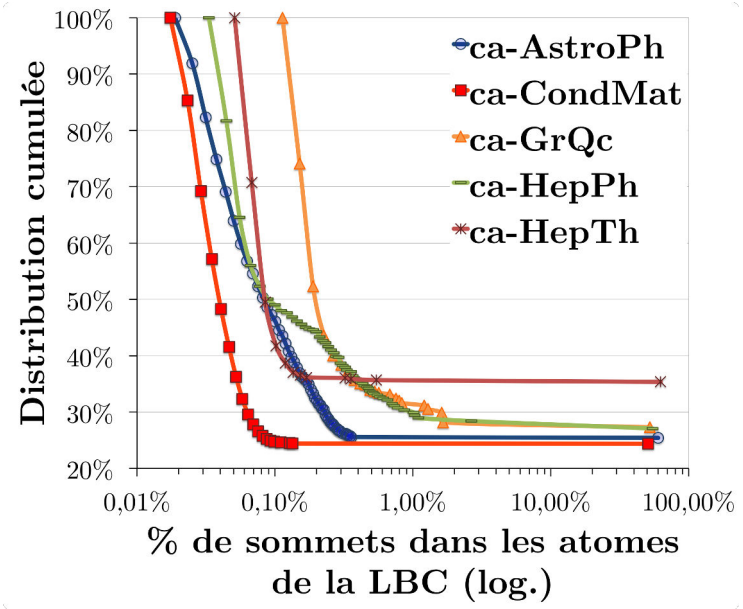
Ces expérimentations sur les graphes `ca-*` mettent en lumière l'importance de la taille initiale du plus grande atome à partir duquel le substitut est calculé. Les trois graphes `ca-CondMat`, `ca-GrQc` et `ca-HepPh` ayant le plus grand ratio R_{LA} sont aussi ceux ayant le plus grand ratio R_{LS}. Cependant, un faible ratio R_{LA} peut être compensé par un moindre coût en terme d'ajout de sommets simpliciaux comme c'est le cas pour `ca-HepPh`. En effet, ce graphe possède un ratio R_{LA} plus faible que celui de `ca-GrQc`, mais au final son ratio R_{LS} est plus élevé. Enfin, pour tous les graphes, on observe une distribution du degré des sommets simpliciaux similaire à celle des cliques-minimales-séparatrices. En particulier, les sommets simpliciaux de degré deux et trois représentent de 50,16% à 88,63% des sommets ajoutés pour la construction du substitut.

6 Conclusion

Dans ce chapitre, je montre la relation serrée entre l'hyperbolicité du graphe et la plus grande valeur trouvée parmi les atomes de sa décomposition par les cliques-séparatrices. En plus de ce premier résultat, il a été possible d'en déduire une nouvelle approche basée sur le calcul de substitut des atomes afin de déterminer la valeur exacte de l'hyperbolicité d'un graphe. Cette nouvelle méthode présente l'avantage de conserver une complexité temporelle en $O(nm)$, similaire à celle de l'algorithme de calcul des atomes dans le graphe. Cependant, seuls des graphes d'hyperbolicité au moins 1 peuvent être évalués. Les expérimentations menées sur les graphes de divers réseaux de collaborations scientifiques montrent qu'un arbre des atomes peut être calculé en quelques minutes. On observe aussi une forte relation entre la taille finale des substituts et le nombre de cliques-séparatrices de petite taille présentes dans les atomes. Il serait intéressant de pouvoir étendre cette méthode à d'autres décompositions de graphes.

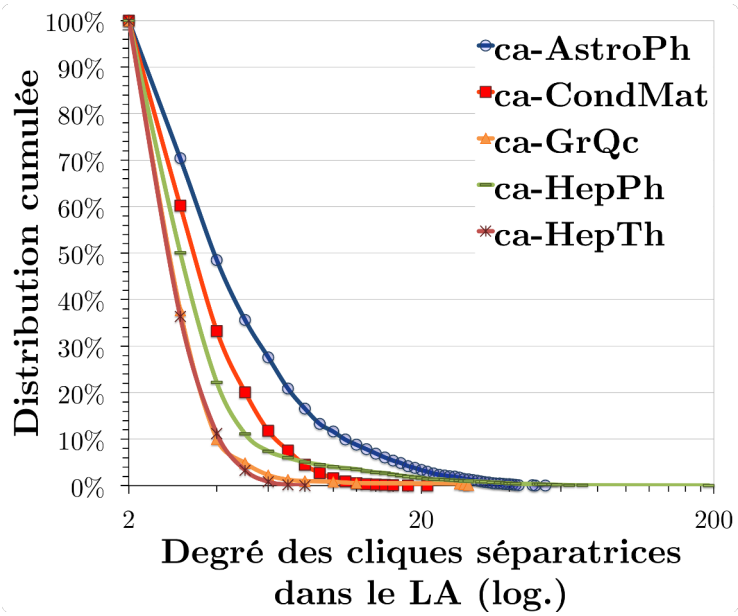


(a) Composantes biconnexes.

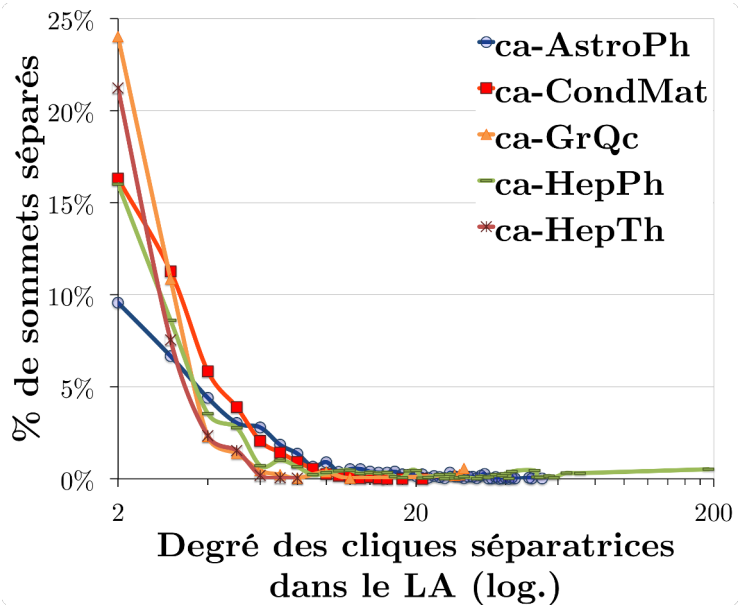


(b) Atomes de la LBC.

FIGURE 1.19 – En figure 1.19a, la distribution cumulée de la taille des composantes biconnexes. En figure 1.19b, celle des atomes dans la LBC.

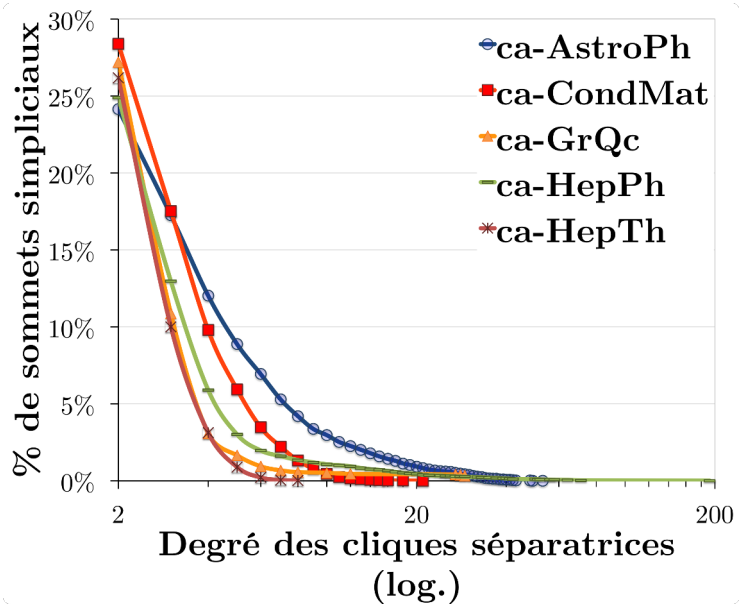


(a) Cliques-séparatrices dans le LA.

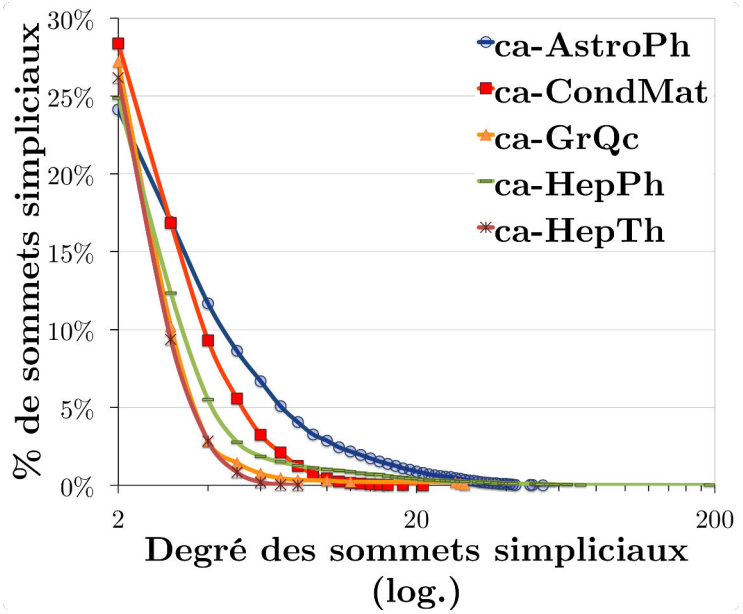


(b) Sommets séparés du LA.

FIGURE 1.20 – En figure 1.20a, la distribution cumulée de la taille des cliques-séparatrices dans le LA. En figure 1.20b, celle du pourcentage de sommets séparés en fonction de la taille des cliques-séparatrices du LA.



(a) Sommets simpliciaux connectés au LA en fonction de la taille des cliques-séparatrices.



(b) Distribution des degrés des sommets simpliciaux du LA.

FIGURE 1.21 – En figure 1.21a, la distribution cumulée du nombre de sommets simpliciaux connectés au LA. Cette distribution est normalisée par la taille de la LBC en fonction de la taille des cliques-séparatrices auxquelles les sommets simpliciaux sont connectés. En figure 1.21b, celle du nombre de sommets simpliciaux connectés au LA normalisée par la taille de la LBC.

Chapitre 2

Algorithme exact pour le calcul de l'hyperbolicité dans de grands graphes

Sommaire

1 Algorithme exact pour le calcul de l'hyperbolicité 32
1.1 Algorithme pour le calcul de l'hyperbolicité 33
1.2 Utilisation des paires éloignées 34
1.3 Algorithme parallèle pour le calcul de l'hyperbolicité 34
2 Résultats expérimentaux 35
2.1 Méthodes de pré-traitement des graphes 36
2.2 Évaluation des performances de l'algorithme 37
2.3 Décomposition et calcul de l'hyperbolicité dans les grands graphes . 38
2.4 Coût du pré-traitement du graphe pour le calcul de l'hyperbolicité . 42
2.5 Évaluation des performances de l'algorithme parallèle 42
3 Heuristique pour le calcul de l'hyperbolicité 44
3.1 Conception de l'heuristique 44
3.2 Calculs expérimentaux de l'heuristique 45
3.3 Analyse des performances de l'heuristique. 46
4 Conclusion 47

Dans le chapitre 1, nous avons vu qu'en pratique, le seul algorithme testé expérimentalement a une complexité temporelle en $O(n^4)$. Dans [ASHM13], les auteurs proposent une implémentation massivement parallèle de l'algorithme en $O(n^4)$, sur les 1 015 cœurs d'un SGI Altix UV 1000 équipé de processeurs Intel Nehalem EX 2.0 GHz. Plus de 3 heures de calcul leurs ont été nécessaires pour obtenir la valeur exacte de l'hyperbolicité d'un graphe à 8 104 sommets. Cette complexité reste donc bien trop importante dans le cas de graphes en $O(10k)$ sommets.

Cependant, nous avons aussi vu qu'il est possible d'associer les décompositions en composantes biconnexes et par les cliques-séparatrices afin de diviser le graphe en sous-graphes de plus petite taille. Il est aussi possible d'utiliser la notion de *paires éloignées* afin de réduire le nombre de quadruplets à visiter. Un algorithme performant en pratique permettrait de tirer partie de ces nouvelles méthodes.

C'est pourquoi je présente un nouvel algorithme qui calcule l'hyperbolicité de graphes de l'ordre de 58 000 sommets en quelques heures en combinant les méthodes présentées dans le chapitre 1. Cet algorithme tire avantage de la borne supérieure sur l'hyperbolicité d'un graphe G, plus précisément telle que $\delta \leq D(G)/2$. Bien que sa complexité temporelle soit en $O(n^4)$, nous verrons qu'en pratique celle-ci dépend de l'hyperbolicité du graphe et de sa distribution des distances.

Dans ce chapitre, je présente quatre contributions :

1. la proposition d'un nouvel algorithme séquentiel pour le calcul de l'hyperbolicité surclassant l'implémentation massivement parallèle de [ASHM13] par un facteur 1 000 sur un seul CPU ;
2. une version parallèle de l'algorithme permettant de diviser par 4 le temps de calcul de la version séquentielle avec une exécution sur 16 cœurs :
3. une analyse des performances de l'algorithme séquentiel et parallèle sur divers types de graphes ;
4. une nouvelle heuristique pour approcher en quelques secondes l'hyperbolicité de graphes composés de plusieurs millions de sommets.

Le code que j'ai développé en language Java de l'algorithme exact utilisant les méthodes de décomposition du graphe en composantes biconnexes, de calcul des substituts et des paires éloignées ainsi que l'heuristique est libre et disponible [Lanb]. Deux autres implémentations libres de l'algorithme exact uniquement sont aussi disponibles [Cou14, S+].

Je commence en section 1 par détailler ce nouvel algorithme, sa version parallèle et l'utilisation des paires éloignées. En section 2, j'évalue ses performances sur divers graphes de référence. En particulier, je montre son efficacité, comparativement aux autres méthodes existantes, sur les cartes CAIDA [CAI13b] et DIMES [SS05] des systèmes autonomes (ou AS, Autonomous System) et divers réseaux de collaboration [CML11, LKF07]. Je montre aussi la réduction du temps de calcul qu'il est possible d'obtenir en utilisant des outils pour la parallélisation de l'algorithme. Enfin, je propose en section 3 une nouvelle heuristique pour le calcul de l'hyperbolicité pouvant atteindre des graphes composés de plusieurs millions de sommets.

1 Algorithme exact pour le calcul de l'hyperbolicité

L'idée principale pour le développement d'un nouvel algorithme est d'essayer d'identifier et de calculer uniquement les quadruplets pouvant avoir la plus grande valeur d'hyperbolicité. Nous avons vu dans le chapitre 1, que lors de la décomposition d'un graphe en composantes biconnexes, les quadruplets dont les sommets appartiennent à deux composantes différentes n'ont pas besoin d'être évalués. De la même manière, lors d'une décomposition par les cliques-séparatrices, le calcul de substituts permet d'ignorer certains quadruplets dont les sommets appartiennent à différents atomes. Ces résultats sont obtenus en prouvant que la *borne supérieure* de l'hyperbolicité sur les quadruplets ignorés est inférieure ou égale à la *borne inférieure* des quadruplets composés uniquement de sommets d'une même composante biconnexe ou d'un même substitut.

Utiliser les bornes inférieures et supérieures de l'hyperbolicité des quadruplets est donc utile dans le cas de l'application de méthodes de pré-traitement du graphe. Celles-ci permettant de réduire la taille des instances sur lesquelles calculer l'hyperbolicité. La question est maintenant de savoir si des bornes sur l'hyperbolicité peuvent être utilisées lors de l'exécution d'un algorithme pour le calcul de l'hyperbolicité afin de ne visiter que les quadruplets les plus «prometteurs», c'est-à-dire de plus grande hyperbolicité.

Une première idée est d'utiliser le lemme 1 (p. 7). Nous allons voir que ce lemme montre qu'en visitant les paires de sommets par ordre de distance décroissante dans le graphe, une borne supérieure sur l'hyperbolicité peut être calculée. Cette observation a permis d'élaborer un nouvel algorithme pour le calcul de la valeur exacte de l'hyperbolicité. C'est un algorithme simple qui peut être facilement implémenté, testé et qui donne de bons résultats en pratique (voir section 2). Dans un premier temps, je décris formellement ce

nouvel algorithme. Puis je précise certains aspects concernant sa complexité temporelle et explique comment l'utiliser pour approcher la valeur de l'hyperbolicité d'un graphe. Enfin, je propose une version parallèle de l'algorithme permettant de réduire encore d'avantage les temps d'exécution.

1.1 Algorithme pour le calcul de l'hyperbolicité

L'idée de l'algorithme est de parcourir les quadruplets en commençant par ceux pouvant fournir la plus grande valeur d'hyperbolicité. Plus précisément, nous parcourons les quadruplets de façon à ce que le quadruplet a, b, c, d soit testé avant le quadruplet a', b', c', d' si $\min\{\mathtt{d}(a,b), \mathtt{d}(c,d)\} > \min\{\mathtt{d}(a',b'), \mathtt{d}(c',d')\}$. Ceci repose sur le lemme 1 (p. 7) donné en chapitre 1 qui montre que $\delta \leq D/2$, avec D le diamètre du graphe.

Notons `paires` la liste des $\binom{n}{2}$ paires de sommets triée par ordre de distance décroissante, nous obtenons l'algorithme 1 :

ALGORITHME 1 : Hyperbolicité

Données : G un graphe biconnexe.
Données : `paires` la liste des $\binom{n}{2}$ paires triée par distance décroissante.
Résultat : δ, l'hyperbolicité de G (notez que $2\delta = \mathrm{h}^*$).

```
1 Soit h* := 0;
2 pour 1 ≤ i < (n 2) faire
3     (a, b) := paires[i];
4     pour 0 ≤ j < i faire
5         (c, d) := paires[j];
6         h* := max {h*, δ(a, b, c, d)};
7         si d(a, b) ≤ h* alors
8             ALLER À LA ligne 9
9 retourner h*/2
```

Grâce au lemme 1 (p. 7), à chaque étape de l'algorithme, $\mathtt{d}(c,d)$ est une borne supérieure de la valeur de $\delta(a,b,c,d)$. Si la borne inférieure courante est δ^*, aucun quadruplet tel que $\mathtt{d}(c,d) \leq \delta^*$ ne peut être utilisé pour améliorer la borne inférieure. L'exploration des quadruplets est ainsi coupée. Comme notre algorithme peut avoir à parcourir les $\binom{n}{2}$ paires de sommets (notamment lorsque G est chordal), sa complexité dans le pire des cas est en $O(n^4)$. L'exécution de notre algorithme requiert le calcul de la matrice des distances des plus courts chemins et le tri des paires de sommets par ordre de distances décroissantes (ceci revient à associer à chaque entier ℓ la liste des paires de sommets à distance ℓ). Ces opérations ont respectivement une complexité temporelle en $O(n(n+m))$ et $O(n^2)$ pour des graphes à n sommets et m arêtes.

La complexité théorique de l'algorithme peut être paramétrée par la valeur optimale de l'hyperbolicité et la distribution des distances. Notons P_ℓ le nombre de paires (a,b) telles que $\mathtt{d}(a,b) = \ell$. L'algorithme ne considère que les paires de paires $((a,b),(c,d))$ telles que $D \geq \mathtt{d}(c,d) = \ell_2 \geq \mathtt{d}(a,b) = \ell_1 \geq 2\delta$. En particulier, pour les grilles $n \times n$ où l'hyperbolicité est donnée par le quadruplet $(0,0),(0,n-1),(n-1,0),(n-1,n-1)$, l'algorithme termine après avoir testé un seul quadruplet.

Proposition 2. *Étant donné un graphe δ-hyperbolique de diamètre D avec P paires de sommets à distance 2δ ou plus, la complexité de l'algorithme 1 est en $O(P^2)$.*

Démonstration. Comme la liste `paires` des paires de sommets est triée par ordre décroissant de distances, et comme l'algorithme 1 utilise le lemme 1, il ne considère que les paires de paires de sommets $((a,b),(c,d))$ telles que $D \geq \mathtt{d}(c,d) \geq \mathtt{d}(a,b) \geq 2\delta$. □

Prenons comme exemple de graphe en entrée de notre algorithme une grille $n \times n$, avec diamètre $2n-2$ et hyperbolicité $\delta = n-1$ (donc $\mathrm{h}^* = 2n-2$), la valeur de l'hyperbolicité sera obtenue à partir du premier quadruplet considéré et l'exécution de l'algorithme sera immédiatement arrêtée. Cependant, si le graphe en entrée est une grille $n \times 2$, avec diamètre n et hyperbolicité $\delta = 1$ (donc $\mathrm{h}^* = 2$), presque tous les quadruplets seront considérés.

Notons qu'à chaque étape de l'algorithme, $\mathbb{d}(c,d)/2$ est une borne supérieure et $\delta^*/2$ une borne inférieure. Nous pouvons donc obtenir une approximation du résultat à un facteur multiplicatif ou à une constante additive près en stoppant l'exécution de l'algorithme lorsque l'approximation demandée est prouvée. Plus précisément, nous pouvons ajouter l'une des affirmations suivantes à la ligne 6 de l'algorithme 1.

— «Si le temps de calcul écoulé est supérieur au temps de calcul autorisé, alors arrêter les calculs et retourner $\mathrm{h}^*/2$ et $\mathbb{d}(a,b)/2$». Nous obtenons alors $\mathrm{h}^* \leq 2\delta \leq \mathbb{d}(a,b)$.

— «Si $\mathbb{d}(a,b) \leq apx \cdot \mathrm{h}^*$, alors arrêter les calculs et retourner $\mathrm{h}^*/2$». Cela amène à une approximation de la valeur δ de l'hyperbolicité avec un facteur multiplicatif apx prouvé (i.e., $\mathrm{h}^* \leq 2\delta \leq apx \cdot \mathrm{h}^*$).

— «Si $\mathbb{d}(a,b) - \mathrm{h}^* \leq 2apx$, alors arrêter les calculs et retourner $\mathrm{h}^*/2$». Cela amène à une approximation de la valeur δ de l'hyperbolicité avec une constante additive apx prouvée (c'est-à-dire $\frac{\mathrm{h}^*}{2} \leq \delta \leq \frac{\mathrm{h}^*}{2} + apx$).

En fait, la majeure partie du temps d'exécution de l'algorithme consiste à réduire l'écart entre la borne inférieure, qui est généralement trouvée très rapidement, et la borne supérieure (voir section 2).

1.2 Utilisation des paires éloignées

Comme vu en section 2.1 (p. 9) du chapitre 1, il est possible de réduire le nombre de quadruplets à visiter en ne calculant que ceux issus des paires éloignées du graphe. Pour quelques classes particulières de graphes, la complexité temporelle du nouvel algorithme 1 est petite (excluant le temps de calcul de la matrice des distances des plus courts chemins et des paires éloignées, le tout étant calculé en temps $O(nm)$).

Proposition 3. *Le temps de calcul de l'algorithme 1 à partir des* ***paires éloignées*** *est en :*

1. $O(1)$ *quand* G *est une grille* $p \times q$ *avec* $p, q \geq 2$ *;*
2. $O(n^2)$ *quand* G *est un cycle d'ordre* $n \geq 4$.

Dans le cas de la grille, il n'existe que deux paires de sommets éloignés formées par les sommets situés aux quatre coins de la grille. Le calcul de l'hyperbolicité est donc immédiat. Dans le cas du cycle d'ordre n, avec $\{i \mid 0 \leq i < n\}$ l'ensemble de ses sommets, une paire éloignée sera formée des deux sommets à indices $i = 0$ et $j = \frac{n}{2}$. L'ensemble des paires éloignées sera donc $L_{2p} = \{(i, i+p) \mid 0 \leq i < p \leq n\}$ résultant en une complexité temporelle en $O(n^2)$.

1.3 Algorithme parallèle pour le calcul de l'hyperbolicité

Comme nous le verrons en section 2, l'algorithme 1 permet un gain significatif du temps de calcul de l'hyperbolicité. Ces performances seront comparées à l'algorithme proposé en [FIV12] qui a une complexité temporelle en $O(n^{3.69})$ mais aussi à l'implémentation massivement parallèle utilisée en [ASHM13]. Cependant, les temps de calcul restent importants sur certaines instances de graphes comme celles des cartes CAIDA les plus grandes. Je propose donc une version parallèle de l'algorithme 1 pour laquelle d'importantes améliorations en terme de temps de calcul ont été obtenues.

Observons tout d'abord que toutes les opérations effectuées par l'algorithme 1 sont indépendantes les unes des autres, exceptées celles pour la mise à jour de la borne inférieure h^* qui est rarement nécessaire. Ainsi, il est possible de se baser, tout comme dans [ASHM13], sur un outil de parallélisation automatique des boucles tel que OpenMP [Opeb] pour distribuer les calculs sur plusieurs processeurs ou cœurs. Je présente dans l'algorithme 2 les instructions OpenMP données pour paralléliser la première boucle (ligne 2), pour déclarer la mise à jour de la variable h^* comme critique (ligne 9) et propager la nouvelle borne (ligne 10).

ALGORITHME 2 : Algorithme parallèle

Données : G un graphe biconnexe.
Données : `paires` la liste des $\binom{n}{2}$ paires triée par distance décroissante.
Résultat : δ, l'hyperbolicité de G (notez que $2\delta = \mathrm{h}^*$).

```
1  Soit h* := 0;
2  #pragma omp parallel shared(h*, paires) private(h, j)
3      pour 1 ≤ i < (n 2) faire
4          (a, b) := paires[i];
5          pour 0 ≤ j < i faire
6              (c, d) := paires[j];
7              Soit h := δ(a, b, c, d);
8              si h < h* alors
9                  #pragma omp critical;
10                 #pragma omp flush(h as h*);
11             si d(a, b) ≤ h* alors
12                 ALLER À LA ligne 14
13 #pragma omp barrier;
14 retourner h*/2
```

2 Résultats expérimentaux

Dans la suite de ce chapitre, les résultats expérimentaux de l'algorithme 1 ont été obtenus à partir d'une implémentation en langage C de l'algorithme 1 accessible depuis [Cou14]. Le calcul des plus grands substituts ou LS (Largest Substitute) se base sur l'implémentation Java que j'ai développé et qui est disponible depuis [Lana]. Deux autres implémentations, qui n'ont pas été utilisées pour le calcul de l'algorithme 1 sont néanmoins disponibles. L'une d'elle peut être obtenue à partir de la suite mathématique libre Sage [S+] et l'autre que j'ai développé en Java à partir de la librairie Grph [H+14] est disponible depuis [Lanb]. Ainsi, ces expérimentations peuvent être reproduites par quiconque le souhaite. Tous les calculs ont été effectués sur des calculateurs équipés de 2 processeurs hexa-core 2.93GHz Intel Xeon X5670 et de 96 GB de RAM. Excepté pour la section 1.3, tous les calculs ont été effectués sur un seul cœur.

La plupart des comparaisons faites dans ce chapitre sont en termes de nombre de *quadruplets visités*, c'est-à-dire le nombre de quadruplets pour lequel $\delta^*(a, b, c, d)$ a été effectivement calculé. Ce nombre est indépendant de la qualité de l'implémentation de l'algorithme et du calculateur utilisé. Il peut ainsi être réutilisé pour comparer différentes méthodes de calcul de l'hyperbolicité. Les temps de calcul sont donnés en seconde et peuvent certainement être améliorés à partir d'une meilleure implémentation ou de calculateurs plus puissants. Néanmoins, ces temps de calcul donnent une idée de l'ordre de grandeur des gains obtenus par l'utilisation de l'algorithme 1 en comparaison avec d'autres implémenta-

tions. Plus particulièrement, seuls deux algorithmes de la littérature ont pu être comparés à l'algorithme 1.

Algorithme massivement parallèle [ASHM13]. Les auteurs ont utilisés OpenMP [Opeb] pour calculer en parallèle les boucles de l'algorithme de base en $O(n^4)$ sur une machine prenant en charge jusqu'à 1 015 cœurs. Remarquons que l'objectif principal de leur étude était de comparer les performances des modèles multi-tâches et de partage du travail avec OpenMP, utilisant le calcul de l'hyperbolicité seulement comme un cas d'étude. Cependant, il est intéressant de comparer les temps d'execution donnés dans [ASHM13] avec ceux de l'algorithme 1. Le nombre de quadruplets total et les temps de calcul pour les graphes évalués dans [ASHM13] sont donnés pour l'algorithme de base en $O(n^4)$.

Algorithme en $O(n^{3.69})$ [FIV12]. Aucune implémentation de cet algorithme ne semble disponible dans la littérature. Son temps de calcul a dû être estimé en se basant sur le nombre d'opérations par seconde pouvant être effectué par le calculateur. Afin de ne pas sous-estimer les performances de cet algorithme, le nombre d'opérations a été surévalué à 10^{10} par seconde. Le temps de calcul total pour cet algorithme est donc estimé à $n^{3.69}/10^{10}$ secondes. Cela semble particulièrement équitable puisque lors des expérimentations, l'algorithme 1 n'a jamais évalué plus de 300 millions de quadruplets par seconde.

2.1 Méthodes de pré-traitement des graphes

Comme détaillé en chapitre 1, deux méthodes de pré-traitement du graphe sont employées pour réduire la taille des instances lors du calcul de l'hyperbolicité.

Décomposition en composantes biconnexes. Une première méthode consiste à calculer les composantes biconnexes d'un graphe G et d'appliquer ensuite l'algorithme 1 sur ses composantes. En effet, à partir du théorème 4 (p. 11) donné dans le chapitre 1, nous savons que l'hyperbolicité d'un graphe est la valeur maximale de l'hyperbolicité parmi toutes ses composantes biconnexes. Les grandes étapes de calcul de l'hyperbolicité peuvent donc se résumer ainsi :

1. le graphe G est décomposé en composantes biconnexes ;
2. les composantes biconnexes sont triées par nombre de sommets décroissant ;
3. l'algorithme 1 est exécuté de manière séquentiel sur chacune des composantes biconnexes. Si le diamètre D_H de la composante H est inférieur ou égal à la valeur maximale h^* calculée à partir des composantes précédentes, l'exécution est arrêtée (rappelons que $\delta(H) \leq D_H/2$).

Observons, que lors des expérimentations, la valeur maximale de l'hyperbolicité $\delta(G)$ est toujours trouvée à partir de la plus grande composante biconnexe du graphe (LBC).

Décomposition par les cliques-minimales-séparatrices. La seconde méthode pour le pré-traitement du graphe se base sur le calcul des substituts des atomes obtenu à partir de la décomposition par les cliques-minimales-séparatrices présentée dans le chapitre 1. En effet, l'hyperbolicité de G est la valeur maximale de l'hyperbolicité calculée à partir des substituts. Cette méthode a donc été appliquée lors des expérimentations à chacune des composantes biconnexes du graphe G dont le diamètre D_H est inférieur ou égal à la valeur maximale h^* déjà calculée. La complexité temporelle de cette seconde méthode est en $O(nm)$, ce qui est identique au calcul de la matrice des distances des plus courts chemins. Rappelons que cette décomposition par les cliques-minimales-séparatrices ne peut être utilisée que pour des graphes tels que $\delta(G) \geq 1$. Cela sera toujours le cas dans les expérimentations présentées

ci-après. De même qu'observé pour les composantes biconnexes, $\delta(G)$ est toujours obtenu à partir du premier et plus grand substitut LS.

2.2 Évaluation des performances de l'algorithme

Afin de mieux comprendre le comportement de l'algorithme 1, l'hyperbolicité de graphes de différentes origines et de différentes tailles a été calculée.

— Les cartes des relations entre les systèmes autonomes de l'Internet. Ces graphes ont été collectés par l'association de coopération pour l'analyse des données de l'Internet (CAIDA), de janvier 2000 à novembre 2013 [CAI13b] et les graphes du projet pour les simulations et mesures distribuées de l'Internet (DIMES) depuis 2010 à 2012 [SS05].

— 5 réseaux de collaboration de différentes communautés scientifiques, `ca-AstroPh` pour le graphe de la communauté en astrophysique, `ca-CondMat` pour le graphe de la communauté en physique de la matière condensé, `ca-GrQc` pour le graphe de la communauté de la relativité générale et de la cosmologie quantique, `ca-HepPh` pour le graphe de la communauté en phénoménologie de la physique des particules et `ca-HepTh` pour le graphe de la communauté en physique théorique des hautes énergies (voir [LKF07]).

— Le graphe `Gnutella09` représentant une prise de vue du 9 août 2002 du réseau pair-à-pair de partage de fichiers Gnutella (voir [LKF07]).

— Le réseau social de géolocalisation `loc-Brightkite`. Ce graphe, composé de 58 228 sommets est le plus grand pour lequel l'hyperbolicité a été calculée (voir [LKF07]).

Dans le tableau 2.1 sont seulement reportés les temps d'exécution obtenus dans [ASHM13, Table 2] pour le modèle multi-tâches sur 512 et 1015 cœurs. Les meilleurs résultats ayant toujours été obtenus à partir du modèle multi-tâches comparé au modèle de partage du travail. Les meilleurs temps d'exécution de l'algorithme 1 ont aussi été reportés pour les mêmes graphes mais à partir du LS. On observe très clairement le gain en terme de réduction du temps de calcul. Dans un premier temps, la méthode de construction des substituts permet de réduire la taille du graphe en entrée. Dans un second temps, la réduction de l'espace de recherche est opérée par l'algorithme 1 avec une réduction significative du nombre de quadruplets visités. Des gains de 3 à 6 ordres de magnitude ont été obtenus sur ces graphes en utilisant seulement un seul cœur.

Algorithme [ASHM13]	ca-GrQc	as20000102	Gnutella09
n	4 158	6 474	8 104
δ	3.5	2.5	3
Temps (en sec.) sur 512 cœurs	**1 916**	**9 039**	26 888
Temps (en sec.) sur 1015 cœurs	3 691	11 419	**13 295**
Algorithme 1	LS	LS	LS
n	2 107	2 991	5 606
Nb. total de quadruplets	$8.2\,10^{10}$	$3.3\,10^{12}$	$4.1\,10^{13}$
Quadruplets visités	$4.0\,10^{8}$	$2.3\,10^{8}$	$9.8\,10^{6}$
Temps (en sec.)	**1**	**0.57**	**0.028**

TABLE 2.1 – Comparaison des temps d'exécution de l'algorithme 1 avec [ASHM13].

Afin de mieux comprendre les performances de l'algorithme 1, je donne dans le tableau 2.2 pour la LBC de chaque graphe le nombre de sommets et le nombre $P_{=\ell}$ de paires distantes à distance ℓ. Suivant la Proposition 2 (p. 33), la complexité temporelle de l'algorithme dans le pire cas est en $O(P^2_{\geq 2\delta})$. Cependant, on peut observer sur un nombre important de graphes qu'un quadruplet (a, b, c, d) tel que $\delta(a, b, c, d) = \delta(G)$ est trouvé très

Algorithme 1	ca-GrQc				as20000102				Gnutella09	
	LBC		LS		LBC		LS		LBC = LS	
n	2 651		2 107		4 009		2 991		5 606	
Nb. total de quadruplets	$2.0\,10^{12}$		$8.2\,10^{10}$		$1.1\,10^{13}$		$3.3\,10^{12}$		$4.1\,10^{13}$	
ℓ	$P_{=\ell}$	$F_{=\ell}$	$P_{=\ell}$	$F_{=\ell}$	$P_{=\ell}$	$F_{=\ell}$	$P_{=\ell}$	$F_{=\ell}$	$P_{=\ell}$	$F_{=\ell}$
1	10 480	1 129	7 135	182	10 101	–	7 993	–	23 510	–
2	50 508	5 538	38 484	2 293	1 231 628	519 224	412 563	64 871	292 458	3
3	209 027	24 879	160 255	12 028	3 721 215	2 235 874	1 873 697	846 725	2 101 638	1 999
4	625 349	109 394	463 779	59 800	2 569 590	1 943 404	1 765 412	1 248 401	7 222 588	451 789
5	1 111 919	354 964	752 072	204 664	475 366	413 824	388 989	334 668	5 420 722	3 376 888
6	983 190	512 472	567 187	274 411	25 648	24 017	22 453	21 001	645 387	629 722
7	418 341	287 452	193 841	127 957	485	472	435	422	4 490	4 404
8	91 502	73 153	32 858	25 825	3	3	3	3	22	22
9	11 332	9 800	2 890	2 441	–	–	–	–	–	–
10	886	811	165	150	–	–	–	–	–	–
11	41	41	5	5	–	–	–	–	–	–
$P_{\geq 2\delta}$	522 102	371 257	229 759	156 378	501 502	438 316	411 880	356 094	649 899	634 148
$P_{\geq 2\delta+1}$	103 761	**83 805**	35 918	**28 421**	26 136	**24 492**	22 891	**21 426**	4 512	**4 426**
Quadruplets visités	$5.4\,10^{9}$	$3.5\,10^{9}$	$6.4\,10^{8}$	$4.0\,10^{8}$	$3.4\,10^{8}$	$3.0\,10^{8}$	$2.6\,10^{8}$	$2.3\,10^{8}$	$1.0\,10^{7}$	$9.8\,10^{6}$
Temps (en sec.)	13	9	1.58	1	0.85	0.74	0.65	0.57	0.029	0.028

TABLE 2.2 – Performances détaillées de l'algorithme 1.

tôt, l'algorithme finit donc dès que $\mathfrak{d}(a,b) = 2\delta(G)$ (ligne 8 de l'algorithme 1). Ainsi, dans les expérimentations présentées dans le tableau 2.1, le nombre de quadruplets visités est toujours $\binom{P_{\geq 2\delta+1}}{2}$. Cela explique pourquoi, en regardant la distribution des distances des plus courts chemins et l'hyperbolicité des graphes de `Gnutella09` et `ca-GrQc`, le temps d'exécution de l'algorithme est 400 fois plus petit sur `Gnutella09` que sur `ca-GrQc` alors que `Gnutella09` possède deux fois plus de sommets.

Dans le tableau 2.2 est aussi donné le nombre $F_{=\ell}$ de paires éloignées à distance ℓ dans la LBC de chacun des graphes. Ce nombre est très largement inférieur au nombre $P_{=\ell}$ de paires dans le graphe, ce qui est cohérent étant donné la définition des paires éloignées. L'espace de recherche est donc très largement réduit. Au final, on observe une réduction substantielle du nombre de quadruplets effectivement visités et donc du temps de calcul.

Considérons maintenant les calculs sur le LS de la décomposition en cliques-séparatrices pour chacun des graphes. Dans le tableau 2.2 sont donnés le nombre de sommets ainsi que le nombre de paires $P_{=\ell}$ et de paires éloignées $F_{=\ell}$ à distance ℓ. Cette méthode de pré-traitement du graphe permet de réduire encore davantage le temps de calcul excepté pour le graphe `Gnutella09` dont la taille du LS est exactement la même que celle de sa LBC. Pour le graphe `ca-GrQc`, en combinant la méthode de décomposition par les cliques-séparatrices et l'utilisation des paires éloignées, le nombre de quadruplets visités est 13 fois inférieur comparé au nombre visité sur l'ensemble des paires de la LBC. Concernant le graphe `as20000102`, le nombre de quadruplets visités est réduit d'un facteur 1.5.

2.3 Décomposition et calcul de l'hyperbolicité dans les grands graphes

Dans la suite de ce chapitre sont détaillées les expérimentations sur des instances de grands graphes obtenues depuis les cartes des systèmes autonomes de CAIDA [CAI13b] et DIMES [SS05], les réseaux de collaboration (`ca-*`) [LKF07] et un réseau social de géolocalisation (`loc-brightkite`) [CML11]. Pour chacun des graphes considérés sont donnés dans le tableau 2.3 :

— le nombre n de sommets dans le graphe ;

— l'hyperbolicité δ ;
— le nombre de sommets n_B dans la plus grande composante biconnexe ;
— le nombre de sommets n_A dans le LS ;
— le nombre total de quadruplets (Tot.) dans le LS ;
— le nombre de quadruplets visités (Vis.) par l'algorithme 1 dans le LS ;
— le temps d'exécution T_{seq} en seconde de l'algorithme 1 exécuté sur un cœur ;
— les temps d'exécutions parallèles T_4, T_8, and T_{16}, détaillés en section 1.3.

			Nombre de sommets			Quadruplets		Temps de calcul (en sec.)			
Nom du graphe		δ	n	n_B	n_A	Tot.	Vis.	T_{seq}	T_4	T_8	T_{16}
CAIDA	05/01/2004	2.5	16 301	10 424	7 963	$1.7\,10^{14}$	$\mathbf{2.2\,10^{10}}$	57	25	13	7
	07/07/2004	2	17 306	11 100	8 568	$2.2\,10^{14}$	$\mathbf{4.5\,10^{12}}$	14 442	5 606	3 726	2 242
	05/09/2005	3	20 344	12 957	9 751	$3.8\,10^{14}$	$\mathbf{9.0\,10^{8}}$	2	1	0.5	0.5
	20/01/2010	2	33 508	20 940	17 057	$3.5\,10^{15}$	$\mathbf{5.6\,10^{13}}$	201 795	166 101	48 852	38 571
	16/01/2011	2	36 878	23 214	18 934	$5.4\,10^{15}$	$\mathbf{4.7\,10^{13}}$	185 191	67 148	41 836	38 543
	01/01/2012	2	40 109	25 614	20 768	$7.7\,10^{15}$	$\mathbf{1.0\,10^{14}}$	334 078	156 973	88 799	84 902
	01/06/2012	2	41 203	25 815	21 196	$8.4\,10^{15}$	$\mathbf{1.1\,10^{14}}$	352 480	152 552	95 575	89 513
	01/01/2013	2.5	43 274	27 454	22 347	$1.0\,10^{16}$	$\mathbf{4.0\,10^{11}}$	1 563	449	235	135
	01/06/2013	2.5	44 611	28 654	23 331	$1.2\,10^{16}$	$\mathbf{1.0\,10^{14}}$	386 574	164 369	114 896	86 990
	01/11/2013	2.5	45 427	29 432	23 978	$1.4\,10^{16}$	$\mathbf{1.1\,10^{14}}$	363 770	151 634	98 518	90 419
DIMES	12/2010	2	26 235	18 764	14 930	$2.0\,10^{15}$	$\mathbf{7.0\,10^{11}}$	2 171	825	436	246
	10/2011	2	25 683	17 137	13 366	$1.3\,10^{15}$	$\mathbf{7.5\,10^{10}}$	214	89	45	23
	04/2012	2	25 367	16 907	13 217	$1.3\,10^{15}$	$\mathbf{2.3\,10^{11}}$	717	299	137	75
loc-brightkite		3	58 228	33 187	32 040	$4.4\,10^{16}$	$\mathbf{1.9\,10^{12}}$	7 710	3 274	1 424	858
ca-AstroPh		3	18 772	15 929	13 407	$1.3\,10^{15}$	$\mathbf{7.8\,10^{9}}$	22	9	5	3
ca-CondMat		3.5	23 133	17 234	13 643	$1.4\,10^{15}$	$\mathbf{3.6\,10^{10}}$	101	42	24	12
ca-GrQc		3.5	5 242	2 651	2 107	$8.2\,10^{12}$	$\mathbf{4.0\,10^{8}}$	1	0.4	0.2	0.1
ca-HepPh		3	12 008	9 025	7 170	$1.1\,10^{14}$	$\mathbf{1.9\,10^{10}}$	50	31	11	6
ca-HepTh		4	9 877	5 898	5 236	$3.1\,10^{13}$	$\mathbf{1.2\,10^{8}}$	0.3	0.1	0.07	0.04

TABLE 2.3 – Performances des algorithmes sur des graphes de grande taille.

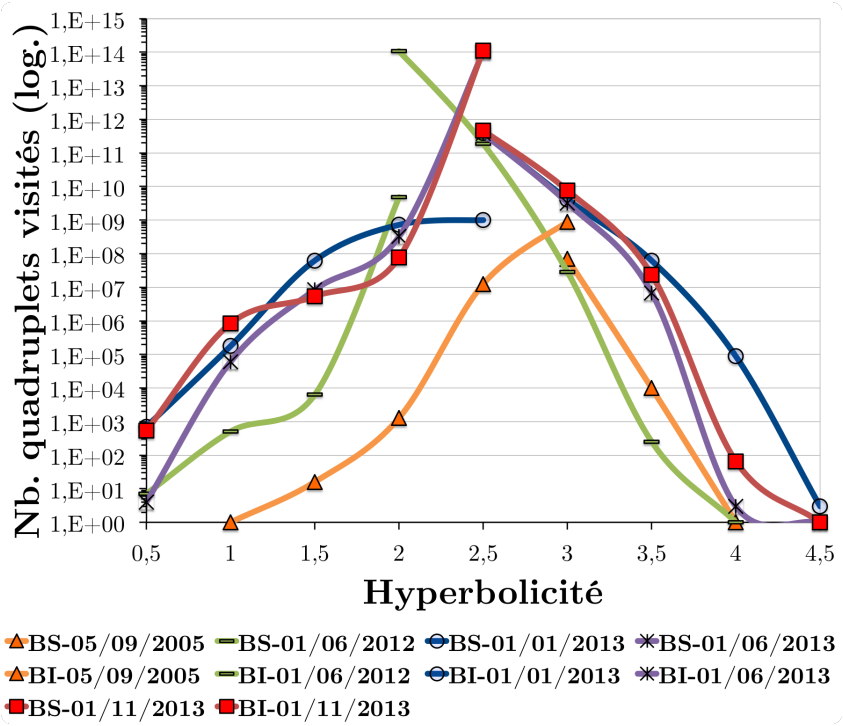
Analyse générale du nombre de quadruplets visités. Pour le graphe de `loc-brightkite`, l'algorithme visite moins de quadruplets que pour la carte CAIDA du 07/06/2004 alors que cette dernière a pourtant quatre fois moins de sommets. D'importantes variations en terme de nombre de quadruplets visités sont également observées entre graphes de même taille et de même hyperbolicité. Trois fois moins de quadruplets sont visités dans la carte DIMES du 04/2012 comparée à celle du 10/2011. Ces différences peuvent être expliquées à partir de la distribution des distances des plus courts chemins des paires éloignées. La carte du 04/2012 a un nombre supérieur de paires à distance $\geq 2\delta + 1$ que la carte du 10/2010.

De plus, excepté pour les cartes CAIDA du 05/09/2005, 01/06/2013, 01/11/2013 et DIMES du 12/2010, le nombre de quadruplets visités par l'algorithme 1 sur le LS est égal à $\binom{F_{\geq 2\delta+1}}{2}$. Nous noterons $F_{\geq \ell}$ le nombre de paires distantes à distance au moins ℓ. Cela montre qu'un quadruplet tel que $\delta(a,b,c,d) = \delta$ est trouvé avant que l'algorithme commence à traiter une paire éloignée (a',b') telle que $\mathtt{d}(a',b') = 2\delta$. En effet, dans l'algorithme 1, $\mathtt{d}(a,b)$ est une borne supérieure sur δ, et $\mathrm{h}^*/2$ est la borne inférieure courante. Dès qu'une telle paire (a',b') est traitée, l'optimalité de la solution est prouvée, l'algorithme finit et

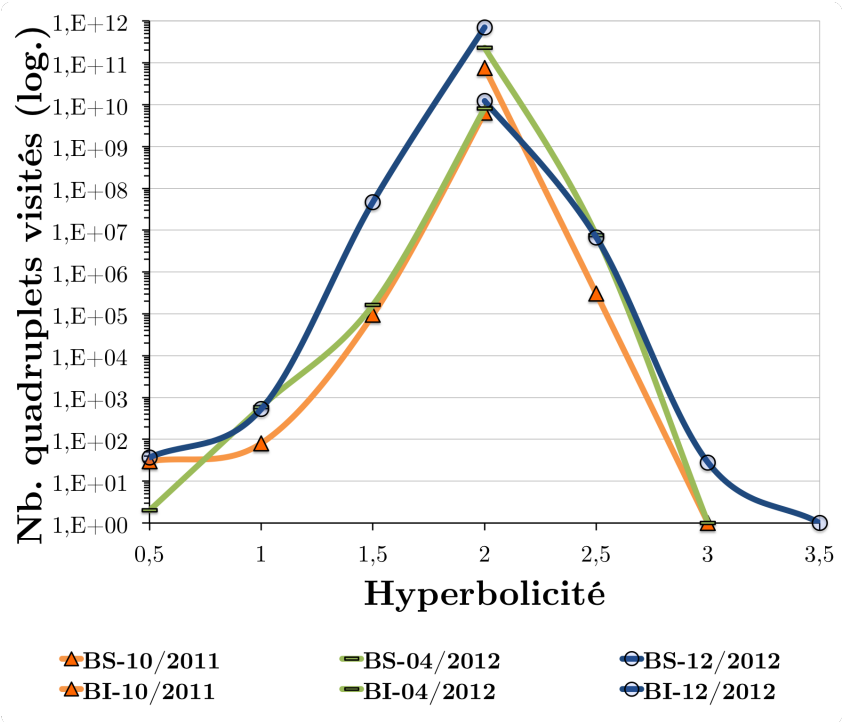
retourne le résultat. Cela implique que seuls les quadruplets tels que $\mathbb{d}(a,b) \geq 2\delta + 1$ sont visités.

Analyse des bornes inférieures et supérieures. Dans les figures 2.1 (p. 41), le nombre de quadruplets visités est donné pour quelques graphes choisis quand les bornes inférieures et supérieures sont trouvées. Rappelons que dans l'algorithme 1, $\mathbb{d}(a,b)$ est la borne supérieure courante de δ et $\mathrm{h}^*/2$ est la borne inférieure courante. Ainsi, dans la figure 2.1b (p. 41), la borne inférieure des cartes DIMES du 10/2011 et du 04/2012 croît à $\mathrm{h}^*/2 = 2$ avant que la borne supérieure puisse être réduite à 2. Le nombre de quadruplets visités pour ces graphes est donc $\binom{F_{\geq 2\delta+1}}{2}$. Cependant, pour la carte DIMES du 12/2010, la borne supérieure est réduite à 2 avant qu'un quadruplet tel que $\delta(a,b,c,d) = 2$ soit trouvé. Le nombre de quadruplets visités est bien plus important dans ce cas. Le même comportement est observé pour les cartes CAIDA du 05/09/2005, 06/01/2013 et du 01/11/2013 dans la figure 2.1a (p. 41). Cela explique par exemple l'importante différence de quadruplets visités et de temps de calcul entre les cartes CAIDA du 01/01/2013 et du 01/06/2013. Le nombre de sommets de leur LS ne diffère que de 5% (voir tableau 2.3), les deux cartes ont une même hyperbolicité de 2.5, environ $1.6\,10^7$ de paires distantes à distance $\geq 2\delta = 5$ et $8.9\,10^5$ à distance $\geq 2\delta + 1 = 6$. Ces deux cartes ont donc une structure similaire. Dans le cas de la carte du 01/01/2013, l'algorithme 1 est capable de trouver un quadruplet tel que $\delta(a,b,c,d) = 2.5$ à partir d'une paire distante (a,b) avec $\mathbb{d}(a,b) \geq 6$. Pour la carte du 01/06/2013, les paires distantes à distance $\mathbb{d}(a,b) \geq 5$ doivent être visitées. Cela représente au final 10^{13} quadruplets supplémentaires à visiter pour la carte du 01/06/2013.

Dans la figure 2.1 (p. 41), les bornes inférieures et supérieures se rejoignent très rapidement pour ne laisser qu'un petit intervalle. On peut donc utiliser l'algorithme comme une heuristique et borner le temps d'exécution ou le nombre de quadruplets à visiter comme expliqué en section 1. Il en résultera de très bonnes approximations. Pour les graphes donnés en figure 2.1 (p. 41), une heure de calcul suffit pour que soit prouvée l'optimalité de la solution ou pour retourner la solution optimale additive à 1/2 près.



(a) Cartes CAIDA depuis 2004.



(b) Cartes DIMES depuis 2010.

FIGURE 2.1 – Nombre de quadruplets visités pour atteindre la borne inférieure et supérieure. Le nombre de quadruplets pour atteindre la borne inférieure est tracé de gauche à droite, celui pour atteindre la borne supérieure de droite à gauche.

2.4 Coût du pré-traitement du graphe pour le calcul de l'hyperbolicité

Afin de réduire la taille des instances de graphe, et donc réduire le nombre de quadruplets à visiter, il est nécessaire de calculer une version modifiée des atomes. Les temps de calcul de l'hyperbolicité sur le LS et sur la LBC sont comparés dans le tableau 2.3. Sur la plupart des graphes étudiés, ce temps de calcul supplémentaire est entièrement compensé par la réduction du temps de calcul de l'hyperbolicité sur le LS.

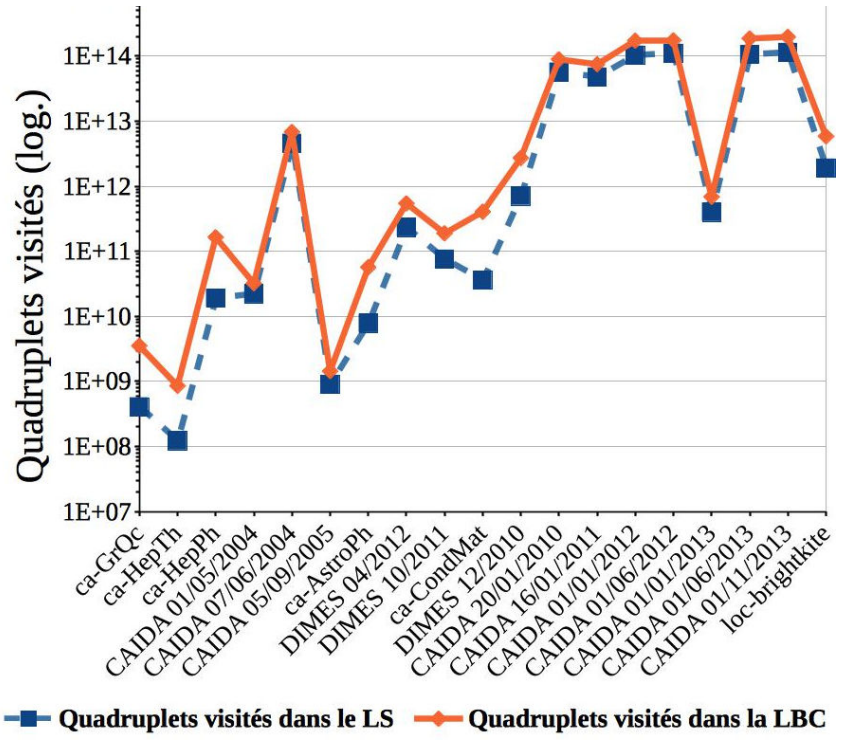
Le nombre de quadruplets visités lors du calcul de l'hyperbolicité sur le LS et la LBC est donné en figure 2.2a (p. 43). Dans cette figure, les graphes sont ordonnés par ordre croissant en fonction de leur nombre de sommets dans la LBC (voir tableau 2.3). Pour chacun des graphes, l'algorithme 1 visite au minimum 10 fois moins de quadruplets dans le LS que dans la LBC, même dans le cas des cartes CAIDA du 05/09/2005 et DIMES du 12/2012. Cela s'explique simplement à partir du tableau 2.3 par la réduction significative du nombre de sommets entre le LS et la LBC des graphes.

Les temps de calcul du LS et de son hyperbolicité sont donnés en figure 2.2b (p. 43). Leur somme et le temps de calcul de l'hyperbolicité sur la LBC sont visibles séparément afin de faciliter leur comparaison. Un temps de calcul non négligeable est requis pour le calcul du LS. Dans quelques cas, il peut dépasser le temps de calcul de l'hyperbolicité. Cela arrive notamment pour les graphes `ca-HepTh`, `ca-AstroPh` et les cartes CAIDA du 05/01/2004 et du 05/09/2005. De plus, le temps de calcul de l'hyperbolicité dans la LBC de ces quatre graphes est plus faible que dans le LS. Cependant, dans cette étude, les graphes de petite taille sont les seuls cas où calculer directement l'hyperbolicité dans la LBC est plus rapide que calculer le LS puis son hyperbolicité. L'impact est donc minime puisque le temps de calcul reste très faible quelle que soit la méthode utilisée. Concernant les graphes restants, comme on peut le voir en figure 2.2b (p. 43), la somme des temps de calcul du LS et de son hyperbolicité est généralement plus faible que le temps de calcul de l'hyperbolicité sur la LBC. Seule 1 h 38 min a été nécessaire pour calculer le LS de `loc-brightkite` et 2 h 8 min son hyperbolicité, pour un total de 3 h 46 min quand le calcul de l'hyperbolicité sur la LBC dure 6 h 28 min. Utiliser la méthode de pré-traitement pour calculer le LS a permis de réduire le temps de calcul de 3 heures. Concernant la carte CAIDA du 01/06/2013, le calcul du LS a duré 21 minutes et son hyperbolicité 5 jours quand 11 jours ont été requis pour calculer l'hyperbolicité de la LBC. Le temps de calcul a été divisé par 2 avec un gain de 6 jours.

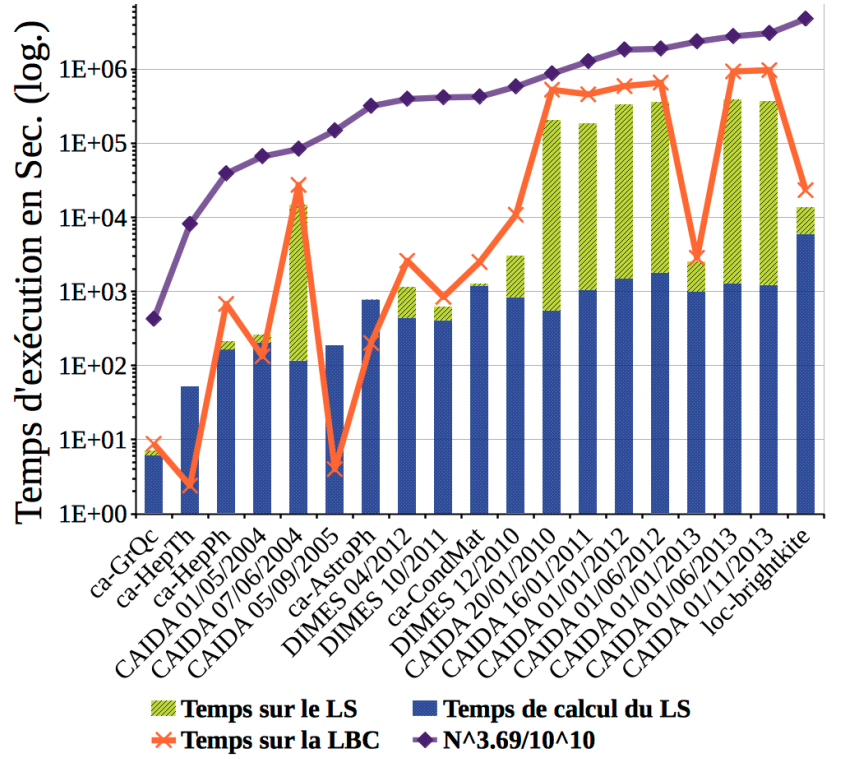
Comparons maintenant les résultats expérimentaux de l'algorithme 1 avec l'algorithme théorique proposé dans [FIV12] d'une complexité temporelle en $O(n^{3.69})$. Les temps de calcul de ce dernier ont été estimés à $n^{3.69}/10^{10}$ secondes et tracés en figure 2.2b (p. 43). Notons que l'algorithme 1 ne visite pas plus de $3.0\,10^8$ quadruplets par seconde. Même avec une telle estimation, l'algorithme 1 reste significativement plus rapide que celui proposé dans [FIV12].

2.5 Évaluation des performances de l'algorithme parallèle

Les résultats sont donnés dans le tableau 2.3 (p. 39) pour le LS de chaque graphe. Les temps de calcul de l'algorithme 2 donnés en colonnes T_4, T_8 et T_{16} correspondent respectivement à l'utilisation de 4, 8 et 16 cœurs. Comparée à la version séquentielle, l'implémentation parallèle offre un gain de temps de calcul intéressant. Rappelons que le gain d'un algorithme parallèle est défini par $S_p = \frac{T_1}{T_p}$ et son efficacité par $E_p = \frac{T_1}{pT_p}$, avec T_i le temps d'exécution de l'algorithme sur p unités de calcul. Le gain moyen obtenu dans notre cas est de 2.13 avec 4 cœurs, 3.7 avec 8 cœurs et 4.3 avec 16 cœurs. Ces résultats sont moins bons que ce que l'on pourrait attendre. Une analyse plus complète du code pour optimiser la parallélisation de l'algorithme pourrait certainement amener à de meilleures performances.



(a) Nombre de quadruplets visités dans la LBC et dans le LS.



(b) Temps de calcul de l'hyperbolicité et du LS.

FIGURE 2.2 – Comparaison des temps de calcul de l'hyperbolicité sur le LS (incluant le temps de calcul du LS lui-même) et celui sur la LBC et le temps estimé donné dans [FIV12].

3 Heuristique pour le calcul de l'hyperbolicité

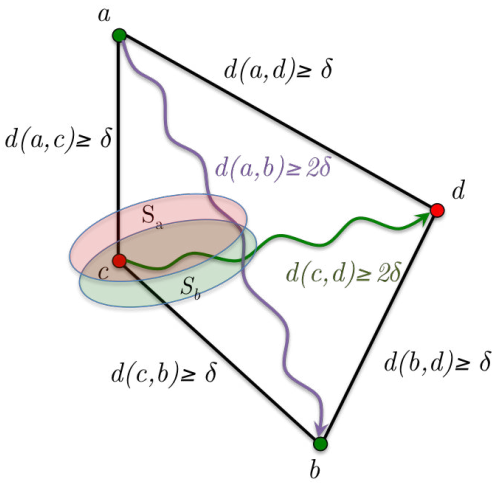


FIGURE 2.3 – Représentation du certificat de sommets (a, b, c, d) recherché par l'heuristique et maximisant l'hyperbolicité.

L'algorithme 1 permet une réduction significative des temps de calcul de l'hyperbolicité. Néanmoins, dans le cas de grands graphes, ces temps peuvent être conséquents. Une seconde limitation de l'algorithme est son empreinte mémoire. Considérons que la plus grande distance dans le graphe puisse être gardée en mémoire en un nombre constant de bits. Alors, le calcul de la matrice des plus courts chemins du graphe nécessite un espace mémoire en $O(n^2)$. Cette même limitation tient aussi pour l'algorithme proposé dans [FIV12]. Pour pallier à ce problème et pouvoir estimer l'hyperbolicité de graphes à plusieurs millions de sommets, je présente une nouvelle heuristique pour le calcul de l'hyperbolicité. Cette heuristique a une complexité spatiale en $O(n)$ et une complexité temporelle en $O(k^2(n+m))$ ou k est un paramètre de l'algorithme.

3.1 Conception de l'heuristique

Lors des expérimentations de l'algorithme 1, nous avons observé que le quadruplet de plus grande hyperbolicité contient deux paires de sommets (a, b) et (c, d) telles que $2\delta \leq \mathrm{d}(a, b) \leq \mathrm{d}(c, d)$. De plus, il a été prouvé dans [SG11] que $\delta(a, b, c, d) \leq \min_{x,y\in\{a,b,c,d\}} \mathrm{d}(x, y)$. En d'autres termes, $\delta(u, v, x, y)$ est petit si la distance entre les sommets de l'une des paires du quadruplet (a, b, c, d) est petite. En cherchant des quadruplets composés de deux paires de sommets (a, b) et (c, d) éloignés les uns des autres, on cible les quadruplets qui sont potentiellement de grande hyperbolicité. Cependant, il est nécessaire de trouver de tels quadruplets sans avoir à calculer et garder en mémoire la matrice des distances des plus courts chemins.

La figure 2.3 illustre la forme du certificat recherché, c'est-à-dire un quadruplet composé des sommets (a, b, c, d) maximisant l'hyperbolicité. Deux paires de sommets (a, b) et (c, d), dont les sommets sont les plus éloignés possibles, doivent être trouvées. Cependant, les sommes $S_2 = \mathrm{d}(a, c) + \mathrm{d}(b, d)$ et $S_3 = \mathrm{d}(a, d) + \mathrm{d}(b, c)$ doivent aussi être minimisées puisque $\delta = S_1 - \max\{S_2, S_3\}$ avec S_1 la plus grande des sommes. Ainsi, la première paire (a, b) doit être composée de sommets les plus éloignés possible. Une telle paire peut être calculée grâce à l'heuristique 2-sweep proposée dans [MLH09]. Une solution pour minimiser la somme S_2 est ensuite de choisir des sommets d à distance $\mathrm{d}(a, b)/2$ de a et b. Enfin, un sommet d le

plus éloigné de c doit être trouvé. Le meilleur sommet d sera celui maximisant la distance $\mathtt{d}(c,d)$ et minimisant une des sommes S_2 ou S_3.

Je propose une nouvelle approche reposant sur l'heuristique *2-sweep* permettant de trouver rapidement une paire de sommets les plus éloignés possible l'un de l'autre. Considérons un graphe $G(V,E)$, dans la suite, le choix aléatoire d'un sommet dans V sera toujours uniforme. Les différentes étapes de l'heuristique sont données dans l'algorithme 3.

ALGORITHME 3 : Heuristique pour le calcul de l'hyperbolicité

1. Choisir aléatoirement un sommet $x \in V$, calculer les distances des plus courts chemins à partir de x en utilisant un BFS. Soit a un des sommets le plus éloigné de x (c'est-à-dire le dernier sommet visité par le BFS). Calculer les distances du sommet a en utilisant un BFS. Soit b un des sommets le plus éloigné de a. Les sommets a et b sont donc choisis suivant l'heuristique 2-sweep. Enfin, soit S_a l'ensemble des sommets à distance $\mathtt{d}(a,b)/2$ de a ;
2. calculer les distances des plus courts chemins à partir de b en utilisant un troisième BFS et soit S_b l'ensemble des sommets à distance $\mathtt{d}(a,b)/2$ de b.
3. Pour au plus k sommets (choisis aléatoirement) $c \in S_a \cap S_b$, calculer les distances des plus courts chemins à partir de c avec un quatrième BFS. Pour chaque sommet d visité par le BFS calculer $\delta(a,b,c,d)$.
4. répéter les étapes précédentes k fois et retourner la plus grande valeur δ_h.

La complexité temporelle de cette heuristique est en $O(k^2(n+m))$. Notons que l'exploration au départ du sommet x, choisi aléatoirement, est arrêté si $\min\{\mathtt{d}(a,b),\mathtt{d}(a,c),\mathtt{d}(b,c)\} \leq \delta_h$, avec δ_h la meilleure solution trouvée jusqu'à présent. Le lemme 2 permet d'arrêter l'exploration si $\max\{\mathtt{d}(a,b),\mathtt{d}(a,c),\mathtt{d}(b,c)\} \leq 2\delta_h$. De plus, pour s'assurer que $S_a \cap S_b$ est suffisamment grand, un paramètre ε permet aux ensembles S_a et S_b de contenir des sommets à distance ℓ tels que $\mathtt{d}(a,b)/2 - \varepsilon \leq \ell \leq \mathtt{d}(a,b)/2 + \varepsilon$ de a et b. Cette heuristique utilise deux tableaux d'entiers de taille n pour garder en mémoire les distances de a à b. Un troisième tableau de taille n est requis pour garder en mémoire $S_a \cap S_b$. Les distances depuis c ne sont pas enregistrées. Au final, la complexité spatiale de l'heuristique est en $O(n)$.

3.2 Calculs expérimentaux de l'heuristique

Les résultats de l'évaluation de l'heuristique sont donnés dans le tableau 2.4. Les calculs ont été faits à partir de la LBC des graphes utilisés en section 2.3 pour les expérimentations de l'algorithme 1 ainsi que sur la LBC des trois réseaux routiers (`roadNet-*`) obtenus depuis [Lok09]. Pour chacun des graphes, le paramètre k a été initialisé à $k = 50$ répétant ainsi l'heuristique à partir de 50 sommets choisis aléatoirement. L'ensemble $S_a \cap S_b$ a été borné à 50 sommets. Le paramètre ε a été initialisé à $\varepsilon = 1$. La colonne $\langle\delta_h\rangle$ donne la valeur moyenne de l'hyperbolicité trouvée par l'heuristique sur 100 exécutions. La colonne $\hat{\delta}_h$ donne la meilleure valeur de l'hyperbolicité trouvée parmi toutes les exécutions. La colonne T_h donne les temps d'exécution moyens de l'heuristique.

Afin de confirmer les choix d'une première paire de sommets éloignés l'un de l'autre et d'un ensemble de sommets à mi-distance, l'heuristique est comparée aux résultats d'une heuristique aléatoire. Cette seconde heuristique choisit uniformément aléatoirement trois sommets a, b, et c et calcule ensuite la valeur $\delta(a,b,c,d)$ pour tous les sommets $d \in V \setminus \{a,b,c\}$. Ce processus aléatoire est répété jusqu'à ce que son temps de calcul égale le temps de calcul de la première heuristique. La plus grande valeur δ_r trouvée est ensuite retournée. La plus grande des valeurs parmi 100 exécutions est donnée dans la colonne $\hat{\delta}_r$ et la moyenne dans la colonne $\langle\delta_r\rangle$ du tableau 2.4.

Les résultats de l'heuristique sont comparés avec ceux obtenus par un autre processus

d'échantillonnage proposé dans [KNS13]. Cette autre heuristique extrait un échantillon de quadruplets jusqu'à ce qu'un certain niveau de confiance dans la valeur moyenne de l'hyperbolicité soit obtenu. Ensuite, la plus grande valeur δ_s est retournée. Les valeurs disponibles dans [KNS13] sont données dans la colonne δ_s du tableau 2.4, cependant aucune indication concernant la taille des échantillons ou les temps d'exécution n'est disponible.

Afin de compléter les informations du tableau 2.4, la taille de la plus grande composante biconnexe des graphes est donnée dans la colonne n_b ainsi que la valeur exacte de l'hyperbolicité δ quand celle-ci est connue (valeurs calculées à partir de l'algorithme 1).

Nom du graphe		n_b	δ	δ_s	$\hat{\delta}_h$	$\langle\delta_h\rangle$	$\hat{\delta}_r$	$\langle\delta_r\rangle$	T_h (en sec.)
CAIDA	05/01/2004	10 424	2.5	–	2	2	2	1.5	3.0
	07/06/2004	11 100	2	–	2	1.6	2	1.5	3.2
	05/09/2005	12 957	3	–	2.5	2.2	2	1.5	4.0
	20/01/2010	20 940	2	–	2	1.7	2	1.5	10.7
	16/01/2011	23 214	2	–	2	2	1.5	1.5	14.5
	01/01/2012	25 614	2	–	2	1.5	1.5	1.5	16.6
	01/06/2012	25 815	2	–	2	1.5	2	1.5	19.6
	01/01/2013	27 454	2.5	–	2.5	2.5	2	1.5	11.5
	01/06/2013	28 654	2.5	–	1.5	1.5	1.5	1.5	20.1
	01/11/2013	29 432	2.5	–	2	1.5	1.5	1.5	19.9
DIMES	12/2010	18 764	2	–	1.5	1.5	1.5	1.1	14.7
	10/2011	17 137	2	–	1.5	1.5	1.5	1.1	11.3
	04/2012	16 907	2	–	1.5	1.5	1.5	1.2	12.4
loc-brightkite		33 187	3	–	2.5	2.4	2	1.8	29.9
ca-AstroPh		15 929	3	2	2.5	2.5	2.5	2	39.8
ca-CondMat		17 234	3.5	2.5	3	3	2	2.3	14.9
ca-GrQc		2 651	3.5	3	3.5	3	3	2.6	1.1
ca-HepPh		9 025	3	2	3	3	2.5	2.1	12.7
ca-HepTh		5 898	4	3	3.5	3.3	3	2.6	2.5
roadNet-PA		863 105	–	195.5	166.5	158.5	158	143.8	329.2
roadNet-TX		1 050 434	–	222	225	203.3	221.5	205.4	371.9
roadNet-CA		1 563 362	–	208.5	220	218.7	223	202.9	582.4

TABLE 2.4 – Comparaisons des valeurs d'hyperbolicité trouvées avec la nouvelle heuristique proposée (colonne δ_h), le processus d'échantillonnage de [KNS13] (colonne δ_s) et d'une heuristique aléatoire (colonne δ_r)

3.3 Analyse des performances de l'heuristique.

Considérons tout d'abord dans le tableau 2.4 les graphes pour lesquels la valeur exacte de δ est connue. L'heuristique est capable de trouver la valeur exacte de l'hyperbolicité pour 8 d'entre eux et la valeur moyenne de l'heuristique est égale à celle de l'hyperbolicité pour 3 d'entre eux. En comparant les valeurs moyennes $\langle\delta_h\rangle$ avec $\langle\delta_r\rangle$, on observe l'impact positif de la sélection initiale faite par l'heuristique de paires de sommets les plus éloignés possible. En effet, la valeur $\langle\delta_h\rangle$ est généralement plus grande que $\langle\delta_r\rangle$. L'heuristique trouve aussi en moyenne de meilleures valeurs pour les graphes `ca-*` en comparaison avec δ_s. Ces résultats ont été obtenus avec un temps de calcul bien plus faible que ceux donnés dans le

tableau 2.3, obtenus à partir de l'algorithme 1. D'autant plus que les résultats du tableau 2.3 ont été calculés à partir de graphes ayant une taille inférieure à celle de la LBC. Notons que le temps d'exécution de l'heuristique sur `ca-AstroPh` est plus grand que pour les autres graphes de taille similaire. Cela s'explique par le degré moyen du graphe de 24 qui accroit le nombre d'arêtes à visiter lors de l'exécution des BFS en comparaison avec celui de la carte DIMES du `4/2012` qui a un degré moyen de 8. Quant à la carte CAIDA du `01/01/2013`, un plus faible temps d'exécution est obtenu grâce à un meilleur usage des règles de réduction de l'espace de recherche.

La valeur de l'hyperbolicité des réseaux routiers (`roadNet-*`) est inconnue. Seule une estimation obtenue à partir du processus d'échantillonnage dans [KNS13] nous permet de comparer les performances de notre heuristique. Un très bon résultat obtenu par l'heuristique est la valeur moyenne $\langle \delta_h \rangle$ calculée pour `roadNet-CA` qui est plus grande que celle de δ_s et $\langle \delta_r \rangle$. Cependant, la plus grande valeur, 223, a été obtenue par l'heuristique aléatoire. Pour ce graphe en particulier, cela indique que les quadruplets maximisant $\delta(a, b, c, d)$ peuvent ne pas impliquer de paires de sommets très éloignés les uns des autres. Une plus grande diversité dans le choix des paires, telle qu'obtenue par l'heuristique aléatoire, est donc nécessaire. Concernant `roadNet-PA`, en regardant la valeur $\hat{\delta}_h$, le résultat est encore plus mitigé puisque bien inférieur à δ_s. Cependant, l'heuristique est bien meilleure que celle basée sur le processus aléatoire. De plus, avec $\langle \delta_h \rangle > \hat{\delta}_r$, les quadruplets les plus prometteurs sont bien ceux impliquant des paires de sommets éloignés. Enfin, de meilleurs résultats sont obtenus à partir de l'heuristique aléatoire pour le graphe de `roadNet-TX`.

Globalement, les résultats obtenus par cette nouvelle heuristique sur les graphes des réseaux routiers `roadNet-*` sont bons pour une heuristique conçue pour cibler des quadruplets spécifiques à partir des observations faites sur les cartes CAIDA. D'autant plus que les cartes CAIDA ont des propriétés structurelles différentes. En particulier, une distribution des degrés suivant une loi de puissance avec un petit diamètre. Les graphes `roadNet-*` sont eux quasi-planaires avec une distribution des degrés suivant une loi de poisson et un grand diamètre (voir [KNS13]). Au final, de nouvelles bornes sur l'hyperbolicité de ces graphes ont été trouvées améliorant les résultats de [KNS13]. Cette heuristique est simple et obtient de bons résultats sur des graphes de très grande taille avec un temps d'exécution et une complexité spatiale linéaires.

4 Conclusion

Dans ce chapitre j'ai proposé un nouvel algorithme pour le calcul de l'hyperbolicité sur de grands graphes. Cet algorithme offre de bonnes performances et a permis le calcul de la valeur exacte de l'hyperbolicité sur des tailles de graphes encore jamais atteintes. J'ai ensuite analysé l'impact de la valeur de l'hyperbolicité et de la distribution des distances des plus courts chemins sur le temps de calcul de l'algorithme. Un gain significatif a pu être obtenu en combinant ce nouvel algorithme avec la méthode de décomposition de graphe par des cliques-séparatrices proposée dans le chapitre 1. Grâce à l'utilisation d'outils d'automatisation pour la parallélisation d'algorithmes il est maintenant possible de calculer la valeur exacte de l'hyperbolicité sur des graphes de plusieurs dizaines de milliers de sommets. Afin de pallier la nécessité de calculer la matrice des distances des plus courts chemins et atteindre des graphes à plusieurs millions de sommets, une heuristique rapide donnant de bons résultats a été proposée. Cette étude ouvre les portes à de nouvelles problématiques.

Optimisation de l'utilisation de la mémoire. Les gains obtenus à partir de la version parallèle de l'algorithme 1 ne sont pas satisfaisants et pourraient certainement être grandement améliorés. Par exemple, un placement attentif des données en mémoire ou encore l'ordre dans lequel les instructions sont exécutées, et donc l'ordre des éléments dans la

liste `paires`, pourraient particulièrement aider à l'amélioration des temps d'exécution de l'algorithme.

Réduction de l'espace de recherche. Les performances de l'heuristique pourraient aussi être améliorées. En particulier, une méthode rapide pour couper l'espace de recherche lors de la sélection du quatrième sommet d pourrait réduire le temps d'exécution de l'ensemble du processus.

Amélioration de la borne supérieure. Des outils pour évaluer la distance avec la valeur optimale (autre que la distance avec la borne supérieure, qui est $D_G/2$) aideraient à décider quand arrêter les calculs.

Deuxième partie

Simulation de modèles de routage dynamiques dans de grands réseaux

Chapitre 3

Introduction au routage dans l'Internet

Sommaire

1 Routage dans les réseaux 52
1.1 Le protocole IP 52
1.2 Architecture d'un routeur IP 52
2 Routage dans l'Internet 54
2.1 L'architecture d'Internet 54
2.2 Le routage intra-domaine 56
2.3 Le routage inter-domaine 57
2.4 Le protocole de routage BGP 59
3 Problématiques du routage dans l'Internet 63
3.1 Évolution de l'Internet 63
3.2 Conception de nouveaux systèmes de routage pour l'Internet 65
4 Conclusion 69

Le nombre toujours croissant de systèmes autonomes dans Internet pousse dans ses retranchements son architecture de routage et plus particulièrement le protocole de routeur frontière [RLH06] (ou BGP, Border Gateway Protocol). Ces dernières années, de nombreux travaux ont portés sur l'adaptation de BGP afin de permettre son passage à l'échelle. D'autres travaux étudient la conception de nouveaux schémas de routage afin de remplacer BGP.

Dans cette deuxième partie de thèse, mon travail se concentre sur le développement du simulateur DRMSim pour l'évaluation des performances de ces nouveaux schémas et leur comparaison avec BGP. Cependant, avant d'aborder le simulateur DRMSim et plus particulièrement les problématiques liées à la simulation à grande échelle, je rappelle dans ce chapitre les notions de routage et plus précisément de routage dans Internet. L'objectif étant de fournir au lecteur un exposé du fonctionnement actuel de l'architecture de routage d'Internet et des problématiques qui lui sont liées sans aller dans des détails qui ne sont pas utiles à la compréhension des prochains chapitres de cette thèse.

Je commence en section 1 par rappeler les bases du routage dans les réseaux et notamment le fonctionnement du protocole IP et des routeurs. En section 2, je présente l'architecture d'Internet, les relations économiques entre AS qui régissent son évolution et les notions de routage intra-domaine et inter-domaine. Dans cette section, je rappelle également les principes de BGP sans détailler toutes ses fonctionnalités. En section 3, j'énonce les principaux défis auxquels fait face la communauté scientifique pour maintenir l'architecture de routage d'Internet fonctionnelle. Enfin, toujours en section 3, j'expose la méthodologie proposée dans [DBI+11a] pour la conception de nouveaux schémas de routage. Les notions présentées et la terminologie utilisée dans cette méthodologie seront reprises dans les chapitres suivants. Finalement, je conclus en section 4 sur les perspectives d'évolution de l'architecture de routage d'Internet et les besoins de développement de nouveaux outils pour l'évaluation et la comparaison des performances de nouveaux schémas de routage.

1 Routage dans les réseaux

Les réseaux informatiques sont constitués d'un ensemble d'équipements, aussi appelés *nœuds*, reliés entre eux pour échanger des informations. Par exemple, un nœud du réseau peut être un ordinateur personnel, une imprimante, un routeur ou encore un serveur. Les nœuds situés au sein d'un même réseau peuvent communiquer directement entre eux à partir d'un protocole de la couche liaison du modèle OSI (ou Open Systems Interconnection model) (par exemple Ethernet [MB76]). Dans le cas de nœuds situés dans des réseaux différents, un itinéraire doit être calculé pour transmettre un message d'un nœud source vers un nœud destination. Cet itinéraire, ou *route*, est calculé par un nœud particulier appelé *routeur*. On distinguera donc deux types de nœuds :

— les *hôtes*, ils sont à l'origine des messages ou sont leur destination finale ;

— les *routeurs*, ils transmettent uniquement les messages entre les hôtes.

La topologie physique des liens interconnectant les réseaux entre eux définit les routes possibles pour acheminer un message entre les hôtes. Les routeurs permettant l'interconnexion des réseaux entre eux. Leur rôle et celui de leur protocole de routage étant de déterminer quel chemin prendre pour acheminer un message entre deux hôtes.

Dans Internet, afin de pouvoir identifier les réseaux, leurs nœuds et assurer la transmission des messages, le protocole IP (ou Internet Protocol [Pos81]) est utilisé. Dans la suite de ce chapitre, nous nous intéresserons donc uniquement au fonctionnement des réseaux IP.

1.1 Le protocole IP

Dans le protocole IP, un nœud d'un réseau se voit affecter une adresse IP qui l'identifie de manière unique dans son réseau. Cette adresse IP est codée sur 32 bits (dans la version 4 d'IP) que nous pouvons représenter sous un format décimal découpé en quatre octets A.B.C.D (ex. 138.96.0.1). Un réseau est donc constitué d'un ensemble d'adresses IP. Ces adresses IP sont allouées à un réseau par blocs contigüs. Elles partagent un certain nombre de premiers bits identiques. Le nombre de ces bits identiques forme le *masque réseau*. Un réseau est identifié par un *préfixe*. Ce préfixe est composé de la première adresse IP du bloc associée à son masque. Le préfixe d'un réseau peut être représenté au format CIDR (ou Classless Inter-Domain Routing), l'adresse IP et son masque sont séparés par une barre oblique inversée (ex : 138.96.0.0/16). Un préfixe peut identifier un réseau ne contenant que des hôtes. Il peut aussi être constitué de plusieurs sous-réseaux. Par exemple, le réseau 138.96.0.0/16 peut être subdivisé en deux sous-réseaux 138.96.0.0/17 et 138.96.128.0/17 pouvant contenir chacun 32766 hôtes ou être à leur tour divisés en sous-réseaux. Plus le masque est petit, plus le nombre de sous-réseaux ou d'hôtes dans le réseau est grand.

Dans le protocole IP, les unités d'informations transmises entre les routeurs sont appelées *datagrammes*. Ces datagrammes contiennent l'adresse IP des nœuds source et destination. Ainsi, lorsqu'un routeur reçoit un datagramme, il extrait l'adresse IP de destination, détermine son réseau d'appartenance et fait suivre le datagramme au prochain routeur permettant d'atteindre le réseau destination.

1.2 Architecture d'un routeur IP

Un routeur IP possède plusieurs interfaces réseau (au moins deux) auxquelles sont connectés les hôtes ou les routeurs des réseaux adjacents. Chacune de ces interfaces est identifiée par un numéro de port et se voit attribuer une adresse IP. Lors de la réception d'un datagramme sur une interface, le routeur calcule le port de l'interface par laquelle les informations reçues vont transiter pour joindre le réseau destination. Ce choix est déterminé par son algorithme de routage et se base sur les données contenues dans la *table de routage*

construite à partir de *messages de contrôle* échangés entre les routeurs par leur protocole de routage.

L'architecture d'un routeur, illustrée en figure 3.1, peut être représentée en deux parties logiques :

— la partie *routage* construit une table de routage à partir de messages de contrôle échangés entre les routeurs, calcule et sélectionne les routes. Les procédures de calcul des routes, les données enregistrées dans la table de routage et l'échange de messages dépendent du protocole de routage utilisé par le routeur. Les informations de la table de routage sont ensuite utilisées pour construire la table de transmission.

— la partie *transmission* détermine l'interface de sortie sur laquelle le datagramme IP entrant doit être transféré en direction du prochain routeur pour atteindre sa destination. Ce choix est effectué en recherchant dans la table de transmission le plus grand préfixe IP du réseau contenant l'adresse de destination du datagramme IP.

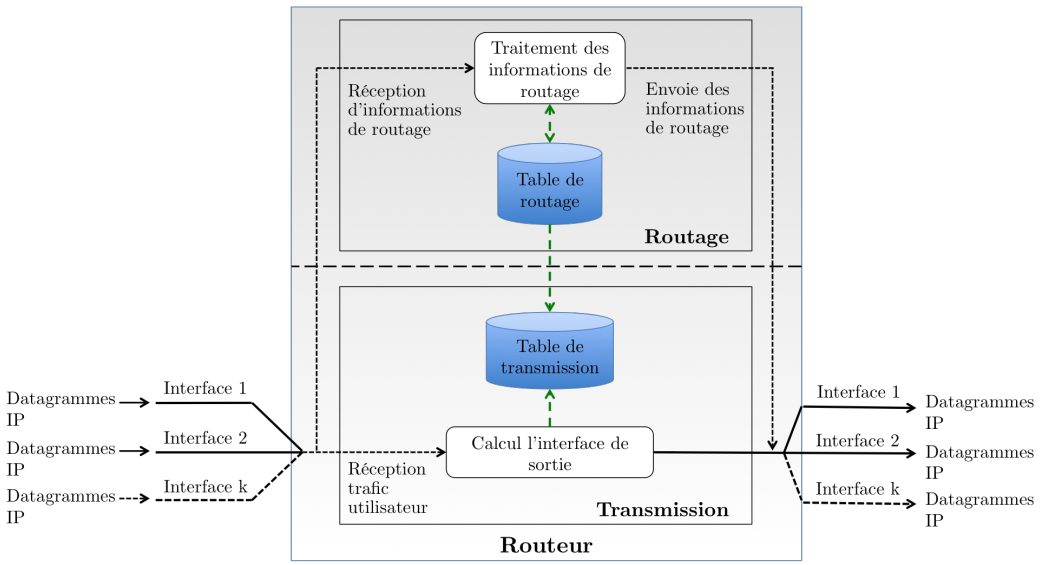


FIGURE 3.1 – Architecture d'un routeur

La table de transmission représente une base de données indexée par le préfixe des réseaux auquel est associée une unique interface de sortie du routeur. La table de transmission doit contenir l'ensemble des préfixes des réseaux atteignables par le routeur.

Lors de la réception d'un datagramme IP, le routeur doit déterminer si le datagramme contient des informations de routage devant être traitées par la partie routage ou si il correspond à des données utilisateur qui doivent être simplement transmises vers l'hôte destination. La méthode utilisée pour déterminer si un datagramme est un message de contrôle dépend du protocole de routage. Certains protocoles de routage se basent sur les protocoles de la couche transport tels que UDP (ou User Datagram Protocol) ou TCP (ou Transmission Control Protocol) pour échanger des informations de routage. Le numéro de port défini par ces protocoles de transport est utilisé pour identifier un message de contrôle. D'autres protocoles utilisent directement le champ de numéro de protocole disponible dans l'entête des datagrammes IP.

2 Routage dans l'Internet

Internet n'est pas un réseau unique, mais, comme son nom l'indique, un réseau de réseaux. Ces réseaux sont la propriété d'opérateurs (ou ISP, Internet Service Provider) qui les administrent selon leurs besoins. Un réseau administré de façon unique est appelé *domaine*. Dans Internet, un domaine est aussi appelé *système autonome* (ou AS, Autonomous System), il est identifié par un numéro d'AS unique, l'ASN (ou Autonomous System Number) attribué par les Registres Internet Régionaux (RIR [HCHC13]). Un AS peut être composé de plusieurs *sous-réseaux*. Ces sous-réseaux sont identifiés entre les AS par un préfixe IP.

Internet peut être comparé à un continent, les AS en sont les pays et leurs sous-réseaux les régions. Les AS sont séparés par des frontières, avec chacun sa législation. Un grand ISP peut posséder plusieurs AS, à l'instar d'un état fédéral. Internet est constitué d'environ 48 000 AS (septembre 2014) [Bat].

2.1 L'architecture d'Internet

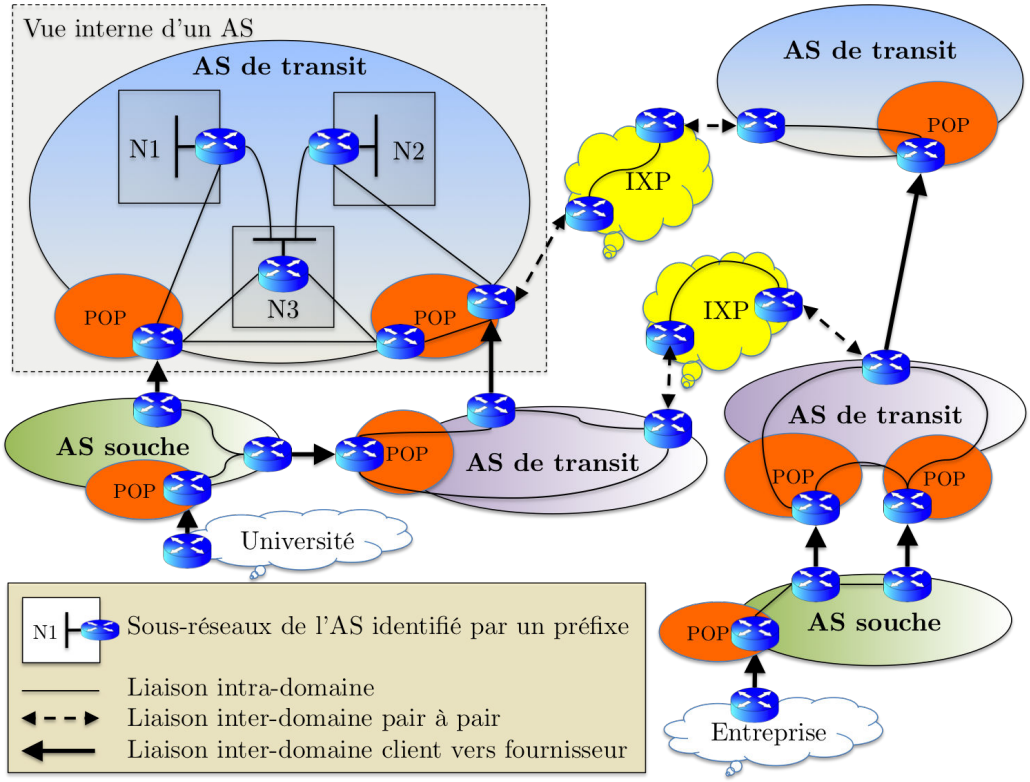


FIGURE 3.2 – Exemple d'interconnexion entre AS.

À l'intérieur d'un AS, les routeurs appartiennent à l'ISP ou à ses clients et sont administrés selon les règles qu'il a fixé, on parlera de routage *intra-domaine* (ou Intra-Routing Domain). Les routeurs et les liens assurant le routage intra-domaine forment le *coeur de réseau* de l'AS. Un exemple de coeur de réseau d'un AS est donné en figure 3.2 dans la vue interne de l'AS entre les sous-réseaux N1, N2 et N3.

Les clients se connectent à l'AS par des *liens d'accès*. Ces clients peuvent être des particuliers, des universités ou des entreprises privées. Pour que Internet existe et puisse

fonctionner, il faut établir des communications entre les AS d'ISP différents. Ce «passage de frontière» est possible grâce aux routeurs frontières des systèmes autonomes dont disposent les ISP. On parlera de routage *inter-domaine* (ou Inter-Domain Routing). Ces routeurs frontières sont directement reliés aux routeurs des ISP avec lesquels une relation doit être établie. Ces liens sont appelés *liens frontières*. Un AS ne contrôle qu'une extrémité d'un lien frontière. Les emplacements géographiques des routeurs frontières et des liens d'accès d'un AS sont appelés points de présence (ou PoP, Points of Presence). C'est à partir de ces PoP que les clients se connectent ou que d'autres AS s'interconnectent directement entre eux. Dans [GR00], deux types de relations entre AS ont été identifiées, des relations *client-fournisseur* et des relations *pair-à-pair*.

Relation client-fournisseur. Un client achète de la bande-passante à un fournisseur. Ce fournisseur est un ISP de plus grande taille. Le fournisseur se charge de la transmission des données en provenance et à destination du client.

Relation pair à pair. Les règles qui régissent la transmission de données entre les ISP dans le cas d'accords *pair à pair* (ou *peering*) sont généralement basées sur la réciprocité et postulent un certain équilibre. Ils ne donnent pas lieu à facturation tant que l'équilibre est respecté, mais des indemnités sont versées dès que la symétrie des échanges est trop fortement perturbée. On comprend que de tels accords, qui sont le plus souvent gardés secrets, ne sont possibles qu'entre des ISP de taille comparable [Bar00].

Afin de faciliter l'interconnexion des AS et l'établissement de liaisons pair à pair, les routeurs frontières sont regroupés dans des points d'échange (ou IXP, Internet eXchange Point). Les frais de connexion et de maintenance sont ainsi réduits puisque mutualisés entre les AS connectés à l'IXP.

En figure 3.2 est illustré un exemple de l'interconnexion entre AS de différentes catégories. Les AS sont connectés directement entre eux par une liaison entre leurs PoP ou par l'intermédiaire d'IXP. On pourra distinguer deux catégories d'AS [SARK02] :

— AS de transit, ils forment le coeur d'Internet. Leur infrastructure peut être utilisée par certains domaines pour transmettre leurs données vers d'autres domaines ;

— AS souche, aussi dénommé *stub domain*, généralement de plus petite taille que les AS de transit. La transmission de données en provenance d'autres AS par leur infrastructure n'est pas autorisée. Ils sont connectés à au moins un AS de transit mais peuvent l'être avec plusieurs afin d'assurer leur connectivité en cas de panne d'une des liaisons.

Les AS souches peuvent être différenciés à leur tour en fonction du service qu'ils fournissent. Si ils délivrent d'importants volumes de données on les nommera *fournisseurs de contenu*. On pourra citer Google, Apple ou encore Amazon.

Actuellement, le plus grand AS (AS 3356, LEVEL3) fournit ses services à approximativement 25 318 AS clients (55% des AS de l'Internet) représentant 322 403 préfixes réseau (64% des préfixes annoncés dans Internet) et 1 562 430 335 adresses IP (72% des adresses IPv4 dans Internet). Ces statistiques ont été obtenues à partir du rang des AS calculé par CAIDA [CAI14a] (ou Center for Applied Internet Data Analysis).

Plus globalement, à partir de la carte des relations inter-AS (septembre 2014) obtenue depuis [CAI14b] et composée de 46 063 AS et 172 987 liens, on remarque que 22,4% des AS sont des AS de transit et 77,6% des AS souches. Environ 54% des liens sont des liaisons client-fournisseur pour 46% de liaisons pair à pair. Les liens entre AS de transit concentrent 72% des liens. Les AS souches maintiennent donc peu de liens en comparaison avec leur nombre. Seuls un ou deux liens sont établis avec un AS fournisseur, le deuxième lien étant

utilisé afin de garantir la connectivité de l'AS en cas de panne. Remarquons aussi que 15 AS de transit forment une *clique* au coeur du réseau, chacun de ces AS maintient au moins une liaison pair à pair avec tous les autres AS. Ces AS concentrent à eux seuls 14% des liens.

Il est intéressant d'observer l'évolution dans le temps de cette connectivité. Dans [SARK02], l'étude de la topologie des AS de 2002 montre 95% de relations client-fournisseur avec 15% d'AS transit. On observe donc un accroissement important du nombre de liaisons pair à pair, confirmant une logique d'économie des coûts par les AS transit. Cependant, cette comparaison est à relativiser suite à l'évolution des méthodes utilisées pour établir la carte des AS et inférer les relations inter-AS [LHc+13].

2.2 Le routage intra-domaine

Le routage à l'intérieur d'un domaine est assuré par la catégorie des protocoles de routage *intra-domaine* (ou IGP, Interior Gateway Protocol). Ces protocoles transmettent les données en provenance ou à destination des routeurs frontières du domaine. Les objectifs d'un tel protocole sont de maintenir la connectivité du réseau, s'adapter facilement et rapidement aux changements topologiques ou aux pannes et optimiser les resources du réseau. Les routes entre les nœuds du réseau sont optimales. Les boucles ne sont pas autorisées.

Il existe plusieurs méthodes pour assurer le routage intra-domaine :

Table de routage statique. Les tables de routage sont éditées manuellement par l'administrateur du domaine. Cette méthode est adaptée pour de petits domaines avec peu de changements topologiques (< à 10 routeurs). Elle devient très compliquée à mettre en œuvre dans des réseaux de plus grande taille et peut amener à des erreurs de configurations.

Protocole à vecteur de distances. Ils se basent sur les algorithmes de Bellman-Ford, Ford–Fulkerson ou encore DUAL FSM [GLA93]. Un routeur diffuse des messages de contrôle à ses voisins uniquement. Un message contient une métrique associée à l'identifiant de la destination. Cet identifiant peut par exemple être un préfixe IP. La métrique détermine le coût pour atteindre la destination à partir du routeur ayant émis le message de contrôle. Plus la métrique est faible, meilleure sera la route pour atteindre la destination. Une métrique peut être définie par le délai des liens ou le nombre de sauts (routeurs) pour atteindre la destination.

Un routeur construit sa table de routage à partir des messages de contrôle reçus. Les entrées de la table de routage sont composées de l'identifiant de la destination associé à la métrique et au routeur adjacent vers qui transmettre les messages. Lorsqu'un routeur reçoit un message de contrôle, pour une destination donnée, il compare la métrique avec celle contenue dans sa table de routage. Si la métrique est plus faible alors la table est mise à jour avec les informations reçues.

Les protocoles à vecteur de distances posent différents problèmes de routage dans les grands réseaux ce qui rend difficile leur passage à l'échelle.

— Les *boucles de routage* : pour une destination donnée, la route suivie par un paquet rejoint son point de départ avant d'atteindre sa destination. Le paquet passe de façon répétée par les mêmes routeurs, la route suivie forme une boucle. Au final, la transmission des données ne peut être arrêtée sans un mécanisme dédié comme le temps à vivre d'un paquet (ou TTL, Time To Live).

— Les *mesures infinies* : elles résultent de boucles de routage et engendrent des échanges d'informations contradictoires entre les routeurs. La métrique de la route concernée dans la table de routage des routeurs augmente par incrément jusqu'à atteindre l'infini, surchargeant le réseau de messages et pouvant rendre les routeurs inopérants.

Ces problèmes peuvent être résolus en utilisant différentes techniques comme *l'empoisonnement de route*, le mécanisme de *split horizon* ou encore le *compteur de retenue* (Hold-down Timer). Chacun de ces mécanismes ayant ses avantages et inconvénients. Différents protocoles à vecteur de distances existent, on pourra citer RIP [Mal98] et IGRP [Cis81].

Protocole à état de liaisons. Un routeur diffuse à *tous* les autres routeurs du domaine des messages contenant leur voisinage. Un message est émis dès que l'état d'un lien évolue. Chaque routeur maintient un graphe orienté du domaine, il a une vision globale de la topologie. Ce graphe doit être identique pour tous les routeurs. La table de routage est dérivée à partir du calcul des plus courts chemins dans le graphe. L'algorithme de Dijkstra est généralement utilisé. Un plus court chemin est déterminé à partir de la métrique des liens. Cette métrique est définie par l'administrateur du domaine, elle est en général proportionnelle à la bande passante des liens.

Le principal avantage de cette catégorie de protocole de routage est l'impossibilité de créer des boucles de routage. Cependant, le volume d'information échangé pour la construction du graphe par chacun des routeurs peut être important dans de grands réseaux. De plus, chaque routeur garde en mémoire le graphe complet du réseau pour le calcul de la table de routage à partir de l'algorithme des plus courts chemins.

Afin de pallier à ces inconvénients, un domaine de grande taille peut être découpé en *aires*. Dans ce cas, une aire exécute son propre protocole IGP. Les routeurs appartenant à une aire différente ne sont pas visibles. Des routeurs frontières entre les aires sont chargés de la communication inter-aires. On pourra citer les protocoles à état de liaisons IS-IS [Ora90] et OSPF [Moy98].

2.3 Le routage inter-domaine

Le routage d'information entre domaines est assuré par un protocole dit de routage *inter-domaine* (ou EGP, External Gateway Protocol). Un seul protocole inter-domaine est actuellement utilisé dans Internet, le protocole de routeur frontière (ou BGP, Border Gateway Protocol). Un protocole inter-domaine doit permettre aux AS de contrôler les données entrantes et sortantes dans le réseau en fonction de son origine. Contrairement aux protocoles à vecteur de distances ou à état de liaisons, le choix d'une route ne se base pas uniquement sur une métrique de plus court chemin pour atteindre une destination. Ce choix se base sur un ensemble de règles afin de décider de qui peut utiliser le réseau de l'AS et faire transiter des données. L'ensemble de ces règles définit les *politiques de routage* de l'AS.

De plus, un protocole de routage inter-domaine doit pouvoir passer à l'échelle, le nombre d'AS dans Internet ne cessant d'augmenter depuis sa création. Les coûts en mémoire et de calcul des routes ainsi que le volume de communication nécessaire à la construction des tables de routage doivent être maîtrisés. C'est pourquoi une troisième catégorie de protocole a été proposée, celle des protocoles à vecteur de chemins.

Politiques de routage

Les relations économiques inter-AS régissent les politiques de routage appliquées par les AS qui elles-mêmes déterminent le choix des routes à suivre pour le routage des données. Dans le cas d'une relation client-fournisseur, l'AS fournisseur autorise l'utilisation de son réseau par le client pour envoyer ou recevoir des données, ce dernier payant pour ce service. Le fournisseur annonce l'ensemble des destinations qui lui sont connues au client et l'ensemble des préfixes du client à son voisinage.

Un AS préfère utiliser une liaison pair à pair pour échanger des données afin de limiter le coût des transmissions via un fournisseur. Cependant, un AS refuse tout transit de données

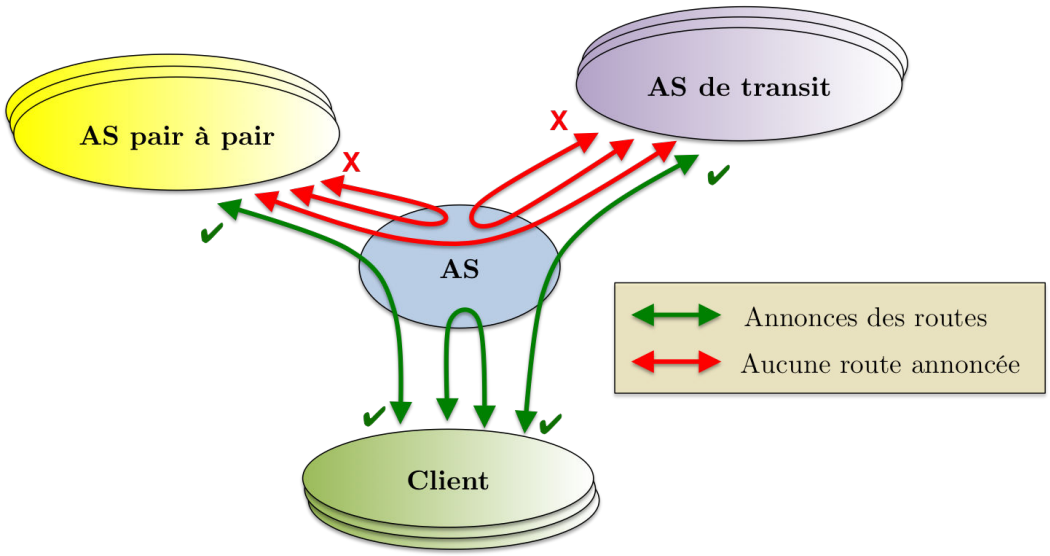


FIGURE 3.3 – Politiques de routage entre AS clients, fournisseurs et pairs.

par son réseau entre une liaison pair à pair et ses fournisseurs. En effet, le transit de ces données via un fournisseur engendre un coût de communication dont l'AS n'est pas à l'origine. De plus, dans le cas de données en provenance d'un fournisseur, le volume de communication peut déséquilibrer la liaison pair à pair. De telles relations sont illustrées en figure 3.3. Le transit d'informations par le réseau de l'AS central est interdit entre les liaisons pair à pair et les fournisseurs. Seul celui du client est autorisé.

Afin d'appliquer ses politiques de routage, un AS doit contrôler les informations de routage reçues et envoyées par le protocole EGP aux AS voisins. En effet, si le protocole EGP annonce au voisinage la possibilité d'utiliser le réseau de l'AS pour atteindre une destination donnée, celui-ci devra ensuite assurer le transit des communications. Le contrôle des informations échangées par le protocole EGP peut être assuré à plusieurs niveaux :

- le filtrage en entrée des informations de routage permet de préférer l'utilisation d'une liaison pair à pair plutôt que celle vers un fournisseur pour atteindre une destination annoncée par un AS voisin ;
- la sélection des routes à enregistrer dans la table de routage du protocole EGP ;
- le filtrage en sortie des informations de routage évite d'informer certains AS de la possibilité de transit de données vers certaines destinations.

Protocole à vecteur de chemins

Comme nous l'avons vu en section 2.2, les protocoles à vecteur de distances impliquent l'utilisation de mécanismes complexes afin d'éviter les boucles et ne sont pas utilisables dans l'Internet. Les protocoles à état de liaisons permettent le routage sans boucle sur des topologies de plus grande taille. Cependant, le maintien en mémoire par les routeurs d'une carte globale de la topologie, le coût du calcul des plus courts chemins et la diffusion de messages de contrôle vers l'ensemble des routeurs de la topologie rendent difficilement utilisable ce type de protocole dans le cadre du routage dans l'Internet.

Le protocole à vecteur de chemins se base sur celui à vecteur de distances. Les messages de contrôle sont échangés de proche en proche. Cependant, en plus de la destination et

de sa métrique associée, la route complète vers la destination est incluse dans le message. Lorsqu'un routeur reçoit un message de contrôle, il ajoute son identifiant au chemin reçu puis le transmet à ses voisins. Les routeurs maintiennent donc une table de routage dont chaque entrée contient l'identifiant de la destination, la métrique et la route à suivre pour atteindre la destination. Ainsi, au prix de tables de routage de plus grande taille (comparativement au protocole à vecteur de distances), il est possible de détecter les boucles de routage. Il suffit pour un routeur recevant une information de routage, de contrôler sa présence dans la route annoncée. Si tel est le cas, le message est ignoré. Un protocole à vecteur de chemins se caractérise donc par :

— l'annonce du chemin vers une destination ;

— l'annonce de ces chemins uniquement de proche en proche ;

— la sélection des chemins les plus courts en fonction d'une métrique donnée.

Le protocole à vecteur de chemins corrige les principaux problèmes du protocole à vecteur de distances et évite de maintenir en mémoire une carte de la topologie comme dans le cas des protocoles à état de liens. Cependant, afin d'être utilisable dans l'Internet, un protocole à vecteur de chemins doit être adapté afin de répondre aux particularités du routage dans l'Internet (utilisation des préfixes IP, ASN, etc) et fournir les fonctionnalités nécessaires à la mise en place de politiques de routage par les AS. Dans ce but, le protocole de routage BGP a été développé.

2.4 Le protocole de routage BGP

BGP [RLH06] est le protocole à vecteur de chemins actuellement utilisé dans l'Internet. Il permet l'échange d'informations de routage entre les AS afin de déterminer le chemin à suivre pour joindre une destination donnée. Dans Internet, les destinations sont les sous-réseaux hébergés par les AS. Chacun de ces sous-réseaux est identifié par un préfixe IP. Un chemin pour atteindre un préfixe donné est constitué des numéros d'AS (ASN) à suivre. BGP se basant sur le protocole à vecteur de chemins, un routeur BGP annonce à ses voisins un préfixe associé au chemin des AS pour l'atteindre. Cette annonce est une *promesse* de l'AS auquel appartient le routeur d'autoriser le routage des données vers le préfixe annoncé à travers son réseau.

Sessions BGP internes et externes

Chaque paire de routeurs BGP communique au moyen d'une *session* ouverte à partir d'une connexion TCP sur le port 179. Le choix de TCP autorise une liaison entre deux routeurs en mode connecté, une liaison virtuelle est établie entre les routeurs. La transmission des messages de contrôle et sa fiabilité sont ainsi assurées par les protocoles de niveaux inférieurs.

Dans le cas d'un AS composé d'un seul routeur BGP, les échanges d'informations de routage se font directement avec le routeur situé dans l'AS voisin. Les données reçues à destination d'un préfixe hébergé par l'AS sont directement transmises au protocole IGP. Celui-ci assurant l'acheminement des données jusqu'a l'hôte.

Dans le cas d'un AS composé de plusieurs routeurs BGP, chacun connecté à différents AS voisins, deux types de sessions BGP sont possibles :

— *eBGP*, les routeurs appartiennent à deux AS différents ;

— *iBGP*, les routeurs appartiennent au même AS.

Les sessions iBGP assurent l'échange des informations de routage au sein de l'AS. Les annonces des routes échangées avec les AS voisins doivent pouvoir être transmises à l'ensemble des routeurs BGP de l'AS. En prenant l'exemple de la figure 3.4, les routeurs R1 et R2 établissent une session eBGP avec leur homologue dans l'AS1 et l'AS2. Si l'AS3 souhaite

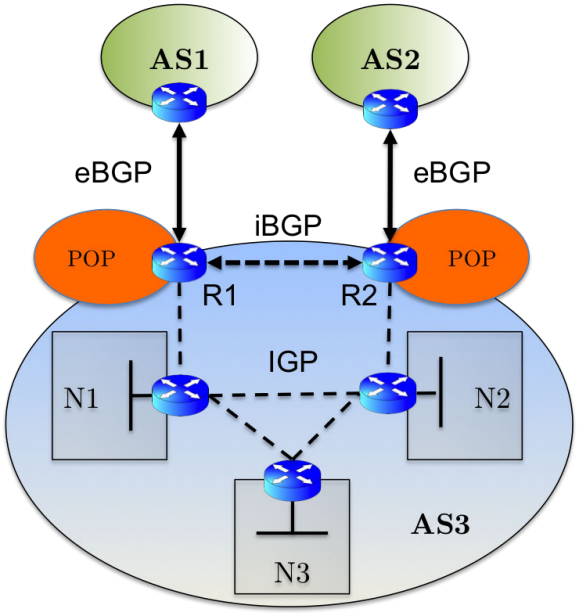


FIGURE 3.4 – Sessions entre routeurs iBGP et eBGP et protocole IGP.

autoriser le transit par son réseau de données entre les AS1 et AS2, les routeurs R1 et R2 doivent annoncer entre eux une route vers leur AS voisin respectif. Ainsi, R1 saura que pour atteindre AS2, il doit transmettre les données à R2 et inversement pour R2 vers AS1. La transmission des données sera ensuite assurée par le protocole IGP. Cependant, l'annonce des routes entre R1 et R2 est seulement possible à partir de leur session iBGP.

Les communications entre les routeurs BGP par des sessions iBGP et eBGP sont régies par trois règles :

1. un routeur recevant des informations de routage d'un AS voisin par sa session eBGP pourra les retransmettre uniquement aux routeurs BGP de son AS avec lesquels une session iBGP est directement établie ;
2. un routeur recevant des informations de routage par une session iBGP pourra les annoncer uniquement aux routeurs des AS voisins par une session eBGP ;
3. un routeur recevant des informations de routage par une session iBGP ne peut pas les retransmettre vers un autre routeur par une autre session iBGP.

Ces règles imposent un maillage complet des routeurs BGP à l'intérieur de l'AS, ce qui peut poser problème dans le cas d'un grand nombre de routeurs. Il est possible d'utiliser des *réflecteurs de route* afin de créer une hiérarchie au sein de ces routeurs [BCC06] ou encore subdiviser l'AS en plus petites entités afin de créer une confédération [TMS07].

Messages BGP

Divers types de messages assurent la robustesse du protocole en cas de panne des liens ou l'intégration de nouveaux routeurs dans le réseau. Un message OPEN est échangé entre deux routeurs lors de l'ouverture d'une session BGP, chacun informant l'autre de son AS d'appartenance, de la version de BGP exécutée et du temps maximum sans activité à partir duquel la session sera fermée. Des messages KEEPALIVE sont envoyés régulièrement afin de garder la session ouverte. Un message NOTIFICATION permettant lui de clore la session.

Le message UPDATE, est échangé entre les routeurs lors de la mise à jour des informations de routage. Ces informations sont composées d'*attributs*, les principaux étant :

— l'origine de la route (ORIGIN) qui détermine si la route a été apprise via eBGP, iBGP ou par un autre moyen (route statique) ;

— le chemin des AS à suivre pour atteindre un préfixe (AS-PATH) ;

— l'adresse IP du prochain routeur (NEXT-HOP) vers lequel envoyer les données utilisateur pour atteindre le préfixe IP destination ;

— une mesure décimale (LOCAL-PREF) pour décider de la préférence entre deux routes vers un même préfixe. La route de plus grande valeur LOCAL-PREF sera préférée.

L'attribut LOCAL-PREF est une mesure interne à l'AS qui est transmise uniquement entre routeurs connectés par une session iBGP. Dans le cas d'un message UPDATE, les attributs ORIGIN, AS-PATH et NEXT-HOP doivent être obligatoirement présents. L'attribut LOCAL-PREF étant optionnel.

Enfin, un message particulier, ROUTE-REFRESH, permet à un routeur de demander à son voisin de lui envoyer l'ensemble de ses entrées dans un message UPDATE. Ce message peut ne pas être supporté par la version de BGP exécutée par un routeur. Deux routeurs BGP s'assurent de la prise en charge de ce message lors de l'envoi du message OPEN à l'ouverture de la session.

Compteurs BGP

BGP utilise un compteur de retenue afin de déterminer si une session doit être fermée faute d'échange de message. D'autres compteurs sont utilisés, les principaux sont les suivants :

— *ConnectRetryTimer*, ce compteur définit le temps avant de renvoyer un message OPEN lors de l'ouverture d'une session. Le temps par défaut est de 120 sec.

— *HoldTimer*, c'est le compteur de retenue utilisé afin de déterminer si une session doit être maintenue ouverte ou fermée. Le temps recommandé est de 90 sec et doit être au minimum de 3 sec. Ce temps peut être configuré à 0 sec dans le cas d'une session qui ne peut pas expirer.

— *KeepAliveTimer*, ce compteur détermine l'intervalle de temps entre deux messages KEEPALIVE afin de maintenir la session active. Sa valeur correspond à un tiers de celle de HoldTimer.

— *MinRouteAdvertisementIntervalTimer (MRAI)*, ce compteur détermine l'intervalle de temps entre deux messages UPDATE pour l'annonce ou le retrait d'une route vers un routeur BGP voisin particulier pour un préfixe donné. On nommera cet intervalle une *instance* MRAI. Dans le cas d'une session iBGP, la valeur recommandée est de 5 sec, pour eBGP la valeur recommandée est de 30 sec.

— *MinASOriginationIntervalTimer*, ce compteur détermine l'intervalle de temps avant la diffusion d'un message UPDATE par le routeur BGP si le préfixe concerné est interne à l'AS.

Un paramètre clé de BGP est la valeur du compteur MRAI. Ce mécanisme empêche l'envoi successif de messages de mise à jour et limite donc leur fréquence. Le temps d'attente imposé par le compteur MRAI permet à un routeur de recevoir plusieurs routes pour un nouveau préfixe annoncé. À l'expiration du compteur, seule la meilleure route reçue sera annoncée, réduisant le nombre de messages de mise à jour nécessaire pour converger. Ce mécanisme a cependant ses limites qui seront détaillées en section 3.1.

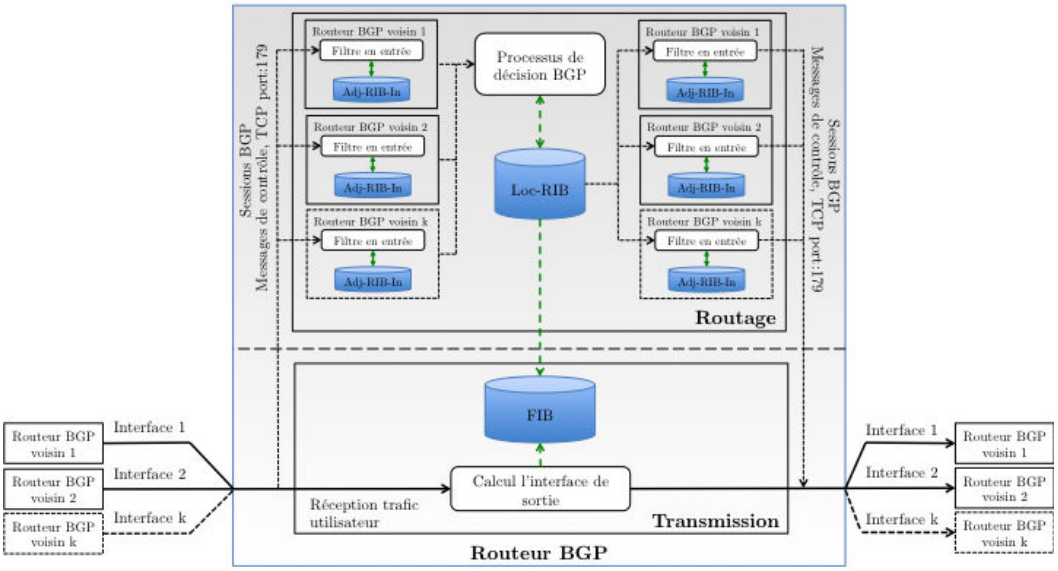


FIGURE 3.5 – Architecture d'un routeur BGP

Processus de décision BGP

Lors de la réception d'un message UPDATE par un routeur BGP, celui-ci applique le processus de décision BGP. Ce processus filtre les messages de mise à jour reçus, décide quels chemins associés aux préfixes annoncés doivent être enregistrés dans la table de routage et filtre ceux devant être annoncés aux AS voisins. La table de routage de BGP (ou RIB, Routing Information Base) est composée de trois structures de données illustrées en figure 3.5 et utilisées par le processus de décision BGP :

— *Adjacent RIBs-In (Adj-RIBs-In)*, cette structure de données garde en mémoire les informations de routage contenues dans les messages UPDATE envoyés par chacun des routeurs voisins et pour chaque préfixe. Plusieurs tables Adj-RIBs-In sont donc maintenues, une par routeur BGP avec lequel une session est ouverte. Plus d'un chemin par préfixe IP peut être gardé en mémoire. Si une route précédemment sélectionnée devient indisponible, une nouvelle peut être choisie parmi les tables Adj-RIBs-In. Ces routes sont ensuite filtrées puis transmises au processus de sélection.

— *Loc-RIB*, cette structure de données garde en mémoire une route par préfixe. La sélection des routes est obtenue après application du processus de sélection. Ce processus de sélection permet de choisir une route à enregistrer dans la Loc-RIB lorsque plusieurs routes à destination d'un même préfixe sont disponibles. Une seule table Loc-RIB est maintenue par le routeur. La table de transmission est calculée à partir de la Loc-RIB.

— *Adjacent RIBs-Out (Adj-RIBs-Out)*, cette structure de données garde en mémoire les routes possibles à annoncer aux routeurs BGP voisins. Une table Adj-RIBs-out est maintenue par routeur voisin. Des règles de filtrage sont ensuite appliquées afin de déterminer quelles seront les routes envoyées aux routeurs voisins.

Le processus de sélection des routes intervient lorsque, après application des règles de filtrage sur les tables Adj-RIBs-In, plusieurs routes sont encore possibles pour atteindre un même préfixe destination. Ce processus permet la sélection d'une seule route et peut être résumé en sept étapes :

1. choisir la route ayant le plus grand attribut LOCAL-PREF ;
2. choisir la route avec le plus petit nombre d'AS à parcourir ;
3. choisir la route en fonction de l'attribut ORIGIN ;
4. choisir la route avec l'attribut MED le plus petit ;
5. choisir la route en provenance d'un routeur eBGP au lieu d'un routeur iBGP ;
6. choisir la route dont le routeur NEXT-HOP est le plus proche en accord avec la métrique du protocole IGP ;
7. choisir le routeur iBGP avec le plus petit identifiant.

Les trois structures de données Adj-RIBs-In, Loc-RIB et Adj-RIBs-Out permettent l'application par BGP des politiques de routage présentées en section 2.3. Les phases de filtrage en entrée et sortie de l'AS utilisent les structures Adj-RIBs-In et Adj-RIBs-Out et la Loc-RIB permet la sélection des préfixes et de leur attribut NEXT-HOP pour la mise à jour de la table de transmission.

3 Problématiques du routage dans l'Internet

L'augmentation du nombre d'AS et de préfixes annoncés dans Internet a motivé de nombreux travaux portant sur l'amélioration et l'ajout de fonctionnalités à BGP afin de garantir son fonctionnement à court et moyen terme. En effet, à l'échelle d'Internet, avec environ 48 000 AS et 500 000 entrées actives dans la table de transmission, les coûts mémoire et de calcul du processus de décision de BGP sont particulièrement importants. De nouveaux routeurs, toujours plus puissants, doivent être sans cesse développés par les constructeurs afin d'assurer la pérennité de l'architecture de routage d'Internet.

En plus de l'augmentation de la taille de la table de routage de BGP, des problèmes de *temps de convergence* peuvent apparaitre. Le temps de convergence est défini comme le temps nécessaire pour l'obtention d'une vue *cohérente* des tables de routage par l'ensemble des routeurs du réseau. Lorsque plus aucune information de routage est nécessaire à un routeur, sa table de routage est cohérente avec l'état du réseau. Quand tous les routeurs ont atteint cet état, on dira que le protocole a *convergé*. Le *temps de convergence* peut être mesuré en terme d'unité de temps mais aussi en terme de nombre de messages échangés par les routeurs pour le calcul des tables de routage. Dans le cas de changements topologiques réguliers ou d'erreurs de manipulation des politiques de routage engendrant l'annonce de nouvelles routes, le temps de convergence du protocole peut être infini. Ces erreurs de manipulation des politiques de routage apparaissent régulièrement dans l'Internet et peuvent causer d'importantes perturbations dans tout le réseau.

En conséquence, un certain nombre de travaux se sont concentrés sur l'étude de nouveaux systèmes de routage pour l'Internet afin de remplacer BGP. De nouvelles méthodologies pour la conception de ces systèmes ont été mises en place. Celles-ci permettent de tenir compte dès la conception du système des différents besoins et d'identifier les contraintes à respecter [DBI+11a].

3.1 Évolution de l'Internet

Depuis le début des années 1990 et l'ouverture d'Internet, le nombre d'AS n'a cessé de croître. Son évolution est illustrée en figure 3.6. On peut observer une progression du nombre d'AS quasi-exponentielle entre les années 1999 et 2002 puis une progression linéaire jusqu'à nos jours. Avec l'augmentation du nombre d'AS, le nombre de routes actives utilisées dans la table de transmission atteint maintenant les 500 000 entrées ce qui dépasse les prévisions de croissance.

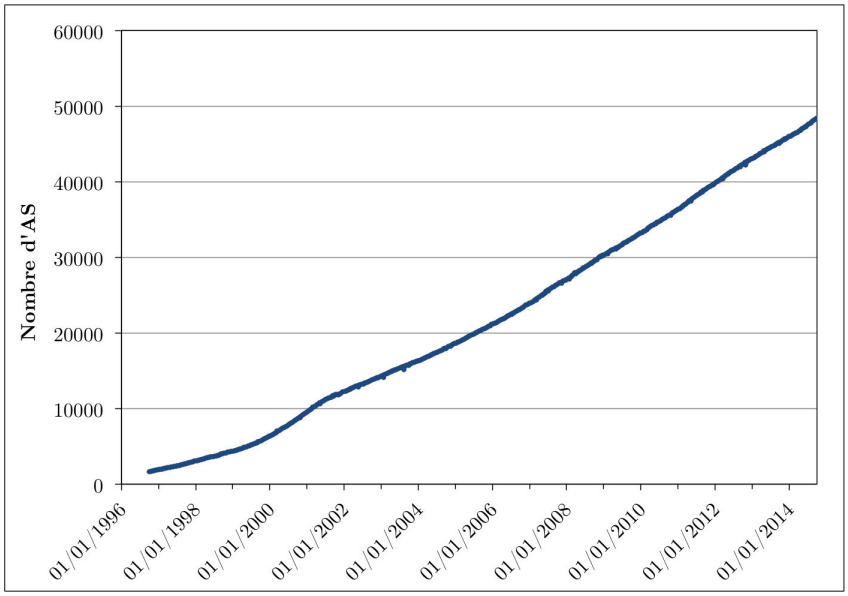


FIGURE 3.6 – Évolution du nombre d'AS dans l'Internet.

L'évolution du nombre d'entrées dans la table de transmission de l'AS 65000 est illustrée en figure 3.7. On peut observer depuis 2000 une augmentation d'environ 15% des routes chaque année. En ne considérant que la progression du nombre de routes depuis 2011, l'augmentation approche plutôt les 12% d'entrées supplémentaires chaque année. À partir d'une estimation intermédiaire de cette évolution pour 2020, on peut s'attendre à un doublement du nombre d'entrées pour atteindre 1 000 000.

Dans ce contexte, les deux principaux problèmes pour le passage à l'échelle de BGP sont la taille de ses structures de routage et le temps de convergence du protocole.

Taille des structures de routage de BGP. Comme nous l'avons vu en section 2.4, plusieurs structures de données forment la table de routage. Les tables Adj-RIBs-In et Adj-RIBs-Out permettent le filtrage en entrée et sortie du routeur des routes depuis ou vers les routeurs BGP voisins. Une table en entrée et sortie est maintenue par voisin. La table Loc-RIB maintient les routes sélectionnées qui seront utilisées pour le calcul de la table de transmission. Le coût en mémoire de ces tables est donc en $O(k.n)$ bits avec k la taille d'un chemin et de ses attributs et n le nombre de préfixes destination. De plus, chacune des tables Adj-RIBs-In peut stocker plusieurs chemins par préfixe. Au final, dans l'AS65000, l'ensemble des structures de données de routage de BGP est environ 3 fois plus volumineux que la table de transmission [Bat].

Une telle évolution du nombre d'entrées entraine une augmentation du nombre de messages échangés sur le réseau et un allongement du temps de convergence. La question est de savoir si les évolutions technologiques proposées par les constructeurs permettront aux routeurs BGP d'assurer le bon fonctionnement du système de routage et pour combien de temps.

Temps de convergence de BGP Environs $2.6\,10^6$ messages de mise à jour BGP ont été échangés en 6 jours durant le mois de septembre 2014, avec une moyenne de 3.86 messages par seconde et un maximum de 5294. Les routeurs BGP doivent donc faire face à de subites montées en charge.

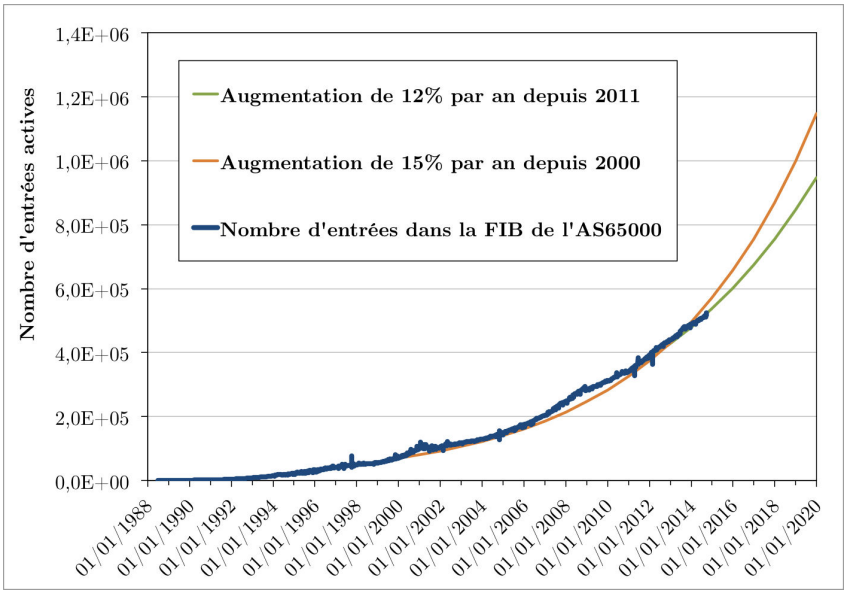


FIGURE 3.7 – Évolution du nombre d'entrées dans la FIB de l'AS 65000.

Le nombre de messages échangés et le temps de convergence sont directement impactés par le choix de la valeur du compteur MinRouteAdvertisementIntervalTimer. Il a été montré que la stabilisation des routes n'intervient que 3 à 15 minutes après la mise hors service d'un nœud du réseau [LABJ01]. Cette limitation introduit une synchronisation des envois de messages de mise à jour par préfixe de destination.

Si le réseau est connexe, alors [LABJ01] ont démontré que la borne inférieure concernant le temps de convergence est donnée par $(N-3) \times MRAI$ avec N le nombre d'AS dans le réseau. On peut penser qu'il est possible de diminuer le temps de convergence en réduisant la valeur de MRAI. En pratique, une fois la valeur de MRAI réduite en-dessous d'un certain seuil, d'autres effets contradictoires apparaissent en termes de coût de communication et temps de convergence [GP01]. Le choix optimal de cette valeur dépend particulièrement des propriétés du réseau de l'AS, une valeur de MRAI doit donc être choisie au cas par cas. Un travail en cours à l'IETF [Jak11] prévoit de ne plus recommander de valeur. Le choix de celle-ci étant laissé à la libre appréciation des administrateurs du réseau.

3.2 Conception de nouveaux systèmes de routage pour l'Internet

Dans l'idéal, un système de routage devrait pouvoir passer à l'échelle en termes de mémoire et de messages échangés et garantir des routes de faible longueur. Un tel système devrait donc avoir une complexité sous-linéaire en mémoire pour le calcul des tables de routage et nécessiter un faible nombre de messages pour converger tout en laissant la possibilité aux administrateurs de configurer les politiques de routage de l'AS.

Diverses catégories de *schémas de routage* sont proposées pour remplacer BGP. Nous parlerons d'un schéma de routage pour définir un algorithme permettant de calculer une route vers une destination du réseau. Par exemple, les schémas de routage compacts permettent d'atteindre des tables de routage sous-linéaires au prix d'un allongement des routes dans le réseau. Les paquets n'empruntent pas le plus court chemin pour aller d'une source à une destination. Une étude complète de ces schémas est proposée dans la thèse de Christian Glacet [Gla13]. Le routage géographique permet quant à lui d'atteindre une destination à partir du calcul des coordonnées d'un nœud dans le réseau. Il a été montré que le plonge-

ment du graphe de l'Internet dans un espace hyperbolique permet un routage glouton fiable et efficace [PKBV09].

Cependant, la conception de ces nouveaux schémas de routage et l'intégration des contraintes de routage dans l'Internet (politiques de routage, indépendance des noms, traitement rapide des messages) est particulièrement complexe. Dans [DBI+11a], les auteurs proposent une méthodologie afin de faciliter cette tâche, un schéma est décomposé en différents modèles plus simples à appréhender. Un réseau, ses équipements, les protocoles de routage utilisés ainsi que les interactions entre tous ces éléments composent l'*architecture* d'un système de routage.

Dans la suite de cette thèse, nous nous référons à la terminologie et aux définitions proposées dans [DBI+11a] des différents éléments de l'architecture d'un système et plus particulièrement d'un système de routage.

Architecture d'un système

Plus globalement, sans nécessairement parler de système de routage, l'architecture d'un système est définie par :

— un ensemble de *fonctions* du système, auquel est associé un modèle fonctionnel ;

— des *éléments d'information*, auxquels est associé un modèle d'information ;

— des *états*, auxquels est associé un modèle à états.

À chacun de ces éléments est associé un comportement, une structure, une composition et une répartition spatio-temporelle (ex. les routeurs, leur localisation et leurs relations dans l'Internet). Les relations entre ces éléments sont représentées en figure 3.8. Les spécifications de l'architecture d'un système incluent aussi les principes et règles concernant sa conception et son évolution dans le temps.

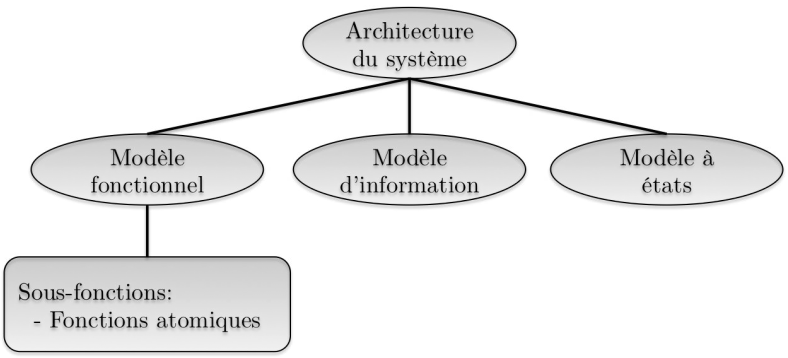


FIGURE 3.8 – Architecture d'un système.

Le modèle fonctionnel. Il caractérise le système en définissant sa fonction, les entrées et sorties ainsi que les interfaces qui le composent. Il a pour but de décrire le traitement nécessaire effectué par un système complexe pour transformer les entrées en sorties désirées. Dans le modèle fonctionnel, les fonctions complexes de haut-niveau sont décomposées en un ensemble de sous-fonctions simplifiées pouvant aller jusqu'à des *fonctions atomiques*, c'est-à-dire des fonctions qui ne peuvent plus être davantage décomposées.

Par exemple, le protocole BGP peut être décomposé en trois fonctions principales plus générales, indépendantes du protocole :

— un modèle fonctionnel de découverte des informations ;

— un modèle fonctionnel de sélection des meilleures routes ;

— un modèle fonctionnel de diffusion des informations.

Dans BGP, le modèle de découverte des informations correspond au traitement des messages de mise à jour reçus par un routeur (stockage dans la Adj-RIB-In et filtrage). Celui de sélection des meilleures routes correspond au processus de sélection des routes et à leur stockage dans la Loc-RIB. Celui pour la diffusion de informations correspond au traitement et filtrage des routes à annoncer aux routeurs voisins.

Le modèle d'information. Il a pour but de décrire formellement les données utilisées, leurs relations, leurs propriétés, les contraintes, règles et opérations dans le système.

Dans le cas de BGP, ce modèle décrit les différentes structures de la table de routage telles que l'Adj-RIB-In, l'Adj-RIB-Out et la Loc-RIB ainsi que les relations qui les lient avec les opérations de filtrage et de sélection des routes.

Le modèle à états. Il décrit le comportement du système par l'intermédiaire d'une machine avec un nombre fini d'états. Cette machine est composée d'états, d'événements, de transitions entre les états et d'actions.

La machine avec un nombre fini d'états de BGP a par exemple été formellement décrite dans [RLH06]. Celle-ci est composée de six états qui sont : l'état d'attente (Idle), l'état connecté (Connect), l'état actif (Active), l'état d'ouverture de session (OpenSent), l'état de confirmation d'ouverture de session (OpenConfirm) et l'état de session établie (Established). Les événements peuvent être l'ouverture manuelle d'une session par un administrateur BGP ou un compteur qui a expiré. Les transitions d'un état à un autre sont initiées à la réception d'événements. Par exemple, une transition entre l'état d'attente d'un routeur et son état connecté arrive lorsque que l'événement de connection manuelle d'un routeur avec un routeur homologue par l'administrateur apparaît.

Ainsi, ces modèles permettent l'application d'une méthode générale de décomposition d'un système. Cette méthode est indépendante du système décrit et permet la comparaison de différents systèmes et l'identification de fonctions, d'informations, d'états ou d'événements communs entre eux.

Architecture d'un système de routage

À partir de ces définitions, il nous est maintenant possible de décrire plus précisément ce qu'est un système de routage et de le décomposer en modèles plus simples à utiliser. Ces modèles simplifiés seront décrits par la suite en tant que *schéma* de routage, *modèle* de routage et *protocole* de routage. L'architecture d'un système de routage peut se composer d'un ou plusieurs schémas de routage. Un schéma de routage peut lui-même être à son tour composé de plusieurs modèles de routage comme illustré en figure 3.9.

Schéma de routage. Il représente un ensemble de modèles de routage qui se basent sur les mêmes *principes* et qui partagent donc les mêmes éléments de structures et caractéristiques essentielles.

Modèle de routage. Il est composé d'un modèle fonctionnel associé à un modèle de données, un modèle à états et un modèle de communication. Les modèles de données et de communication forment le modèle d'information comme vu en section 3.2. Le modèle de communication définit les interactions (messages, appel de procédure, etc) entre les éléments du modèle de routage (modèle fonctionnel, à états et de données).

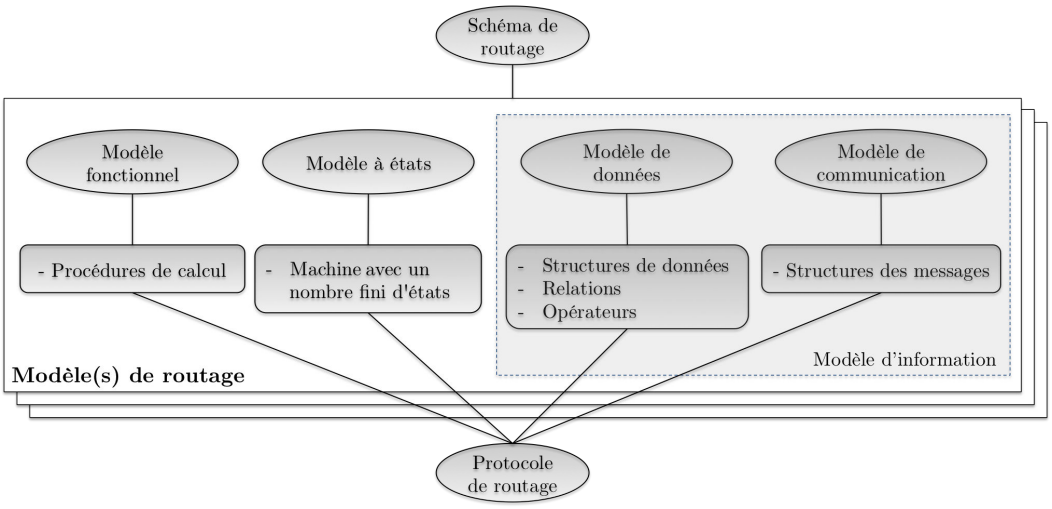


FIGURE 3.9 – Modélisation du schéma et du protocole de routage.

Protocole de routage. Il est composé d'une procédure de routage décrite par un algorithme à laquelle sont associés des structures de données, une machine avec un nombre fini d'états et le format des messages.

Dans le cas général, on préférera le terme de schéma de routage à celui de modèle de routage. Néanmoins, dans un contexte précis, le terme de modèle de routage pourra être utilisé. On notera aussi la différence entre un modèle de routage et son protocole. Le premier permet une décomposition hiérarchique des fonctions de routage (associées aux informations de routage) du protocole. Une telle modélisation permet d'isoler les blocs fonctionnels d'un protocole et d'identifier ceux posant problèmes. Ces problèmes pouvant être le passage à l'échelle du protocole, le coût (temporel, en mémoire ou en complexité de calcul) de convergence ou encore les aspects de sécurité du protocole. Cette approche n'a donc pas pour objectif de spécifier les procédures à utiliser ni comment les implémenter, mais d'identifier où les modifications doivent être effectuées et éventuellement comment concevoir un nouveau schéma de routage alternatif.

La proposition dans [DBI+11a] de cette méthodologie a été motivée par le constat suivant : le développement de nouveaux schémas de routage pour l'Internet suit la même approche que celle de BGP ; l'algorithme de routage détermine le comportement du système de routage. Or, les besoins du système devraient déterminer quelles classes d'algorithmes permettent d'obtenir les résultats voulus.

Dans la suite de ce manuscrit nous utiliserons la terminologie venant d'être proposée. Le terme *schéma de routage* sera le plus souvent utilisé. Dans le cas de BGP, nous utiliserons naturellement le terme de *protocole*. Cependant, suivant le contexte, un schéma ou modèle de BGP pourra être considéré. Ces termes feront référence au protocole BGP auquel aura été appliqué un découpage en modèles fonctionnel et d'information. Un *modèle de BGP* sera plus particulièrement utilisé dans le cas du simulateur DRMSim présenté dans le chapitre 4.

4 Conclusion

Comme nous l'avons vu, Internet évolue rapidement, les relations entre ses systèmes autonomes sont complexes et son protocole de routage frontière montre ses limites. Un défi majeur pour assurer la pérennité du fonctionnement d'Internet est le passage à l'échelle de son architecture de routage.

De nouveaux schémas de routage sont développés afin de pallier aux limites de BGP et une nouvelle méthodologie pour l'analyse fonctionnelle de ces schémas a été proposée dans [DBI+11a]. Cette méthode vise à faciliter le choix des algorithmes en fonction des contraintes imposées par les besoins de l'architecture de routage d'Internet. Elle permet aussi l'identification des points bloquant dans le processus de développement de ces nouveaux schémas de routage.

Cependant, ces schémas de routage, bien que prometteurs, sont complexes à mettre en œuvre et à tester directement dans Internet. À partir des premières spécifications et analyses théoriques d'un schéma, un protocole de routage doit en être dérivé. Le cycle de développement d'un nouveau protocole de routage impose la validation de ses performances par simulation puis par émulation. Son comportement à différentes situations, tels que l'ajout ou la suppression de nœuds et liens dans le réseau, la modification des politiques de routage ou l'application de règles d'ingénierie de trafic, doit être testé. C'est seulement après la validation de chacune de ces étapes qu'il pourra être standardisé.

Un ensemble d'outils est donc nécessaire à l'étude de ces nouveaux schémas de routage. C'est dans ce contexte que je présente dans le chapitre 4, un nouveau simulateur de modèles de routage dynamiques pour l'évaluation de leurs performances et leur comparaison dans les grands réseaux.

Chapitre 4

Méthodes de simulation de modèles de routage dans de grands réseaux dynamiques

Sommaire

1 Introduction 72
1.1 Émulation du protocole BGP 72
1.2 Simulation de protocoles de routage 74
1.3 Motivations pour le développement de DRMSim 76
2 Principes et méthodologie pour la simulation 77
2.1 Définitions et terminologie 77
2.2 Modèle de simulation d'un système 78
2.3 Simulation à événements discrets 78
3 DRMSim, un simulateur de modèles de routage dynamiques . . . 79
3.1 Motivations et objectifs 79
3.2 Architecture de DRMSim 80
3.3 Exemple d'exécution 89
4 Le protocole de routeur frontière dans DRMSim 91
4.1 Description du modèle 92
4.2 Transmission des messages de mise à jour BGP 93
4.3 Métriques dans BGP 93
4.4 Validation du modèle 94
4.5 Complexité de la simulation de BGP dans DRMSim 94
5 Expérimentations de modèles de routage avec DRMSim 96
5.1 Un schéma de routage compact efficace dans les graphes k-chordaux 96
5.2 Expérimentations de schémas de routage compacts universels ou dédiés aux graphes sans échelle 97
6 Conclusion 100

Dans ce chapitre, je présente DRMSim [DRM09], un simulateur à événements discrets de modèles de routage dynamiques sur de grandes topologies. Ce simulateur a pour objectif premier de fournir une plate-forme commune pour la simulation et la comparaison de différents modèles de routage sur des topologies composées de plusieurs milliers de nœuds. L'idée sous-jacente est de simuler uniquement le modèle de routage et non son protocole afin d'éviter l'exécution de procédures bas niveau coûteuses en resources processeur et mémoire. Dans ce chapitre, je présente trois contributions :

— le développement d'une architecture modulaire basée sur le principe de *composants* afin de faciliter la modification et l'ajout de nouvelles fonctionnalités et de nouveaux modèles de routage au simulateur.

— l'intégration d'un modèle de mesure permettant à l'utilisateur de définir ses propres métriques, d'un modèle de dynamique permettant de configurer des scénarios d'ajout et de suppression de nœuds ou liens, d'un modèle de transmission prenant en compte le délai des liens et la dynamique de la topologie et un modèle système pouvant se baser sur plusieurs librairies de graphe pour la représentation de la topologie et le calcul de ses propriétés ;

— l'intégration de deux nouveaux modèles de routage, le modèle de routage à vecteur de distances sans boucle LFR [DDFM12] et le modèle de routage géographique [KD14].

Je commence en section 1 par présenter les différentes techniques existantes pour l'évaluation de protocoles de routage. Je dresse ensuite l'état de l'art dans le domaine de la simulation de protocoles de routage pour l'Internet et je détaille plus particulièrement mes contributions dans le projet du simulateur DRMSim. En section 2, j'introduis les principes de résolution de problèmes à partir de la simulation. Puis, en section 3, je présente DRMSim, les objectifs du simulateur, son architecture et ses différents composants. En section 4, je détaille les choix d'implémentation du modèle de BGP dans DRMSim. Ce modèle sera ensuite utilisé dans le chapitre 5 pour l'étude de faisabilité de la distribution des simulations de BGP. Finalement, en section 5, je donne quelques exemples d'expérimentations menées avec le simulateur et je conclus en section 6 sur les travaux effectués dans DRMSim et ses perspectives d'évolution.

1 Introduction

Deux méthodes sont principalement utilisées pour évaluer les performances des schémas de routage. La première repose sur le principe d'émulation permettant des scénarios d'expérimentation particulièrement réalistes, ciblant des implémentations bas niveau des schémas de routage. La seconde utilise les techniques de simulation discrète ou continue. L'implémentation des schémas de routage peut être facilement adaptée en fonction des objectifs permettant notamment des expérimentations à grande échelle.

Ces méthodes d'expérimentation s'inscrivent dans un cycle plus global pour le développement de nouveaux protocoles de routage ou la vérification de nouvelles fonctionnalités dans le cadre d'un protocole existant. Ce cycle de développement débute par une phase théorique dans laquelle est décrit le nouveau schéma de routage ou une nouvelle fonctionnalité. Une évaluation par simulation à une échelle macroscopique est ensuite menée. Si la simulation valide le modèle théorique proposé, alors une spécification détaillée du modèle est proposée. La phase d'émulation permet une évaluation du modèle au niveau microscopique. C'est la dernière étape expérimentale avant la standardisation du protocole.

Les méthodes d'expérimentations par simulation et émulation sont donc complémentaires. Chacune permettant d'évaluer les aspects du protocole à différents niveaux de granularité.

1.1 Émulation du protocole BGP

L'*émulation* d'un matériel informatique, dans notre cas un routeur BGP, consiste à reproduire son comportement par un logiciel. Le comportement d'un routeur BGP étant caractérisé par son protocole de routage, on parlera par la suite de l'émulation du protocole de routage BGP. L'émulation de BGP permet d'analyser finement son comportement et d'évaluer précisément certaines métriques. Ces métriques peuvent être par exemple le temps d'établissement d'une nouvelle connexion, le temps de transmission d'un paquet ou encore le temps de décision de transmission (choix de l'interface de sortie). Une topologie est créée à partir de machines et de liens physiques. Une machine physique exécute un logiciel,

nommé *démon*, qui émule le comportement d'un protocole de routage au plus près de ses spécifications. On pourra citer le démon Zebra [Zeb] supportant les protocoles RIP, OSPF et BGP. Plus récemment, Quagga [Qua] a été développé à partir de Zebra. Il utilise l'API de Zebra permettant à un ensemble de modules clients, représentant les protocoles de routage, de communiquer avec le noyau Unix et accéder aux interfaces réseaux. Quagga fournit une API complète pour faciliter le développement de nouveaux modules clients. On pourra aussi citer BIRD [Bir], un autre démon dont l'implémentation BGP semble avantageusement remplacer celle de Quagga. Finalement, le démon OpenBGP [Opea] est disponible pour les plate-formes *BSD. On pourra se référer au rapport technique [DBI+11b] du projet Européen Euler [EUL14] dans lequel une comparaison détaillée des moyens existants pour l'émulation de BGP est disponible.

Si un seul démon est exécuté par machine, ce qui limite l'intérêt de l'approche, alors chacune des machines représente un nœud de la topologie simulée. Il est possible d'exécuter plusieurs démons sur une même machine grâce à la technologie de virtualisation du système d'exploitation et ainsi démultiplier la taille maximale des topologies. Différents projets tels que User Mode Linux [ULK], VNUML [VNU], VNX [VNX], NetKit [Net] ou encore Imunes [IMU] permettent d'exécuter jusqu'à une centaine de démons sur une même machine.

Afin de simplifier la mise en place de ces expérimentations, il est possible d'utiliser une plate-forme de test telle que celle de iLAB.t [iLa]. Cette dernière possède deux cents machines chacune équipée de 4 à 11 interfaces réseaux. Il est possible d'installer sur ces machines un nombre varié de systèmes d'exploitation et de mettre en place dynamiquement la topologie désirée grâce au logiciel Emulab [Emu]. Récemment, le projet Européen BonFIRE [Bon] a été mis en place afin de fédérer autour d'une même interface de contrôle sept plate-formes de test similaires à celle d'iLAB.t. Cependant, chacune des plate-formes de test ne fournit qu'une partie de ses machines. Ainsi, le projet BonFire met en permanence à disposition à travers les différentes plate-formes de test une centaine de machines et permet sur demande de réquisitionner plus de 300 machines.

Il existe donc un nombre varié de démons dans lesquels les principaux protocoles de routage utilisés en pratique sont fidèlement implémentés. Différentes infrastructures pour la mise en place des expérimentations sont aussi disponibles. Cependant, malgré les efforts pour regrouper ces plate-formes de test, le nombre de machines mis à disposition reste limité à quelques centaines, limitant au mieux l'émulation d'une topologie d'un millier de nœuds. Ce nombre est donc insuffisant dans le cadre d'expérimentations à grande échelle de nouveaux protocoles de routage.

De plus, la mise en place d'expérimentations pour l'émulation de BGP ou de nouveaux protocoles de routage demande un temps de préparation très important. De manière générale, le scénario d'émulation requiert le développement d'outils et la préparation des plate-formes d'émulation. Ces plate-formes sont le plus souvent partagées entre différents utilisateurs, les machines devant être réservées à l'avance. Ces machines sont ensuite configurées, la topologie désirée créée et l'exécution du scénario validée. Dans le cas d'expérimentations de nouveaux protocoles de routage, les spécifications de ces protocoles doivent être établies en détail. À partir de ces spécifications, une implémentation du protocole est développée dans l'environnement d'émulation.

Il n'est donc pas possible de reconstituer en laboratoire l'ensemble de l'architecture de routage de l'Internet. De plus, même sur des topologies de taille limitée, la mise en oeuvre d'expérimentations par émulation reste compliquée. L'émulation de BGP ou de nouveaux protocoles de routage à l'échelle microscopique sur des topologies de grande taille est donc impossible. Ainsi, il est nécessaire d'utiliser des outils de simulation pour évaluer ces nouveaux algorithmes à grande échelle.

1.2 Simulation de protocoles de routage

Afin de faire face aux limitations de BGP et évaluer de nouveaux paradigmes de routage, plusieurs projets pour le développement de plate-formes de simulation ont été initiés. Ces simulateurs peuvent être classés en trois catégories dont les principales caractéristiques sont résumées dans le tableau 4.1.

1. Les simulateurs de *protocoles de routage*. Ils permettent des simulations particulièrement réalistes mais sont limités en termes de taille de topologies.
2. Les simulateurs de *configurations de routage*. Ils permettent d'atteindre des tailles de topologies semblables à celle de l'Internet. Cependant, ils se concentrent sur l'évaluation du processus de sélection des routes et non sur les problèmes de convergences.
3. les simulateurs de *modèles de routage*. Ils simulent un modèle plus abstrait du protocole de routage et donc moins réaliste. Ils permettent la simulation de modèles de routage sur des topologies de grandes tailles.

Détaillons maintenant les différents logiciels de simulation disponibles pour ces deux catégories.

Simulateurs de protocoles de routage

Ces simulateurs sont conçus pour inclure les procédures et formats des protocoles au niveau microscopique, ils peuvent être répartis en deux sous-catégories :

— les simulateurs dédiés à BGP ;

— les simulateurs plus généraux qui permettent l'exécution de plusieurs protocoles de routage différents.

BGP++ [DR04]. Il appartient à la première sous-catégorie. Il repose sur le simulateur à événements discrets NS-2 [NS-] et propose une implémentation de BGP basée sur le démon BGP Zebra [Zeb]. Il est utilisé pour construire des scénarios particulièrement réalistes de simulation de routage inter-domaines. Cependant, l'exécution de procédures bas niveau empêche son utilisation sur de grandes topologies, celles-ci étant limitées à une centaine de nœuds.

NS-2, SSFNet [SSF] et J-Sim [JS]. Ces simulateurs à événements discrets appartiennent à la seconde sous-catégorie. SSFNET et J-Sim proposent une implémentation détaillée du protocole BGP. La machine à états finis et les compteurs du protocole sont entièrement modélisés. Une implémentation du protocole TCP/IP, sur laquelle repose les sessions BGP, est fournie.

Tout comme BGP++, l'implémentation bas niveau de BGP dans ces simulateurs requièrent d'importantes resources mémoire et processeur. La taille des topologies sur lesquelles BGP est simulé est donc du même ordre de grandeur que pour BGP++, une centaine de nœuds.

Simulateurs de configurations de routage

Ils incluent les spécificités du protocole BGP. Ils permettent de calculer le résultat du processus de sélection des routes dans BGP. Ce processus de sélection tient compte de la configuration des routeurs, des informations de routage et des caractéristiques de la topologie. Ces simulateurs peuvent être utilisés par les chercheurs ou les opérateurs réseaux pour évaluer l'impact d'une modification dans la configuration de BGP. Ces modifications peuvent concerner le processus de décision, l'ajout d'attributs BGP ou encore les changements topologiques ou logiques au niveau des tables de routage. Les changements topologiques représentent une présélection de liens ou de routeurs tombant en panne. Les

changements logiques représentent des modifications de la configuration des routeurs telles que la politique de routage en entrée ou sortie du routeur ou encore le poids des liens IGP. Ces simulateurs sont spécialisés et optimisés en termes de structures de données et procédures pour exécuter BGP. Des simulations sur de grandes topologies, du même ordre de grandeur que celle d'Internet, sont possibles.

C-BGP [QU05]. L'approche suivie par C-BGP est la simulation d'un modèle simplifié de BGP. Plusieurs choix sont à l'origine des possibilités de simulation à grande échelle de C-BGP.

1. Les couches sous-jacentes au transport de données ne sont pas modélisées. Plus particulièrement TCP, le protocole de transport utilisé pour établir les sessions BGP.
2. La machine à états finis de BGP est simplifiée, les messages d'ouvertures et de fermetures de sessions et de détection d'erreurs sont ignorés.
3. Les compteurs BGP ne sont pas modélisés (MRAI, RFD, CRT, HT, etc).
4. Aucune dynamique n'est possible au sein de la simulation (pas de délai de transmission).

Dans BGP, les messages de contrôle sont transmis de proches en proches. En ne considérant aucun délai de transmission, il est possible de simplifier la gestion des messages dans le simulateur. Ainsi, seule une queue FIFO est nécessaire. Celle-ci suffit à garantir la causalité des messages avec l'avantage de l'ajout et la suppression d'événements dans la queue en temps constant. L'ensemble de ces optimisations dans C-BGP permettent la simulation à grande échelle de BGP à partir de ressources de calcul limitées.

Ces simulateurs incluent l'ensemble du processus de décision de BGP. L'importation et l'exportation de filtres, de réflecteurs de route, le traitement d'attributs du chemin des AS ainsi que des règles spécifiques pour la sélection de route est possible. Ces simulateurs sont mis à jour pour incorporer au fur et à mesure les nouvelles fonctionnalités de BGP mais sont particulièrement complexes à utiliser hors du champs de BGP.

Simulateurs de modèles de routage

Seuls les modèles de routage sont considérés et non les protocoles, les procédures de bas niveau ne sont pas implémentées. Ce modèle doit être suffisamment simple pour être performant sur des topologies de grande taille tout en restant représentatif du protocole simulé. Cette nouvelle catégorie complète celles des simulateurs de protocoles de routage et de configurations de routage en permettant la simulation à grande échelle.

SimBGP [Sim]. Il se base sur le principe d'événements discrets et permet des simulations dynamiques dédiées à BGP sur des topologies du même ordre de grandeur que celle d'Internet. Des délais aléatoires de transmission et de traitement peuvent être configurés. Le compteur MRAI est modélisé.

DRMSim [DRM09]. Il permet la simulation de différents modèles de routage au sein d'une même plate-forme. Un modèle de BGP est disponible. La simulation de scénarios dynamiques est possible ainsi que la configuration de délais de transmission. Ce simulateur sera présenté en section 3.

Comparaison des simulateurs

La plupart des simulateurs proposent des modèles de simulation bas niveau. BGP++ est dédié à la simulation de BGP, les procédures et les structures de données sont optimisées.

Simulateurs		Taille des topologies	Dynamique temporelle	Dédié à BGP
Protocoles	SSFNet	$O(100)$	Oui	Non
	NS-2	$O(100)$	Oui	Non
	J-Sim	$O(100)$	Oui	Non
	BGP++	$O(100)$	Oui	Oui
Configurations	C-BGP	$O(10k)$	Non	Oui
Modèles	SimBGP	$O(10k)$	Oui	Oui
	DRMSim	$O(10k)$	Oui	Non

TABLE 4.1 – Comparaison des simulateurs de protocoles de routage.

BGP++, SSFNet ou encore J-Sim modélisent les couches bas niveau du modèle OSI. De ce fait, de tels simulateurs ne peuvent atteindre des topologies de grande taille. En pratique, ces simulations sont limitées à quelques centaines de nœuds. En simplifiant ces procédures, il est possible d'atteindre des topologies composées de plusieurs milliers de nœuds (SimBGP, C-BGP, DRMSim).

Au final, mis à part DRMSim, aucun simulateur ne permet l'expérimentation à grande échelle de différents schémas de routage au sein d'une même plate-forme de simulation dans des scénarios dynamiques.

1.3 Motivations pour le développement de DRMSim

La motivation principale pour le développement d'un nouveau simulateur est de proposer une plate-forme de simulation à événements discrets pour de multiples modèles de routage. Cette plate-forme de simulation doit pouvoir atteindre des topologies de plusieurs milliers de nœuds tout en conservant l'aspect dynamique des topologies et des modèles de routage. Le projet pour le développement du simulateur DRMSim a donc été initié sur ces bases.

La conception de DRMSim a débuté avant mon implication dans son développement. Une première version du simulateur était donc déjà disponible, validée par une première expérimentation [HPTM10]. Afin de clarifier mes contributions, je détaille tout d'abord l'état initial du simulateur à mon arrivée dans le projet. Je présente ensuite mes principales contributions. Celles-ci seront détaillées en section 3.

La version initiale de DRMSim

Deux modèles de routage sont fournis avec DRMSim. Un modèle simplifié du protocole de routage BGP et un modèle de routage compact dans les graphes k-chordaux [NRS12]. Tous deux ayant servis à la validation du simulateur dans [HPTM10]. Afin de faciliter le développement de nouveaux modèles de routage, diverses structures de base pour les tables de routage sont disponibles. Une API permet de gérer la transmission de messages émis par le modèle de routage simulé. Les délais de transmission sont calculés aléatoirement à partir d'un intervalle donné dans le fichier de configuration.

Les principales caractéristiques du simulateur sont :

— une architecture orientée objets ;

— l'initialisation de DRMSim à partir d'un fichier de configuration décrivant le scénario de simulation ;

— la gestion des événements discrets à partir d'une queue de priorité ;

— la possibilité d'automatiser plusieurs exécutions d'un même scénario pour l'évaluation statistique du modèle de routage simulé.

Un inconvénient majeur du simulateur à ce stade du développement était l'impossibilité d'étendre facilement ses possibilités. L'intégration de nouvelles fonctionnalités était particulièrement problématique et la prise de mesure lors des simulations devait être gérée spécifiquement pour chacun des modèles de routage.

Contributions au développement de DRMSim

Mes contributions dans le développement de DRMSim sont articulées autour de trois axes :

— le développement d'une architecture modulaire basée sur le principe des *composants* ;

— l'ajout de fonctionnalités afin d'étendre les possibilités d'intégration de nouveaux modèles de routage et de scénarios de simulation ;

— la possibilité d'une intégration aisée de contributions extérieures au projet.

Dans une architecture basée sur les composants, chacun des aspects de la simulation est séparé et géré par un composant distinct. Un composant regroupe les structures de données et procédures nécessaires afin d'assurer une fonction précise dans le logiciel. DRMSim possède par exemple un composant distinct pour la gestion des délais de transmission des messages, la prise de mesures, la dynamique de la topologie, la gestion de la topologie ou encore le contrôle du scénario de simulation. Ces composants peuvent être étendus par l'utilisateur afin de répondre à des besoins spécifiques. Cette possibilité permet la simulation de tous types de scénarios.

Ces évolutions ont permis une étude poussée des performances de divers modèles de routage compacts présentée dans [Gla13]. De nouvelles classes de modèles de routage sont aussi en cours d'intégration, notamment un modèle de routage géographique [KD14] et de routage sans boucle [DDFM12]. Enfin, à partir de ces améliorations, une étude présentée dans le chapitre 5 a pu être menée sur la faisabilité de la distribution des simulations de BGP.

2 Principes et méthodologie pour la simulation

Dans la suite, j'introduis le principe de processus de simulation pour la résolution de problèmes complexes. Je donne les définitions puis la terminologie nécessaire pour la suite de ce chapitre.

2.1 Définitions et terminologie

Le *processus de simulation* peut être vu comme la modélisation d'un système complexe pour caractériser son comportement dans le temps.

Définition 10 (Processus de simulation [Sha75])**.** *Un processus qui construit le modèle d'un système réel et qui mène des expériences sur ce modèle afin de comprendre le comportement du système ou d'évaluer différentes stratégies (en respectant les limites imposées par certains critères) pour le fonctionnement du système.*

On pourra définir la notion de *système* par un ensemble d'objets et leurs interactions. Cette définition peut être étendue aux événements extérieurs pouvant influer sur son comportement. Dans ce cas, ces événements formeront l'*environnement du système*. On définira l'*état du système* comme étant l'ensemble minimal d'informations à partir duquel le comportement futur du système pourra être prédit. Ces informations seront aussi appelées *variables d'état* du système. On appellera *activité*, un événement qui implique un changement dans l'état du système. Deux types d'activités existent, celles internes au système qui seront appelées activités *endogènes*, celles externes au système qui seront appelées activités *exogènes*.

L'analyse, la compréhension et la caractérisation du comportement d'un système complexe par la simulation amène la possibilité de dériver de nouvelles théories ou hypothèses et prédire les effets produits par un changement dans le système simulé.

2.2 Modèle de simulation d'un système

Un système peut être classifié en plusieurs catégories. Celles-ci sont déterminées en fonction des transitions d'un état à un autre ou suivant la présence d'activités et leur nature temporelle.

Une simulation sera *déterministe* ou *stochastique* suivant la nature des transitions des états du système. Une simulation sera déterministe si un nouvel état du système dépend entièrement de l'état précédent et de l'activité ayant engendré cette transition. Elle sera *stochastique* si la transition entre deux états dépend d'une certaine probabilité, l'état du système suivant une activité ne peut donc être prédit.

La notion de simulation *continue* ou *discrète* se réfère à la nature temporelle des changements des variables d'état dans le système. Une simulation sera continue si les variables d'état du système évoluent de façon continue. Le nombre d'états du système est donc infini. Ces variables sont généralement décrites par des équations différentielles. Dans le cadre de simulations à événements discrets, les variables d'état changent instantanément à chaque occurrence d'un événement représentant une activité du système.

2.3 Simulation à événements discrets

En simulation à événements discrets, le fonctionnement d'un système est représenté par une séquence chronologique d'événements associés à tout changement dans l'état du système. Trois structures de données sont nécessaires :

— un ensemble de variables d'état ;

— une liste d'événements ;

— une horloge globale.

Un événement est une structure de données. Elle inclut le temps d'occurrence de l'événement, que nous appellerons *estampille* et la procédure à exécuter. Le modèle de simulation inclut la gestion des événements discrets à partir d'une queue de priorité et d'une horloge globale.

Le principe de simulation à événements discrets (DES) se caractérise par un modèle :

— déterministe ou stochastique, certaines variables d'état peuvent être aléatoires ;

— dynamique, qui évoluent dans le temps ;

— à événements discrets, les variables d'état changent instantanément à des moments distincts dans le temps.

De façon similaire, les réseaux de communication se caractérisent aussi par leur dynamique représentée par des activités exogènes (ajout et suppression de nœuds ou liens, évolution des débits, etc) et une évolution temporelle discrète des activités du réseaux. La simulation à événements discrets est donc particulièrement bien adaptée pour la simulation des réseaux de communication. Comme vu en section 1, de nombreux simulateurs pour l'évaluation de BGP ou de nouveaux schémas de routage se basent sur ce principe.

3 DRMSim, un simulateur de modèles de routage dynamiques

DRMSim est un simulateur de modèles de routage dynamiques à événements discrets écrit en Java. C'est un logiciel libre et disponible sur [DRM09]. Ce simulateur propose une plate-forme commune à l'expérimentation et l'évaluation statistiques des performances de divers modèles de routage sur de grandes topologies.

Dans cette section, je présente les choix architecturaux que j'ai mis en oeuvre dans le cadre de l'évolution du simulateur. Les modèles de simulation et système présentés par la suite ont été repris et remaniés afin de servir de base au développement d'une nouvelle version de DRMSim.

3.1 Motivations et objectifs

DRMSim a été développé avec comme principaux objectifs :

1. être une plate-forme de simulation pour de multiples modèles de routage ;
2. simuler des modèles de routage sur des topologies composées de plusieurs milliers de nœuds ;
3. simuler des scénarios dynamiques à partir d'activités exogènes ;
4. être modulaire et facilement extensible à l'aide de composants ;
5. automatiser les mesures de performance des modèles de routage.

La combinaison des trois premiers objectifs permet d'offrir une nouvelle catégorie de simulateur (voir tableau 4.1). Le chercheur peut se concentrer sur les aspects essentiels de son prototype, c'est-à-dire développer uniquement les procédures et structures de données propres à son prototype. Détaillons maintenant chacun de ces objectifs.

Une plate-forme de simulation multi-modèles

Une API claire et extensible doit faciliter le développement et l'expérimentation de modèles de routage variés. Différents modèles de routage sont actuellement supportés tels que ceux à vecteurs de distances ou de chemins, de routage compact ou encore géographique.

Des exemples d'expérimentations dans les graphes k-chordaux ou à partir de modèles de routage compacts universels ou dédiés sont donnés en section 5.

Des modèles de routage pour la simulation à grande échelle

DRMSim simule un modèle abstrait du protocole de routage. Les procédures bas niveau du protocole ne sont pas exécutées et les formats simplifiés. Un modèle abstrait de procédure, de données et d'état les remplace. Les choix de simplification de ces modèles sont laissés au développeur. Seule une API fournissant diverses structures de données et méthodes pour la transmission de données entre les nœuds de la topologie doit être fournie.

Des scénarios dynamiques et des activités exogènes

Différentes activités exogènes doivent être supportées nativement par DRMSim. Celles-ci peuvent être planifiées à partir de la configuration du simulateur. L'utilisateur doit pouvoir développer ses propres activités exogènes. Le simulateur doit permettre au minimum :

— l'ajout et la suppression de nœuds ;

— l'ajout et la suppression de liens ;

— la modification des délais de transmission affectés aux liens.

Des scénarios de simulation doivent pouvoir être définis par l'utilisateur afin de planifier précisément l'exécution des activités exogènes et contrôler le déroulement de la simulation.

Une architecture modulaire et extensible

Afin de répondre à un maximum de besoins, l'architecture du simulateur doit pouvoir évoluer et intégrer facilement de nouveaux composants. Ces mêmes composants peuvent eux-mêmes être complétés par de nouvelles fonctionnalités définies par l'utilisateur. Ainsi, de nouveaux modèles de routage, scénarios de simulation, librairies de graphe ou encore de nouvelles métriques doivent pouvoir être intégrés sans avoir à modifier le coeur du simulateur.

L'automatisation des mesures de performance

DRMSim doit intégrer nativement un ensemble de métriques pour l'évaluation des performances des modèles de routage. Celles-ci doivent se composer de :

— l'étirement des chemins (stretch) ;

— la taille des tables de routage ;

— le nombre et la taille des messages de contrôle nécessaires au modèle simulé pour converger ;

— le nombre et la taille des messages de contrôle nécessaires pour converger après une activité exogène.

La conception modulaire de DRMSim doit permettre l'intégration de nouvelles métriques définies par l'utilisateur.

3.2 Architecture de DRMSim

L'architecture globale de DRMSim permettant de remplir les objectifs cités en section 3.1 est illustrée en figure 4.2 (p. 82). Par la suite, nous décrirons cette architecture par les termes de *modèle* et de *composant*. Un modèle étant une description abstraite d'une fonction du simulateur et de ses entrées et sorties. Nous pourrons nous référer à la définition de *modèle fonctionnel* donnée en section 3.2 (p. 65) pour davantage de détails. Un composant assurera quant à lui l'exécution de la fonction du modèle. L'architecture de DRMSim comprend six modèles, chacun décrivant un aspect du simulateur. Chaque modèle est associé à un composant chargé de son exécution :

— le modèle de simulation, initie le scénario de simulation et exécute les événements discrets ;

— le modèle du système, génère la topologie, exécute les instructions d'ajout et suppression de nœuds et liens, transmet les messages entre les nœuds ;

— le modèle de dynamique, génère les événements discrets représentant les activités exogènes définies dans le scénario de simulation ;

— le modèle de transmission, calcule les délais de transmission des messages entre les nœuds de la topologie et modifie les délais de transmission des liens en fonction du scénario de simulation ;

— le modèle de routage, calcule les informations de routage d'un paquet d'un nœud source à un nœud destination ;

— le modèle de mesure, définit les métriques, mesure les performances du modèle de routage et génère les fichiers de mesure.

Cette segmentation des fonctionnalités permet leur évolution indépendamment des autres. Un nouveau modèle et son composant associé peuvent être facilement ajoutés au simulateur. Seules les interactions avec les autres composants sont à développer. L'architecture globale de DRMSim n'ayant pas besoin d'être modifiée.

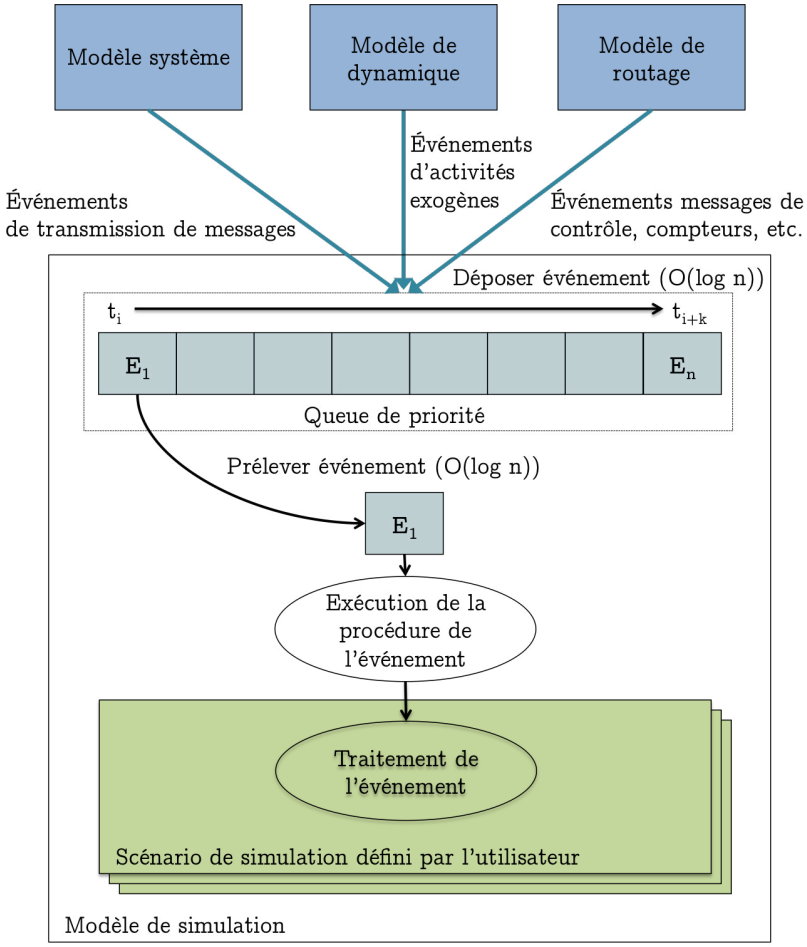


FIGURE 4.1 – Modèle de simulation.

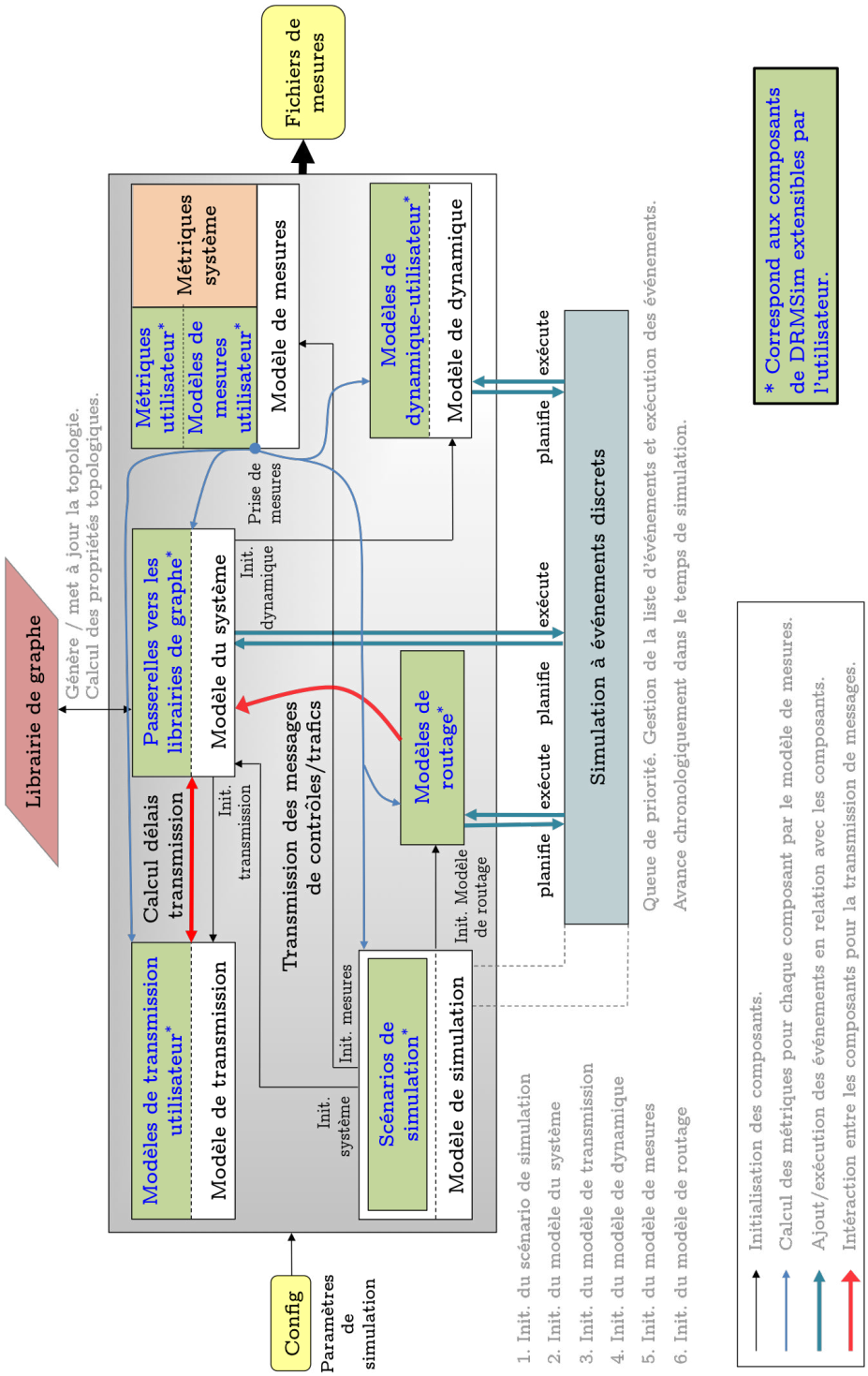


FIGURE 4.2 – Architecture de DRMSim

Modèle de simulation

Détaillons chacun des modèles de l'architecture de DRMSim et leurs relations.

Ce modèle est représenté en figure 4.1 (p. 81). DRMSim se base sur le principe de simulation à événements discrets. Le dépôt et le prélèvement d'événements se fait en temps $O(log(n))$. Lorsque l'événement de plus petite estampille est prélevé, sa procédure est exécutée. Celle-ci peut modifier les variables d'état du simulateur et créer de nouveaux événements qui seront déposés dans la queue de priorité. Ces événements représentent les activités endogènes et exogènes du système simulé. Trois modèles peuvent être source d'activités et donc déposer de nouveaux événements dans la queue :

— le modèle du système, lors de la transmission de messages entre les nœuds de la topologie ;

— le modèle de dynamique, lors de la génération des activités exogènes ;

— le modèle de routage, par l'envoi de messages via le modèle du système ou leur dépôt directement dans la queue de priorité et lors de la génération de compteurs.

Une fois la procédure exécutée, l'événement est traité par le scénario de simulation défini par l'utilisateur. Cette possibilité de personnalisation du traitement au cas par cas des événements permet à l'utilisateur de contrôler finement le déroulement de la simulation.

Un exemple de scénario est la simulation de BGP jusqu'à convergence avec un modèle de dynamique configuré pour mettre hors service un ensemble de liens et de nœuds à un temps défini pendant l'exécution de BGP. La détection de convergence est permise en contrôlant le nombre de messages de mise à jour émis entre deux instances de MRAI (voir section 4 pour d'avantage d'informations sur l'implémentation de BGP dans DRMSim).

Lors de l'initialisation du simulateur, le modèle de simulation est le premier à être chargé en mémoire. Les modèles suivants sont ensuite initialisés par le modèle de simulation :

1. le modèle du système ;
2. le modèle de mesure ;
3. le modèle de routage ;
4. le scénario de simulation.

Le fichier de configuration lu par le modèle de simulation est ensuite transmis aux différents modèles à leur initialisation. Le scénario est le dernier modèle à être chargé, c'est lui qui contrôlera le début de la simulation.

Modélisation du système

Le modèle du système est illustré en figure 4.3. Il permet le contrôle de la topologie et la transmission de messages entre les nœuds.

Par l'intermédiaire d'une librairie de graphe, le réseau est créé, les informations topologiques calculées (par exemple la matrice des plus courts chemins) et des modifications structurelles telles que la suppression de liens ou de nœuds effectuées. Les opérations de contrôle de la topologie sont accessibles via une interface nommée *Topologie*. Cette interface de contrôle est notamment utilisée par les activités exogènes prévues dans le scénario de simulation. Cette interface se base sur des passerelles pour communiquer avec les librairies de graphe. Une passerelle doit être développée pour chaque librairie de graphe. Cette passerelle définit les commandes déclarées dans l'interface de contrôle nécessaires au fonctionnement de DRMSim. Ainsi, une nouvelle librairie de graphe peut être intégrée à DRMSim par le développement de la passerelle associée.

Outre la gestion de la topologie, le modèle du système permet l'accès aux propriétés du système. Ces propriétés peuvent être celles calculées à partir de la librairie de graphe, ou peuvent être celles directement chargées à partir de sources extérieures (fichiers) fournies par l'utilisateur.

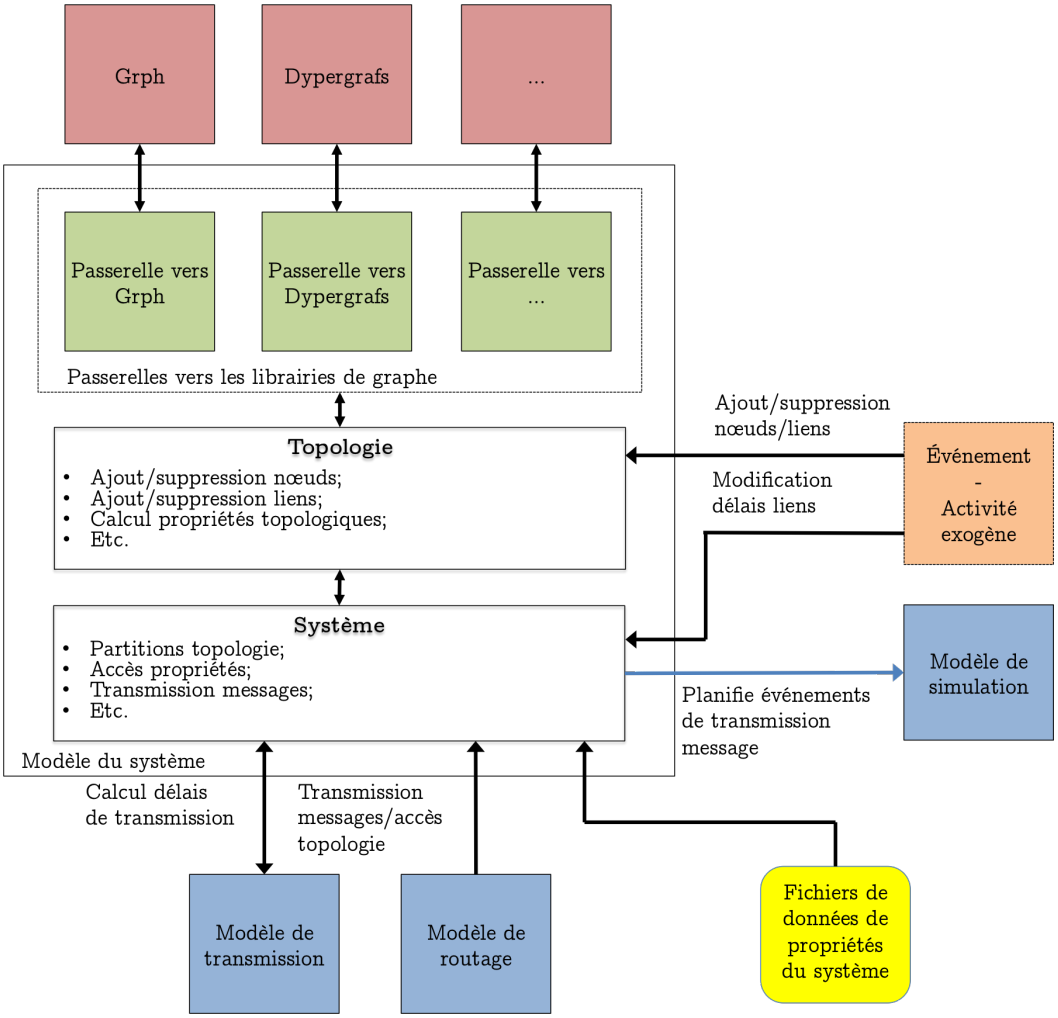


FIGURE 4.3 – Modèle du système.

Le modèle du système assure la transmission des messages entre les nœuds de la topologie. Un message est composé d'une en-tête et d'un contenu. Ce contenu peut être personnalisé selon les besoins du modèle de routage. Il est aussi possible d'accéder au lien et au nœud dont provient le message. L'en-tête de routage est composée du nœud d'origine du message, de l'ensemble des destinations, d'une mesure de temps à vivre (TTL).

Le modèle de transmission calcule le délais de transmission d'un message à travers un lien. Un événement représentant la transmission du message à travers le lien est planifié. La date d'exécution de l'événement est calculée à partir de la somme de la date courante de simulation et du délai de transmission. En cas de dynamique de la topologie, si le lien ou le routeur destination devient hors service pendant le transit du message, alors un événement d'échec d'envoi du message est notifié. Si le message est bien reçu par un nœud relais, alors le TTL est décrémenté de un. Si cette valeur est à zéro avant d'arriver au nœud destination, alors un événement d'échec d'envoi du message est notifié.

Lors de l'initialisation du système, les modèles de transmission et de dynamique sont chargés en mémoire.

Modèle de transmission

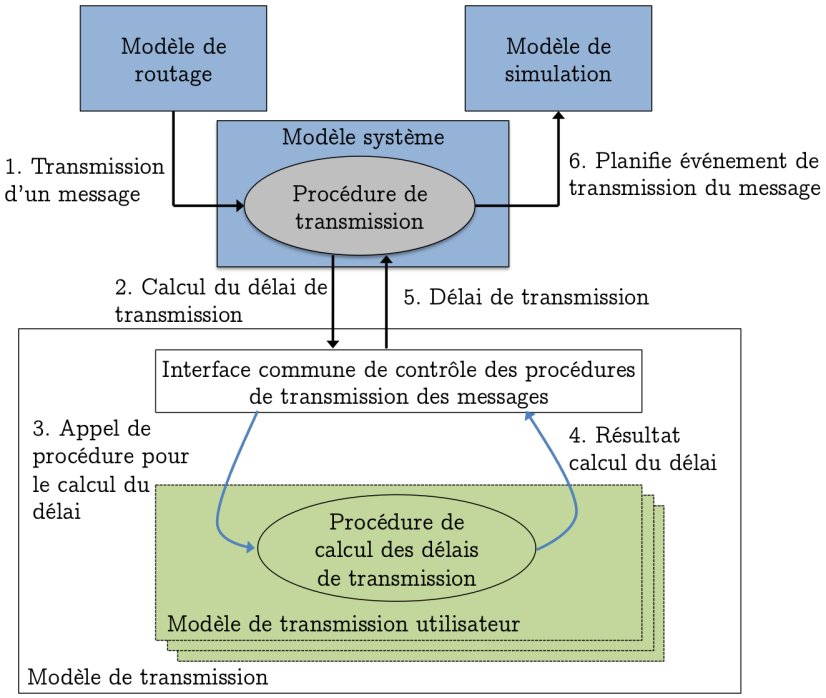


FIGURE 4.4 – Modèle de transmission et séquence de transmission d'un message.

Le modèle de transmission et la séquence de transmission d'un message sont illustrés en figure 4.4. Le modèle de transmission permet de calculer le délai de transmission d'un message. L'interface de contrôle des procédures de transmission des messages est appelée par le modèle du système. Cette interface se base sur la procédure de transmission chargée à l'initialisation du simulateur. Plusieurs procédures peuvent être développées mais une seule est chargée par le simulateur. Ces procédures prennent en paramètre le message, le nœud relais et le lien par lequel le message doit transiter. À partir de ces informations, le délai de transmission du message est calculé. DRMSim est actuellement composé de trois procédures.

1. Calcul du délai à partir de la capacité du lien. Le lien est considéré comme bidirectionnel. À chaque lien sont attribués une capacité en bits par seconde et un nombre de bits en cours de transmission pour chacune des directions de transmission. Le délai est ensuite calculé à partir de la taille en bits du message suivant la formule :

$$délai = \frac{bits\ en\ cours\ de\ transmission + taille\ du\ message}{capacité}.$$

2. Calcul aléatoire des délais. À l'initialisation de la simulation, une valeur est affectée à chacun des liens. Ce délai est constitué de l'addition de deux valeurs décimales. La première valeur est constante et la seconde est calculée aléatoirement et uniformément dans un intervalle spécifié dans la configuration de la simulation.
3. Calcul du délai à partir de la capacité du lien en fonction de l'appartenance des nœuds à un groupe. La capacité des liens est déterminée selon le groupe auquel appartient les nœuds à leur extrémité. La première procédure est ensuite appliquée pour calculer le délai.

Modèle de dynamique

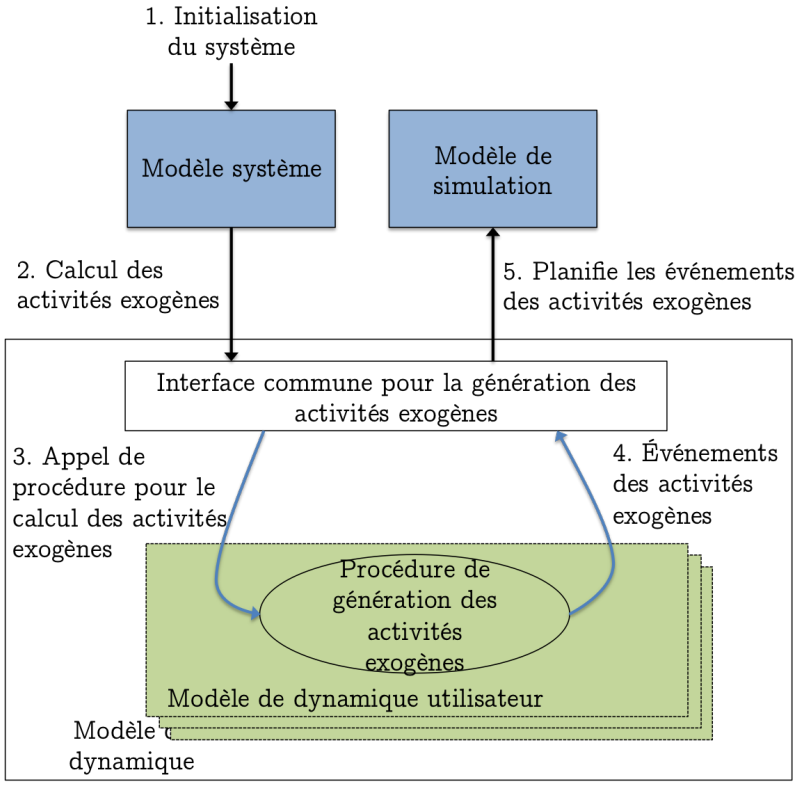


FIGURE 4.5 – Modèle de dynamique et séquence de génération des activités exogènes.

Le modèle de dynamique et la séquence de génération des activités exogènes sont illustrés en figure 4.5. Le modèle de dynamique permet de planifier les événements représentant des activités exogènes dans le système simulé. Ces activités représentent typiquement les opérations de maintenance sur l'infrastructure réseau. La planification de ces événements est exécutée une fois le système initialisé. L'interface de génération des activités exogènes est appelée par le système. Cette interface se base sur une procédure de calcul des activités exogènes chargée à l'initialisation du simulateur. Plusieurs procédures peuvent être développées mais une seule est chargée par le simulateur. Celle-ci retourne l'ensemble des événements représentant les activités exogènes de la simulation. Ces événements sont ensuite planifiés dans la queue de priorité du simulateur.

Une procédure de génération d'activités exogènes est fournie dans DRMSim. Elle génère les événements de mise hors service ou de rétablissement des nœuds ou liens dans la topologie. Quatre activités exogènes sont représentées :

— la mise hors ligne d'un nœud ;

— la mise hors ligne d'un lien ;

— le rétablissement d'un nœud précédemment mis hors ligne ;

— le rétablissement d'un lien précédemment mis hors ligne.

Les intervalles de temps durant lesquels un lien ou un routeur est mis hors service puis rétabli sont distribués aléatoirement. Plusieurs types de distribution sont possibles telles que les distributions uniformes, binomiales, de poisson, etc. Les paramètres de ces distributions

ainsi que la date de début et de fin des activités exogènes sont donnés dans la configuration de la simulation.

Les intervalles de temps sont ensuite mémorisées par le modèle de dynamique et utilisées par le système pour déterminer la faisabilité de la transmission d'un message.

Modèle de mesure

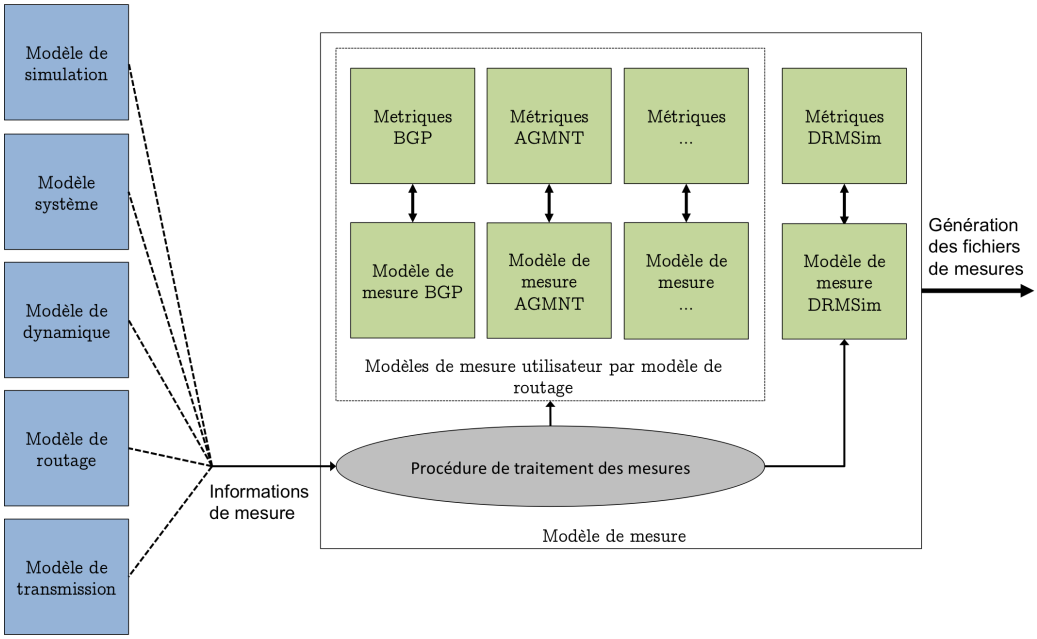


FIGURE 4.6 – Modèle de mesure.

Le modèle de mesure est illustré en figure 4.6. Il se met à l'écoute des modèles de simulation, du système, de dynamique, de routage et de transmission pour la prise de mesures. Ces informations de mesures sont ensuite traitées par une procédure. Cette procédure aiguille les informations vers le modèle de mesure défini par l'utilisateur et celui de DRMSim. En effet, le modèle de mesure de DRMSim calcule automatiquement un certain nombre de métriques.

- L'étirement multiplicatif et additif des chemins de routage. Ils sont respectivement définis par le ratio et la différence de la longueur de la route suivie pour le routage d'un paquet par la longueur optimale.
- La taille mémoire occupée par les tables de routage du modèle de routage.
- Le nombre et la taille des messages de contrôle nécessaires pour converger.
- Le temps de convergence.

Ces métriques sont calculées globalement pour l'ensemble des nœuds de la topologie à la fin de la simulation. Pour chaque modèle de routage, l'utilisateur de DRMSim peut définir son modèle de mesure et ses métriques associées. Les informations de mesures sont automatiquement redirigées vers le modèle considéré.

Les besoins en mémoire et puissance de calcul dépendent du type de mesures à effectuer lors de la simulation. Le nombre de nœuds et de liens pour lesquels les métriques doivent être calculées, l'intervalle de temps de prise de mesures et leur complexité de calcul sont déterminants. Il est ainsi possible de sélectionner un sous-ensemble de nœuds et de liens

pour les prises de mesure. En cas de partition de la topologie, des métriques spécifiques pour les nœuds et liens frontières peuvent être calculées.

En fin de simulation, l'ensemble des mesures sont écrites dans des fichiers distincts.

Modèle de routage

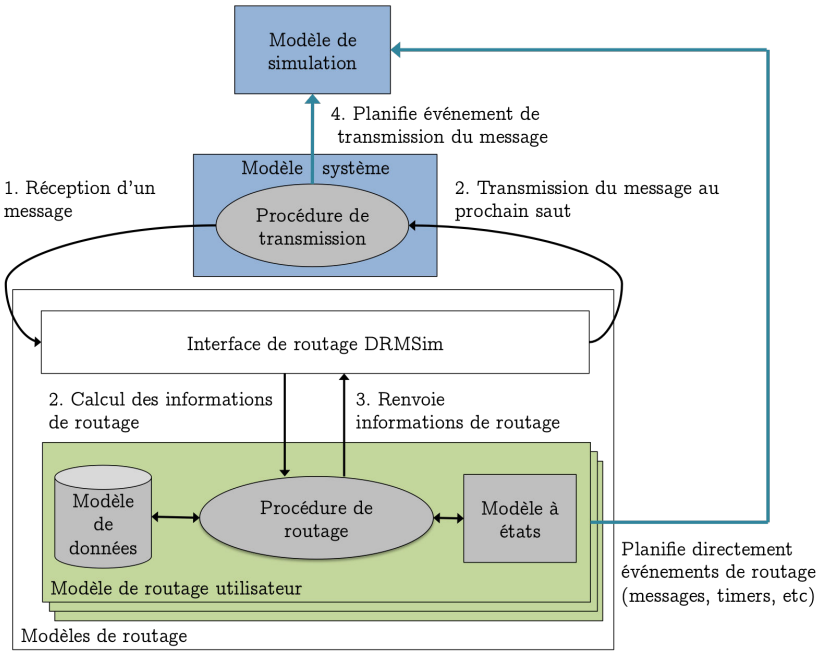


FIGURE 4.7 – Modèle de routage et séquence de transmission d'un message.

Le modèle de routage et la séquence de transmission d'un message sont illustrés en figure 4.7. Chaque modèle de routage comprend une procédure de routage, un modèle de données et un modèle de communication à simuler. DRMSim fournit un ensemble de modèles de routage basiques (routage à la source, routage aléatoire, diffusion, etc). Ces modèles servent à la vérification du simulateur et de référence pour la comparaison des performances de nouveaux modèles. D'autres modèles, utilisés de nos jours dans les réseaux ou issus de travaux de recherche récents, sont inclus :

- un modèle du protocole de routeur frontière BGP [RLH06] ;
- un modèle du protocole d'information de routage RIP [Mal98] ;
- un modèle de routage compact dans les graphes k-chordaux [NRS12] ;
- un modèle de routage géographique [KD14].

Une interface de routage est fournie par DRMSim permettant au modèle de routage de transmettre des messages de contrôle ou de trafics entre les nœuds de la topologie. Cette interface se base sur le modèle du système pour la transmission de ces messages en fonction du délai des liens. Le modèle de simulation est ensuite appelé pour la planification des événements associés à ces messages. Il est possible de planifier directement des événements à partir du modèle de routage. Cela est particulièrement utile dans le cadre de compteurs. Il est aussi possible d'utiliser cette méthode pour simplifier la transmission de messages entre nœuds. Dans ce cas, le modèle de transmission est ignoré et ces messages ne sont pas comptabilisés dans les métriques de DRMSim. Des métriques utilisateurs pouvant être définies afin de les prendre en compte.

3.3 Exemple d'exécution

Une simulation est exécutée à partir d'un fichier de configuration décrivant les paramètres de chacun des composants. Ce fichier est donné en argument de la commande d'exécution de DRMSim. Prenons l'exemple d'une simulation de BGP. Le scénario de simulation doit exécuter le modèle de routage de BGP jusqu'à ce que celui-ci ait convergé. La topologie doit être chargée à partir d'un fichier et un modèle de mesure spécifique à BGP est utilisé. Un exemple partiel du fichier de configuration est donné dans le listing 4.1. Les clés de configuration et leurs valeurs associées sont regroupées en sections et sous-sections. Quatre clés particulières permettent de charger les composants nécessaires à l'exécution de cette simulation.

— `simulation-class` (ligne 3), permet de charger le scénario de simulation. Dans notre exemple, celui-ci est spécifique à BGP et permet d'arrêter la simulation une fois que BGP a convergé. La section `logging` (ligne 8) permet quant à elle l'écriture automatique des journaux de simulation pouvant contenir le détail des événements exécutés durant la simulation.

— `user-metric-model-class` (ligne 15), est le modèle de mesure spécifique à BGP. Il permet de mesurer le nombre de messages de mise à jour émis et leur taille et, si nécessaire, le nombre d'instances de MRAI durant la simulation. Il est complété par `user-metric-class`, décrivant chacune des métriques de BGP et par `result-class` pour l'écriture des fichiers de mesures en fin de simulation.

— `routing-model-class` (ligne 22), est le modèle de routage à simuler, dans notre cas BGP. Des paramètres spécifiques à BGP peuvent être décrits dans le fichier de configuration. Ici, la simulation de BGP sera avec compteur MRAI (ligne 25). La valeur de certains compteurs peut être spécifiée (lignes 27 à 29) ainsi que les délais de transmission des messages UPDATE et d'établissement des sessions (lignes 31 à 34).

— `topology-bridge-class` (ligne 24), permet de spécifier la passerelle utilisée pour créer la topologie. La clé `load-graph` permettant de charger une topologie à partir d'un fichier.

Le contenu du fichier de configuration est transmis à chacun des composants de DRMSim. Un utilisateur peut définir ses propres sections et entrées dans le fichier de configuration et les utiliser pour paramétrer les modèles et procédures qu'il a défini lors de leur initialisation.

Ces composants et procédures sont chargés en mémoire dans un ordre précis lors de l'exécution de DRMSim.

1. Le modèle de simulation et le scénario associé sont d'abord initialisés.
2. Le modèle de simulation initialise ensuite le modèle du système. La topologie est créée et les paramètres externes sont chargés en mémoire à partir de la configuration donnée dans la section `topology` du listing.
3. Les modèles de transmission et de dynamique sont ensuite initialisés par le système. Les événements représentant les activités exogènes du système sont planifiés dans la queue de priorité.
4. Le modèle de mesure est initialisé par le modèle de simulation.
5. Enfin, le modèle de routage est initialisé par le modèle de simulation à partir des paramètres donnés en section `routing-model` de la configuration. Le scénario de simulation est ensuite exécuté.

Simulation déterministe. En spécifiant une valeur d'initialisation au générateur de nombres aléatoires, DRMSim permet une exécution déterministe des simulations. Cette valeur, aussi appelée *graine*, est spécifiée dans le fichier de configuration par la clé `seed`

```
1  section simulation
2    section experimentation
3      simulation-class={bgp.scenario.RunsUntilConvergence}
4      number-of-runs={1}
5      seed={1}
6    end of section
7
8    section logging
9      print-debug-events={false}
10     log-file={false}
11   end of section
12
13   section metric
14     result-class={org.drmsim.DrmResult}
15     user-metric-model-class={bgp.metric.BGPMetricModel}
16     user-metric-class={bgp.metric.BGP_METRIC}
17     metric-model-settings={true, false, true, false, true}
18   end of section
19 end of section
20
21 section routing-model
22   routing-model-class={bgp.BGPRoutingAlgorithm}
23
24   section bgp
25     mrai={true}
26
27     mrai-delay={30000}
28     keep-alive-delay={30000}
29     hold-timer-delay={90000}
30
31     update-random-delay={100}
32     update-constant-delay={1}
33     session-random-delay={0.01}
34     session-constant-delay={0}
35   end of section
36 end of section
37
38 section topology
39   topology-bridge-class={org.drmsim.system.bridge.grph.
       DrmGrphBridge}
40   load-graph={line, Demo/topologies/1000.glp}
41 end of section
```

Listing 4.1 – Extrait partiel d'un fichier de configuration pour la simulation de BGP jusqu'à convergence sans compteur MRAI.

(ligne 5 du listing 4.1). Ainsi, pour une même configuration, les résultats de l'exécution de plusieurs simulations seront identiques. En modifiant cette valeur, la part aléatoire des délais de transmission, la planification des activités exogènes ou toutes procédures utilisant le générateur de nombres aléatoires de DRMSim verront leurs résultats modifiés.

Calcul statistique. La variation de la valeur de la graine est automatisée lors de l'exécution d'une série de simulations identiques. Il est ainsi possible d'obtenir une évaluation statistique des performances d'un modèle de routage pour un scénario donné. La clé `number-of-run` (ligne 4 du listing 4.1) permet la configuration du nombre d'exécutions d'une même configuration. Chaque exécution se fera à partir d'une valeur différente de la graine. Les fichiers de mesures de chacune des simulations seront écrits dans un répertoire distinct et leur moyenne automatiquement calculée.

4 Le protocole de routeur frontière dans DRMSim

Comme expliqué en section 3 du chapitre 3, l'architecture de routage dans l'Internet montre ses limites. Plus particulièrement BGP, le protocole de routeur frontière utilisé par les systèmes autonomes (AS) de l'Internet pour communiquer. Les routeurs BGP échangent entre eux des informations de routage depuis lesquels la construction d'un graphe d'AS sans boucle est possible. Les informations échangées par BGP permettent d'effectuer un routage basé uniquement sur l'adresse de destination d'un paquet. Chaque AS a la possibilité d'appliquer des politiques de routage. Il est donc souhaitable de pouvoir évaluer les performances et le comportement de BGP.

BGP est un protocole de routage complexe, sa simulation requiert d'importantes ressources mémoire et processeur. Différents aspects de BGP sont à l'origine de cette complexité.

— Afin de garantir un routage sans boucle, chaque routeur BGP doit garder dans sa table de routage le chemin calculé pour chaque destination. Le coût en mémoire d'une table de routage est en $O(n.k.log(n))$ bits avec k la taille d'un chemin et n le nombre de préfixes destination.

— Les politiques de routage imposent l'exécution d'une procédure, coûteuse en temps de calcul, à la réception et à l'envoi d'un message de contrôle.

— Le nombre de messages de contrôle pour calculer les tables de routage peut être important. Ces messages contiennent les chemins vers les destination annoncées. Un grand nombre de messages peut être cumulé dans la queue du simulateur. Dans le pire cas, un nombre exponentiel de messages peut être requis pour la convergence du protocole.

D'après l'état de l'art dressé en section 1, à l'exception de C-BGP et de SimBGP, aucun simulateur ne permet d'atteindre des topologies composées de plusieurs milliers de nœuds. Ces deux simulateurs reposent sur le développement d'un modèle de BGP. Les procédures et variables d'états sont simplifiées afin de réduire les ressources nécessaires à la simulation du protocole. Seuls certains aspects précis du protocole sont simulés.

Cette approche est similaire à celle suivie dans DRMSim avec le développement de modèles de routage. DRMSim se démarquera de C-BGP par la simulation des aspects de dynamiques temporelles du protocole et de la topologie. Ces aspects sont pris en compte dans SimBGP, mais ce dernier se concentre sur la simulation de BGP quand DRMSim propose une plate-forme commune à la simulation de différents modèles de routage.

4.1 Description du modèle

Une première version du modèle de BGP était initialement disponible dans DRMSim. Le développement de cette version auquel je n'ai pas participé a permis l'exécution des expérimentations présentées :

— en section 4.4, pour la validation de BGP dans DRMSim ;

— en section 5.1, pour l'évaluation et la comparaison avec BGP du modèle de routage compact dans les graphes k-chordaux.

J'ai repris cette implémentation de BGP afin de l'intégrer dans la nouvelle version du simulateur. Mon travail s'est principalement concentré sur son optimisation et la possibilité d'évaluer le coût des communications dans une partition de la topologie. Cette nouvelle implémentation de BGP sera utilisée dans le cadre de l'étude présentée en chapitre 5 sur la faisabilité de la distribution des simulations de BGP.

Modélisation des informations de routage. Le modèle utilisé dans DRMSim pour les simulations de BGP se base sur une topologie des AS : chaque nœud représente un système autonome. Seule la version inter-AS du protocole de routeur frontière, aussi appelé eBGP (External Border Gateway Protocol), sera simulée. BGP segmente sa table de routage RIB (Routing Information Base) en trois structures logiques (voir la figure 3.1 de l'architecture d'un routeur BGP en section 1.2 du chapitre 3) :

— Loc-RIB, contient toutes les routes sélectionnées localement par le routeur et qui sont utilisées par la table de transmission ;

— Adj-RIB-in, permet d'appliquer un filtre sur les informations de routage reçues en fonction des voisins du routeur ;

— Adj-RIB-out, permet d'appliquer un filtre sur les informations de routage envoyées aux voisins du routeur.

Afin de simplifier les calculs effectués au niveau des nœuds, seule la Loc-RIB est implémentée dans DRMSim. Seul le numéro d'AS et sa route associée sont stockés dans la table. Aucun attribut BGP n'est stocké par défaut. Cependant, il est possible d'en ajouter de nouveaux si nécessaire.

Simplifications du modèle. Comme expliqué en début de section, la simulation de BGP requiert d'importantes ressources en termes de mémoire et temps de calcul. Différentes techniques de simplification du protocole ont donc été appliquées.

— Réduction du nombre de messages en ne considérant que les deux états possibles IDLE ou ESTABLISHED. Le temps d'établissement des sessions BGP est réduit.

— Utilisation uniquement de la table de routage. La table de transmission pour le routage d'un paquet est omise et remplacée par une fonction de calcul à la demande de la prochaine étape.

— Une structure efficace pour implémenter la table de routage. Les AS constituant le réseau sont identifiés en interne par un entier de 1 à n (avec n le nombre d'AS dans le réseau). Chacune des entrées est indexée dans la table de routage par l'identifiant de la destination.

— Utilisation de vecteur de bits pour déterminer si une route existe pour une destination donnée. Permet grâce à des opérations bit à bit rapides et efficaces de déterminer si une entrée reçue dans un message de mise à jour BGP est utilisée ou non.

— Simulation uniquement des compteurs MRAI et HoldTimer ;

— Simulation uniquement des messages de mise à jour (UPDATE) pour l'annonce de nouvelles routes ou de routes à retirer.

La simplification du modèle de BGP dans DRMSim ne permet pas l'évaluation des politiques de routage. L'intégration de cette fonctionnalité est prévue dans les développements futurs de DRMSim.

4.2 Transmission des messages de mise à jour BGP

DRMSim propose un modèle de transmission des messages d'un routeur source à un routeur destination. Ce modèle impose une certaine complexité en terme de procédures à exécuter pour garantir l'ordre de transmission des messages suivant le délai des liens et la dynamique de la topologie. Ces messages doivent suivre un chemin lui-même calculé par le modèle de routage. L'en-tête de routage des messages impose un coût mémoire supplémentaire.

Simplification des transmissions. Dans le cas d'un scénario sans ajout ou suppression de liens ou de routeurs dans la topologie, le fonctionnement de BGP permet de simplifier les procédures de transmission des messages de contrôle. Les temps d'exécution de procédures superflues pour le routage du message et la gestion de la dynamique sont évités. La mémoire requise est limitée aux stricts besoins de BGP.

En effet, les messages de contrôle BGP sont transmis de proche en proche. L'en-tête de routage dans le message et l'exécution des procédures pour le calcul d'une route ne sont pas nécessaires. Seul l'ordre d'arrivée des messages et le délai de transmission du lien vers le voisin doivent être considérés. DRMSim permet la planification directe de tout type d'événement dans la queue de priorité. Celle-ci garantit l'ordre d'arrivée des messages. Le modèle de transmission permet le calcul du délai de transmission. Ainsi, il est possible de définir un événement propre à BGP pour la transmission d'un message de mise à jour directement à un voisin.

Simplification des procédures. La planification de messages de mise à jour peut représenter un nombre important d'événements à stocker dans la queue de priorité. Ces messages doivent contenir les routes à annoncer, augmentant l'empreinte mémoire des simulations de BGP. Afin de réduire cet impact, les routes ne sont pas directement incluses dans le message. Seuls les index des préfixes des routes à annoncer dans la Loc-RIB du routeur source sont enregistrés dans le message. Le modèle de mémoire partagé de DRMSim permet aux nœuds l'accès aux variables d'états du système, y compris celles des nœuds eux-mêmes. Les routeurs peuvent accéder directement entre eux à leur Loc-RIB.

Considérons l'envoi d'un message de mise à jour par un routeur R_{source} à destination d'un routeur voisin R_{cible}. Seuls les index de la Loc-RIB des routes à annoncer sont copiés dans le message. R_{cible} lit les routes directement depuis la table de routage de R_{source} à partir des index. Cette optimisation ne peut fonctionner que dans le cas d'un scénario statique. Les délais de transmission doivent être uniformes et aucune dynamique de la topologie n'est possible.

4.3 Métriques dans BGP

La principale métrique pour la mesure des performances de BGP dans DRMSim est le nombre de messages de mise à jour nécessaire pour converger. Dans ce but, le modèle de mesure de DRMSim a été étendu afin de calculer :

— le nombre de messages de mise à jour ;

— le nombre d'entrées par message ;

— la taille des messages.

Ces métriques sont mesurées durant toute la simulation. Il est possible de les détailler par nœud et lien. Dans le cas d'une simulation de BGP avec le compteur MRAI initialisé, il est possible de comptabiliser ces métriques pour chaque nœud, à chaque instance de MRAI.

4.4 Validation du modèle

La validation du modèle de BGP a fait l'objet d'une publication dans [HPTM10] par les auteurs à l'origine du développement de DRMSim. L'idée de cette expérimentation est de reproduire avec DRMSim les mesures théoriques et simulées observées dans [DS04]. Cette étude mesure l'impact du traitement des routes et des compteurs MRAI sur le temps de convergence.

Plus précisément, dans [DS04], les auteurs cherchent à déterminer la probabilité que la phase de convergence, à l'initialisation des pairs BGP ou après diffusion d'un nouveau préfixe, implique un nombre k d'instances MRAI. Un réseau de N routeurs BGP est considéré. Une session BGP entre deux routeurs existe avec probabilité p. Les délais sur chacun des liens sont calculés à partir d'une même distribution. Les auteurs proposent une nouvelle méthodologie afin de déterminer théoriquement la probabilité que la phase de convergence soit terminée après k instances MRAI. Ils en dérivent deux expressions pour le calcul précis de cette probabilité dans les cas ou $k = 1$ et $k = 2$. Les expressions pour de plus grandes valeurs de k pouvant être déterminées à partir de la méthodologie proposée. Afin de compléter ces résultats, les auteurs ont validé par simulation leur résultat théorique dans le cas ou $k = 1$. Cette expérience a été menée avec le simulateur SSFNet à partir d'un graphe aléatoire de 10 nœuds.

Cette même expérimentation a donc été reconduite avec l'implémentation de BGP proposée dans DRMSim. Les résultats de simulation obtenus dans [HPTM10] sont donnés dans le tableau 4.2.

Probabilité d'un lien entre deux nœuds	$p = 0.15$	$p = 0.3$	$p = 0.45$
Théorique	0.22	0.94	0.99
SSFNet	0.35	0.92	1.00
DRMSim	0.24	0.93	0.99

TABLE 4.2 – Comparaison des probabilités de convergence pour $k = 1$ en fonction de la probabilité p d'une session entre deux nœuds de la topologie.

Les résultats obtenus par DRMSim sont très proches de ceux obtenus théoriquement. Dans ce sens, ils sont aussi meilleurs que ceux obtenus avec SSFNet. L'implémentation de BGP dans DRMSim est donc fiable et permet de reproduire le comportement de paramètres clés du protocole de routage.

4.5 Complexité de la simulation de BGP dans DRMSim

Lors des expérimentations du modèle de BGP dans DRMSim, les limites d'utilisation mémoire du simulateur ont été atteintes à partir de topologies de l'ordre de $O(10k)$ nœuds. Celles-ci ayant requis 25 Go de mémoire pour une durée de simulation de 3 heures.

L'objectif de ces expérimentations était de faire converger le protocole à partir d'un scénario de pire cas :

— les tables de routage sont vides en début de simulation ;

— les délais de transmission aléatoires ;

— l'ensemble des sessions BGP ouvertes quasi-simultanément à l'initialisation de la simulation.

L'ouverture des sessions BGP quasi-simultanément génère un nombre important de messages de mise à jour BGP. Tous les routeurs s'annonçant au même moment. Les délais aléatoires provoquent l'annonce de mauvaises routes, allongeant le temps de convergence et augmentant encore le nombre de messages échangés.

En terme d'empreinte mémoire, la simulation à grande échelle de BGP pose deux problèmes :

— la taille des tables de routage pour l'ensemble des nœuds de la topologie est en $O(n^2.k.log(n))$ bits avec k la taille d'une entrée et n le nombre de nœuds exécutant BGP ;

— le nombre et la taille des routes contenues dans les messages de mise à jour durant une instance de MRAI.

Taille des tables de routage. Prenons l'exemple d'un graphe connexe $G(V, E)$ composé de 10 000 nœuds avec une longueur moyenne des plus courts chemins de 3.6 (proche de celle d'Internet) et un numéro d'AS dans le chemin codé sur 4 octets. Le calcul de l'ensemble des tables de routage représente $10\,000 \times 10\,000 \times 3.6 \times 4 = 1.4$ Go de mémoire. En réalité, due à des contraintes d'implémentation et à la possibilité d'étendre les attributs des routes, la mémoire requise dans DRMSim avoisine plutôt les 2 Go dans ce cas. Les machines actuelles, et plus particulièrement les calculateurs, sont couramment équipés de plusieurs dizaines de gigaoctets de mémoire. La complexité mémoire des tables de routage de BGP dans DRMSim peut donc être supportée pour des topologies de plusieurs dizaines de milliers de sommets.

Les messages de contrôle. Le problème majeur pour la simulation de BGP dans DRMSim est l'empreinte mémoire due au nombre et à la taille des messages de mise à jour nécessaires pour la construction des tables de routage. La durée d'un MRAI affecte la taille et le nombre de messages nécessaire pour converger. Considérons une simulation sans politique de routage, celles-ci n'étant pas disponibles dans DRMSim. Par la suite, G est non orienté et non pondéré. Nous noterons n le nombre de nœuds et m le nombre de liens. Deux cas de valeurs de MRAI engendrent différents comportement de BGP.

MRAI = 0 seconde. BGP est similaire à l'algorithme distribué de Bellman-Ford. Dans un réseau asynchrone, le nombre de messages pour converger est en $O(n^2m)$ [Gal82]. Ces messages sont composés d'une seule route.

MRAI > 0 secondes. Le réseau peut être considéré comme synchrone. Le nombre de messages est en $O(nm)$ [BBCK+07]. Moins de messages sont nécessaires. Cependant, plusieurs routes peuvent être annoncées par message, augmentant sa taille.

Dans le scénario de simulation de pire cas présenté, la valeur du compteur MRAI était initialisée à 0 seconde, générant un grand nombre de messages. Ces messages sont stockés simultanément dans la queue de priorité et la consommation mémoire explose. Dans le cas ou $MRAI = 30$ secondes, moins de messages sont stockés en mémoire, mais le nombre de routes annoncées par message est plus important. Au final, la consommation mémoire de DRMSim est doublée.

5 Expérimentations de modèles de routage avec DRMSim

Plusieurs expérimentations ont pu être menées avec DRMSim pour évaluer les performances de différentes catégories de modèles de routage. Plus généralement, les modèles de routage peuvent être classés en deux grandes familles : celle des modèles *dédiés* à certaines classes de graphe (chordaux, planaire, sans échelle, etc), un modèle ne fonctionne pas ou ne garantit pas ses performances si le graphe ne possède pas certaines propriétés ; celles des modèles *universels*, un modèle garantit ses performances pour tout type de graphe.

Outre la validation de l'implémentation de BGP, un premier résultat concernant un nouveau modèle de routage compact dédié aux graphes k-chordaux a aussi été présenté dans [HPTM10]. Une seconde étude menée dans [Gla13] a permis l'évaluation complète des performances de différents modèles de routage compacts distribués universels.

Afin d'illustrer les capacités de DRMSim à simuler différents modèles de routage sur des topologies de plusieurs milliers de nœuds, je présente succinctement ces différentes études.

5.1 Un schéma de routage compact efficace dans les graphes k-chordaux

Un nouveau modèle de routage compact distribué synchrone particulièrement efficace dans les graphes k-chordaux a été proposé dans [NRS09]. Ce modèle tire avantage des propriétés de faible *diamètre* (logarithmique dans le nombre de nœuds du graphe) et de fort *coefficient de clustering* observées dans certains grands réseaux tels que celui de l'Internet.

La propriété de fort coefficient d'agglomération implique l'existence d'une petite quantité de *cycles* de grande taille dans le graphe. La classe des graphes k-chordaux permet de borner la longueur de ces grands cycles à k. En effet, on définira un graphe k-chordal si aucun cycle de longueur k ne possède une *corde*, c'est-à-dire une arête reliant deux sommets non-adjacents du cycle.

Ce modèle de routage renomme les nœuds du graphe par la construction d'arbres particuliers à partir de l'algorithme de recherche en profondeur. En considérant un graphe avec n nœuds et de diamètre D, le calcul distribué des tables de routage à partir de messages de taille en $O(\log n)$ est effectué en temps $O(D)$. Le modèle de transmission des messages est synchrone. Tout nœud est capable d'émettre et de recevoir des messages depuis ou vers ses voisins en une seule étape de communication. Dans le cadre du routage d'un paquet entre deux nœuds, l'étirement additif du chemin parcouru sera d'au plus $k-1$. Au prix d'une moins bonne complexité temporelle en $O(\min\{\Delta D, n\})$ avec Δ le plus grand degré du graphe, le calcul de nouvelles tables de routage permet un étirement additif de 1 dans les graphes chordaux. La mémoire requise localement par les nœuds avec des adresses codées sur $\log n$ bits est en $2(d-1)\log n$ bits avec d le degré du nœud.

L'étirement additif moyen des routes calculées par ce modèle de routage compact a été mesuré sur des graphes générés à partir de GLP [BT02]. Ce générateur se base sur le principe d'attachement préférentiel pour leur construction. Ces graphes ont la particularité d'avoir une distribution des degrés suivant une loi de puissance. Cette caractéristique est aussi partagée par les graphes de l'Internet [FFF99]. La taille des graphes générés varient entre 500 et 10000 nœuds. Pour chaque taille de topologie, 30 simulations ont été exécutées à partir d'une graine différente lors de l'initialisation du générateur de nombres aléatoires.

L'étirement additif moyen et l'écart type ont été calculés à partir d'une centaine de nœuds choisis aléatoirement. L'évolution de l'étirement en fonction de la taille des topologies a pu être calculée. Celui-ci est donné en figure 4.8. On observe un étirement particulièrement faible, entre 0.4 et 0.8 permettant de valider les très bonnes performances du modèle dans ces graphes.

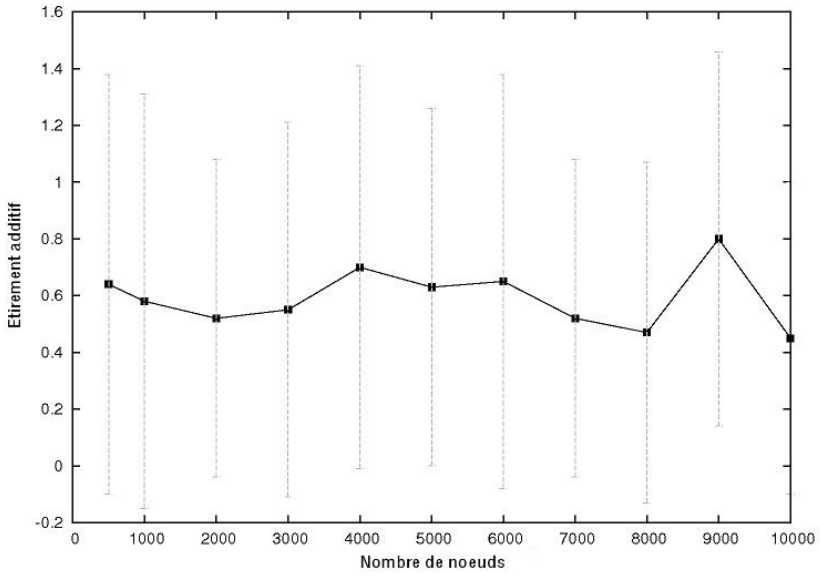


FIGURE 4.8 – Évolution de l'étirement additif du schéma de routage compact dans les graphes k-chordaux. Figure obtenue dans [HPTM10].

Chacune de ces simulations a été exécutée en moins de 15 min sur un ordinateur équipé d'un processeur Intel Core 2 Duo T7500 2.2 GHz avec 4 GB de RAM et la JVM limitée à 2 GB.

5.2 Expérimentations de schémas de routage compacts universels ou dédiés aux graphes sans échelle

Le schéma de routage compact précédent repose sur un modèle de communication synchrone, nécessite un renommage des nœuds dans le graphe et est particulièrement efficace dans les graphes chordaux. Dans l'idéal, le schéma de routage recherché sera asynchrone, universel, avec indépendance des noms, conservant de bonnes performances en termes de nombre de messages échangés ($o(n^2)$) pour la construction des tables de routage, une faible consommation mémoire ($\log n$), un temps de convergence en $O(D)$ et un étirement des routes le plus faible possible.

Le modèle de routage AGMNT a été proposé dans [AGM+04] pour répondre à ces objectifs. Cet algorithme est universel, ne nécessite pas de renommage des nœuds et garantit un étirement multiplicatif des routes de 3 avec des tables de routage de taille en $O(\sqrt{n})$. Cependant, cet algorithme est centralisé. Afin de combler ce manque, les schémas de routage compacts, distribués et sans renommage DCR, LOMNI, HDLBR et CLUSTER sont proposés dans [Gla13].

Principes d'AGMNT. Les schémas de routage DCR, LOMNI et CLUSTER reposent tous sur les mêmes principes utilisés dans AGMNT :

— une coloration aléatoire des nœuds du graphe en k couleurs ;

— une répartition des sommets en k groupes de couleurs mixtes ;

— chaque sommet est responsable des informations de routage vers les sommets de couleur identique ;

— dans chacun des groupes, un sommet particulier, appelé *landmark*, sera responsable des informations de routage pour tous les nœuds de son groupe.

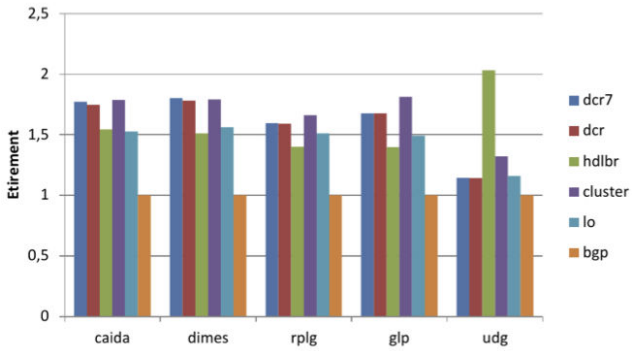
Cette segmentation des informations permet de borner efficacement la taille des différentes tables de routage utilisées.

Les schémas DCR et LOMNI. Ces schémas sont universels. Les coûts de communication sont similaires pour les deux schémas. Cependant, DCR offre les meilleurs performances en termes de mémoire avec des tables de routage composées d'au plus $O(k \log k + n/k)$ entrées alors que celles de LOMNI peuvent atteindre n entrées. Ce surcoût mémoire permet néanmoins de limiter l'étirement multiplicatif à 3 dans le cas de LOMNI alors que celui de DCR peut atteindre 5, la meilleure borne prouvée à ce jour.

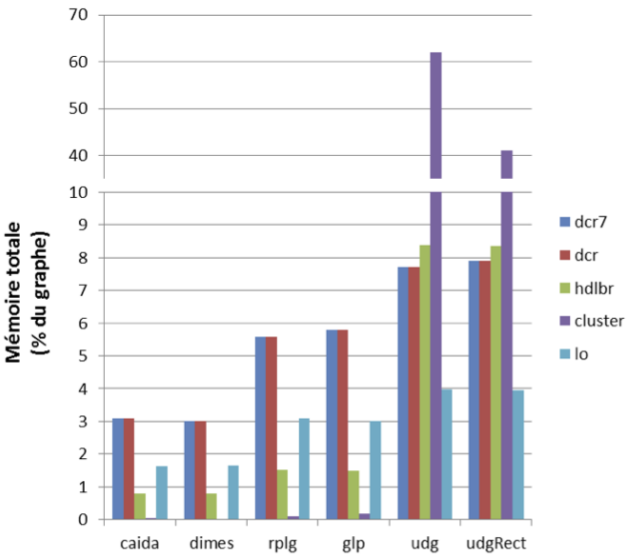
Les schémas HDLBR et CLUSTER. Ils sont conçus pour les graphes sans échelle, c'est-à-dire dont la distribution des degrés suit une loi de puissance. HDLBR est basé sur l'algorithme donné dans [TZLL13] tout en reprenant certains aspects précis de DCR afin de le rendre indépendant des noms. CLUSTER reprend quant à lui les techniques de routage de HDLBR et DCR. Ces deux schémas permettent d'obtenir un meilleur étirement multiplicatif des routes, respectivement de 2 et 3. CLUSTER aura cependant une empreinte mémoire plus faible en $O(\sqrt{n})$ alors que celle de HDLBR sera de n.

Les performances de ces schémas ont été vérifiées par simulation avec DRMSim et sont données en figure 4.9. L'étirement moyen, la taille des tables de routage et le coût des communications sont mesurés sur les graphes petit monde GLP, RPLG, CAIDA et DIMES et sur des graphes aléatoires de plus grand diamètre dont la distribution des degrés suit une loi de poisson. La taille de ces graphes est comprise entre 5 000 et 17 144 nœuds. Ces expérimentations montrent que tous les schémas de routage proposés ont en pratique un étirement multiplicatif moyen inférieur à 2 (figure 4.9a). Le nombre d'entrées dans les tables de routage ne représente que 10% des nœuds excepté pour le schéma CLUSTER, qui dans le cas des graphes aléatoires peut atteindre 60% des nœuds (figure 4.9b). Quant aux coûts de communication, ils évoluent de façon similaire à l'empreinte mémoire des schémas (figure 4.9c).

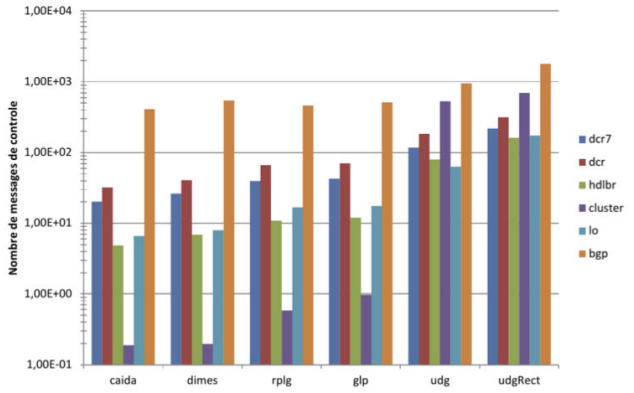
Sur les graphes GLP, RPLG eT DIMES, ces simulations ont requis entre 1 et 4 heures de calcul sur un ordinateur équipé d'un bi-processeur hexa-core Westmere Intel® Xeon® X5670 2,93 GHz avec 32 GB de RAM. Cependant, sur la plus grande carte CAIDA composée 17 144 nœuds, une machine équipée de 300 GB de RAM a été nécessaire.



(a) Étirement multiplicatif.



(b) Taille des tables de routage en pourcentage des destinations stockées.



(c) Coût des communications.

FIGURE 4.9 – Résultats des simulations des schémas DCR, OMNI, HDLBR et CLUSTER obtenus à partir de DRMSim. Figures obtenues dans [Gla13].

6 Conclusion

L'évolution rapide d'Internet ces deux dernières décennies a exhibé les problèmes de passage à l'échelle inhérent à BGP. La taille des tables de routage linéaires en le nombre d'AS dans l'Internet, le nombre quadratique de messages de contrôles nécessaires pour converger et un temps de convergence de plusieurs dizaines de minutes en sont quelques illustrations caractéristiques.

Un premier axe d'étude consiste donc à comprendre d'avantage le comportement de BGP. De nouvelles corrections ou fonctionnalités doivent être proposées afin d'assurer le fonctionnement opérationnel du protocole. J'ai présenté en section 1.1 les outils d'émulation dédiés à BGP. Ces outils permettent une étude approfondie de BGP mais sont limités à des topologies de quelques centaines de nœuds. De plus, de telles études requièrent la mise en oeuvre de plate-formes d'expérimentation complexes et coûteuses à maintenir. Afin de faciliter ces expérimentations, des logiciels de simulation dédiés à l'étude de BGP ont été développés. Ces logiciels, présentés en section 1.2, permettent une simulation détaillée du protocole mais se limitent aussi à des topologies de quelques centaines de nœuds. D'autres se concentrent sur certains aspects du protocole. Au prix de cette simplification, des simulations peuvent être exécutées sur des topologies de taille similaire à celle de l'Internet.

Un deuxième axe d'étude aborde l'aspect conceptuel de BGP. Les performances des routeurs BGP évoluent moins vite que les resources requises par l'évolution d'Internet [Gla13]. À plus ou moins long terme, il n'est pas certain que BGP puisse continuer à assurer son rôle dans l'architecture de routage de l'Internet [1]. Dans cette perspective, de nouveaux prototypes de schémas de routage sont développés. Les performances de ces schémas doivent être évaluées mais aussi comparées à celles de BGP dans des scénarios de simulation identiques. Afin de répondre à cet objectif, le simulateur de modèles de routage dynamiques DRMSim a été développé.

Ce simulateur dont j'ai repris le développement, a été présenté en section 3. Une première étape dans mon travail a été de modifier l'architecture de DRMSim afin qu'elle soit basée sur une structure modulaire et évolutive permettant une intégration aisée de futures fonctionalités et la possibilité d'étendre celles existantes. À partir de cette nouvelle architecture, plusieurs évolutions majeures ont été intégrées.

1. De nouvelles passerelles ont été développées pour l'utilisation de diverses librairies de graphe.
2. Un modèle dédié aux mesures permettant d'automatiser l'évaluation des performances des modèles de routage. De nouvelles métriques propres à chaque modèle de routage peuvent être intégrées. Les fichiers de mesures sont automatiquement générés en fin de simulation.
3. Une API simple est fournie pour l'intégration de nouveaux modèles de routage et l'utilisation du modèle de transmission de messages.
4. Un modèle de dynamique est proposé pour le contrôle des événements exogènes ;
5. Un modèle pour la gestion des délais et la transmission des messages est fourni.

L'intégration de ces évolutions dans DRMSim a permis une première série d'expérimentations. Les performances de différents modèles de routage compacts universels et dédiés aux graphes sans échelle présentées en section 5.2 ont été évaluées. Ces expérimentations montrent les capacités de DRMSim à intégrer divers schémas de routage et à exécuter différents scénarios de simulation sur des topologies de plusieurs milliers de nœuds.

Cependant, la plus grande taille de topologie atteinte reste de l'ordre de 17 000 nœuds. Ce qui est bien trop faible pour espérer approcher celle de l'Internet (environ 48 000 nœuds).

1. Notons que même si une alternative viable à BGP est proposée dans le futur, les considérations pratiques d'une migration laisse croire qu'elle sera repoussée le plus longtemps possible.

De plus, il est nécessaire d'anticiper les besoins futurs de ces nouveaux modèles par des simulations sur des topologies de l'ordre de 100 000 nœuds.

L'accumulation dans la queue de priorité d'événements représentatifs des messages de contrôle du modèle de routage est à l'origine de l'impossibilité pour DRMSim de simuler des topologies de très grande taille. Ce problème peut être abordé de différentes façons.

Optimisation des structures de données. Les structures de données et les procédures des modèles de routage doivent être optimisées au maximum. Cependant, d'importantes modifications dans l'implémentation du modèle peuvent être nécessaires et apporter une complexité non souhaitable.

Écriture des données sur le disque. Les techniques d'écriture sur le disque (swap) d'une partie des données de la queue de priorité et leur chargement en mémoire à la demande peuvent grandement limiter l'empreinte mémoire des simulations. La technologie SSD (Solid State Drive) offre des accès en lecture et écriture sur le disque particulièrement intéressants. Le rapport entre le coût en termes de temps de simulation supplémentaire et taille de topologie est à déterminer. Une première utilisation de cette technique dans DRMSim a été faite dans [Gla13] lors des simulations les plus coûteuses en termes de nombre de messages de contrôle échangés. Cependant, cette technique doit être étendue à l'ensemble des modèles de routage de DRMSim.

Simulations parallèles et distribuées. Le grand nombre de messages échangés lors des simulations, et plus particulièrement celles de BGP, est la principale limite du simulateur. Il est possible de distribuer la simulation de ces messages à travers différentes unités de calcul dotées de leur propre mémoire et communiquant en réseau. Ainsi, il est théoriquement possible de diviser la mémoire requise par le nombre d'unités de calcul (à mémoire égale). Un algorithme de partition de la topologie est nécessaire pour répartir les nœuds de la topologie entre ces unités de calcul. Cet algorithme doit minimiser le nombre de messages échangés entre les parties afin de limiter la communication réseau entre les unités de calcul. Le nombre de messages échangés est déterminant pour la faisabilité de telles simulations. Une étude dans ce sens est proposée en chapitre 5 pour le modèle de BGP dans DRMSim.

Chapitre 5

Étude de faisabilité de la distribution des simulations de BGP

Sommaire

1 Simulation discrète parallèle et distribuée 104
1.1 Coût des communications entre LP 104
1.2 Erreurs de causalités 104
1.3 Protocole de synchronisation, l'approche conservatrice 105
1.4 Paralysie du système 107
2 Distribution des simulations de BGP dans DRMSim 107
2.1 Modèles de communication 107
2.2 Algorithme de partition 110
3 Simulation des solutions A et B 112
3.1 Scénarios de simulation 112
3.2 Environnement d'exécution 113
3.3 Résultats de simulation 113
4 Conclusion 116

L'implémentation actuelle de BGP dans DRMSim, dans un scénario de pire cas (exécution à partir de tables de routage vides et ouverture des sessions BGP simultanément, délais de transmission aléatoires), ne permet pas de simulation sur des topologies de taille similaire à celle de l'Internet (environ 48 000 AS). Cette limitation n'est pas due au temps de calcul, mais à la quantité de mémoire requise pour la simulation de BGP. Un travail d'optimisation et de réécriture du code permettrait probablement d'atteindre de plus grandes tailles. Cependant, il est souhaitable de pouvoir anticiper le comportement de BGP sur des topologies de taille bien plus importante, de l'ordre de $O(100k)$ nœuds. De telles simulations doivent donc être envisagées.

Afin d'atteindre des topologies de cet ordre, une possibilité est d'étendre DRMSim au principe de simulation parallèle et distribuée. Cependant, le coût des communications entre les différentes unités de calcul distribuées peut être important. Le temps de simulation devenant trop long et rendant impossible la simulation distribuée de BGP. Dans ce chapitre, je présente une étude dont l'objectif est de déterminer si ce coût supplémentaire, induit par la distribution des simulations de BGP, est suffisamment faible pour rendre faisable une telle approche. Deux modèles de communication entre les processus logiques de la simulation distribuée sont évalués. Les topologies considérées sont comprises entre 1 000 et 5 000 nœuds. Le temps nécessaire pour la transmission des messages est ensuite extrapolé dans le cas de topologies de l'ordre de $O(100k)$.

Je commence en section 1 par présenter le principe de simulation à événements discrets parallèle et distribuée. En section 2, je donne les modèles de distribution qui seront évalués, l'algorithme de partition utilisé et les coûts de communication attendus. En section 3, je présente les différents scénarios de simulation, leurs résultats, puis je conclus en section 4.

1 Simulation discrète parallèle et distribuée

D'après Fujimoto [Fuj90], le principe de simulation à événements discrets parallèle et distribuée (ou PDES, Parallel Discrete Event Simulation) concerne en premier lieu la simulation de systèmes asynchrones avec des événements qui ne sont pas synchronisés par une horloge globale. C'est-à-dire, totalement distribués par nature. Les systèmes de routage, et notamment celui de l'Internet, étant totalement distribués, le principe de PDES semble donc parfaitement adapté à DRMSim.

De manière générale, le modèle de simulation parallèle et distribuée est décomposé hiérarchiquement en processus logiques (ou LP, Logicial Process). Ces LP sont distribués parmi les ressources de calcul et exécutés en parallèle. Les événements de la simulation sont traités de façon concurrentes entre LP. Au final, lors de l'exécution d'une simulation, la topologie est découpée en parties chacune affectée à un LP. Ces LP sont équipés de leur propre unité de mémoire et de calcul communiquant par réseau. En théorie, il est possible de diviser la mémoire nécessaire par le nombre de parties. Cependant, la structure de la topologie et l'algorithme de partition permettent rarement un tel équilibre. De plus, le calcul parallèle et distribué de simulations discrètes pose différents problèmes de coût de communication, d'erreurs de causalités et de paralysie du système.

1.1 Coût des communications entre LP

L'utilisation de processus logiques nécessite une partition des variables d'états du simulateur en un ensemble disjoint. Dans le cas ou des interactions existent entre les parties dans un système sans mémoire partagé, une charge supplémentaire de communication apparaît. Une première approche naturelle est d'émuler la mémoire partagée en affectant un LP à chaque partie. Un modèle de communication permet à chaque LP l'envoi de messages d'écriture et de lecture pour accéder aux informations partagées. Cependant, cette approche est limitée par le nombre de messages échangés entre LP et par les performances du modèle de communication. Une meilleure méthode est de dupliquer les informations partagées entre LP qui le nécessitent. L'accès à ces informations se fait directement à partir de la mémoire du LP. Les performances sont bien meilleures mais cette méthode induit une plus grande empreinte mémoire. De plus, un protocole est requis pour assurer la cohérence entre les différentes copies des variables d'états partagées. Ces deux approches et leur coût de communication sont détaillés en section 2.1.

1.2 Erreurs de causalités

Rappelons qu'un simulateur à événements discrets est composé de variables d'états représentant le système simulé et d'une liste contenant les futurs événements ordonnés suivant leur estampille. Un événement peut modifier l'état du système et ajouter ou supprimer d'autres événements à la liste. Si les événements ne sont pas exécutés dans le bon ordre, les résultats de la simulation peuvent être faussés. De telles erreurs sont appelées *erreurs de causalités.*

Dans une simulation à événements discrets parallèle et distribuée à mémoire partagée, plusieurs LP se partagent l'exécution des événements. L'ordre dans lequel ces événements doivent être exécutés peut ne pas être respecté. De telles erreurs sont particulièrement difficiles à éviter ou corriger.

Pour illustrer ce problème, prenons l'exemple donné en figure 5.1. On dispose d'un ensemble de processus logiques. Chaque LP est une instance du simulateur qui exécute à partir de certaines ressources de calcul des événements liés à un sous-ensemble des variables d'état du système. Considérons un processus logique LP_i pouvant communiquer avec LP_j. Si un message contenant l'événement $E2$ créé par LP_i modifie la partie du système gérée par LP_j, alors LP_i doit l'informer. LP_i doit transmettre par message l'événement $E2$ à

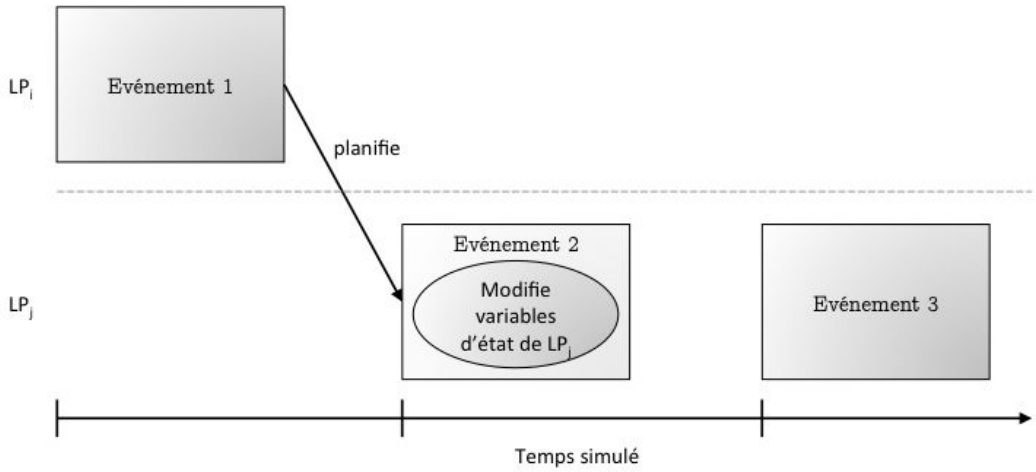


FIGURE 5.1 – L'événement E1 affecte l'événement E3 en planifiant un troisième événement E2 qui modifie la variable d'état utilisée par E3. Une exécution séquentielle des trois événements est donc nécessaire.

LP_j. Si LP_j ne considère pas la possibilité de recevoir des messages provenant d'autres processus logiques, alors l'événement $E3$ est directement exécuté. Une erreur de causalité est provoquée.

Afin d'éviter les erreurs de causalités, un protocole de synchronisation doit être établi entre les LP. Deux approches sont possibles :

— *l'approche conservatrice* [CM79], empêche toute erreur de causalité en bloquant la simulation ;

— *l'approche optimiste* [Jef85], beaucoup plus compliquée à mettre en oeuvre. Elle autorise les erreurs tout en les détectant, puis les corrige dans un second temps.

1.3 Protocole de synchronisation, l'approche conservatrice

Une contrainte suffisante mais pas forcément nécessaire pour éviter toute erreur de causalité est la *contrainte de causalité locale*.

Définition 11 (Contrainte de causalité locale)**.** Dans une simulation à événements discrets, les LP respectent la contrainte de causalité locale *si, et seulement si*, les messages échangés entre les LP sont traités dans leur ordre chronologique.

Le non respect de cette condition n'induit pas obligatoirement une erreur de causalité. En effet, deux événements indépendants l'un de l'autre peuvent être traités par un processus logique indépendamment de leur date d'estampillage sans produire d'erreur dans la simulation.

Afin de garantir la contrainte de causalité locale, nous utilisons l'approche conservatrice proposée dans [CM79] comme protocole de synchronisation des LP dans les modèles de distribution présentés en section 2.1. Dans cette approche, nous nommerons *influent*, un LP pouvant *influencer* l'exécution d'un autre LP par l'envoi de messages.

— Pour chaque LP, considérons d LP influents. Alors, $d+1$ listes ordonnées d'événements devront être gardées en mémoire par le LP. Une liste des événements locaux au LP et une liste de messages envoyés par chacun des LP influents.

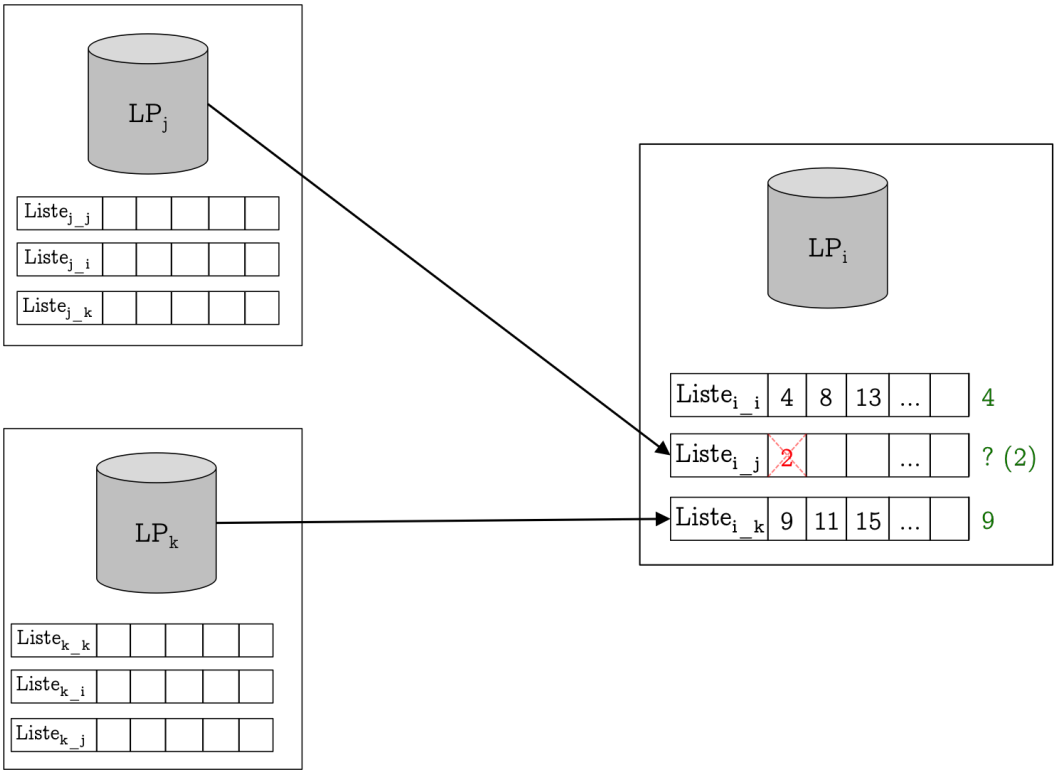


FIGURE 5.2 – Exemple d'exécution d'un protocole de synchronisation conservateur.

— Une horloge est associée à chaque liste. La valeur de cette horloge est initialisée à zéro. Elle est ensuite modifiée par l'estampille de l'événement le plus ancien de la liste. Dans le cas d'une liste vide, l'horloge sera égale à l'estampille du dernier événement traité.

Les événements de LP_i sont traités du plus ancien au plus récent. Dans le cas d'une liste vide $Liste_{i\_k}$, avec k l'index d'un LP influençable, LP_i ne peut poursuivre son traitement. Il doit attendre l'arrivée de nouveaux messages de la part de LP_k, même si d'autres événements sont présents dans les autres listes. C'est ce qui garantit la contrainte de causalité locale. LP_i pourra exécuter un événement d'une autre liste uniquement quand LP_k aura envoyé un message dont l'estampille sera supérieure à l'horloge d'une des autres listes.

Un exemple d'exécution de ce protocole de synchronisation est donné en figure 5.2. LP_i est influencé par LP_j et LP_k. Il possède donc trois listes d'événements :

— $Liste_{i\_i}$ est la liste des événements locaux traités par LP_i ;

— $Liste_{i\_j}$ est la liste des messages envoyés par LP_j à LP_i ;

— $Liste_{i\_k}$ est la liste des messages envoyés par LP_k à LP_i.

La date de l'événement le plus ancien traité par LP_i est 2 et provient de $Liste_{i\_j}$. Aucun autre événement n'étant présent dans cette liste, LP_i se met en attente de messages de la part de LP_j. Plus tard, LP_j envoie un message à LP_i avec une estampille de 5. L'horloge associée à $Liste_{i\_j}$ est initialisée à 5 et la procédure de l'événement exécutée. À partir de ce moment, la liste des événements locaux $Liste_{i\_i}$ à la plus petite valeur d'horloge 4. Cette liste n'étant pas vide, LP_i exécute l'événement de plus petite estampille dans celle-ci, celui dont la valeur est 4. Le même principe est ensuite appliqué au reste des événements.

1.4 Paralysie du système

Du respect de la contrainte de causalité locale peut résulter une paralysie du système. Un ensemble de LP nommé S sera paralysé si les conditions suivantes, définies dans [Mis86], sont validées :

— tout LP dans S est terminé ou en attente de messages ;

— au moins un LP dans S est en attente de messages ;

— pour tout LP_i dans S en attente de messages depuis un LP_j, LP_j est dans S et aucun message n'est en transit entre LP_j et LP_i.

Dans un système complètement décentralisé, quand tous les LP essayent de respecter la contrainte de causalité locale, il est possible qu'aucun d'entre eux ne possède suffisamment d'information pour continuer. Les LP sont en attente de messages les uns envers les autres. Les conditions de paralysie sont remplies et le système est alors bloqué.

null-message. Des messages particuliers, appelés *null-message* sont utilisés pour débloquer le système. Ces messages ont la particularité d'avoir une routine d'exécution vide, ils ne correspondent à aucune activité dans le système simulé. Considérons un null-message envoyé de LP_j à LP_i et d'estampille T_x. Ce message est une promesse de LP_j à LP_i qu'aucun message d'estampille inférieure à T_x sera envoyé dans le futur. LP_i peut reprendre le traitement de ses événements dont l'estampille est inférieure à T_x.

Ces null-messages doivent être envoyés régulièrement de la part des LP influents. Les LP influençables devant continuer le traitement de leurs événements sans rester bloqués trop longtemps. Un LP influent peut envoyer des null-messages à chaque incrément de l'horloge de plus petite valeur. Cependant, une telle stratégie peut générer un nombre important de messages et donc un trop gros volume de communication réseau ralentissant l'exécution de la simulation.

Une autre approche consiste à envoyer des null-messages à la demande. Lorsqu'un LP est bloqué, il envoie une demande au LP influent associé à la liste d'événements vide. Celui-ci répond par un null-message dont l'estampille est égale à sa plus petite valeur d'horloge. Deux messages sont donc nécessaires, impliquant un plus long délai avant la réception du null-message mais réduisant le volume des communications.

Un des objectifs pour la simulation distribuée de BGP dans DRMSim est de minimiser les communications réseaux afin de réduire le temps de simulation. La seconde approche est donc la plus adaptée.

2 Distribution des simulations de BGP dans DRMSim

Un modèle de communication doit assurer les échanges d'informations entre les routeurs BGP situés dans différents LP. Une certaine quantité de mémoire supplémentaire peut être nécessaire pour assurer une communication efficace entre les LP. De plus, un algorithme de partition de la topologie, à partir duquel les routeurs et les liens sont affectés aux LP, doit minimiser la charge des communications entre LP. Le surcoût mémoire et la quantité d'informations échangées dépendent donc du modèle de communication et de l'algorithme de partition choisis.

2.1 Modèles de communication

Le système est modélisé par un graphe connexe $G(V, E)$, non orienté, non pondéré avec V l'ensemble des nœuds et E l'ensemble des liens. Une partition des nœuds de V, de même taille que le nombre de LP, est calculée. Chacune des parties est attribuée à un LP. Un lien entre deux nœuds d'une même partie est seulement connu du LP gérant cette partie. Un

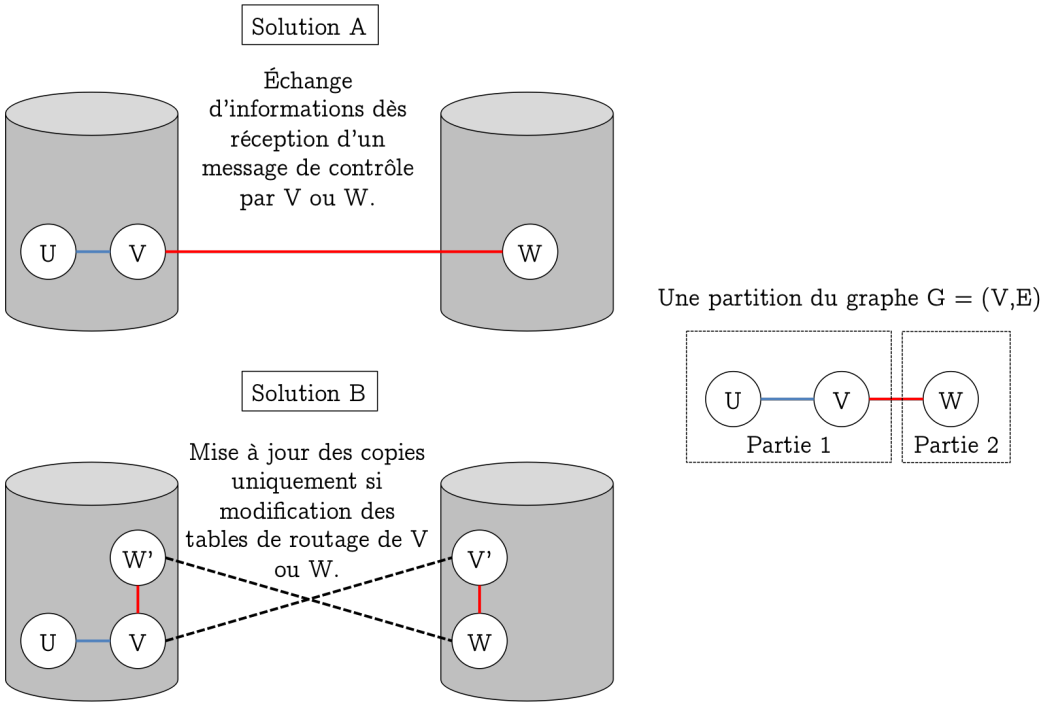


FIGURE 5.3 – Modèles de communication. La solution A prend en charge l'échange direct des informations entre les routeurs de différents LP. La solution B recopie les routeurs frontières et assure leur mise à jour.

nœud de la topologie est assimilé à un routeur BGP. Un lien entre deux routeurs correspond à une session BGP bidirectionnelle. Un routeur dont certains voisins sont situés dans des LP différents est appelé *routeur frontière*. Un lien connectant deux routeurs frontières est appelé *lien frontière*.

Comme expliqué en section 4.5 (p. 94), les messages de mise à jour BGP pour l'annonce d'une nouvelle route constituent le problème principal en termes de ressource mémoire. Les autres types d'événements ayant un effet négligeable sur le système.

Les solutions A et B sont proposées pour gérer les communications entre les routeurs frontières. Dans ces solutions, les messages de mise à jour sont échangés de deux façons.

— Échanges *internes*, c'est la méthode utilisée dans la version séquentielle de DRMSim pour échanger les messages de mise à jour BGP. Un événement contenant l'index de la route à annoncer dans la Loc-RIB du routeur source est planifié. Le routeur destination accède à la route annoncée directement en mémoire dans la Loc-RIB du routeur source.

— Échanges *externes*, un message échangé entre deux routeurs frontières situés dans des LP différents transite par le réseau. Les LP fournissent une API de communication afin que les routeurs frontières puissent transmettre des messages entre eux.

Ces solutions sont décrites à partir de l'exemple donné en figure 5.3. Un routeur frontière v situé dans LP_1 annonce une nouvelle route à son voisin u situé dans le même LP et à son voisin w situé dans LP_2.

Solution A.

— Un routeur frontière utilise une communication externe pour transmettre un message. Le message contient la route à annoncer et transite par le réseau. Dans l'exemple, v

envoie directement son message à w en utilisant l'API de communication de LP_1. Le message transite par le réseau à destination de LP_2 qui en informe w.

— Une communication interne est utilisée entre routeurs d'un même LP. Dans l'exemple, v planifie un événement de mise à jour à destination de u. La procédure de l'événement est exécutée. Le message de mise à jour est traité par u.

Solution B.

— Soit $N(v)$ les routeurs frontières voisins de v. Ces routeurs sont répartis dans des LP différents de celui de v. Une copie des routeurs de $N(v)$ est gardée en mémoire dans le LP de v. Ces copies sont synchronisées entre elles uniquement lorsqu'un changement dans la Loc-RIB intervient dans l'une des copies. Une communication externe est utilisée pour leur synchronisation. Les messages échangés dans le réseau contiennent uniquement les routes modifiées. Dans la figure 5.3, w' est la copie de w dans LP_1 et v' la copie de v dans LP_2. Lorsqu'un changement intervient dans v, un message est envoyé à v' et inversement. Le même procédé s'applique pour w et w'. Le message est dupliqué et envoyé autant de fois qu'il y a de copie du routeur.

— Soit $N'(v)$ un ensemble de routeurs frontières voisins de v appartenant à un même LP. Alors un seul message est nécessaire entre le LP de v et le LP auquel appartient $N'(v)$ pour annoncer une route.

— Une communication interne est utilisée entre routeurs et copies de routeurs dans un même LP. Si un message de mise à jour n'implique pas de changement dans la Loc-RIB d'une copie d'un routeur frontière, alors aucune communication réseau est nécessaire.

Coût des communications

Les notions suivantes sont utilisées :

— $G(V, E) \triangleq$ le graphe du réseau ;

— $V = \cup_{i=1}^{K} V_i \triangleq$ la partition des nœuds du graphe ;

— $N_{V'}(v) \triangleq$ le voisinage d'un nœud v dans le sous graphe induit par V' ;

— $ME(v) \triangleq$ l'ensemble des entrées modifiées dans le routeur v ;

— $e(v, V') = 1$ si v n'est pas dans V' et a un voisin dans V', sinon 0 ;

— $Esize \triangleq$ la taille moyenne d'une entrée de la table de routage.

Les coûts de communication et de mémoire des solutions A et B sont donnés dans les tableaux 5.1 et 5.2.

Solution A	$Esize.\sum_{i=1}^{k}\sum_{v\in V_i}(\lvert N_{V\setminus V_i}(v)\rvert.\lvert ME(v)\rvert)$
Solution B	$Esize.\sum_{i=1}^{k}\sum_{v\in V_i}[(\sum_{j=1}^{k} e(v, V_j)).\lvert ME(v)\rvert]$

TABLE 5.1 – Volume des communications entre les nœuds frontières des solutions A et B.

Dans la solution B, seules les entrées modifiées dans la table de routage des routeurs frontières sont échangées dans le réseau. De plus, si plusieurs routeurs frontières appartiennent à un même LP, un seul message vers ce LP est nécessaire. Dans le cas de la solution A, un

Solution A	pas de coût mémoire
Solution B	$Esize.\|V\|.\sum_{i=1}^{k}\sum_{v\in V_i}[(\sum_{i=1}^{k} e(v,V_j))]$

TABLE 5.2 – Coût mémoire des solutions A et B.

message par routeur frontière est envoyé. Le coût des communications entre les LP dans la solution B est donc réduit.

Cependant, un nombre important de liens frontières dans la coupe donnée par l'algorithme de partition implique de nombreuses copies de routeurs frontières. La duplication de ces routeurs peut être coûteuse en mémoire.

2.2 Algorithme de partition

Il existe une technique simple pour le calcul d'une partition des nœuds d'un graphe. Étape après étape, une bipartition du sous-ensemble de nœuds de l'étape précédente est calculée. La partition des nœuds en 2^q sous-ensembles requiert q étapes. Pour calculer une bipartition, nous utiliserons une formulation du problème en programmation linéaire en nombres entiers (ou IP, Integer Programming). Celle-ci peut être exacte, chacune des parties à la même taille. Une différence à 2ε près entre les tailles des deux bipartitions peut aussi être acceptée.

Une bonne bipartition doit minimiser le coût des communications entre les parties. Ce problème de minimisation peut être modélisé en ajoutant des poids dans le graphe. Soit au niveau des nœuds, soit au niveau des liens. Une bipartition est ensuite calculée en minimisant une fonction de ces poids. À partir d'une bipartition optimale, il est possible d'obtenir une borne inférieure sur le volume de communication échangé entre les parties.

Pour calculer une telle bipartition optimale, le volume de communication sur les liens frontières ou pour chaque nœuds frontières doit être estimé pour les solutions A et B. Le nombre d'entrées échangées pour chaque lien et chaque nœud est donc calculé à partir d'une topologie sans partition. Il est ensuite possible d'obtenir une bipartition optimale à partir des deux programmes IP proposés par la suite.

Solution A - bipartition optimale

Dans la solution A, on assigne à chaque lien $uv \in E(G)$ un poids W_{uv} qui représente approximativement le nombre d'entrées échangées sur ce lien. Le meilleur algorithme de bipartition minimise la somme des poids sur tous les liens frontières. Soit $(S, \bar{S})$ une bipartition et considérons deux types de variables binaires :

$$\alpha_u = \begin{cases} 1, & \text{Si } u \in S \\ 0, & \text{sinon} \end{cases} \qquad \forall u \in V$$

$$\beta_{uv} = \begin{cases} 1, & \text{Si } (u,v) \in S\text{x}\bar{S} \text{ ou } (v,u) \in S\text{x}\bar{S} \\ 0, & \text{sinon} \end{cases} \qquad \forall uv \in E$$

On peut calculer une bipartition optimale à partir de la formulation **IP1** suivante :

$$\min \sum_{uv \in E} W_{uv}\beta_{uv} \tag{5.1a}$$

$$s.t.\ \ \alpha_u + \alpha_v \leq 2 - \beta_{uv} \qquad \forall uv \in E \tag{5.1b}$$

$$\alpha_u + \alpha_v \geq \beta_{uv} \qquad \forall uv \in E \tag{5.1c}$$

$$\beta_{uv} \geq |\alpha_u - \alpha_v| \qquad \forall uv \in E \tag{5.1d}$$

$$\sum_{u \in V} \alpha_u \geq n/2 - \varepsilon \tag{5.1e}$$

$$\sum_{u \in V} \alpha_u \leq n/2 + \varepsilon \tag{5.1f}$$

Dans **IP1**, la fonction objective 5.1a est le volume de communication des liens frontières à minimiser. Les contraintes 5.1b, 5.1c et 5.1d vérifient que a_u et a_v sont bien dans des parties différentes et connectés par un lien frontière uv. Les contraintes 5.1e et 5.1f assurent le calcul d'une bipartition dont le nombre de sommets dans chacune des parties est égal à 2ε près.

Solution B - bipartition optimale

Dans la solution B, on assigne à chaque nœud $v \in V(G)$ un poids W_v qui est approximativement $|ME(v)|$. Dans ce cas, la meilleure bipartition minimise la somme des entiers sur tous les nœuds frontières. Ajoutons deux nouveaux types de variables binaires :

$$\gamma_u = \begin{cases} 1, & \text{Si } u \text{ a un voisin dans un sous-ensemble différent} \\ 0, & \text{sinon} \end{cases} \qquad \forall u \in V$$

$$\gamma_v = \begin{cases} 1, & \text{Si } v \text{ a un voisin dans un sous-ensemble différent} \\ 0, & \text{sinon} \end{cases} \qquad \forall v \in V$$

On peut calculer une bipartition optimale à partir de la formulation **IP2** suivante :

$$\min \sum_{u \in V} W_u\gamma_u \tag{5.2a}$$

$$s.t.\ \ \alpha_u + \alpha_v \leq 2 - \beta_{uv} \qquad \forall uv \in E \tag{5.2b}$$

$$\alpha_u + \alpha_v \geq \beta_{uv} \qquad \forall uv \in E \tag{5.2c}$$

$$\beta_{uv} \geq |\alpha_u - \alpha_v| \qquad \forall uv \in E \tag{5.2d}$$

$$\beta_{uv} \leq \gamma_u \qquad \forall uv \in E \tag{5.2e}$$

$$\beta_{uv} \leq \gamma_v \qquad \forall uv \in E \tag{5.2f}$$

$$\sum_{u \in V} \alpha_u \geq n/2 - \varepsilon \tag{5.2g}$$

$$\sum_{u \in V} \alpha_u \leq n/2 + \varepsilon \tag{5.2h}$$

Dans **IP2**, la fonction objective 5.2a est le volume de communication des nœuds frontières à minimiser. Les contraintes 5.2b, 5.2c, 5.2d, 5.2d et 5.2f vérifient que a_u et a_v sont bien dans des parties différentes et connectés par un lien frontière uv. Les contraintes 5.2g et 5.2h assurent le calcul d'une bipartition dont le nombre de sommets dans chacune des parties est égal à 2ε près.

3 Simulation des solutions A et B

Pour rappel, l'objectif est de déterminer le coût des communications causées par une implémentation distribuée et parallèle de DRMSim lors de la simulation de BGP. Nous sommes principalement intéressés par le nombre de messages de mise à jour BGP échangé sur les liens frontières. Ces événements sont connus comme étant les plus nombreux et coûteux en mémoire. Le coût temporel des communications peut être dérivé en estimant le temps de transmission et de propagation d'un message de mise à jour BGP entre deux LP.

3.1 Scénarios de simulation

Plusieurs paramètres du modèle de routage de BGP influencent le nombre de messages de mise à jour échangés. Un premier paramètre est la valeur de MRAI choisie. Nous considérons le paramètre $MRAI = 0$ seconde. Cette valeur correspond à une borne supérieure sur le volume de communication entre les routeurs BGP. Un second paramètre est le délai d'établissement d'une session BGP entre deux pairs. Si toutes les sessions BGP sont établies au même moment, un grand nombre de messages sera généré. Si les délais de transmission sont aléatoires, des messages annonçant des routes trop longues peuvent être propagés. Le temps de convergence et le nombre de messages augmentent. Trois scénarios de simulation ont été élaborés.

Scenario 1. Les sessions BGP sont établies avant le début des échanges des messages de mise à jour. Une fois toutes les sessions établies, les messages sont traités dans l'ordre d'arrivée. Ce scenario simule un réseau où tous les routeurs ont déjà leurs sessions BGP établies avec leurs voisins. Les délais de communication des messages entre les routeurs sont négligeables.

Scenario 2. Les sessions BGP sont établies avant le début des échanges de messages de mise à jour. Cependant, une fois reçus, les messages sont traités dans un ordre aléatoire. Ce scénario simule un réseau avec des délais de communication variables.

Scenario 3. Après chaque établissement d'une session BGP entre deux routeurs, le message de mise à jour qui en résulte est transmis. Le protocole converge puis la prochaine session BGP planifiée est établie. Ce scénario simule l'ajout, les uns après les autres, de routeurs dans le réseau. Celui-ci converge avant l'ajout d'un nouveau routeur. Les délais de transmission sont négligeables.

Les simulations de ces trois scénarios sont exécutées sur des topologies de 2.5K, 3K, 3.5K, 4K, 4.5K et 5K routeurs. Le programme d'entier mixte ne pouvant calculer sur de plus grande taille une bipartition optimale. Chaque scénario est exécuté une fois par topologie. Ces topologies sont générées à partir du modèle préférentiel linéaire généralisé (ou GLP, Generalized Linear Preference) [BT02]. Ce modèle permet d'obtenir des graphes dont certaines propriétés sont proches de celles d'Internet, notamment la distribution des degrés et le coefficient d'agglomération.

Ce modèle construit un graphe de n nœuds à partir d'un arbre aléatoire de taille m_0 donné en paramètre. À chaque itération i, avec probabilité p, $m \leq m_0$ liens sont ajoutés entre deux nœuds existants. Le paramètre m définit le nombre moyen de liens ajoutés à la fin du processus. Avec probabilité $1 - p$, un nouveau nœud est ajouté et connecté à un nœud existant. Les nœuds sont choisis par une fonction du degré du nœud. Cette fonction est paramétrée par $\beta \in]-\inf, 1]$. Plus la valeur de β est faible, plus la probabilité de choisir un nœud de fort degré est faible. L'algorithme s'arrête une fois le nombre de nœuds voulu atteint.

Les paramètres du modèle GLP utilisés pour générer les graphes lors des simulations sont :

— $m_0 = 6$;
— $p = 0.4669$;
— $beta = 0.6753$;
— $m = 1.15$.

Ces paramètres se basent sur les valeurs données dans [BT02]. Ils permettent d'obtenir une distribution des degrés et un coefficient d'agglomération similaire à ceux mesurés dans l'Internet lors de leur étude.

3.2 Environnement d'exécution

Les simulations sont exécutées sur un calculateur équipé d'un processeur Intel Xeon W5580 3.20 Ghz avec 64 GB de RAM (la JVM est limitée à 50 GB) sur une distribution 64-bits Fedora 12, le noyau Linux 2.6.32 et le JRE v1.6.0. Les temps de simulation les plus longs sont observés sur les topologies de 5 000 routeurs. L'exécution du scénario 1 dure 52 minutes, le scénario 2 dure 48 minutes et le scénario 3 dure 16 minutes.

3.3 Résultats de simulation

Je présente maintenant les résultats des simulations des trois scénarios définis précédemment pour les solutions A et B.

Le nombre total d'entrées annoncées dans les messages de mise à jour BGP est calculé sur les topologies sans partition. Elle permet de quantifier les informations transmises entre les routeurs pour les différents scénarios. Cette mesure nous servira par la suite comme *valeurs de références* pour déterminer le poids des liens et des nœuds pour le calcul des bipartitions en fonction des solutions A et B. À partir des bipartitions, le nombre d'entrées annoncées à travers les liens frontières est calculé. Rappelons que ces bipartitions sont optimales. Le volume de communication mesuré dans les liens frontières sera inférieur à tout autre algorithme de partition. Une borne inférieure du coût de communication est ainsi obtenue.

Pour les solutions A et B, les expérimentations sont exécutées sur la bipartition des topologies. Seul le nombre d'entrées annoncées transmises sur les liens frontières est calculé. Il est donné pour les différents scénarios et solutions proposés dans le tableau 5.3. Son évolution en fonction de la taille des topologies est donnée en figure 5.4.

Résultats de références sur les topologies sans partition

Globalement, dans le cas des topologies sans partition, suivant les scénarios, le nombre d'entrées varie entre $1\,10^8$ et $2.5\,10^8$ pour 5 000 routeurs. On observe en moyenne une seule entrée annoncée par message de mise à jour. L'augmentation du nombre d'entrées est linéaire en leur racine carrée. Il est donc possible par la suite d'extrapoler ce nombre sur des topologies de grande taille (de 40K à plus de 100K routeurs) et ainsi déterminer le temps requis pour l'échange des messages.

Scénario 1. Il est le plus efficace en terme de messages échangés comparé aux autres scénarios. Un seul message est reçu à la fois. Un état stable du système est donc perturbé au fur et à mesure de la réception des messages. La concentration spatiale des messages lors de l'initialisation des routeurs et la faible variation des délais de transmission permet de recevoir rapidement les meilleures routes. L'annonce successive de routes de plus grande longueur est évitée.

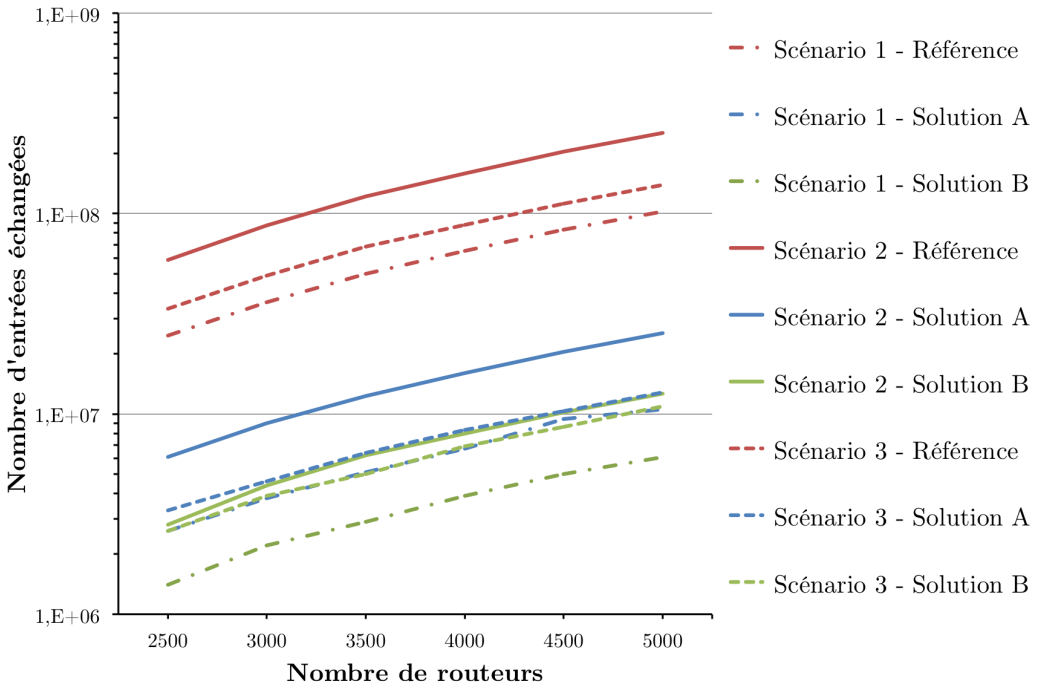


FIGURE 5.4 – Évolution du nombre d'entrées annoncées dans les messages de mise à jour BGP.

Scénario 2. Il est le moins efficace. La combinaison entre la concentration spatiale des messages lors de l'initialisation des routeurs et les délais aléatoires de transmission des messages expliquent ces résultats. Ce scénario est équivalent à une synchronisation retardée de toutes les paires de routeurs. Les mises à jour interférant les unes avec les autres, de mauvaises routes sont annoncées plusieurs fois.

Scénario 3. Les routeurs sont ajoutés les uns après les autres dans le réseau. Les perturbations du système sont causées par un ensemble de messages concentrés à partir d'un routeur. Une route annoncée et enregistrée dans la Loc-RIB d'un routeur peut devenir obsolète à l'arrivée d'un nouveau routeur. Un nombre de messages plus important que dans le scénario 1 est nécessaire pour converger, mais reste limité en comparaison avec le scénario 2.

Au final, le scénario 2 produit le plus grand nombre d'entrées échangées alors que les résultats des scénarios 1 et 3 sont de même ordre de magnitude.

Résultats des simulations sur la bipartition des topologies avec la solution A

Dans la solution A, pour les trois scénarios et l'ensemble des topologies, environ 10% du nombre total d'entrées (valeurs de références) transitent entre les partitions.

Contrairement à la solution B, aucune copie des routeurs n'est conservée par les LP. Cette solution est donc plus économe en mémoire mais un plus grand nombre de messages doit transiter par le réseau.

	2.5k	3k	3.5k	4k	4.5k	5k
Scénario 1 ($\times 10^6$)						
Sans partition	24,6	36,1	50,1	65,0	83,1	102,4
Sol A avec bipartition	2,6	3,8	5,1	6,7	9,4	10,5
Sol B avec bipartition	1,4	2,2	2,9	3,9	5	6,1
Scénario 2 ($\times 10^6$)						
Sans partition	58,7	87,0	121,7	158,8	204,3	252,8
Sol A avec bipartition	6,1	9,0	12,3	16	20,4	25,2
Sol B avec bipartition	2,8	4,4	6,2	8	10,2	12,6
Scénario 3 ($\times 10^6$)						
Sans partition	33,5	49,0	68,4	88,0	111,7	138,8
Sol A avec bipartition	3,3	4,6	6,4	8,3	10,3	12,8
Sol B avec bipartition	2,6	3,9	5	6,9	8,6	10,9

TABLE 5.3 – Nombre d'entrées annoncées dans les messages de mise à jour BGP.

Résultats de simulation sur la bipartition des topologies avec la solution B

La solution B a été proposée afin de réduire d'avantage les communications entre les LP comparé à la solution A.

Nous observons pour les trois scénarios une amélioration du nombre d'entrées transmises entre les parties. Avec la solution B, en comparaison avec les valeurs de références, seules 5,6% des entrées sont transmises avec le scénario 1, 5% avec le scénario 2 et 8% avec le scénario 3.

Coût des communications

À partir du nombre d'entrées échangées et de leur taille moyenne, le temps de transmission des messages peut être calculé. D'après le tableau 5.3, nous savons combien d'entrées transitent entre les routeurs frontières. Leur taille correspond au nombre d'AS formant la route pour atteindre un préfixe donné, elle varie en moyenne entre 3.6 et 4.9 suivant les scénarios.

Le temps de transmission des entrées entre deux LP est déterminé à partir des performances du réseau local du laboratoire de INRIA. Chaque LP est exécuté sur un ordinateur différent connecté par une interface Ethernet de 1 Gbps. La transmission des informations entre les LP repose sur le protocole TCP/IP et les sockets JAVA. Le temps de transmission d'un paquet TCP (incluant les temps d'attente des paquets ACK et TCP) contenant un entier entre deux ordinateurs est de 2,6 ms. Le temps d'envoi de 1 000 paquets les uns après les autres est de 0,26 ms par paquet en moyenne. Nous savons que la taille moyenne des entrées est comprise entre 3.6 et 4.9. Un seul paquet TCP est donc nécessaire pour transmettre une entrée. Nous considérons donc un temps de transmission par paquet de 0,26 ms.

Solution A - Temps de transmission Comparons les différents scénarios avec la solution A sur une topologie de 5k routeurs.

— Scénario 1, $10\,10^6$ entrées sont transmises, représentant un temps de transmission de 2 745 s (environ 45 minutes).

— Scénario 2, $25\,10^6$ entrées sont transmises, représentant un temps de transmission de 6 561 s (environ 110 minutes).

— Scénario 3, $12\,10^6$ entrées sont transmises, représentant un temps de transmission de 3 328 s (environ 55 minutes).

Solution B - Temps de transmission Comparons les différents scénarios avec la solution B sur une topologie de 5K routeurs.

- Scénario 1, $6.1\,10^6$ entrées sont transmises, représentant un temps de transmission de 1 586 s (environ 27 minutes).
- Scénario 2, $12.6\,10^6$ entrées sont transmises, représentant un temps de transmission de 3 276 s (environ 54,6 minutes).
- Scénario 3, $10.9^{\,6}$ entrées sont transmises, représentant un temps de transmission de 2 834 s (environ 47 minutes).

Considérons maintenant le pire cas, la solution A appliquée au scénario 2 avec 5K routeurs. Comme le nombre d'entrées mesurées et leur taille est linéaire en leur racine carrée, il est possible d'extrapoler ce résultat. Dans le cas d'une topologie de 10K routeurs, 13 122 s supplémentaires seraient nécessaires et 131 220 secondes (environs 36 heures) pour une topologie de 100K routeurs.

Ces résultats montrent la faisabilité de la distribution des simulations de BGP dans DRMSim. Certains scénarios peuvent générer un volume important de communications. Cependant, un délai de 36 heures pour la transmission des messages reste raisonnable dans la cas d'une simulation sur 100K routeurs. De plus, en supposant que les LP soient connectés entre eux par des liens de 10 Gbps, le temps de transmission tomberait à 3 heures.

4 Conclusion

L'évolution rapide d'Internet nécessite le développement d'outils adaptés pour l'étude de BGP et des nouveaux schémas de routage proposés pour le remplacer. L'évolution de DRMSim vers le paradigme de simulation à événements discrets distribuée et parallèle semble être une technique prometteuse. Il est possible de faire abstraction de la taille des topologies, mais au prix d'un temps de calcul plus long.

L'objectif principal de cette étude a été de quantifier ce temps supplémentaire pour la simulation de BGP dans une architecture distribuée sur des topologies composées de 10K à 100K routeurs. Les messages de mise à jour de BGP lors de l'annonce de nouvelles routes sont l'une des principales sources de communication entre routeurs lors de l'initialisation des tables de routage. Le nombre d'entrées transmises entre les routeurs a été mesuré pour différents scénarios d'exécution de BGP. À partir de ces informations, une bipartition optimale des topologies a été calculée. Le scénario 2 (scénario 1 avec des délais aléatoires) est celui qui engendre le plus d'échanges entre les routeurs frontières. Dans ce cas, le temps additionnel peut atteindre jusqu'à 36 heures de calcul pour une topologies de 100K routeurs. Dans le cas d'un réseau équipé d'interfaces Ethernet de 10 Gbps, ce temps de calcul tombe à 3 heures.

Deux modèles de communication ont été évalués. La synchronisation des routeurs frontières présentée dans la solution B permet une réduction significative des communications. Avec cette méthode, il a été possible de réduire de moitié le temps évalué pour la transmission des messages dans les scénarios 1 et 2.

Méthodes de réduction du nombre de messages. Un facteur permettant la réduction du nombre de messages de mise à jour échangés est le paramètre de MRAI. Dans [GP01], il a été montré qu'avec un $MRAI = 0$ seconde, le nombre de messages nécessaires pour converger augmente significativement. Plus la valeur du MRAI augmente, plus le nombre de messages de mise à jour diminue. Le temps de convergence atteint alors une moyenne minimale correspondant à la valeur optimale du MRAI. Au-delà de cette valeur, le temps de convergence augmente à nouveau alors que le nombre total de messages de mise à jour ne change plus de manière significative. Dans le contexte de l'étude de la distribution des

simulations de BGP, il est critique de pouvoir déterminer la valeur de MRAI pour laquelle décroître le taux de messages de mise à jour n'augmente pas le temps de convergence tout en limitant la quantité de mémoire supplémentaire requise.

Il est aussi possible de réduire le volume de messages échangés en considérant des techniques plus avancées. Le Path Exploration Damping (PED) [HRA10] est utilisé dans BGP dans ce but. Cependant, cette technique n'est pas disponible dans le modèle de BGP inclus dans DRMSim et devra lui être intégrée.

Au final, un tel simulateur parallèle et distribué semble réalisable pour la simulation de BGP. DRMSim pourrait être étendu aux modèles de distribution proposés dans cette étude.

Limitations de l'étude. Cette étude s'est concentrée sur le temps minimal auquel on peut s'attendre pour la transmission des messages de BGP entre les LP. Seuls trois scénarios ont été simulés à partir d'un seul modèle de topologie. D'autres scénarios et types de topologie pourraient engendrer des volumes de communication bien plus importants.

De plus, une bipartition optimale a été calculée à partir d'un programme d'entier mixte. Un tel calcul n'a été possible que pour des topologies d'au plus 5 000 nœuds. Pour de plus grandes tailles, il est nécessaire d'utiliser des algorithmes de partition tels que ceux proposés dans la librairie METIS [KK98a]. Cependant, le calcul des partitions sera moins bon, d'avantage de messages seront échangés entre les routeurs frontières.

Enfin, l'empreinte mémoire de la solution B est plus grande que la solution A. En pratique, une plus grande partition de la topologie devra être calculée et un plus grand nombre de LP sera nécessaire pour atteindre des topologies en $O(100k)$.

Conclusion et perspectives

Dans cette thèse, deux axes d'étude en lien avec la problématique du routage dans les grands réseaux ont été discutés. Bien que distincts, ces deux axes d'étude sont complémentaires pour le développement de nouvelles solutions de routage dans l'Internet. Dans un premier temps, cette thèse s'est concentrée sur le calcul de la propriété métrique de l'hyperbolicité au sens de Gromov dans les grands graphes. En prouvant une petite valeur d'hyperbolicité dans le graphe on s'assure de l'efficacité de certains schémas de routage. Dans un second temps, une nouvelle plate-forme de simulation de modèles de routage a été présentée. Les performances de ces nouveaux schémas de routage peuvent ainsi être évaluées.

Calcul de l'hyperbolicité

Les méthodes de calcul de l'hyperbolicité présentées dans le chapitre 1 et le chapitre 2 ont montré de bons résultats en pratique sur de grands graphes. Ces résultats ont été obtenus en travaillant en parallèle sur deux aspects du calcul de l'hyperbolicité. Un premier aspect porte sur les méthodes de pré-traitement de graphe. Elles permettent une réduction significative de la taille des graphes sur lesquels calculer l'hyperbolicité. Un second aspect porte sur le parcours efficace des quadruplets à évaluer pour le calcul de la valeur exacte de l'hyperbolicité. Les principales contributions apportées dans cette première partie de thèse sont :

1. une nouvelle méthode pour la construction de substituts en $O(nm)$ basée sur la décomposition du graphe par les cliques-séparatrices ;
2. un nouvel algorithme pour le calcul exact de l'hyperbolicité, efficace sur des graphes composés de 58 000 sommets, combinant la méthode de décomposition par les cliques-séparatrices et la notion de paires-éloignées ;
3. une heuristique capable d'approcher l'hyperbolicité de graphes composés de plusieurs millions de sommets.

Méthodologie par la recherche de bornes inférieures et supérieures. L'idée essentielle suivie dans ces nouvelles méthodes de calcul de l'hyperbolicité est la recherche de certificats permettant de rapidement réduire l'écart entre les bornes inférieures et supérieures de l'hyperbolicité afin de prouver au plus tôt l'optimalité de la solution. Cette méthode de recherche des minimaux et maximaux dans le but de réduire l'espace de recherche d'un problème pourrait être réutilisée dans d'autres problématiques. Bien que cette méthode ne permette pas, dans le cas de l'hyperbolicité, de réduire la complexité théorique de l'algorithme, c'est néanmoins la raison de ses bonnes performances en pratique.

Un algorithme pour les graphes de petite hyperbolicité. L'algorithme 1 est particulièrement efficace dans les graphes de grande hyperbolicité. L'ordre de visite des quadruplets permet à l'algorithme de trouver rapidement la valeur optimale de l'hyperbolicité. L'écart entre les bornes inférieures et supérieures peut être rapidement réduit. À l'opposé, lorsque cette valeur est petite, l'algorithme va devoir visiter un grand nombre de quadruplets avant de trouver la valeur optimale de l'hyperbolicité et la prouver.

Ainsi, il est intéressant de savoir déterminer rapidement si un graphe à une grande valeur d'hyperbolicité. Une méthode pourrait être la recherche de grands cycles dans le graphe. Une heuristique dans ce sens est d'ailleurs proposée dans [KLNS].

Un second axe d'étude est le développement d'un algorithme performant dans les graphes de petite hyperbolicité. Les travaux dans [CD14] ont permis de caractériser les graphes $\frac{1}{2}$-hyperboliques. Les auteurs ont montré que chercher si un graphe est $\frac{1}{2}$-hyperbolique est équivalent à chercher un C_4 dans le graphe. Ils ont aussi amélioré le meilleur algorithme en un nouveau d'une complexité temporelle en $O(n^{3.26})$.

Un algorithme exact dans les très grands graphes. Une limitation majeure de l'algorithme 1 est sa complexité quadratique en mémoire induite par le calcul de la matrice des plus courts chemins. Des graphes composés de plusieurs millions de sommets ne sont pas atteignables. Pourtant, dans le cas d'une grande valeur d'hyperbolicité et d'une distribution des distances adéquate (peu de paires de sommets à distance $\geq 2\delta$), l'algorithme pourrait rapidement calculer l'hyperbolicité d'un tel graphe. Il est donc important de diminuer sa complexité en mémoire.

Une autre possibilité serait de travailler à un accès partagé de la matrice des distances et de la liste triée des paires de sommets distribuées à travers plusieurs unités de calcul chacune équipée de leur propre mémoire.

Heuristique dans les graphes planaires. L'heuristique qui a été proposée dans le chapitre 2 montre de bons résultats dans les cartes CAIDA, DIMES et les graphes de collaboration. Cependant, de moins bons résultats sont obtenus sur les graphes des réseaux routiers roadNet. Ces graphes ont la particularité d'être quasi-planaires et de grande hyperbolicité. Or, notre heuristique a été conçue à partir de graphes sans échelle. Il est donc intéressant de réfléchir au développement d'heuristiques tirant partie des propriétés structurelles de différentes catégories de graphes.

Évaluation des performances des schémas de routage

La plate-forme de simulation DRMSim permet l'évaluation des performances de divers schémas de routage et leur comparaison à BGP. Un nombre varié de modèles de routage compacts a pu être expérimenté sur divers types de topologies jusqu'à des tailles de l'ordre de $O(10k)$ sommets. Dans cette seconde partie de thèse, les trois principales contributions sont :

1. le développement d'une architecture permettant l'extension des fonctionnalités du simulateur selon les besoins du modèle de routage et des scénarios de simulation ;
2. l'aide à la préparation des expérimentations présentées dans [Gla13] et au développement des modèles à vecteur de distances sans boucle [DDFM12] et de routage géographique [KD14] ;
3. l'étude pour le développement d'une version parallèle et distribuée de DRMSim dans le cadre de la simulation de BGP afin d'anticiper les futurs besoins de simulations à grande échelle.

Un modèle de BGP plus complet. Le modèle de BGP présent dans le simulateur ne propose pas de politiques de routage. L'intégration des politiques de routage peut reposer dans un premier temps sur un modèle simplifié des relations inter-AS [Gao01]. Un AS annonce toujours ses meilleures routes à ses AS clients. Si une meilleure route est annoncée depuis un AS client, alors celle-ci est de nouveau annoncée aux AS pairs et aux AS fournisseurs d'accès.

Dans un second temps, il est possible d'intégrer au sein du simulateur le langage de spécification des politiques de routage (ou RPSL, Routing Policy Specification Language) [ABG+97]. Des règles spécifiques à chaque AS pourraient être chargées à l'initialisation de la topologie.

Expérimentations de nouveaux schémas de routage. DRMSim a démontré sa capacité pour la simulation de schémas de routage compacts et à vecteur de chemins. Il est possible d'intégrer d'avantage de modèles de routage appartenant à d'autres catégories. Dans ce but, deux modèles de routage sont en cours d'intégration. Un premier modèle de routage géographique [KD14] en collaboration avec l'université catholique de Louvain-la-Neuve, un second pour le routage sans boucle [DDFM12] en collaboration avec l'université de L'Aquila.

Amélioration des possibilités de simulation. La configuration du simulateur repose sur l'écriture d'un fichier de configuration avec une structure et une syntaxe définie. Celui-ci est constitué d'un ensemble de clés associées aux paramètres de simulation. Bien que permettant une configuration complète de la simulation, l'édition d'un tel fichier n'est pas aisée et requiert un certain temps d'apprentissage. Afin de simplifier l'exécution des expérimentations présentées en section 5.2 du chapitre 4, l'équipe du Labri [1] a développé une interface graphique de contrôle et d'exécution des scénarios de simulation et de visualisation de la topologie [AMM+10]. Leur intégration à la branche officielle de développement de DRMSim et leur adaptation aux particularités des modèles de routage déjà présents dans le simulateur en simplifieront l'usage.

1. Laboratoire Bordelais de Recherche en Informatique

Bibliographie

[ABG+97] Cengiz Alaettinoglu, Tony Bates, Elise Gerich, Daniel Karrenberg, David Meyer, Marten Terpstra et Curtis Villamizer : Routing Policy Specification Language (RPSL), 1997.

[AD14] Muad Abu-Ata et Feodor F. Dragan : Metric tree-like structures in real-life networks : an empirical study. *CoRR*, abs/1402.3364, 2014.

[AGM+04] Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, Noam Nisan et Mikkel Thorup : Compact Name-independent Routing with Minimum Stretch. *In Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '04, pages 20–24, New York, NY, USA, 2004. ACM.

[AMM+10] David Auber, Patrick Mary, Morgan Mathiaut, Jonathan Dubois, Antoine Lambert, Daniel Archambault, Romain Bourqui, Bruno Pinaud, Maylis Delest et Guy Melançon : Tulip : a Scalable Graph Visualization Framework. *In* Sadok Ben Yahia et Jean-Marc Petit, éditeurs : *Extraction et Gestion des Connaissances (EGC) 2010*, volume RNTI E-19, pages 623–624, Hammamet, Tunisia, janvier 2010. RNTI.

[ASHM13] Aaron B. Adcock, Blair D. Sullivan, Oscar R. Hernandez et Michael W. Mahoney : Evaluating OpenMP Tasking at Scale for the Computation of Graph Hyperbolicity. *In 9th International Workshop on OpenMP (IWOMP)*, volume 8122 de *Lecture Notes in Computer Science*, pages 71–83, Canberra, Australia, 2013. Springer.

[Bar00] S Bartholomew : The Art of Peering. *BT Technology Journal*, 18(3):33–39, 2000.

[Bat] Tony Bates : CIDR Reports. http://www.cidr-report.org/as2.0/.

[BBCK+07] Anat Bremler-Barr, Nir Chen, Jussi Kangasharju, Osnat Mokryn et Yuval Shavitt : Bringing order to BGP : decreasing time and message complexity. *In* Indranil Gupta et Roger Wattenhofer, éditeurs : *PODC*, pages 368–369. ACM, 2007.

[BC03] Hans-Jürgen Bandelt et Victor Chepoi : 1-Hyperbolic Graphs. *SIAM Journal on Discrete Mathematics*, 16(2):323–334, 2003.

[BCC06] T. Bates, E. Chen et R. Chandra : RFC 4456 : BGP Route Reflection : An Alternative to Full Mesh Internal BGP (IBGP). Rapport technique, IETF, 2006.

[Bir] Bird : The BIRD Internet Routing Daemon. http://bird.network.cz/.

[BKM01] Gunnar Brinkmann, Jack H. Koolen et Vincent Moulton : On the Hyperbolicity of Chordal Graphs. *Annals of Combinatorics*, 5(1):61–69, 2001.

[BM86] H.-J. Bandelt et H.M. Mulder : Distance-hereditary graphs. *Journal of Combinatorial Theory, Series B*, 41(2):182–208, 1986.

[Bon] BonFIRE : Building service testbeds for Future Internet Research and Experimentation. http://www.bonfire-project.eu/.

[BPK10] Marián Boguñá, Fragkiskos Papadopoulos et Dmitri V. Krioukov : Sustaining the Internet with Hyperbolic Mapping. *Nature Communications*, 1(62):1–18, octobre 2010.

[BPS10a] Anne Berry, Romain Pogorelcnik et Geneviève Simonet : An introduction to clique minimal separator decomposition. *Algorithms*, 3(2):197–215, 2010.

[BPS10b] Anne Berry, Romain Pogorelcnik et Geneviève Simonet : Efficient clique decomposition of a graph into its atom graph. Rapport technique RR-10-07, INRIA, mars 2010.

[BT02] Tian Bu et Donald F. Towsley : On Distinguishing between Internet Power Law Topology Generators. *In INFOCOM*, volume 2, pages 638–647, 2002.

[CAI13b] The Center for Applied Internet Data Analysis CAIDA : The CAIDA UCSD AS Relationships Dataset, 2000 to 2014. http://www.caida.org/data/active/as-relationships/, 2013.

[CAI14a] The Center for Applied Internet Data Analysis CAIDA : The CAIDA AS Ranking. http://as-rank.caida.org/, 2014.

[CAI14b] The Center for Applied Internet Data Analysis CAIDA : The CAIDA UCSD AS Relationships Dataset, 2014-09-01. http://data.caida.org/datasets/as-relationships/serial-1/20140901.as-rel.txt.bz2, 2014.

[CD14] David Coudert et Guillaume Ducoffe : On the recognition of C_4-free and 1/2-hyperbolic graphs. Rapport de recherche RR-8458, INRIA, janvier 2014.

[CDE+08] Victor Chepoi, Feodor F. Dragan, Bertrand Estellon, Michel Habib et Yann Vaxès : Notes on diameters, centers, and approximating trees of δ-hyperbolic geodesic spaces and graphs. *Electronic Notes in Discrete Mathematics*, 31:231–234, 2008.

[CDE+12] Victor Chepoi, Feodor F. Dragan, Bertrand Estellon, Michel Habib, Yann Vaxès et Yang Xiang : Additive Spanners and Distance and Routing Labeling Schemes for Hyperbolic Graphs. *Algorithmica*, 62(3-4):713–732, 2012.

[CE80] William H. Cunningham et Jack Edmonds : A combinatorial decomposition theory. *Canadian Journal of Mathematics*, 32(3):734–765, 1980.

[CH10] John Chakerian et Susan Holmes : distory : Distance Between Phylogenetic Histories, 2010.

[CH12] John Chakerian et Susan Holmes : Computational Tools for Evaluating Phylogenetic and Hierarchical Clustering Trees. *Journal of Computational and Graphical Statistics*, 21(3):581–599, 2012.

[Cis81] Cisco : Interior Gateway Routing Protocol (IGRP). Rapport technique, Cisco, 1981.

[CM79] K. Mani Chandy et Jayadev Misra : Distributed Simulation : A Case Study in Design and Verification of Distributed Programs. *IEEE Trans. Software Eng.*, 5(5):440–452, 1979.

[CML11] Eunjoon Cho, Seth A. Myers et Jure Leskovec : Friendship and mobility : user movement in location-based social networks. *In ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1082–1090, San Diego, CA, 2011. ACM.

[Cou14] David Coudert : Gromov hyperbolicity of graphs : C source code. http://www-sop.inria.fr/members/David.Coudert/code/hyperbolicity.shtml, May 2014.

[Cun82] William H. Cunningham : Decomposition of directed graphs. *SIAM Journal on Algebraic Discrete Methods*, 3(2):214–228, 1982.

[DBI+11a] D.Papadimitriou, B.Sales, I.Caragiannis, P.Canellopoulos, S.Sahhaf, W.Tavernier, N.Hanusse, C.Glacet, J.-C.Delvenne, D.Careglio, L.Fabrega, P.Vila, M.Camelo et F.Tarissan : EULER Deliverable D2.2, Routing Scheme and Design Specification, 2011.

[DBI+11b] D.Papadimitriou, B.Sales, I.Caragiannis, P.Canellopoulos, S.Sahhaf, W.Tavernier, N.Hanusse, C.Glacet, J.-C.Delvenne, D.Careglio, L.Fabrega, P.Vila, M.Camelo et F.Tarissan : EULER Deliverable D4.6, Routing Scheme and Design Specification, 2011.

[DBKMS07] Mohamed Didi Biha, Bangaly Kaba, Marie-Jean Meurs et Eric SanJuan : Graph decomposition approaches for terminology graphs. *In MICAI 2007 : Advances in Artificial Intelligence*, volume 4827 de *Lecture Notes in Computer Science*, pages 883–893. Springer, 2007.

[DCL+12] D.Coudert, C.Glacet, L.Hogie, A.Lancin, N.Nisse, D.Papadimitriou, F.Tarissan, M.Latapy, D.Bernardes, A.Medem, W.Tavernier et J. Perellò Muntan : EULER Deliverable D4.2, Experimental methodology, scenarios, and tools, 2012.

[DDFM12] Gianlorenzo D'Angelo, Mattia D'Emidio, Daniele Frigioni et Vinicio Maurizio : Engineering a new loop-free shortest paths routing algorithm. *In 11th International Symposium on Experimental Algorithms (SEA2012)*, volume 7276 de *Lecture Notes in Computer Science*, pages 123–134, Bordeaux, France, juin 2012. Springer.

[DG07] Y. Dourisboure et C. Gavoille : Tree-decompositions with bags of small diameter. *Discrete Mathematics*, 307(16):2008–2029, 2007.

[DHK+11] Andreas Dress, Katharina Huber, Jacobus Koolen, Vincent Moulton et Andreas Spillner : *Basic Phylogenetic Combinatorics*. Cambridge University Press, Cambridge, UK, décembre 2011.

[dMSV11] Fabien de Montgolfier, Mauricio Soto et Laurent Viennot : Treewidth and Hyperbolicity of the Internet. *In 10th IEEE International Symposium on Network Computing and Applications (NCA)*, pages 25–32, Boston, 2011. IEEE.

[DP09] Ran Duan et Seth Pettie : Fast algorithms for (max, min)-matrix multiplication and bottleneck shortest paths. *In 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 384–391, New York, NY, USA, 2009. SIAM.

[DR04] Xenofontas A. Dimitropoulos et George F. Riley : Large-Scale Simulation Models of BGP. *In* Doug DeGroot, Peter G. Harrison, Harry A. G. Wijshoff et Zary Segall, éditeurs : *MASCOTS*, pages 287–294. IEEE Computer Society, 2004.

[Dra13] F.F. Dragan : Tree-Like Structures in Graphs : A Metric Point of View. *In 39th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 8165 de *Lecture Notes in Computer Science*, pages 1–4. Springer, 2013.

[DRM09] DRMSim : A Dynamic Routing Simulator. http://drmsim.gforge.inria.fr/, 2009.

[DS04] Shivani Deshpande et Biplab Sikdar : On the impact of route processing and MRAI timers on BGP convergence times. *In GLOBECOM*, pages 1147–1151. IEEE, 2004.

[Dua14] Ran Duan : Approximation Algorithms for the Gromov Hyperbolicity of Discrete Metric Spaces. *In 11th Latin American Theoretical Informatics Symposium (LATIN)*, volume 8392 de *Lecture Notes in Computer Science*, pages 285–293, Montevideo, Uruguay, 2014. Springer.

[DWP+11] D.Papadimitriou, W.Tavernier, P.Audenaert, N.Hanusse, A.Lancin, I.Caragiannis, D.Careglio, et F.Tarissan : EULER Deliverable D4.1, Performance objectives, evaluation criteria and metrics, 2011.

[Emu] Emulab : Network Emulation Testbed. http://www.emulab.net/.

[EUL14] EULER : EULER. http://www.euler-fire-project.eu/, 2011-2014.

[FFF99] Michalis Faloutsos, Petros Faloutsos et Christos Faloutsos : On Power-Law Relationships of the Internet Topology. *In In SIGCOMM*, pages 251–262, 1999.

[FIV12] Hervé Fournier, Anas Ismail et Antoine Vigneron : Computing the Gromov hyperbolicity of a discrete metric space. *CoRR*, abs/1210.3323, 2012.

[Fuj90] Richard M. Fujimoto : Parallel Discrete Event Simulation. *Commun. ACM*, 33(10):30–53, octobre 1990.

[Gal67] Tibor Gallai : Transitiv orientierbare graphen. *Acta Mathematica Hungarica*, 18(1):25–66, 1967.

[Gal82] Robert G Gallager : Distributed minimum hop algorithms. Rapport technique LIDS-P ; 1175, Massachusetts Institute of Technology. Laboratory for Information and Decision Systems, janvier 1982.

[Gao01] Lixin Gao : On Inferring Autonomous System Relationships in the Internet. *IEEE/ACM Trans. Netw.*, 9(6):733–745, décembre 2001.

[GLA93] J. J. Garcia-Luna-Aceves : Loop-free routing using diffusing computations. *IEEE/ACM Trans. Netw.*, 1(1):130–141, 1993.

[Gla13] Christian Glacet : *Algorithmes de routage : de la réduction des coûts de communication à la dynamique.* Thèse, Université Sciences et Technologies - Bordeaux I, décembre 2013.

[GP01] T. G. Griffin et B. J. Premore : An experimental analysis of BGP convergence time. *In Ninth International Conference on Network Protocols*, pages 53–61, 2001.

[GR00] Lixin Gao et Jennifer Rexford : Stable Internet Routing Without Global Coordination. *SIGMETRICS Perform. Eval. Rev.*, 28(1):307–317, juin 2000.

[Gro87] Micha Gromov : Hyperbolic Groups. *In* S.M. Gersten, éditeur : *Essays in Group Theory*, volume 8 de *Mathematical Sciences Research Institute Publications*, pages 75–263. Springer, New York, 1987.

[H+14] Luc Hogie *et al.* : Grph, the high performance graph library for Java (Version 1.4.17). http://grph.inria.fr, 2014.

[HCHC13] R. Housley, J. Curran, G. Huston et D. Conrad : RFC 7020 : The Internet Numbers Registry System. Rapport technique, IETF, 2013.

[How79] E. Howorka : On metric properties of certain clique graphs. *Journal of Combinatorial Theory, Series B*, 27(1):67–74, 1979.

[HP10] Michel Habib et Christophe Paul : A survey of the algorithmic aspects of modular decomposition. *Computer Science Review*, 4(1):41–59, 2010.

[HPTM10] Luc Hogie, Dimitri Papadimitriou, Issam Tahiri et Frédéric Majorczyk : Simulating Routing Schemes on Large-Scale Topologies. *In PADS*, pages 132–141. IEEE, 2010.

[HRA10] Geoff Huston, Mattia Rossi et Grenville Armitage : A Technique for Reducing BGP Update Announcements Through Path Exploration Damping. *IEEE J.Sel. A. Commun.*, 28(8):1271–1286, octobre 2010.

[iLa] iLab.t : The iMinds technical testing center. http://www.iminds.be/en/succeed-with-digital-research/technical-testing.

[IMU] IMUNES : Integrated Multiprotocol Network Emulator/Simulator. http://imunes.tel.fer.hr/.

[IPD+12] I.Caragiannis, P.Kanellopoulos, D.Coudert, C.Glacet, N. Hanusse, A.Lancin, N.Nisse, F.Tarissan et D.Papadimitriou : EULER Deliverable D3.3, Graph analysis / mining, 2012.

[Jak11] P. Jakma : Internet-Draft : Revisions to the BGP 'Minimum Route Advertisement Interval' draft-ietf-idr-mrai-dep-04. Rapport technique, IETF, 2011.

[Jef85] David R. Jefferson : Virtual Time. *ACM Transactions on Programming Languages and Systems*, 7:404–425, 1985.

[JL04] Edmond A. Jonckheere et Poonsuk Lohsoonthorn : Geometry of network security. *In American Control Conference*, volume 2, pages 976–981, Boston, MA, USA, 2004. IEEE.

[JS] J-Sim : A component-based, compositional simulation environment. https://sites.google.com/site/jsimofficial/.

[KD14] Alok Kumar et Jean-Charles Delvenne : A new route discovery scheme. http://tools.ietf.org/html/draft-kumar-route-discovery-00, 2014.

[KK98a] G. Karypis et V. Kumar : A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[KLNS] Adrian Kosowski, Bi Li, Nicolas Nisse et Karol Suchan : k-Chordal Graphs : from Cops and Robber to Compact Routing via Treewidth. Rapport technique RR-7888, INRIA, -.

[KM02] Jack H. Koolen et Vincent Moulton : Hyperbolic Bridged Graphs. *European Journal of Combinatorics*, 23(6):683–699, 2002.

[KNS13] W. Sean Kennedy, Onuttom Narayan et Iraj Saniee : On the Hyperbolicity of Large-Scale Networks. *CoRR*, abs/1307.0031, 2013.

[KPK+10] Dmitri V. Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat et Marián Boguñá : Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, septembre 2010.

[KPL+07] B. Kaba, N. Pinet, G. Lelandais, A. Sigayret et A. Berry : Clustering gene expression data using graph separators. *In silico biology*, 7(4):433–452, 2007.

[LABJ01] Craig Labovitz, Abha Ahuja, Abhijit Bose et Farnam Jahanian : Delayed Internet Routing Convergence. *IEEE/ACM Trans. Netw.*, 9(3):293–306, juin 2001.

[Lana] Aurélien Lancin : Algorithms to compute the atom decomposition and the substitutes of a graph. ftp://ftp-sop.inria.fr/members/Aurelien.Lancin/substitute/substitute.zip.

[Lanb] Aurélien Lancin : Algorithms to compute the hyperbolicity of a graph. ftp://ftp-sop.inria.fr/members/Aurelien.Lancin/hyperbolicity/HyperbolicityGrph-0.1.jar.

[Lei93] H.-G. Leimer : Optimal decomposition by clique separators. *Discrete Mathematics*, 113(1):99–123, 1993.

[LG12] François Le Gall : Faster Algorithms for Rectangular Matrix Multiplication. *In IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 514–523, New Brunswick, NJ, USA, 2012. IEEE.

[LHc+13] M. Luckie, B. Huffaker, k. claffy, A. Dhamdhere et V. Giotsas : AS Relationships, Customer Cones, and Validation. *In Internet Measurement Conference (IMC)*, pages 243–256, Oct 2013.

[LKF07] Jure Leskovec, Jon Kleinberg et Christos Faloutsos : Graph evolution : Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1):1–41, mars 2007.

[Lok09] Daniel Lokshtanov : Finding the longest isometric cycle in a graph. *Discrete Applied Mathematics*, 157(12):2670–2674, 2009.

[Mal98] G. Malkin : RFC 2453 : RIP Version 2 . Rapport technique, IETF, 1998.

[MB76] Robert M. Metcalfe et David R. Boggs : Ethernet : Distributed Packet Switching for Local Computer Networks. *Commun. ACM*, 19(7):395–404, juillet 1976.

[Mis86] Jayadev Misra : Distributed Discrete-Event Simulation. *ACM Comput. Surv.*, 18(1):39–65, 1986.

[MLH09] Clémence Magnien, Matthieu Latapy et Michel Habib : Fast computation of empirically tight bounds for the diameter of massive graphs. *ACM Journal of Experimental Algorithmics*, 13:10 :1.10–10 :1.9, février 2009.

[Moy98] J. Moy : RFC 2328 : OSPF Version 2. Rapport technique, IETF, 1998.

[Net] Netkit : The poor man's system to experiment computer networking. http://wiki.netkit.org.

[Nog09] Damien Noguès : δ-hyperbolicité et graphes. Mémoire de D.E.A., MPRI, Université Paris 7, 2009.

[NRS09] Nicolas Nisse, Ivan Rapaport et Karol Suchan : Distributed Computing of Efficient Routing Schemes in Generalized Chordal Graphs. *In SIROCCO'09*, pages 252–265, 2009.

[NRS12] Nicolas Nisse, Ivan Rapaport et Karol Suchan : Distributed computing of efficient routing schemes in generalized chordal graphs. *Theor. Comput. Sci.*, 444:17–27, 2012.

[NS-] NS-2 : The Network Simulator. http://www.isi.edu/nsnam/ns/.

[NS11] Onuttom Narayan et Iraj Saniee : The Large Scale Curvature of Networks. *Physical Review E*, 84:066108, décembre 2011.

[NST12] Onuttom Narayan, Iraj Saniee et Gabriel H. Tucci : Lack of spectral gap and hyperbolicity in asymptotic Erdös-Renyi sparse random graphs. *In 5th International Symposium on Communications, Control and Signal Processing (ISCCSP)*, pages 1–4, Rome, Italy, mai 2012. IEEE.

[Opea] OpenBGP : A FREE implementation of the Border Gateway Protocol, Version 4. http://www.openbgpd.org/.

[Opeb] OpenMP Architecture Review Board : OpenMP API version 3.0.

[Ora90] D. Oran : RFC 1142 : OSI IS-IS Intra-domain Routing Protocol. Rapport technique, IETF, 1990.

[par11] All partners : EULER Deliverable D3.1, Graph-based topology modelling, 2011.

[PKBV09] Fragkiskos Papadopoulos, Dmitri V. Krioukov, Marián Boguñá et Amin Vahdat : Greedy forwarding in scale-free networks embedded in hyperbolic metric spaces. *ACM SIGMETRICS Performance Evaluation Review*, 37(2): 15–17, septembre 2009.

[Pos81] Jon Postel : RFC 791 : Internet Protocol - DARPA Inernet Programm, Protocol Specification. Rapport technique, IETF, 1981.

[QU05] Bruno Quoitin et Steve Uhlig : Modeling the routing of an autonomous system with C-BGP. *IEEE Network*, 19(6):12–19, 2005.

[Qua] Quagga : The Quagga Routing Suite. http://www.nongnu.org/quagga/index.html.

[R C14] R Core Team : R : A Language and Environment for Statistical Computing, 2014.

[RLH06] Y. Rekhter, T. Li et S. Hares : RFC 4271 : A Border Gateway Protocol 4 (BGP-4). Rapport technique, IETF, 2006.

[RMK+09] Venugopalan Ramasubramanian, Dahlia Malkhi, Fabian Kuhn, Mahesh Balakrishnan, Archit Gupta et Aditya Akella : On the Treeness of Internet Latency and Bandwidth. *ACM SIGMETRICS Performance Evaluation Review*, 37(1):61–72, juin 2009.

[S+] William A. Stein *et al.* : Sage Mathematics Software (Version 6.1).

[SARK02] Lakshminarayanan Subramanian, Sharad Agarwal, Jennifer Rexford et Randy H. Katz : Characterizing the Internet Hierarchy from Multiple Vantage Points. *In INFOCOM*, 2002.

[SG11] Mauricio A. Soto Gómez : *Quelques propriétés topologiques des graphes et applications à internet et aux réseaux*. Thèse de doctorat, Univ. Paris Diderot (Paris 7), 2011.

[Sha75] Robert E. Shannon : Simulation : A Survey with Research Suggestions. *A I I E Transactions*, 7(3):289–301, 1975.

[Sha11] Yilun Shang : Lack of Gromov-Hyperbolicity in Colored Random Networks. *PanAmerican Mathematical Journal*, 21(1):27–36, 2011.

[Sim] SimBGP : A simple BGP simulator. http://www.bgpvista.com/simbgp.php.

[SS05] Yuval Shavitt et Eran Shir : DIMES : Let the Internet Measure Itself. *ACM SIGCOMM Computer Communication Review*, 35(5):71–74, octobre 2005.

[SSF] SSFNET : Scalable Simulation Framework (SSF). http://www.ssfnet.org/.

[Tar72] Robert Endre Tarjan : Depth-First Search and Linear Graph Algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[Tar85] Robert Endre Tarjan : Decomposition by clique separators. *Discrete Mathematics*, 55(2):221–232, 1985.

[TMS07] P. Traina, D. McPherson et J. Scudder : RFC 5065 : Autonomous System Confederations for BGP. Rapport technique, IETF, 2007.

[TZLL13] Mingdong Tang, Guoqiang Zhang, Tao Lin et Jianxun Liu : HDLBR : A name-independent compact routing scheme for power-law networks. *Computer Communications*, 36(3):351–359, 2013.

[ULK] ULK : The User-mode Linux Kernel. http://user-mode-linux.sourceforge.net/.

[VNU] VNUML : Vitual Network User Mode Linux. http://web.dit.upm.es/vnumlwiki/.

[VNX] VNX : Virtual Networks over Linux. http://www.dit.upm.es/vnx.

[WZ11] Yaokun Wu et Chengpeng Zhang : Hyperbolicity and Chordality of a Graph. *The Electronic Journal of Combinatorics*, 18(1):P43, 2011.

[Zeb] Zebra : Zebra Routing Software. http://www.zebra.org/.

Annexe

Applying clique-decomposition for computing Gromov hyperbolicity

Nathann Cohen, David Coudert, Guillaume Ducoffe, Aurélien Lancin

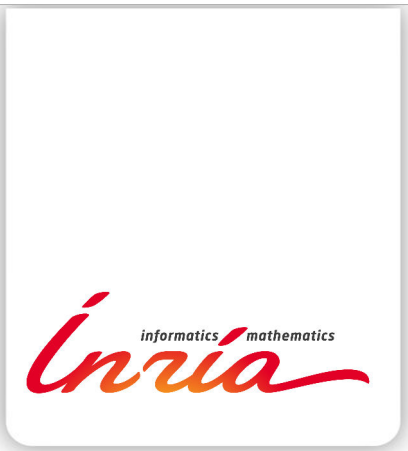
RESEARCH REPORT

N° 8535

May 2014

Project-Team COATI

ISSN 0249-6399 ISRN INRIA/RR--8535--FR+ENG



Applying clique-decomposition for computing Gromov hyperbolicity*

Nathann Cohen†, David Coudert‡§, Guillaume Ducoffe¶, Aurélien Lancin‡§

Project-Team COATI

Research Report n° 8535 — May 2014 — 30 pages

Abstract: The shortest-path metric d of a connected graph G is δ-*hyperbolic* if, and only if, it satisfies $\mathrm{d}(u,v)+\mathrm{d}(x,y) \leq \max\{\mathrm{d}(u,x)+\mathrm{d}(v,y), \mathrm{d}(u,y)+\mathrm{d}(v,x)\}+2\delta$, for every 4-tuple u,x,v,y of G. We investigate some relations between the hyperbolicity of a graph and the hyperbolicity of its *atoms*, that are the subgraphs resulting from the clique-decomposition invented by Tarjan [34, 45]. More precisely, we prove that the maximum hyperbolicity taken over all the atoms is at least the hyperbolicity of G *minus one*. We also give an algorithm to slightly modify the atoms, which is at no extra cost than computing the atoms themselves, and so that the maximum hyperbolicity taken over all the resulting graphs is *exactly* the hyperbolicity of G. An experimental evaluation of our methodology is provided for large collaboration networks. Finally, we deduce from our theoretical results the first *linear-time* algorithm to compute the hyperbolicity of an outerplanar graph.

Key-words: Hyperbolicity, algorithm, graph, decomposition.

* This work has been partially supported by ANR project Stint under reference ANR-13-BS02-0007, ANR program "Investments for the Future" under reference ANR-11-LABX-0031-01, and by European project FP7 EULER (Grant No.258307).

† LRI, Laboratoire de Recherche en Informatique, Université Paris-Sud 11, France.
‡ Inria, France
§ Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, 06900 Sophia Antipolis, France
¶ ENS Cachan, 61 Avenue du Président Wilson, 94230 Cachan, France

RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Décomposition par des cliques-séparatrices pour le calcul de l'hyperbolicité de Gromov

Résumé : La métrique d des plus courts chemins d'un graphe connexe G est *δ-hyperbolique* si, et seulement si, elle satisfait $\mathrm{d}(u,v)+\mathrm{d}(x,y) \leq \max\{\mathrm{d}(u,x)+\mathrm{d}(v,y), \mathrm{d}(u,y)+\mathrm{d}(v,x)\}+2\delta$, pour tout quadruplet u,x,v,y de G. Nous étudions la relation entre l'hyperbolicité d'un graphe et celle de chacun de ses *atomes*. Ces derniers sont les sous-graphes résultant de la décomposition d'un graphe par des cliques-séparatrices [34, 45]. Plus précisemment, nous montrons que l'hyperbolicité d'un atome est au plus l'hyperbolicité de G *moins un*. Nous proposons un algorithme pour modifier les atomes de sorte que la valeur maximale de l'hyperbolicité de ces atomes modifiés soit exactement l'hyperbolicité de G. La complexité de cet algorithme est la même que celle de la décomposition du graphe par des cliques-séparatrices. Nous évaluons expériementalement cette méthode sur des graphes de collaborations (co-auteurs). Enfin, nous proposons un algorithme pour calculer en temps linéaire l'hyperbolicité des graphes planaires extérieurs.

Mots-clés : Hyperbolicité, algorithme, graphe, décomposition.

Contents

1 Introduction **5**

2 Definitions and notations **6**

3 Hyperbolicity and clique-separators **7**
3.1 Hyperbolicity of $(a_1, a_2|b_1, b_2)$ 4-tuples . . . 7
3.2 Hyperbolicity of $(a|b_1, b_2, b_3)$ 4-tuples . . . 8
3.3 Separable sets . . . 9

4 Hyperbolicity and clique-decomposition **10**
4.1 Relating atoms and 4-tuples with large hyperbolicity . . . 10
4.2 Substitution method . . . 11
4.3 Additive approximation for hyperbolicity . . . 13

5 Substitution method for an exact computation **13**
5.1 Substitute graphs . . . 14
5.1.1 Basic step: single disconnection . . . 14
5.1.2 Extension to the atoms . . . 14
5.2 Implementation and complexity analysis . . . 15
5.2.1 Precomputation step and updates . . . 15
5.2.2 Applying simplification rules . . . 17
5.2.3 Complexity analysis . . . 18

6 Hyperbolicity of outerplanar graphs **18**
6.1 Outerplanar graphs with small hyperbolicity . . . 19
6.2 Substitute graphs of cycles . . . 20
6.3 Applying the substitution method in linear-time . . . 22

7 Experimental evaluation **23**
7.1 Datasets . . . 23
7.2 Empirical results . . . 24
7.3 Decomposition analysis . . . 25

8 Conclusion **27**

1 Introduction

In this paper, we primarily aim to study graph hyperbolicity, from an algorithmic point of view. This parameter was first introduced by Gromov in the context of automatic groups [26], then extended to more general metric spaces, including the shortest-path metrics of simple graphs. Hyperbolicity of unweighted graphs, and its tight relations to other metric parameters, has received growing attention over the last decades, especially due to its practical applications in approximation algorithms [13], routing [9], network security [29] and bioinformatics [12, 22], to name a few. The reader may refer to [1, 21] for a recent survey.

However, the computational cost of hyperbolicity has received less attention. So far, the best-known algorithm for computing the hyperbolicity of a graph [23] is impractical for large-scale graphs such as the graph of the Autonomous Systems of the Internet, road maps, etc. This both comes from its challenging implementation, relying on fast square matrix multiplications, and its time complexity which is strictly more than cubic. While practical advances have been made, improving the computational cost on certain graph classes (see [14, 31]), a recent theoretical work [15] suggests that an algorithm for the problem with a significant speed-up for *all graphs* is unlikely to exist. This motivates the study of structural properties that may help to decrease the running time of the computation of hyperbolicity.

Our approach relies on divide-and-conquer techniques, especially *graph decompositions*. Roughly, we aim to reduce the computation of the hyperbolicity of a graph to the computation of the hyperbolicity of (some of) its subgraphs, so that we can decrease the size of the input graphs to deal with. A first step toward this direction was the result in [41], where the author proved that the hyperbolicity of a connected graph is the maximum hyperbolicity taken over all the subgraphs obtained either via the modular decomposition [24], or with the split-decomposition [16].

We here address a similar question for the subgraphs obtained via the *clique-decomposition*, also known as *atoms*. This decomposition was first introduced by Tarjan in [45], then made unique by Leimer in [34]. On the theoretical side, it is already known that clique-decomposition can be applied to speed-up the computation of many graph parameters, including metric parameters that are related to hyperbolicity, such as tree-length [20]. Also, complex networks are expected to be decomposable w.r.t. clique-decomposition, especially when they are related to phylogenetical data, text-data mining, or distance data [6, 18, 30]. This makes our approach practical for large-scale graphs.

We will prove that while the hyperbolicity of a graph cannot be guessed from the hyperbolicity of its atoms (Section 3), it yields an approximation with additive constant 1 of this parameter (Section 4). Additionally, we characterize the cases for which this small additive distortion might happen. In Section 5, we will show how each atom can be modified in order to compute exactly the hyperbolicity, and provide a complexity analysis of the procedure. Experiments in Section 7 show the benefit of our method in terms of size of the graph, when applied to some real collaboration networks.

Finally, we will see in Section 6 that our method is beneficial for the class of outerplanar graphs, as it gives a linear-time algorithm to compute the hyperbolicity of these graphs.

Definitions and notations used in this paper will be introduced in Section 2.

2 Definitions and notations

We essentially rely on the graph terminology of [10, 19]. All graphs considered in this paper are finite, unweighted and simple. We here only emphasize some notions related to metric graph theory. The reader may refer to [3] for a survey about this domain. Given two vertices u, v, a *uv-path* of length $l \geq 0$ is a sequence of vertices $(u = v_0, v_1, \ldots, v_l = v)$, satisfying $\{v_i, v_{i+1}\}$ is an edge for every $0 \leq i \leq l-1$. In particular, G is *connected* if there exists a uv-path for all pair $u, v \in V$, and in such a case the *distance* $\mathrm{d}_G(u, v)$ is defined as the minimum length of a uv-path in G. Note that it yields a discrete metric space (V, d_G), also known as the shortest-path metric space of G. Also, we will write d instead of d_G whenever G is clear from the context. In the sequel, we will abuse of the distance notation to denote the distance $\mathrm{d}(u, X) = \min_{x \in X} \mathrm{d}(u, x)$ between a vertex u and a subset X.

Studying the properties of the metric space (V, d) requires stronger notions than the classical one of *induced subgraph*, as in general those subgraphs fail to be distance-preserving. Indeed, the wheel W_n is a good example, as it contains the cycle C_n as an induced subgraph, yet the shortest-path metric of the cycle is not induced by the shortest-path metric of the wheel (*e.g.* there is a shortcut between cyclic vertices, by using the central vertex of the wheel). We here restrict our study to *isometric subgraphs*. Formally, H is an isometric subgraph of the graph G if, and only if, it is connected and we have $\mathrm{d}_H(u, v) = \mathrm{d}_G(u, v)$, for all $u, v \in V(H)$.

δ-hyperbolic graphs. Graph hyperbolicity provides tight bounds on the worst additive distortion of the distances in a (connected) graph when its vertices are embedded into a weighted tree. Several definitions exist, some of them considering graph metrics that are slightly different than the usual shortest-path metric, but they are known to be equivalent up to a linear-function [5, 17, 26]. Moreover, 0-hyperbolic graphs are exactly the connected graphs whose given metric is a *tree metric*, which makes hyperbolicity a tree-likeness parameter. Especially, the shortest-path metric of a graph is 0-hyperbolic if, and only if, it is a *block-graph*, that is a graph whose biconnected components are complete subgraphs [4, 28]. This class of graphs includes trees and cliques, and a block-graph can be recognized in linear $O(n + m)$-time.

The four-point definition of hyperbolicity is now defined as follows.

Definition 1 (**4-points Condition, [26]**). *Let G be a connected graph. For every 4-tuple u, x, v, y of G, we define $\delta(u, v, x, y)$ as half of the difference between the two largest sums amongst*
$S_1 = \mathrm{d}(u, v) + \mathrm{d}(x, y)$,
$S_2 = \mathrm{d}(u, x) + \mathrm{d}(v, y)$, *and*
$S_3 = \mathrm{d}(u, y) + \mathrm{d}(v, x)$.

The graph hyperbolicity, denoted by $\delta(G)$, is equal to $\max_{u,x,v,y} \delta(u, v, x, y)$. Moreover, we say that G is δ-hyperbolic, *for every $\delta \geq \delta(G)$.*

It is straightforward, by the above definition, to compute graph hyperbolicity in $\theta(n^4)$-time. Currently, the best-known *theoretical* algorithm for the problem runs in $O(n^{3.69})$-time [23], and the best-known *practical* algorithm has $O(n^4)$-time complexity [14]. Recognizing graphs with small hyperbolicity upper-bounded by $\frac{1}{2}$ is computationally equivalent to decide whether there is a chordless cycle of length 4 in a graph, and it can be done in $O(n^{3.256689})$-time by using fast rectangular matrix multiplication [15, 33]. Note also that the hyperbolicity of a connected graph is the maximum hyperbolicity taken over all its biconnected components.

Atoms and clique-separators. Given a connected graph $G = (V, E)$, we name *separator* a subset of vertices $X \subset V$ such that the removal of X disconnects the graph. When the induced subgraph $G[X]$ is a complete graph, then we say that X is a *clique-separator*. More generally, we

say that X is a separator of $S \subseteq V$ if S intersects (at least) two distinct connected components of $G\backslash X$, and S is then said to be *separable*. The clique-decomposition of a graph (see Figure 1) is the collection of all its maximal sets of vertices that do not admit a clique-separator (we will call them *atoms*) [45]. The decomposition is unique [34], and it can be computed in $O(nm)$-time [34, 45].

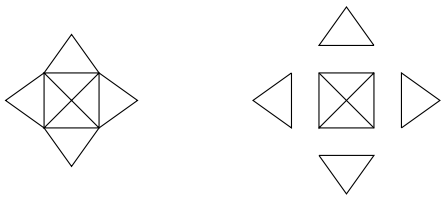


Figure 1: Clique-decomposition of a graph into five atoms.

Notations. Let us fix some notations for the proofs. Given two subsets A, B of vertices, we say that a separator X is an $(A|B)$-separator if it disconnects any $a \in A\backslash X$ from any $b \in B\backslash X$. In particular, any separator X containing A or B is an $(A|B)$-separator, and in such a case we call X a *trivial* $(A|B)$-separator.

We also denote by $(a|b_1, b_2, b_3)$ a 4-tuple such that $a \in A$ and $b_1, b_2, b_3 \in B$. In the same way, we denote by $(a_1, a_2|b_1, b_2)$ a 4-tuple such that $a_1, a_2 \in A$ and $b_1, b_2 \in B$.

Finally, a clique-separator is called a *clique-minimal separator* if there exists some pair u, v such that it is an inclusionwise minimal, non-trivial $(u|v)$-clique-separator. It has to be noted, perhaps counter-intuitively, that a clique-minimal separator may be contained into another one.

3 Hyperbolicity and clique-separators

We analyse in this section the relationship between the hyperbolicity of a graph and a given clique-separator, leading to the approximation with additive constant of Theorem 11. It begins with an observation about $(a_1, a_2|b_1, b_2)$ 4-tuples and the diameter $\mathrm{diam}(X) = \max_{u,v\in X} \mathrm{d}_G(u, v)$ of an $(A|B)$-separator X.

3.1 Hyperbolicity of $(a_1, a_2|b_1, b_2)$ 4-tuples

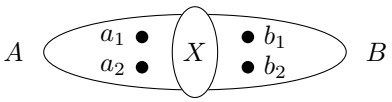


Figure 2: Illustration of an $(A|B)$-separator.

Lemma 2. *Let X be a an $(A|B)$-separator of a connected graph G. For every $(a_1, a_2|b_1, b_2)$ 4-tuple, we have $\delta(a_1, a_2, b_1, b_2) \leq \mathrm{diam}(X)$.*

Proof. Recall that we have:

$$\begin{aligned} S_1 &= \mathrm{d}(a_1, a_2) + \mathrm{d}(b_1, b_2), \\ S_2 &= \mathrm{d}(a_1, b_1) + \mathrm{d}(a_2, b_2), \\ S_3 &= \mathrm{d}(a_1, b_2) + \mathrm{d}(a_2, b_1). \end{aligned}$$

We can assume without loss of generality (w.l.o.g.) that $S_2 \geq S_3$. Writing $\mathrm{d}(v, X) = \min_{x \in X} \mathrm{d}(v, x)$, we also know that for all $i, j \in \{1, 2\}$:

$$\begin{aligned} \mathrm{d}(a_i, b_i) &\geq \mathrm{d}(a_i, X) + \mathrm{d}(b_i, X) \\ \mathrm{d}(a_i, a_j) &\leq \mathrm{d}(a_i, X) + \mathrm{diam}(X) + \mathrm{d}(a_j, X) \end{aligned}$$

If $S_1 \geq S_2$, we get:

$$\begin{aligned} S_2 =& \mathrm{d}(a_1, b_1) + \mathrm{d}(a_2, b_2) \\ \geq& \mathrm{d}(a_1, X) + \mathrm{d}(b_1, X) + \mathrm{d}(a_2, X) + \mathrm{d}(b_2, X) \\ \geq& \big[\mathrm{d}(a_1, X) + \mathrm{d}(a_2, X) + \mathrm{diam}(X)\big] + \\ & \big[\mathrm{d}(b_1, X) + \mathrm{d}(b_2, X) + \mathrm{diam}(X)\big] - 2\,\mathrm{diam}(X) \\ \geq& S_1 - 2\,\mathrm{diam}(X) \end{aligned}$$

Hence we have that $\delta(a_1, a_2, b_1, b_2) \leq (S_1 - S_2)/2 \leq \mathrm{diam}(X)$. It can be shown similarly that $S_3 \geq S_2 - 2\,\mathrm{diam}(X)$, and when $S_2 \geq S_1$ that $S_1 \geq S_2 - 2\,\mathrm{diam}(X)$. Consequently, $\delta(a_1, a_2, b_1, b_2) \leq \mathrm{diam}(X)$ in all cases. □

Corollary 3. *Let X be a an $(A|B)$-clique-separator of a connected graph G. For every $(a_1, a_2|b_1, b_2)$ 4-tuple, we have $\delta(a_1, a_2, b_1, b_2) \leq 1$.*

3.2 Hyperbolicity of $(a|b_1, b_2, b_3)$ 4-tuples

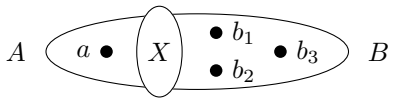


Figure 3: Illustration of an $(a|b_1, b_2, b_3)$-separator.

Note that X being a clique, each vertex $a \in A$ is at distance at least $\mathrm{d}(u, X)$ and at most $\mathrm{d}(u, X) + 1$ from any vertex of X. We now show how this can be used with respect to the hyperbolicity.

Lemma 4. *Let X be an $(A|B)$-clique-separator of a connected graph G, and let $a \in A$. If we add to G a vertex a^* adjacent to $\{x \in X : \mathrm{d}(a, x) = \mathrm{d}(a, X)\}$, then for every $b_1, b_2, b_3 \in B$ we have $\delta(a, b_1, b_2, b_3) = \delta(a^*, b_1, b_2, b_3)$.*

Proof. $\forall x \in X, \mathrm{d}(a, x) \in \{\mathrm{d}(a, X), \mathrm{d}(a, X) + 1\}$ holds as X is a clique. Consequently, for every $b \in B, \mathrm{d}(a, b) = \mathrm{d}(a^*, b) + \mathrm{d}(a, X) - 1$ and replacing a with a^* does not change the hyperbolicity of a, b_1, b_2, b_3. □

Lemma 5. *Let X be an $(A|B)$-clique-separator of a connected graph G. Given an $(a|b_1, b_2, b_3)$ 4-tuple, let $x \in X$ be such that $\mathrm{d}(a, x) = \mathrm{d}(a, X)$. We have $\delta(a, b_1, b_2, b_3) \leq \delta(x, b_1, b_2, b_3) + 1/2$.*

Proof. Let us assume w.l.o.g. that $S_1 \geq S_2 \geq S_3$, where $S_1 = d(a, b_1) + d(b_2, b_3)$, $S_2 = d(a, b_2) + d(b_1, b_3)$ and $S_3 = d(a, b_3) + d(b_1, b_2)$.

We can assume that every vertex $a \in A$ has been replaced by an equivalent vertex a^* as defined in Lemma 4 and so, that all vertices of A have a neighbor in X. In such a situation, any

$b \in B$ satisfies $\mathrm{d}(x,b) \leq \mathrm{d}(a,b) \leq \mathrm{d}(x,b)+1$. Similarly, any sum $S'_i = \mathrm{d}(x,b_i)+\mathrm{d}(b_j,b_k)$, where $\{j,k\} = \{1,2,3\} \setminus \{i\}$, satisfies $S'_i \leq S_i \leq S'_i + 1$. Thus for every $i \in \{2,3\}$:

$$\begin{aligned}\delta(a,b_1,b_2,b_3) \leq & (S_1 - S_i)/2 \\ \leq & (S'_1 + 1 - S'_i)/2 \\ \leq & (S'_1 - S'_i)/2 + 1/2\end{aligned}$$

In particular, if $S'_1 \neq \max\{S'_1, S'_2, S'_3\}$, then we have $\delta(a,b_1,b_2,b_3) \leq \frac{1}{2}$ by the choice of $S'_i = \max\{S'_1, S'_2, S'_3\}$. Otherwise, we have $\delta(a,b_1,b_2,b_3) \leq \delta(x,b_1,b_2,b_3) + \frac{1}{2}$ by the choice of $S'_i = \max\{S'_2, S'_3\}$. □

3.3 Separable sets

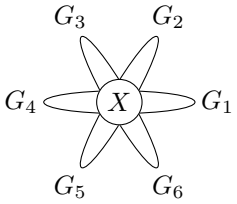


Figure 4: Illustration of separable sets.

Theorem 6. *Let X be a clique-separator of a connected graph G, and let $C_1, \ldots, C_l$ be the connected components of $G \backslash X$. We define $G_i = G[C_i \cup X]$. We have :*

$$\max\{\delta(G_1), \ldots \delta(G_l)\} \leq \delta(G) \leq \max\{\frac{1}{2}, \delta(G_1), \ldots \delta(G_l)\} + 1/2$$

Proof. Note that since a complete subgraph is isometric, then every subgraph G_i is isometric as well. Hence, the lower-bound follows from the four-point definition.

Let us now prove that $\delta(a,b,c,d) \leq \max\{\frac{1}{2}, \delta(G_1), \ldots \delta(G_l)\} + 1/2$ holds for any $a,b,c,d \in V$. We consider a connected component C_i maximizing $|C_i \cap \{a,b,c,d\}|$.

- If $|C_i \cap \{a,b,c,d\}| = 4$ we are done as $\delta(a,b,c,d) \leq \delta(G_i)$.
- If $|C_i \cap \{a,b,c,d\}| = 3$ we can assume that a,b,c,d is an $(a|b_1,b_2,b_3)$ 4-tuple, for the choices of $B = C_i \cup X$ and $A = V \setminus C_i$. By Lemma 5 it follows that $\delta(a,b,c,d) \leq \delta(G_i) + 1/2$.
- Finally, if $|C_i \cap \{a,b,c,d\}| \leq 2$ we can assume that a,b,c,d is an $(a_1,a_2|b_1,b_2)$ 4-tuple, for an appropriate merging of the sets $C_j \cup X$ into two subsets A, B. We know by Corollary 3 that $\delta(a_1,a_2,b_1,b_2) \leq$ 1 in this case.

□

The upper-bound of Theorem 6 is actually tight. It can be shown using the graph in Figure 5, constructed from a cycle C_7 of length 7 to which we add a triangle.

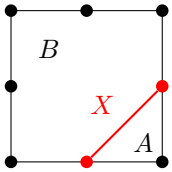


Figure 5: X is an $(A|B)$-clique-separator: we have $\delta(G) = 3/2$, while $\delta(G[B]) = 1$, and $\delta(G[A]) = 0$.

4 Hyperbolicity and clique-decomposition

In Section 3, we explained how a single clique-separator can be used to approximate hyperbolicity. However, this result cannot be used on a whole clique-decomposition as the successive approximations would add up. We thus need to find additional properties to approximate the hyperbolicity of a graph from computations on its atoms in order to prove Theorem 11.

4.1 Relating atoms and 4-tuples with large hyperbolicity

Firstly, we aim to relate to every 4-tuple a, b, c, d with a sufficiently large hyperbolicity, some atom by which all the paths between a, b, c, d go through. Our result involves basic knowledge about *tree-decomposition* (see [8]). We remind the reader that a tree-decomposition of a connected graph G is a tree T whose nodes are labeled by subsets of $V(G)$ (also known as *bags*), and which satisfies the following properties:

- every edge of G is contained into some bag;
- for every vertex $u \in V(G)$, the subgraph induced by the bags containing u is a subtree of T.

Lemma 7. *Let a, b, c, d be a 4-tuple satisfying $\delta(a, b, c, d) \geq \frac{3}{2}$ in a connected graph G. There exists an atom A such that $\forall u \in \{a, b, c, d\} \backslash A$, there is a clique-separator $X_u \subseteq A$ which separates u from $\{a, b, c, d\} \setminus \{u\}$.*

Proof. If the vertices a, b, c, d are contained into a common atom A, then we are done as $\{a, b, c, d\} \setminus A$ is empty. Suppose on the contrary that there is no atom containing the 4-tuple. The authors of [38] proved that there exists a tree-decomposition T_G of G whose bags are exactly the atoms of G. As there is no bag containing a, b, c, d by the hypothesis, there is a unique smallest subtree T of T_G containing a, b, c, d. Note that T is not reduced to a single vertex, and it has at most 4 leaves.

It can be checked that there exists a bag A in T (i.e. an atom of G) such that no connected component of $T \backslash \{A\}$ (and thus no connected component of $G \backslash A$) contains more than two elements among a, b, c, d. Moreover, if a connected component C of $G \setminus A$ contains exactly two elements of a, b, c, d, then $N_G(C) \cap A$ is a clique of G separating two elements of a, b, c, d from the two others, which is impossible by Corollary 3 as we supposed $\delta(a, b, c, d) \geq \frac{3}{2}$.

Thus, for every $u \in \{a, b, c, d\} \backslash A$, the connected component C_u of $G \backslash A$ containing u yields a clique $N_G(C_u) \cap A$, which separates u from $\{a, b, c, d\} \setminus \{u\}$. □

As an illustration, one may notice that the central atom in Figure 6 satisfies the property of Lemma 7 with respect to the 4-tuple v_0, v_1, v_2, v_3. Indeed, none of the four vertices is contained into this atom, but each of them is simplicial and can be separated from the three others by its two neighbours.

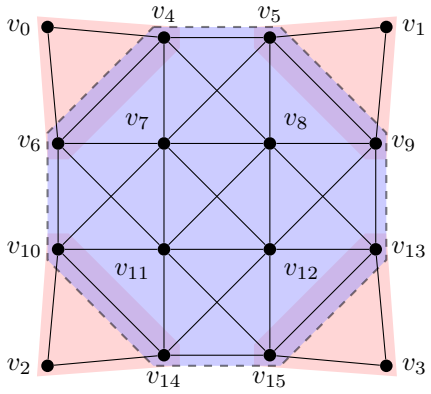


Figure 6: A 2-hyperbolic graph with five atoms: four are 0-hyperbolic, one is 1-hyperbolic.

Moreover, we note that our above result heavily relies on the tree-like structure of the atoms (*e.g* we use a tree-decomposition whose bags are the atoms), and that such a tree-like structure does not exist in general if separators with diameter at least 2 are involved. In fact, our results in this section cannot be extended to decompositions with separators of bounded diameter, as there exist infinitely many graphs that are completely separable with isometric separators of diameter at most 2, and whose hyperbolicity is arbitrarily large [32].

4.2 Substitution method

From Lemma 7 we can associate a specific atom to a 4-tuple of large hyperbolicity. Four applications of Lemma 5 are then sufficient to prove that the hyperbolicity of this 4-tuple and the hyperbolicity of the atom differ by at most 2. The purpose of this section is to prove that this difference is at most 1. To do this, we refine the results of Section 3.2.

Lemma 8. *Let X be an $(A|B)$-clique-separator of a connected graph G. Given an $(a|b_1, b_2, b_3)$ 4-tuple, write:*

$$S_1 = \mathrm{d}(a, b_1) + \mathrm{d}(b_2, b_3),$$
$$S_2 = \mathrm{d}(a, b_2) + \mathrm{d}(b_1, b_3),$$
$$S_3 = \mathrm{d}(a, b_3) + \mathrm{d}(b_1, b_2).$$

Assume w.l.o.g. that $S_1 \geq S_2 \geq S_3$, and let $x_2 \in X$ be such that $\mathrm{d}(a, b_2) = \mathrm{d}(a, x_2) + \mathrm{d}(x_2, b_2) = \mathrm{d}(a, X) + \mathrm{d}(x_2, b_2)$. If $\delta(a, b_1, b_2, b_3) > \delta(x_2, b_1, b_2, b_3)$, then we have

- $S_1 > S_2 = S_3$.
- $\mathrm{d}(a, b_1) = \mathrm{d}(a, x_2) + \mathrm{d}(x_2, b_1)$.

Proof. Recall that by the substitution of Lemma 4, we can assume w.l.o.g. that for every i we have $\mathrm{d}(a, x_i) = \mathrm{d}(a, X) = 1$, where x_i is a vertex of X located on an ab_i-shortest path.

Now, let $\varepsilon_i = \mathrm{d}(x_2, b_i) - \mathrm{d}(x_i, b_i)$. Observe that $\varepsilon_i \in \{0, 1\}$, and that $\varepsilon_i = 0$ if, and only if, x_2 lies on an ab_i-shortest path. In particular we have $\varepsilon_2 = 0$.

Let us denote by S'_i the sum $\mathrm{d}(x_2, b_i) + \mathrm{d}(b_j, b_k)$, where $\{j, k\} = \{1, 2, 3\} \setminus \{i\}$. We aim to exhibit a relation between S_i and S'_i, which would yield in turn a relation between the values

$\delta(a, b_1, b_2, b_3)$ and $\delta(x_2, b_1, b_2, b_3)$. First, we have $\mathrm{d}(a, b_i) = \mathrm{d}(x_2, b_i) + 1 - \varepsilon_i$ and so,

$$\begin{aligned} S_i =& \mathrm{d}(a, b_i) + \mathrm{d}(b_j, b_k) \\ =& \mathrm{d}(x_2, b_i) + \mathrm{d}(b_j, b_k) + 1 - \varepsilon_i \\ =& S'_i + 1 - \varepsilon_i \end{aligned}$$

Since we assume here that $S_1 \geq S_2 \geq S_3$ and $\delta(a, b_1, b_2, b_3) > 0$, we have $S'_1 \geq S_1 - 1 \geq \max\{S_2, S_3\} \geq \max\{S'_2, S'_3\}$. More precisely:

- If $S'_2 \geq S'_3$, then $\delta(a, b_1, b_2, b_3) = \delta(x_2, b_1, b_2, b_3) - \varepsilon_1/2 \leq \delta(x_2, b_1, b_2, b_3)$.
- If $S'_3 > S'_2$, then $\varepsilon_3 = 1$ because $S_2 \geq S_3$, which implies $S_2 = S_3$. This, in turn, implies that $\delta(x_2, b_1, b_2, b_3) = (S'_1 - S'_3)/2 = (S_1 - 1 + \varepsilon_1 - S_3)/2 = (S_1 - S_2)/2 - (1 - \varepsilon_1)/2 = \delta(a, b_1, b_2, b_3) - (1 - \varepsilon_1)/2 \leq \delta(a, b_1, b_2, b_3)$.

  In such a case, $\delta(x_2, b_1, b_2, b_3) < \delta(a, b_1, b_2, b_3)$ if, and only if, we have $\varepsilon_1 = 0$, that is x_2 lies on an ab_1-shortest path.

□

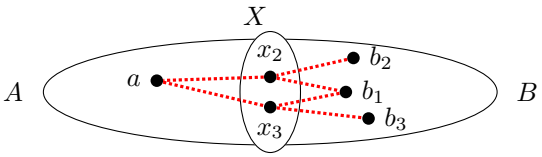


Figure 7: An illustration of the metric property. The dashed lines represent shortest paths.

The metric property of Lemma 8 is illustrated with Figure 7. We use it to extend Lemma 5 as follows.

Lemma 9. *Let a, b, c, d be a 4-tuple of a connected graph G, and two sets $X_a, X_d \subseteq V$ satisfying:*

- *X_a is a $(a|b, c, d)$-clique-separator;*
- *X_d is a $(d|a, b, c)$-clique-separator;*
- *$X_a \backslash X_d$ lies in the same connected component of $G \setminus X_d$ as a, b, c.*

Then there exist $x_a \in X_a$ and $x_d \in X_d$ such that

$$\delta(a, b, c, d) \leq \delta(x_a, b, c, x_d) + 1/2.$$

Proof. Let u_1, u_2, u_3 be such that $\{u_1, u_2, u_3\} = \{b, c, d\}$, $T_1 \geq T_2 \geq T_3$, where $T_i = \mathrm{d}(a, u_i) + \mathrm{d}(u_j, u_k)$ and $\{j, k\} = \{1, 2, 3\} \setminus \{i\}$. We distinguish two cases:

- First, assume that there is $x_a \in X_a$ satisfying $\delta(a, b, c, d) \leq \delta(x_a, b, c, d)$. Then we are done, as Lemma 5 applied to the $(d|x_a, b, c)$-clique separator X_d yields a vertex $x_d \in X_d$ satisfying $\delta(x_a, b, c, d) \leq \delta(x_a, b, c, x_d) + \frac{1}{2}$, hence $\delta(a, b, c, d) \leq \delta(x_a, b, c, x_d) + \frac{1}{2}$.
- Second, assume that no vertex of X_a satisfies the above property. In particular, let $x_a \in X_a$ be such that $\mathrm{d}(a, x_a) = \mathrm{d}(a, X_a)$, and x_a lies on an au_2-shortest path. Note that $\delta(a, b, c, d) \leq \delta(x_a, b, c, d) + \frac{1}{2}$ by Lemma 5 and so, $\delta(a, b, c, d) = \delta(x_a, b, c, d) + \frac{1}{2}$. Moreover,

we deduce from Lemma 8 that $T_1 > T_2 = T_3$, and x_a also lies on an au_1-shortest-path. Let $T'_i = \mathrm{d}(x_a, u_i) + \mathrm{d}(u_j, u_k)$. It thus follows that we have:

$$T'_1 = T_1 - \mathrm{d}(a, X_a), T'_2 = T_2 - \mathrm{d}(a, X_a) \text{ and } T'_3 = T_3 - \mathrm{d}(a, X_a) + 1.$$

So, we have $T'_1 \geq T'_3 > T'_2$. By the contrapositive of Lemma 8 applied to the 4-tuple $(d|x_a, b, c)$, and noticing that the two least sums amongst T'_1, T'_2, T'_3 are different, there exists a vertex $x_d \in X_d$ satisfying $\delta(x_a, b, c, d) \leq \delta(x_a, b, c, x_d)$. □

Corollary 10. *Let a, b, c, d be a 4-tuple of a connected graph G satisfying $\delta(a, b, c, d) \geq 3/2$. There exists an atom A of G such that $\delta(a, b, c, d) \leq \delta(G[A]) + 1$*

Proof. The atom A is obtained by applying Lemma 7. For every vertex $u \in \{a, b, c, d\} \setminus A$, let $X_u \subseteq A$ be a clique-separator disconnecting u from $\{a, b, c, d\} \setminus u$. We claim that $A \setminus X_u$ lies in the same connected component of $G \setminus X_u$ as $\{a, b, c, d\} \setminus u$, because otherwise X_u would separate two elements of $\{a, b, c, d\}$ from the two others, which by Corollary 3 implies that $\delta(a, b, c, d) \leq 1$, a contradiction.

One can then apply Lemmas 9 and 5 to find four vertices $a', b', c', d' \in A$ such that $\delta(a, b, c, d) \leq \delta(a', b', c', d') + 1$. Note that the sets $\{a, b, c, d\}$ and $\{a', b', c', d'\}$ are not necessarily disjoint. □

4.3 Additive approximation for hyperbolicity

Theorem 11. *Let $A_1, \ldots, A_l$ be the atoms of a connected graph G. Then:*

$$\max_i \delta(G[A_i]) \leq \delta(G) \leq \max_i \delta(G[A_i]) + 1$$

Proof. As for Theorem 6, the lower-bound of Theorem 11 follows from the fact that the subgraphs $G_i = G[A_i]$ are isometric subgraphs of G. The upper-bound trivially holds when $\delta(G) \leq 1$. We can thus suppose that $\delta(G) \geq 3/2$ and so, that there exist four vertices a, b, c, d such that $\delta(a, b, c, d) = \delta(G) \geq 3/2$. Corollary 10 then yields an atom A such that $\delta(G) \leq \delta(G[A]) + 1$, which proves the second part of our claim. □

Note that the upper-bound is reached by the graph of Figure 6, and by the 1-hyperbolic chordal graph from Figure 1 whose atoms have hyperbolicity 0.

5 Substitution method for an exact computation

As shown with Theorem 11, the maximum hyperbolicity taken over all the atoms may be slightly lesser than the hyperbolicity of the graph. In the two previous sections, we characterized under which situations both values might differ. We now propose to extend the usual clique-decomposition, by replacing the atoms by some *substitute graphs*, so that both values are always equal. The main drawback of our method is that it requires the hyperbolicity to be at least 1.

Outline of the method. We recall that a *simplicial* vertex is a vertex whose neighborhood induces a complete subgraph. In the spirit of Lemma 4, we add simplicial vertices to the atoms, in order to mimic the maximum $(a|b_1, b_2, b_3)$ 4-tuples that may result from a disconnection. We first introduce our method of substitution in the simpler context of a *single* disconnection. Then we focus on technical details related to the implementation.

5.1 Substitute graphs

5.1.1 Basic step: single disconnection

Let us consider the following algorithm. Given a connected graph G, let A, B be a covering of $V(G)$ satisfying $X = A \cap B$ is an $(A|B)$-clique separator of G.

- Starting from $G_1 = G[A]$, for every vertex b of $B \setminus X$, we consider the subset X_b of vertices which are the closest to b in X. For every such a subset, we add in G_1 a simplicial vertex s_{X_b} whose neighborhood is X_b.
- Starting from $G_2 = G[B]$, for every vertex a of $A \setminus X$, we consider the subset X_a of vertices which are the closest to a in X . For every such a subset, we add in G_2 a simplicial vertex s_{X_a} whose neighborhood is X_a.

The two resulting graphs are denoted G_1^*, G_2^*, and they are called the *substitute graphs* of G_1, G_2.

Lemma 12. *Let G be a connected graph satisfying $\delta(G) \geq 1$, and A, B be a covering of the vertices satisfying $A \cap B = X$, where X denotes an $(A|B)$-clique separator of G. We define $G_1 = G[A]$, $G_2 = G[B]$. We have:*

$$\delta(G) = \max\{1, \delta(G_1^*), \delta(G_2^*)\}.$$

Proof. First, we prove the upper-bound. By construction, the subgraph G_i is an isometric subgraph of G_i^*, for every $i \in \{1, 2\}$. It thus follows that $\delta(G_i^*) \geq \delta(G_i)$. Let us now consider the 4-tuples of G which are separated by the disconnection. We recall that by Corollary 3, the hyperbolicity of any $(a_1, a_2|b_1, b_2)$ 4-tuple of G is bounded by 1. In addition, for every $a \in A \setminus X$, and for every $b_1, b_2, b_3 \in B$, there exists by construction a simplicial vertex a^* of $V(G_2^*) \setminus B$ that is adjacent to $\{x \in X : \mathrm{d}_G(a, x) = \mathrm{d}_G(a, X)\}$; hence, $\delta(a, b_1, b_2, b_3) = \delta(a^*, b_1, b_2, b_3) \leq \delta(G_2^*)$ by Lemma 4. The proof is symmetric for $b \in B \setminus X$ and $a_1, a_2, a_3 \in A$. As a result, $\delta(G) \leq \max\{1, \delta(G_1^*), \delta(G_2^*)\}$.

To prove the lower-bound, let us consider w.l.o.g. an arbitrary 4-tuple of G_2^*. If such a 4-tuple does not contain a simplicial vertex of $V(G_2^*) \setminus B$, then we are done as it exists in the isometric subgraph G_2 of G. If it contains exactly one simplicial vertex $a^* \in V(G_2^*) \setminus B$, then by construction we can replace a^* with a vertex $a \in A \setminus X$, satisfying $N(a^*) = \{x \in X : \mathrm{d}_G(a, x) = \mathrm{d}_G(a, X)\}$; this operation does not modify the hyperbolicity of the 4-tuple by Lemma 4. Finally, if it contains at least two simplicial vertices of $V(G_2^*) \setminus B$, then it is an $(a_1, a_2|b_1, b_2)$ 4-tuple of G_2^* and so, the hyperbolicity is bounded by 1. □

We emphasize that some simple rules can be applied, in order to add fewer vertices into the substitute graphs. In particular, we can remove every pendant vertex, as the hyperbolicity of a connected graph is the maximum hyperbolicity taken over all its biconnected components (this well-known result is a particular case of our proofs in Section 3, as a cut-point is a clique-separator containing a single vertex).

5.1.2 Extension to the atoms

The substitution operation can be naturally extended to the whole clique-decomposition, by mimicking each step of it and applying the basic substitution operation that we describe above at each of these steps. We formalize it by first introducing the following definition of an atom tree.

Definition 13 ([6, 7, 34])**.** *Let G be a connected graph. An* atom tree *of G is a labeled binary rooted tree T, satisfying the following recursive definition:*

- *if G is prime w.r.t. clique-decomposition, then T is reduced to a node labeled with V;*
- *otherwise, the root of T is labeled with a clique-minimal separator X, and there exists a connected component C of $G \setminus X$ satisfying:*
  - $N_G(C) \setminus C = X$;
  - *the left child of the root is labeled with $A = C \cup X$, which does not contain any clique-minimal separator;*
  - *and the right subtree of the root is an atom tree of $G \setminus C$.*

In order to prevent any confusion, the reader has to notice that an atom tree is *not* a tree-decomposition (as defined in Section 4.1). In fact, an atom tree can be seen as the trace of some execution of the algorithm of [34, 45] to compute the clique-decomposition. Indeed, it is proved in [34] that in an atom tree, the leaves are labeled with the atoms of the graph, and each atom labels exactly one leaf of the tree. Given a *fixed* atom tree, this yields a natural ordering of the atoms by increasing depth. We follow this ordering so that we can apply our basic step of substitution sequentially, using at each step $i \in \{1, \ldots, l\}$ the same clique-minimal separator X_i as in the atom tree to separate the connected component containing A_i from the remaining vertices to explore.

The whole process is illustrated with Figure 8. The numbers reported in Tab. 8i illustrate the interest of our pre-processing method for the computation of the hyperbolicity. Indeed, the graph G of Fig. 8a has 28 nodes and so 20 475 4-tuples, while the sum of the numbers of 4-tuples in the graphs G_i^* (Figs. 8c–8h) is 1 800. We thus significantly reduce the search space. Moreover, a simple cutting rule allows us to reduce the number of 4-tuples to consider to 1 575. To do so, we first order the graphs G_i^* by decreasing diameters, then iteratively compute the hyperbolicity of these graphs in this order, and stop exploration as soon as the diameter of a graph G_j^* is less than or equal to twice the largest value of δ computed so far.

5.2 Implementation and complexity analysis

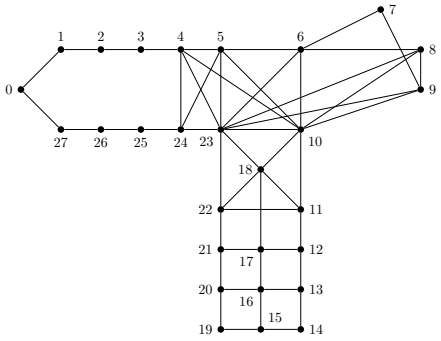
5.2.1 Precomputation step and updates

We first focus on some computational tasks that have to be repeatedly executed at each step of our substitution method. In this section, we provide a high-level description for an implementation of these tasks.

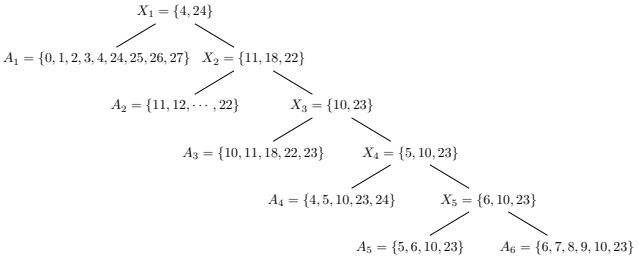
Computation of the distances We heavily rely on the distance-matrix of the graph in our implementation, to compute the distances between some vertex (possibly newly added by our construction) and the vertices of some clique-minimal separator. Since we apply our basic substitution step as many times as there are atoms (*e.g.* Section 5.1.2), it is essential that we can update the distance-matrix of the graph quickly, so that such distances remain computable in constant-time.

Lemma 14. *Let G be a connected graph. We can embed in quadratic-time the distance-matrix of G into a data-structure, supporting:*

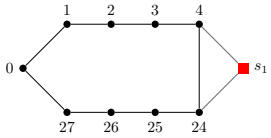
- *constant-time access to the distance between a vertex and a vertex contained into a clique-minimal separator;*
- *constant-time updates when a simplicial vertex is added by our substitution method.*



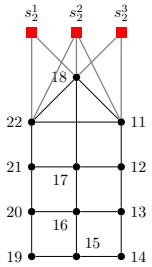
(a) G



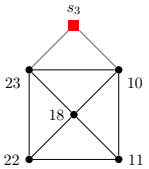
(b) Atom tree of G



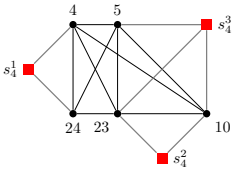
(c) G_1^* built from A_1



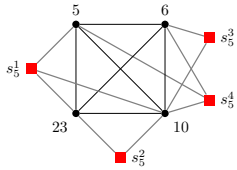
(d) G_2^* built from A_2



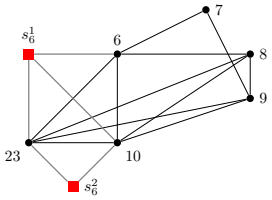
(e) G_3^* built from A_3



(f) G_4^* built from A_4



(g) G_5^* built from A_5



(h) G_6^* built from A_6

	n	m	D	δ	4-tuples
G	28	50	9	2	20 475
G_1^*	10	11	5	2	210
G_2^*	15	25	5	2	1 365
G_3^*	6	10	2	1	15
G_4^*	8	16	3	1/2	70
G_5^*	8	16	2	1/2	70
G_6^*	8	16	3	1	70

(i) Main characteristics

Figure 8: A connected graph G (Fig. 8a), an atom tree of the graph (Fig. 8b), the substitute of the atoms of G (Figs. 8c–8h), and the characteristics of these graphs (Tab. 8i).

Proof. The gist of such a structure is precisely Lemma 4. Let X be an $(A|B)$-clique separator of G, and s be a simplicial vertex added in the substitute graph of $G[B]$. Let $a \in A \setminus X$ satisfying $N(s) = \{x \in X : \mathrm{d}_G(a, x) = \mathrm{d}_G(a, X)\}$. Then we have for every $b \in B$, $\mathrm{d}(b, s) = \mathrm{d}_G(a, b) + (1 - \mathrm{d}_G(a, X))$.

It thus follows that once the substitution operation has been completed, we only need to relate every new simplicial vertex added by our construction to some vertex of $V(G)$ with an offset, so that we can compute the distances in the substitute graphs. The offset can be computed in constant-time by picking a neighbor of the simplicial vertex s, and accessing in constant-time to the distance between this neighbor and a vertex substituted by s. □

Note that the data-structure of Lemma 14 does not support the computation of distances between two vertices added by our construction. We can safely ignore this drawback, as we never require to compute such distances in our method.

Determining the vertex sets of the substitute graphs A similar problem is to compute efficiently, at step i of our algorithm, the connected component containing the atom A_i when we disconnect our current (dynamical) graph using the clique-minimal separator X_i. Note that, due to the (simplicial) vertices and edges possibly added at previous steps, the classical algorithm for this problem cannot be ensured to run in $O(m)$-time. Instead, we propose a method to update this component dynamically, starting from A_i. To do that, let us consider the following partition of the vertex set $V(G)$:

$$V_1 = A_1 \setminus X_1, \ldots, V_i = A_i \setminus X_i, \ldots, V_{l-1} = A_{l-1} \setminus X_{l-1}, V_l = A_l.$$

Lemma 15. *Let G be a connected graph, T be an atom tree of G, and $A_1, \ldots, A_l$ be its atoms. We denote by X_i the clique-minimal separator of G labeling the parent node of A_i in T, and by V_i^* the vertex set of the substitute graph of the atom A_i.*

Then, for every simplicial vertex s_i added at step i of our substitution method, we can decide the vertex set V_j^*, $j > i$, to which we have to add s_i in subsequent steps, in $O(n|X_i|)$-time.

Proof. It can be checked that at step i, the simplicial vertices added into $V_i^* \setminus A_i$ are those of previous steps whose neighborhood partly lies into $V_i = A_i \setminus X_i$. So, at every step of our substitution method, if a simplicial vertex is added by our construction, we consider the minimum index j satisfying V_j contains a neighbour of the vertex, and we dynamically update the vertex set V_j^* containing A_j by adding this new vertex into it. Since only $O(n)$ vertices are added at step i, and that their neighborhood is contained into X_i, the $O(n|X_i|)$-time complexity follows. □

5.2.2 Applying simplification rules

We now focus on some algorithmic aspects of the substitution operation (Section 5.1).

In this section, we will assume that the distance-matrix of the graph is given. Given the set $B \setminus X$, it is straightforward to compute in $O(n|X|)$-time all of the subsets $X_b = \{x \in X : \mathrm{d}_G(b,x) = \mathrm{d}_G(b,X)\}$, for every $b \in B \setminus X$. A naive implementation would then consist in adding a simplicial vertex for every b and by doing so, we would lose all the benefit of the separation in terms of size of the graphs. However, we defined rules in order to avoid this worst-case in some situations, which potentially decrease the number of simplicial vertices to add in the substitute graphs. The goal of this section is to give hints on an efficient way to implement these rules.

Partition refinement techniques Indeed, it may happen that $X_b = X_{b'}$ for some pair $b, b' \in B$, and in such a case we wish to add only one simplicial vertex in the substitute graph G_1^*.

To do that efficiently, we will use the well-known *partition refinement* techniques (*e.g.* see [27, 39]). Given a partition $\mathcal{P}$ of a set V, and a subset $S \subseteq V$ called the *pivot*, the partition refinement of $\mathcal{P}$ w.r.t. S consists in replacing every group V_i of $\mathcal{P}$ by the non-empty groups amongst $V_i \cap S$ and $V_i \cap \bar{S}$. This can be achieved in $O(|S|)$-time, up to the precomputation of an appropriate data-structure in linear $O(|V|)$-time.

We deduce from this standard technique the following result:

Lemma 16. *Let G be a connected graph given by its distance-matrix, and X be a subset of $V(G)$. We define the relation $\equiv_X$ over the set $V(G) \setminus X$ as $u \equiv_X v$ if, and only if,*

$$\{x \in X : \mathrm{d}_G(u,x) = \mathrm{d}_G(u,X)\} = \{x \in X : \mathrm{d}_G(v,x) = \mathrm{d}_G(v,X)\}.$$

The equivalence classes of the relation $\equiv_X$ can be computed in $O(n|X|)$-time.

Proof. Consider the following graph $G_X = (V, \{\{u,x\} : u \notin X, x \in X \text{ and } \mathrm{d}_G(u,x) = \mathrm{d}_G(u,X)\})$. Starting from $\mathcal{P} = \{V \setminus X\}$, we refine the partition with the sets $N_{G_X}(x)$, for every $x \in X$. The total cost is $O(\sum_{x \in X} |N_{G_X}(x)|) = O(n|X|)$. □

5.2.3 Complexity analysis

Finally, to determine the time complexity of our substitution method, we will use the following result:

Lemma 17 ([6]). *Let G be a connected graph, and $A_1, \ldots, A_l$ be its atoms. Then $\sum_i |A_i| \leq n + m$.*

Corollary 18. *The substitute of the atoms of a connected graph G can be computed in $O(nm)$-time.*

Proof. Let T be an atom tree of G, and $A_1, \ldots, A_l$ be the atoms of the graph. For every i, let X_i be the clique-minimal separator of G labeling the father node of leaf A_i in T. By Definition 13, $X_i \subseteq A_i$.

We first precompute the distance-matrix of G in $O(nm)$-time, then we embed it in quadratic-time into the data-structure of Lemma 14. Also, for every i, we initialize the vertex set V_i^* with the atom A_i. The set V_i^* is the vertex set of the substitute graph containing A_i at step i of our substitution method. We then apply each step of our substitution method sequentially.

We can easily check that at step i, there are at most $O(n)$ vertices to consider. By Lemma 16, this allows us to compute the simplicial vertices to add at this step in $O(n|X_i|)$-time. By Lemma 15, we can then update the sets V_j^*, $j > i$, with the same time complexity. Since at most $O(n)$ vertices are added at this step, we finally update the distances in $O(n)$-time by Lemma 14.

Consequently, our modified clique-decomposition can be computed in $O(n \sum_i |X_i|)$-time, that is in $O(n \sum_i |A_i|) = O(nm)$-time by Lemma 17. □

6 Hyperbolicity of outerplanar graphs

Last, we deal in this section with an algorithmic application of the substitution method of Section 5. We use it here to provide a *linear-time* algorithm for computing the hyperbolicity of a large class of graphs.

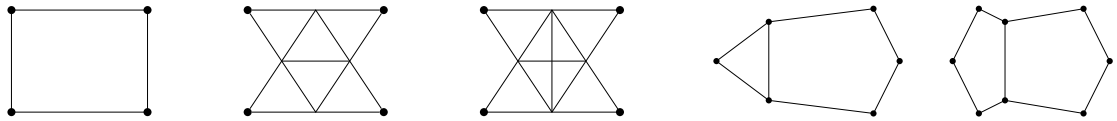
More precisely, we here constrain our study to outerplanar graphs. A *planar graph* is a graph drawable in the Euclidean plane so that edges may only intersect at their endpoints, and an *outerplanar graph* is a graph which keeps the planarity property whenever one adds a universal vertex to it. Equivalently, a graph is outerplanar if it is drawable in the Euclidean plane so that edges may only intersect at their endpoints, and all the vertices lie on a common face which is called the *outerface*. Such a drawing is furthermore called an outerplanar embedding, and it can be computed in linear-time [43]. Outerplanar graphs are stable by the minor operation, and a graph is outerplanar if, and only if, it is K_4-minor-free and $K_{2,3}$-minor-free. The reader may refer to [42] for other characterizations of this class of graphs.

An interesting subclass of outerplanar graphs is the class of cycles. Actually, for an outerplanar graph, the induced cycles are exactly its atoms. Hyperbolicity of cycles is completely characterized, and it can be computed using the length of the cycle:

Lemma 19 ([14, 46]). *Cycles of order $4p + \varepsilon \geq 3$, with $p \geq 0$ and $\varepsilon \in \{0, 1, 2, 3\}$, are $(p - 1/2)$-hyperbolic when $\varepsilon = 1$, and p-hyperbolic otherwise.*

In the sequel, we will use Lemma 19 in order to prove the main result of this section (*e.g.* Theorem 26). Our proofs heavily rely on the notion of *weak dual* [2].

Definition 20. *Let G be a connected outerplanar graph. The weak dual of G is a tree T_G, whose nodes are labeled by the atoms of G, and such that every two nodes adjacent in T_G satisfy the atoms labeling them either share a single edge or a cut-point.*



(a) C_4 (b) H_1 (c) H_2 (d) H_3 (e) H_4

Figure 9: Characterization of 5-chordal $\frac{1}{2}$-hyperbolic graphs, in terms of forbidden isometric subgraphs.

Note that a weak dual is nothing else than a tree-decomposition whose bags are the atoms of an outerplanar graph. If G is biconnected, then starting from an outerplanar embedding of the graph, we construct it by removing from the dual of the graph the universal vertex corresponding to the outerface (see Figure 10 for an example).

6.1 Outerplanar graphs with small hyperbolicity

As observed in Section 5, our substitution method for an exact computation of hyperbolicity requires the hyperbolicity of the graph to be at least 1. To overcome this drawback, we first characterize in this section outerplanar graphs that are $\frac{1}{2}$-hyperbolic. Note that we only consider biconnected graphs, as the hyperbolicity of a graph is the maximum hyperbolicity taken over all its biconnected components, and the biconnected components of a graph are computable in linear-time [44].

Proposition 21. *A biconnected outerplanar graph is $\frac{1}{2}$-hyperbolic if, and only if, either it is isomorphic to C_5, or it is chordal and it does not contain the graph of Figure 9b as a subgraph. Furthermore, these conditions can be checked in linear-time.*

Proof. Let G be an outerplanar biconnected graph. We suppose G to be $\frac{1}{2}$-hyperbolic. By Lemma 19, the graph C_5 is $\frac{1}{2}$-hyperbolic. So, we will assume for the proof that G is not isomorphic to C_5. We then remind that every induced cycle of G is isometric, as the induced cycles of G are exactly its atoms. As a result, we have by Lemma 19 that G only has induced cycles of length 3 or 5. Moreover, by [46], a 5-chordal graph is $\frac{1}{2}$-hyperbolic if, and only if, it does not contain any graph of Figure 9 as an isometric subgraph[1].

By the hypothesis, G is C_4-free and so, it does not contain the graph of Figure 9a as an isometric subgraph. Moreover, we claim that G is C_5-free, as otherwise it would contain the graph of Figure 9d, or the graph of Figure 9e, as an isometric subgraph. Thus G has to be chordal. Also, we check that G cannot contain the graph of Figure 9c, as it is not outerplanar and being outerplanar is a hereditary property. Consequently, G is $\frac{1}{2}$-hyperbolic if, and only if, it is chordal and it does not contain the graph of Figure 9b as an isometric subgraph.

Finally, recall that a graph is outerplanar if, and only if, it is K_4-minor-free and $K_{2,3}$-minor-free. Let H_1 be the graph of Figure 9b. We claim that every induced subgraph of G that is isomorphic to H_1 is isometric, as otherwise it would yield a $K_{2,3}$-minor for G. In the same way, every subgraph of G that is isomorphic to H_1 is an induced subgraph of G, as otherwise it would yield a K_4-minor for G. So, G is $\frac{1}{2}$-hyperbolic if, and only if, it is chordal and it does not contain H_1 as a subgraph. Being chordal can be checked in linear-time [40], and a chordal outerplanar graph contains H_1 as a subgraph if, and only if, there are two adjacent vertices of degree 3 in its weak dual (see Figure 10). □

[1] The characterization of [46] is composed of six forbidden isometric subgraphs, but the sixth one is actually 6-chordal.

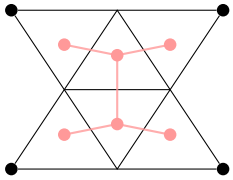


Figure 10: The forbidden subgraph of Figure 9b, and its characterization in the weak dual.

6.2 Substitute graphs of cycles

As we constrain ourselves to outerplanar graphs, recall that the atoms are exactly the induced cycles of the graph. Clearly, a clique-separator contained into a cycle is either a cut-point (in which case our substitution method never adds a simplicial vertex), or it is an edge-separator (in which case our substitution method might only add a single vertex, which has to be adjacent to both endpoints of the edge). Substitute graphs of cycles thus fall into the following subclass of outerplanar graphs:

Definition 22. *A sunshine graph is a biconnected outerplanar graph, containing a dominating induced cycle C satisfying every vertex not in C is a simplicial vertex of degree* 2.

Cycles are a subclass of such graphs, and two other examples of sunshine graphs are given in Figure 11. Moreover, we remark that an edge in a dominating cycle may be the neighborhood of at most one simplicial vertex out of the cycle, as otherwise it would yield a $K_{2,3}$-minor for the graph. It can be noted in addition that except for the particular case of a diamond, there exists a unique dominating cycle in a sunshine graph, and every induced cycle which is not dominating is a triangle. Thus, if G is a sunshine graph and C is a dominating cycle of G, then we have by Theorem 11 that $\delta(C) \leq \delta(G) \leq \delta(C) + 1$. This difference can actually be decreased by $\frac{1}{2}$ as follows.

Lemma 23. *Let G be a sunshine graph, and C be a dominating cycle of G. Then we have:*

$$\delta(C) \leq \delta(G) \leq \delta(C) + \frac{1}{2}$$

Proof. By Theorem 11, we have $\delta(C) \leq \delta(G) \leq \delta(C) + 1$. So, our aim is to prove that no 4-tuple of G has a hyperbolicity greater than $\delta(C) + \frac{1}{2}$. By contradiction, let a, b, c, d be such that $\delta(a, b, c, d) = \delta(C) + 1$.

We arbitrarily orient the cycle C. For every $u \in \{a, b, c, d\} \setminus C$, we denote by $e_u = \{x_u, y_u\}$ the edge of C induced by its neighbours, where x_u denotes the head of the edge w.r.t. the orientation. Observe that for every $u, v \in \{a, b, c, d\} \setminus C$, we have $\mathrm{d}(u, v) = 2 + \min\{\mathrm{d}(x_u, y_v), \mathrm{d}(x_v, y_u)\} = 1 + \mathrm{d}(x_u, x_v) = 1 + \mathrm{d}(y_u, y_v)$.

We then claim that there is exactly one vertex amongst a, b, c, d which belongs to the cycle C. Indeed, not all of a, b, c, d belong to C as the 4-tuple has a hyperbolicity greater than $\delta(C)$. Furthermore, at least three of them are not in C, as otherwise we would have $\delta(a, b, c, d) \leq \delta(C) + \frac{1}{2}$ by Lemmas 5 and 9. On the other hand, if none of them is in C, then we can check that $\delta(a, b, c, d) = \delta(x_a, x_b, x_c, x_d) \leq \delta(C)$, a contradiction. So, the claim is proved.

Finally, assume w.l.o.g. that $a \in C$. In such a case, we can check that $\delta(a, b, c, d) \leq \delta(a, x_b, x_c, x_d) + \frac{1}{2} \leq \delta(C) + \frac{1}{2}$. A contradiction. □

Once the upper-bound for hyperbolicity is refined as above, the hyperbolicity of a sunshine graph can be computed in *linear-time*, using the following characterization.

(a) n odd (b) $n = 4p + 2$

Figure 11: Substitute graphs of the atoms of an outerplanar graph.

Lemma 24. *Let G be a sunshine graph, and C be a dominating cycle for G of length $4p+\varepsilon \geq 3$, with $p \geq 0$ and $\varepsilon \in \{0,1,2,3\}$. Assuming $G \setminus C$ is nonempty we have:*

- *if ε is odd, then $\delta(G) = \delta(C) + \frac{1}{2}$;*
- *if $\varepsilon = 2$, then $\delta(G) = \delta(C) + \frac{1}{2}$ if, and only if, there exist two simplicial vertices out of C which are diametrically opposed;*
- *finally, if $\varepsilon = 0$, then $\delta(G) = \delta(C)$.*

Proof. Recall that by the previous Lemma 24, we have $\delta(G) \leq \delta(C) + \frac{1}{2}$. Thus we only focus on finding 4-tuples u, v, x, y of hyperbolicity (at least) this value, and we choose one, if any, maximizing $|C \cap \{u, x, v, y\}|$. In the sequel, write $S_1 = \mathrm{d}(u, v) + \mathrm{d}(x, y)$, $S_2 = \mathrm{d}(u, x) + \mathrm{d}(v, y)$ and $S_3 = \mathrm{d}(u, y) + \mathrm{d}(v, x)$. We will assume in addition that $S_1 \geq S_2 \geq S_3$.

Case ε odd Equivalently, we have $\varepsilon \in \{1, 3\}$. In such a case, we have $\delta(C) = p + \frac{\min\{0,\varepsilon-2\}}{2}$ by Lemma 19. Figure 11a exhibits a 4-tuple u, v, x, y satisfying:

$$S_1 = (2p + \frac{\varepsilon+1}{2}) + (2p + \min\{1, \varepsilon - 1\}) = 4p + \frac{\varepsilon+1}{2} + \min\{1, \varepsilon - 1\}$$
$$S_2 = (p + 1) + (p + \frac{\varepsilon-1}{2}) = 2p + \frac{\varepsilon+1}{2}$$
$$S_3 = S_2$$

Hence, this 4-tuple has hyperbolicity $p + \frac{\min\{1,\varepsilon-1\}}{2} = \delta(C) + \frac{1}{2}$.

Case $\varepsilon = 2$ In such a case, we have $\delta(C) = p$ by Lemma 19. We assume w.l.o.g. that $u \notin C$, and we claim that it implies $v \notin C$. Indeed, by the metric property of Lemma 8, and noticing that $S_1 \geq S_2 \geq S_3$, the vertex v has to be at equal distance l of both neighbours of u, as otherwise u could be replaced with one of its two neighbours, contradicting the maximality of $|C \cap \{u, x, v, y\}|$. Hence $v \notin C$ is impossible, as it would yield the length of C is $2l + 1 = 4p + 2$. It thus follows that $v \notin C$, and the length of C is in fact $2(l - 1) + 2 = 2l$, yielding $l = 2p + 1$.

Conversely, assume that there exist two simplicial vertices u, v that are diametrically opposed in G. We choose the 4-tuple u, x, v, y as in Figure 11b, and it satisfies:

$$S_1 = (2p + 2) + (2p + 1) = 4p + 3$$
$$S_2 = 2(p + 1) = 2p + 2$$
$$S_3 = S_2$$

So, we have $\delta(u, v, x, y) = p + \frac{1}{2} = \delta(C) + \frac{1}{2}$.

Case $\varepsilon = 0$ Another application of Lemma 19 yields $\delta(C) = p$. Assuming $u \notin C$, we deduce as for the previous case that $v \notin C$, and v is at equal distance $l = 2p$ from both neighbours of u. Thus, C is partitioned by the neighborhoods of u and v, in the same way as in Figure 11b, into two paths of length $l - 1 = 2p - 1$. Furthermore, since the diameter of C is $2p$, those paths are geodesics of the cycle. By the proof of Lemma 23, at least one vertex $z \in \{x, y\}$ amongst the 4-tuple has to be in C so that $\delta(u, v, x, y) > \delta(C)$. As a result, by picking the geodesic containing z we get $\mathrm{d}(u, z) + \mathrm{d}(v, z) = 2 + (l-1) = 2p+1$. So, we have $\min\{\mathrm{d}(u, z), \mathrm{d}(v, z)\} \leq p$, contradicting the fact that $\delta(u, v, x, y) > \delta(C)$, as we have by [41] that $\delta(u, v, x, y) \leq \min\{\mathrm{d}(u, z), \mathrm{d}(v, z)\}$. To sum up, we always have $\delta(G) = \delta(C)$ in such a case. □

6.3 Applying the substitution method in linear-time

Recall that our substitution method consists in constructing a substitute graph for every atom of the graph, then computing the hyperbolicity of every substitute graph, and finally taking the maximum over these hyperbolicity values and 1. In the case of an outerplanar graph, we provided in the previous section a characterization yielding a linear-time computation of the hyperbolicity of the substitute graphs of the atoms (*e.g.* Lemma 24). What now remains to prove is that the substitute graphs can also be computed in linear-time.

Lemma 25. *Let G be an outerplanar biconnected graph. The substitute graphs of the atoms of G can be computed in linear-time.*

Proof. First, we construct in linear-time an outerplanar embedding for G, then we construct from it the weak dual T_G of G. Let $C_1, \ldots, C_l$ be the atoms of G. We root T_G on an atom C_1, which is an induced cycle. Note that $(T_G; C_1)$ naturally yields an atom tree, as defined in Definition 13: starting from a rooted tree $(T; C)$ initialized with $(T_G; C_1)$, we iterate as long as T is not empty, and at each step we disconnect all the atoms being the leaves of T sequentially. Such an ordering is enough to prove the correctness of the following algorithm.

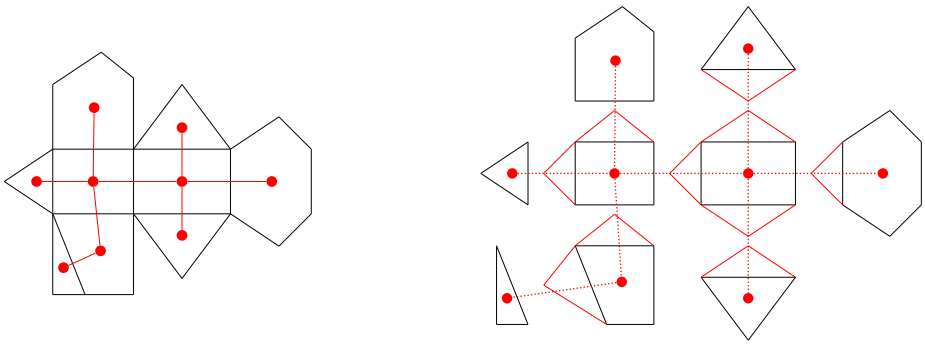


Figure 12: An application of the substitution method to an outerplanar graph.

- For every i, we initialize C_i^* with C_i.
- We start a depth-first search from the root, and for every visited atom C_i, once its subtree has been visited, we consider all its sons C_j, naming $e_{i,j}$ the edge-separator $C_i \cap C_j$. We add in C_i^* a simplicial vertex whose neighborhood is $e_{i,j}$ if, and only if, there is a vertex in C_j^* which is at equal distance to both endpoints of $e_{i,j}$. This is equivalent to have either the length of C_j is odd, or there is a simplicial vertex in $C_j^* \setminus C_j$ whose neighborhood is the edge diametrically opposed to $e_{i,j}$ in C_j.

- Finally, we start a breadth-first search from the root and for every visited atom $C_i \neq C_1$, we consider its parent atom, denoted by C_k, naming $e_{i,k}$ the edge-separator that it shares with it. As before, we add in C_i^* a simplicial vertex whose neighborhood is $e_{i,k}$ if, and only if, either the length of C_k is odd, or there is a simplicial vertex in $C_k^* \setminus C_k$ whose neighborhood is the edge diametrically opposed to $e_{i,k}$ in the atom C_k.

Note that for an atom, the depth-first search here is used to compute the simplicial vertices resulting from the disconnection of its sons, whereas the breadth-first search is used to compute the single vertex resulting from its own disconnection, if any. Using an atom tree as defined above, it can be checked that the resulting $C_1^*, \ldots, C_l^*$ are the substitute graphs of the atoms of G. □

We finally conclude with the following theorem.

Theorem 26. *The hyperbolicity of a connected outerplanar graph is computable in linear-time.*

Proof. We can safely assume G to be biconnected by [44]. By [4, 28], G is 0-hyperbolic if, and only if, G is a clique. If it is not, then by Proposition 21, we can check whether it is $\frac{1}{2}$-hyperbolic in linear-time.

From now on, assume $\delta(G) \geq 1$. By Lemma 25, we can compute the substitute graphs of the atoms of G in linear-time. We can thus conclude by Lemma 12 (*i.e.* the correctness of our substitution method), as these substitute graphs are sunshine graphs and their hyperbolicity is linear-time computable by Lemma 24. □

7 Experimental evaluation

We report in this section on experiments performed with our substitution methodology on the graphs of five collaboration networks. This way, we aim to evaluate the computation time of the substitutes on some empirical graphs, and to better understand the factors impacting their size.

7.1 Datasets

We apply the algorithm presented in Section 5 to the collaboration networks of five different scientific communities [35], namely:

- `ca-AstroPh`, for the astrophysics community;
- `ca-CondMat`, for the condensed matter physics community;
- `ca-GrQc`, for the general relativity and quantum cosmology community;
- `ca-HepPh`, for the high energy physics-phenomenology community;
- and `ca-HepTh`, for the high energy physics-theory community.

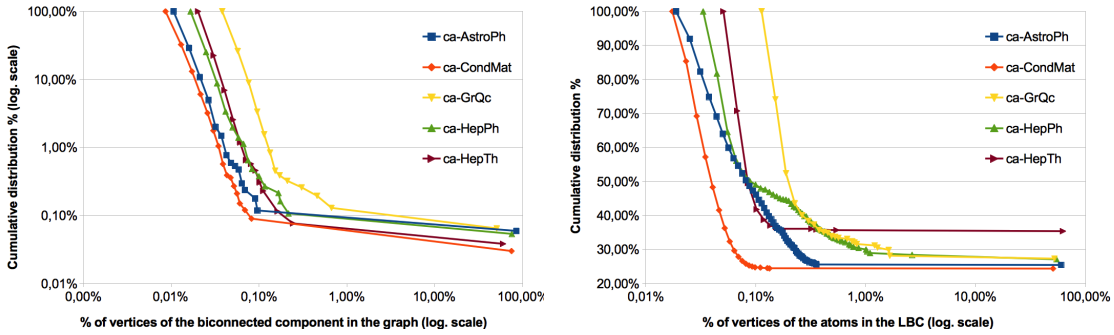
In the `ca-*` graphs, nodes represent scientists and edges represent collaborations (i.e., co-authoring a paper). These graphs are interesting to analyze the behavior of our algorithm and the size of their substitutes graphs because they have many cliques of various sizes. Indeed, a paper co-authored by k scientists induces a clique of size k in the graph. Furthermore, the number of co-authors per papers varies from one community to another. Therefore, we will observe different results in terms of the size of their substitute graphs, despite these graphs share many properties (see [35]).

7.2 Empirical results

We modified the clique-decomposition algorithm of [7] to implementent the substitution method presented in Section 5. We used it to compute, for each graph, the substitute of each atom of the decomposition.

Decomposition into biconnected components We observed that all of the five graphs are composed of one largest biconnected component, that we call LBC. The component includes from 50% to 84,85% of all the vertices. This can be observed from the cumulative distribution of the size of the biconnected components in Fig 13a. The cumulative number of components is given as a percentage of the total number of biconnected components, and the size of the components as a percentage of the total number of vertices in the graph. We noticed that all the biconnected components but the LBC are small: only covering at most 1% of the vertices.

Clearly, the smallest biconnected components can be safely ignored for the computation of hyperbolicity, provided that a) their number of vertices is less than 4, or that b) their diameter is less than two times the hyperbolicity of the LBC, which is always the case for these graphs (see [14]). Thus, we now focus on the clique-decomposition of the LBC, and on its resulting substitute graphs.



(a) Biconnected components (b) Atoms in the LBC

Figure 13: Cumulative distributions of the size of the biconnected components (Fig. 13a) and of the atoms in the LBC of each graph (Fig. 13b).

Clique-decomposition We plotted in Fig 13b the cumulative distribution of the size of the atoms of the LBC. The cumulative number of atoms is given as a percentage of the total number of atoms and the size of the atoms as a percentage of the total number of vertices in the LBC. Again, for all of the graphs, we observed one largest atom, that we call the LA. This atom includes from 50% to 60% of all the vertices, and all the other atoms only represent a small fraction of the overall vertices. In the worst case (`ca-HepPh`), all the atoms but the LA solely cover 2,65% of the vertices of the graph.

Moreover, like for the smallest biconnected components and as reported in [14], the substitute graphs of the smallest atoms can be safely ignored for the computation of hyperbolicity. As a result, the only component of the graphs to deal with for computing their hyperbolicity is the substitute graph of the LA. We will denote it LS in the sequel.

Size of the substitute graphs As explained in Section 5, the size of the LS depends on both the initial size of the LA and the number of added simplicial vertices. We have reported in Table 1 the original size n of each graph, the size n_B of its LBC, the size n_{LA} of the LA, and the size n_{LS} of the largest substitute. We have then computed the percentage R_{LA} of vertices that have been remove from the LBC to obtain the LA, that is $R_{LA} = \frac{n_B - n_A}{n_B}$. We observe a significant reduction rate R_{LA}, varying from 37,40% to 49,22%. We have also computed the reduction rate R_{LS} of the LS with respect to the LBC, that is $R_{LS} = \frac{n_B - n_{LS}}{n_B}$. We observe that this reduction rate falls between 3,79% and 21,95%. This indicates that the substitution method adds many simplicial vertices to the LA when constructing the LS, despite the simplication rules presented in Section 5.2.2.

Instance name	n	n_B	n_{LA}	n_{LS}	R_{LA}	R_{LS}	Cost	*Gain*	Time (in sec.)
`ca-CondMat`	23 133	17 234	8 751	13 451	49,22%	21,95%	27,27%	-	672
`ca-GrQc`	5 242	2 651	1 386	2 343	47,72%	11,62%	36,10%	10,33%	5
`ca-HepPh`	12 008	9 025	4 925	8 287	45,43%	8,18%	37,25%	13,77%	167
`ca-AstroPh`	18 772	15 929	9 561	15 325	39,98%	3,79%	36,19%	18,16%	679
`ca-HepTh`	9 877	5 898	3 692	5 270	37,40%	10,65%	26,75%	11,30%	53

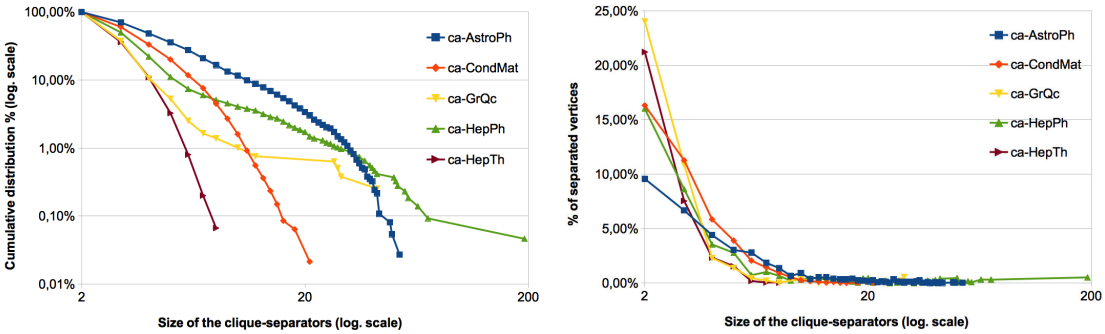
Table 1

We reported in Table 1 as *Cost* the percentage of vertices in the LBC representing the addition of new simplicial vertices. For the graph `ca-CondMat` - with the largest reduction rate R_{LS} -, the addition of simplicial vertices represents 27,27% of the size of its LBC, whereas for the `ca-AstroPh` graph - with the lowest reduction rate -, it goes up to 36,19%. This shows a difference of 18,16% between the R_{LS} of both extremal cases. The difference comes on the one hand, from the addition of 8,91% more simplicial vertices in the `ca-AstoPh` graph, and on the other hand from a difference of 9,25% between the reduction rates R_{LA} of their respective biggest atom LA. In the same way, when comparing the graphs `ca-HepTh`, `ca-HepPh` and `ca-GrQc`, respectively, with the graph `ca-CondMat`, we computed for R_{LA} a difference of 11,82%, 3,79% and 1,5%, respectively, plus a difference of 0,52%, 9,98% and 8,83%, respectively, for the addition of simplicial vertices. We thus concluded that the impact of n_{LA} and of the number of new simplicial vertices on the final size n_{LS} differs greatly depending on the graph.

7.3 Decomposition analysis

Having noticed the heterogeneous results of our empirical section, we are now analyzing in more details the properties causing the asymmetry between the various `ca-*` graphs.

Clique-decomposition We first analyzed the composition of the LA in terms of clique-separators. Let us denote by $\mathcal{X}_{LA} = \{X_1, \ldots, X_l\}$ the clique-minimal separators that are contained into the LA, disconnecting the atoms $\mathcal{A}_{LA} = \{A_1, \ldots, A_l\}$ from the LA, sequentially. We plotted in Fig 14a the cumulative distribution of the size of the clique-separators in the LA as a percentage of the total number of clique-separators. By doing so, we observed smaller clique-separators for the `ca-HepTh` and `ca-CondMat` graphs, with respectively a maximum size of 8 and 21, than for the three others graphs `ca-GrQc`, `ca-AstroPh` and `ca-HepPh`, having respectively clique-separators of maximum size 42, 53 and 192. Also, we reported in Table 2 that the ratio $R_{|\mathcal{X}_{LA}|}$ of the number of clique-minimal separators $|\mathcal{X}_{LA}|$ over the size n_{LA} of the LA, varies from 0.39 for `ca-AstroPh` to 0.54 for `ca-CondMat`. To sum up, there are more clique-separators in



(a) Cumulative distribution of the size of the clique-separators in the LA

(b) Percentage of separated vertices as a function of the size of the clique-separators in the LA

Figure 14

`ca-CondMat` than in `ca-AstropPh`, but there are larger clique-separators in `ca-AstropPh` than in `ca-CondMat`.

To complete our measurements, we related the size of clique-separators with the proportion of vertices that are disconnected by them from the LA. We reported in Table 2 as $\alpha_1 = n_B - n_{LA}$ the total number of vertices separated from the LA in the LBC, as $\alpha_2 = |V(\mathcal{A}_{LA} \setminus \mathcal{X}_{LA})|$ the number of vertices in the subset of atoms $\mathcal{A}_{LA}$ which are not in the LA, and as $|\mathcal{X}_{LA}|$ the number of clique-minimal separators in the LA. Finally, we computed the fraction $\Delta_1 = \frac{\alpha_1 - \alpha_2}{n_B}$, quantifying the percentage of vertices that are neither contained into the LA, nor in any of the atoms in $\mathcal{A}_{LA}$. We reported as $\Delta_2 = R_{LA} - \Delta_1$ the fraction of vertices in some atom of $\mathcal{A}_{LA}$, hence those that are *directly* separated from the LA.

Our results put in evidence that most of the vertices are either contained into the LA, or into some other atom intersecting the LA. Other vertices comprise around 2,88% and 7,03% of the overall vertices. Moreover, as shown with Fig 14b, where we plotted the percentage of separated vertices as a function of the size of clique-minimal separators, smaller clique-separators of size ≤ 5 are responsible for a significant part (w.r.t. Δ_2) of the vertices disconnected from the LA in `ca-CondMat` (37,34% of vertices over 49,22%), whereas in `ca-AstroPh` they solely disconnect 23,67% over 39,98% of vertices. This difference is not balanced with clique-separators of larger size, even though these ones disconnect 13,43% of vertices in `ca-AstroPh`, while only 5,67% in `ca-CondMat`. A comparison of `ca-CondMat` with `ca-HepPh` yielded similar results. In contrast, for the graphs `ca-GrQc` and `ca-HepTh`, we noticed that 6,71% and 4,91% more vertices, respectively, than in `CondMat`, were disconnected by edge-separators. But the rest of the clique-minimal separators, of larger size, only disconnect 16,67% and 11,70% of the vertices, respectively, whereas 26,70% of them are separated in `ca-CondMat`. Therefore, most of the difference for the final size of the subsistute graph LS comes from the number of vertices that are disconnected by clique-minimal separators of *small* size.

Substitute construction As illustrated with Figure 15, the number of additional vertices, resulting from the construction of the substitute graphs, depends on both the number and the size of the clique-minimal separators contained into the LA.

Instance name	α_1	α_2	$\lvert\mathcal{X}_{LA}\rvert$	$R_{\mathcal{X}_{LA}}$	$\Delta_1\%$	$\Delta_2\%$
`ca-CondMat`	8 483	7 413	4 698	0,54	6,21%	43,01%
`ca-GrQc`	1 265	1 079	697	0,5	7,03%	40,69%
`ca-HepPh`	4 100	3 727	2 165	0,44	4,13%	41,3%
`ca-AstroPh`	6 368	5 910	3 709	0,39	2,88%	37,1%
`ca-HepTh`	2 206	1 942	1 503	0,41	4,47%	32,93%

Table 2

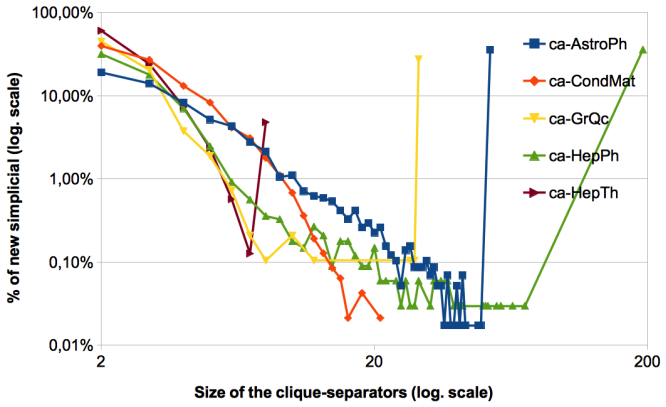


Figure 15: Number of added simplicial vertices

8 Conclusion

In this work, we proved a tight relation between the hyperbolicity of a graph and the maximum hyperbolicity taken over all its atoms. Our result subsumes the one in [11], as it directly implies that chordal graphs are 1-hyperbolic. Also, it implies that two graph classes extending chordal graphs, namely 2-chordal graphs [36, 37] and clique-separable graphs [25], have a bounded hyperbolicity.

In addition of our main result, we deduced from its proof a general substitution method, allowing us to modify the atoms at no extra-cost than the clique-decomposition; for graphs with hyperbolicity at least one, the maximum hyperbolicity taken over all the resulting graphs is exactly the hyperbolicity of the graphs, but the graphs to be considered may have a larger size than the atoms. Experiments suggest that the final size of the substitute graphs is mostly related with the number of clique-minimal separators of small size, and the disconnections resulting from them. We successfully applied this substitution method to outerplanar graphs, providing in this case a linear-time algorithm for computing the hyperbolicity. We let open whether the same can be done for other classes of graphs.

Finally, in the spirit of [2], it would be of interest to take advantage of the linear-time algorithm for outerplanar graphs, in order to yield an efficient computation of the hyperbolicity of planar graphs. Part of our future work will also consist in finding other graph decompositions which are applicable to the computation of this parameter.

References

[1] M. Abu-Ata and F. F. Dragan. Metric tree-like structures in real-life networks: an empirical study. Technical Report arXiv:1402.3364, ArXiv, 2014.

[2] B. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM (JACM)*, 41(1):153–180, 1994.

[3] H.-J. Bandelt and V. Chepoi. Metric graph theory and geometry: a survey. *Contemporary Mathematics*, 453:49–86, 2008.

[4] H.-J. Bandelt and H. Mulder. Distance-hereditary graphs. *Journal of Combinatorial Theory, Series B*, 41(2):182–208, 1986.

[5] S. Bermudo, J. Rodríguez, J. Sigarreta, and J.-M. Vilaire. Gromov hyperbolic graphs. *Discrete Mathematics*, 313(15):1575–1585, 2013.

[6] A. Berry, R. Pogorelcnik, and G. Simonet. Efficient clique decomposition of a graph into its atom graph. Technical Report RR-10-07, LIMOS, Aubière, France, Mar. 2010.

[7] A. Berry, R. Pogorelcnik, and G. Simonet. An introduction to clique minimal separator decomposition. *Algorithms*, 3(2):197–215, 2010.

[8] H. L. Bodlaender. Discovering treewidth. In *31st Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, volume 3381 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2005.

[9] M. Boguñá, F. Papadopoulos, and D. V. Krioukov. Sustaining the Internet with hyperbolic mapping. *Nature Communications*, 1(62):1–18, Oct. 2010.

[10] J. A. Bondy and U. Murty. *Graph theory with applications*, volume 290. Macmillan London, 1976.

[11] G. Brinkmann, J. H. Koolen, and V. Moulton. On the hyperbolicity of chordal graphs. *Annals of Combinatorics*, 5(1):61–69, 2001.

[12] J. Chakerian and S. Holmes. Computational tools for evaluating phylogenetic and hierarchical clustering trees. *Journal of Computational and Graphical Statistics*, 21(3):581–599, 2012.

[13] V. Chepoi, F. F. Dragan, B. Estellon, M. Habib, and Y. Vaxès. Notes on diameters, centers, and approximating trees of δ-hyperbolic geodesic spaces and graphs. *Electronic Notes in Discrete Mathematics*, 31:231–234, 2008.

[14] N. Cohen, D. Coudert, and A. Lancin. Exact and approximate algorithms for computing the hyperbolicity of large-scale graphs. Rapport de recherche RR-8074, Inria, Sept. 2012.

[15] D. Coudert and G. Ducoffe. On the recognition of C_4-free and 1/2-hyperbolic graphs. Research Report RR-8458, INRIA, Jan. 2014.

[16] W. H. Cunningham. Decomposition of directed graphs. *SIAM Journal on Algebraic Discrete Methods*, 3(2):214–228, 1982.

[17] P. de La Harpe and E. Ghys. *Sur les groupes hyperboliques d'après Mikhael Gromov*, volume 83. Progress in Mathematics, 1990.

[18] M. Didi Biha, B. Kaba, M.-J. Meurs, and E. SanJuan. Graph decomposition approaches for terminology graphs. In *MICAI 2007: Advances in Artificial Intelligence*, volume 4827 of *Lecture Notes in Computer Science*, pages 883–893. Springer, 2007.

[19] R. Diestel. *Graph theory, graduate texts in mathematics, vol. 173*. Springer, Heidelberg, 1997.

[20] Y. Dourisboure and C. Gavoille. Tree-decompositions with bags of small diameter. *Discrete Mathematics*, 307(16):2008–2029, 2007.

[21] F. Dragan. Tree-like structures in graphs: A metric point of view. In *39th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 8165 of *Lecture Notes in Computer Science*, pages 1–4. Springer, 2013.

[22] A. Dress, K. Huber, J. Koolen, V. Moulton, and A. Spillner. *Basic Phylogenetic Combinatorics*. Cambridge University Press, Cambridge, UK, Dec. 2011.

[23] H. Fournier, A. Ismail, and A. Vigneron. Computing the gromov hyperbolicity of a discrete metric space. Technical Report arXiv:1210.3323, ArXiv, Oct. 2012.

[24] T. Gallai. Transitiv orientierbare graphen. *Acta Mathematica Hungarica*, 18(1):25–66, 1967.

[25] F. Gavril. Algorithms on clique separable graphs. *Discrete Mathematics*, 19(2):159–165, 1977.

[26] M. Gromov. Hyperbolic groups. In S. Gersten, editor, *Essays in Group Theory*, volume 8 of *Mathematical Sciences Research Institute Publications*, pages 75–263. Springer, New York, 1987.

[27] M. Habib, C. Paul, and L. Viennot. Partition refinement techniques: An interesting algorithmic tool kit. *International Journal of Foundations of Computer Science*, 10(02):147–170, 1999.

[28] E. Howorka. On metric properties of certain clique graphs. *Journal of Combinatorial Theory, Series B*, 27(1):67–74, 1979.

[29] E. A. Jonckheere and P. Lohsoonthorn. Geometry of network security. In *American Control Conference*, volume 2, pages 976–981, Boston, MA, USA, 2004. IEEE.

[30] B. Kaba, N. Pinet, G. Lelandais, A. Sigayret, and A. Berry. Clustering gene expression data using graph separators. *In silico biology*, 7(4):433–452, 2007.

[31] W. S. Kennedy, O. Narayan, and I. Saniee. On the hyperbolicity of large-scale networks. Technical Report arXiv:1307.0031, ArXiv, 2013.

[32] J. H. Koolen and V. Moulton. Hyperbolic bridged graphs. *European Journal of Combinatorics*, 23(6):683–699, 2002.

[33] F. Le Gall. Faster algorithms for rectangular matrix multiplication. In *IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 514–523, New Brunswick, NJ, USA, 2012. IEEE.

[34] H.-G. Leimer. Optimal decomposition by clique separators. *Discrete Mathematics*, 113(1):99–123, 1993.

[35] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1):1–41, Mar. 2007.

[36] S. McCullough. 2-chordal graphs. In *Contributions to Operator Theory and its Applications*, pages 143–192. Springer, 1988.

[37] S. McCullough. Minimal separators of 2-chordal graphs. *Linear algebra and its applications*, 184:187–199, 1993.

[38] K. Olesen and A. Madsen. Maximal prime subgraph decomposition of bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 32(1):21–31, 2002.

[39] R. Paige and R. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[40] D. Rose, R. Tarjan, and G. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on computing*, 5(2):266–283, 1976.

[41] M. A. Soto Gómez. *Quelques propriétés topologiques des graphes et applications à internet et aux réseaux*. PhD thesis, Univ. Paris Diderot (Paris 7), 2011.

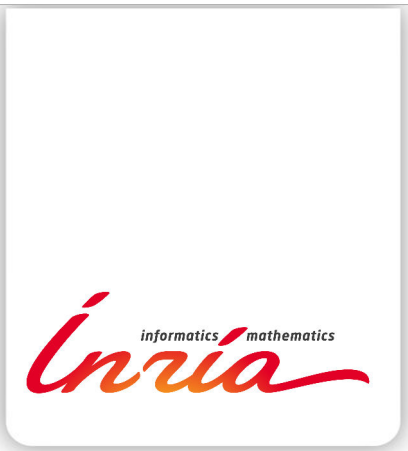
[42] M. Sysło. Characterizations of outerplanar graphs. *Discrete Mathematics*, 26(1):47–53, 1979.

[43] R. Tamassia and I. Tollis. Planar grid embedding in linear time. *IEEE Transactions on Circuits and Systems*, 36(9):1230–1234, 1989.

[44] R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[45] R. E. Tarjan. Decomposition by clique separators. *Discrete Mathematics*, 55(2):221–232, 1985.

[46] Y. Wu and C. Zhang. Hyperbolicity and chordality of a graph. *The Electronic Journal of Combinatorics*, 18(1):P43, 2011.



RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399