LaBRI - INRIA Bordeaux sud-ouest - 4 février 2011

Simulations de routage du schéma AGMaNT

Gavoille Cyril
<cyril.gavoille@labri.fr>

Glacet Christian
<christian.glacet@labri.fr>

Hanusse Nicolas
<nicolas.hanusse@labri.fr>

Ilcinkas David
<david.ilcinkas@labri.fr>

Majorczyk Frédéric
<frederic.majorczyk@irisa.fr>

Résumé

Nous proposons dans ce rapport une étude expérimentale du schéma de routage AGMaNT [AGM+08]. Ce schéma de routage à pour but de réduire la taille des tables de routage. Il garantit simultanément un étirement maximal de 3 et des tables de routage de $O(\sqrt{N}\log N)$ entrées pour tout réseau de N nœuds.

Plusieurs implémentations et optimisations de ce schéma de routage ont été développé. Des simulations sur bon nombre de réseaux tels que le graphe des systèmes autonomes de CAIDA et différents modèles de graphes réalistes (GLP [BT02], PFP [ZM04], etc.) ont notamment montré qu'en pratique :

- L'étirement moyen du routage est aux alentours de 1.5 ;
- Quelques centaines d'entrées par table suffisent pour des réseaux de 10000/15000 nœuds.

Table des matières

1 Introduction

1.1 Préambule

Dans un protocole de routage par plus court chemin classique, chaque nœud du réseau sait router vers tous les autres sommets. Ainsi, pour un réseau arbitraire de N nœuds, toutes les tables de routage peuvent nécessiter $\Omega(N)$ entrées. Informellement, le schéma de routage que l'on nomme AGMaNT [AGM+08], part de l'idée de répartir et factoriser les tables de routages de manière locale tout en préservant des garanties sur les détours effectués lors des routages. En effet, l'information nécessaire pour le routage n'est plus systématiquement stockée dans le nœud courant mais à "proximité" du nœud courant.

Plus précisément, les principales mesures de performances évaluées portent :
- sur la **mémoire** des routeurs : exprimée essentiellement en nombre d'entrées. On peut aussi la mesurer en kilo-octets mais nous n'avons pas chercher à optimiser cette mesure qui nécessite de la compression/décompression des données et entrainerait un surcoût de latence ;
- sur le **temps** de routage : exprimé en nombre de sauts du chemin de routage mais aussi par *l'étirement*, à savoir le rapport entre la longueur du chemin suivi par le routage et la distance entre source et destination.

Il est à noter que AGMaNTest un algorithme universel (valide pour toute topologie, pondérés ou non) et fonctionnant indépendamment du nommage initial du réseau. Dans nos expérimentations, nous proposons parfois des heuristiques d'optimisation liées à la topologie (graphes sans échelle) et les topologies considérées sont non pondérées et peu denses : degré moyen faible.

Description synthétique de AGMaNT .

Les principaux paramêtres du schéma de routage sont :
- le nombre de nœuds N et de liens M ;
- le diamêtre du graphe D ;
- le degré de répartition nommé également le nombre de couleurs K.

Tout nœud possède une couleur parmi les K proposées et partage une même fonction de hashage, qui appliquée à l'identifiant d'un noeud renvoie un entier entre 1 et K. D'autre part, une sélection de nœuds sont des points de repères nommés *landmark*. Un nœud contient suffisamment d'information pour router vers tous les landmarks [1] et vers les sommets de même couleur [2]. Informellement, le routage est basé sur les principes suivants :

- **destination proche** : le nœud courant sait router par plus court chemin vers la destination ;
- **destination éloignée** : le routage est effectué soit par l'intermédiaire d'un landmark ou des informations stockées à proximité de la source.

1.2 Notations et vocabulaire

On appelera B-voisin un nœud v présent dans la boule de voisinage d'un nœud u : on notera $v \in B(u)$. $d(u,v)$ désigne la distance entre deux noeuds u et v. Le schéma de routage AGMaNT [AGM+08] utilise 4 tables de routage différentes.

- La table 1 correspond à la table contenant les informations sur les routeurs de la boule de voisinage. La table 1 d'un nœud u contient donc à la fois le lien pour atteindre un B-voisin de u et les informations de routage de u dans les arbres de ses B-voisins.
- La table 2 correspond à la table contenant les informations permettant de router dans les arbres des landmarks.
- Les tables 3a et 3b contiennent les informations pour router quand le hash de la destination est égal à la couleur du nœud source considéré.
  - La table 3a sert lorsque l'on passe par le landmark le plus proche de la destination
  - la table 3b lorsqu'on passe par les boules de voisinage contigües.

  Lorsque l'on parle dans le texte de l'arbre du nœud u, il s'agit de l'arbre de plus court chemin enraciné en u qui couvre le graphe (s'il y en a plusieurs, celui qui a été calculé lors de la construction des tables). On notera également $h(v)$ le hashé de v et $c(v)$ la couleur du nœud v.

1. En réalité seuls les landmarks "utiles" sont considérés.
2. Plus exactement, tel que la valeur du hashage de la destination correspond à la couleur du nœud courant

1.3 Résultat d'une première implémentation

Le schéma de routage considéré nécessite ainsi :
- d'un choix d'un nombre de couleurs ;
- d'un algorithme d'affectation des couleurs.

En suivant l'analyse théorique de [AGM+08], on sait qu'en prenant $K = \sqrt{n}$, une affectation aléatoire uniforme des couleurs entre 1 et K et en considérant que chaque nœud possède la connaissance des $4K \log K$ plus proches nœuds alors avec grande probabilité :
- pour toute paire de source et destination, le routage atteint la destination avec un étirement maximal de 3 ;
- le nombre d'entrées par table de routage est en moyenne de $4K \log K + 2n/K$.

Ainsi, pour un graphe de 10000 sommets dans le modèle GLP, nous avons obtenu un nombre d'entrées de l'ordre de 3000. Ce résultat décevant a mené à une réflexion et une nouvelle proposition sur la construction des boules de voisinages et une étude fine sur la détermination de la valeur optimale de K. Ainsi, pour le même graphe, nous sommes descendus à un nombre d'entrées de l'ordre de 580. Les optimisations suivantes ont eu pour but de diminuer l'étirement moyen.

1.4 Bilan des expérimentations

Le principal changement effectué pour réduire à la taille des boules de voisinage consiste à ne pas fixer à l'avance la taille des boules mais de les construire de manière minimale en garantissant la propriété de K-mixité : toute boule contient au moins un représentant par couleur. Ceci a également un deuxième avantage :d'avoir une garantie absolue de routage. Ce n'était pas le cas dans l'article initial.

La table 1 résume les principales simulations effectuées. Il est à noter que nous avons réalisé des expériences avec une variante sur l'algorithme de colorations des sommets (*aléatoire vs. degré*). L'objectif était de spécialiser l'algorithme de coloration pour les graphes sans-échelle en tenant compte de propriétés macroscopiques supposées sur les graphes : concentration de sommets de haut degrés dans le cœur du réseau et boules de voisinage intersectant ces sommets. Ceci a permis de réduire la taille des boules de voisinage à considérer et à diminuer le nombre total d'entrées (réduction de 13%) sans avoir d'impact sur l'étirement.

Type de graphe	*(N,M)*	*Coloration*	*Étirement moyen*	*Étirement add. moyen*	*#entrées moyen*
GLP	(10 000, 36 000)	*Aléatoire*	-	-	-
		Degré	1.6	1.9	580
CAIDA	(16 301, 36 465)	*Aléatoire*	1.43	1.52	817
		Degré	1.42	1.48	721
Grille	(2 500, 9 902)	*Aléatoire*	-	-	-
		Degré	1.173	3.85	341
$G_{n,p=\frac{1}{2}}$	(2 500, 1 562 651)	*Aléatoire*	1.742	0.859	274
		Degré	-	-	-

TABLE 1 – Récapitulatif des simulations effectuées

L'étude fine des simulations (cf. sections suivantes) ont également montré des tendances générales qui sont les suivantes :
- pas de corrélation entre *degré* et *taille de tables* ;
- corrélation entre *degré* et *charge des nœuds* lors d'un échange total (N^2 routages possibles). Une embryon d'explication simple peut provenir du fait qu'un sommet reçoit de la part de ses voisins autant de messages que son degré mais aussi que les sommets de haut degrés ont tendance à être les plus centraux. Il serait intéressant que comparer la charge lors de routage de plus court chemins comme le fait BGP ;
- L'*affectation des couleurs* a une grande importance sur le nombre d'entrées/ Il s'agit sûrement du principal bras de levier du schéma pour espérer continuer à baisser le nombre total d'entrées par routeur ;
- L'*étirement moyen* a tendance à être faible lorsque le diamètre du graphe sous-jacent est grand. En cela, le modèle d'Erdös-Rényi peut éventuellement procurer des bornes maximales pour l'étirement ;
- Connaître tous ses voisins peut diminuer drastiquement l'étirement moyen (passage de 2 à 1 pour l'étoile, graphe complet, ...). Ce modèle est peut-être plus réaliste mais ce n'était pas celui envisagé dans l'article initial.

2 Description du schéma AGMaNT

2.1 Analyse algorithmique

Dans le schéma AGMaNT, chaque nœud doit posséder une couleur, le nombre de couleurs différentes étant un paramètre du schéma. Une couleur va être choisie pour être celle des landmarks.

On considère un nœud source u et un nœud destination v. Suivant les caractéristiques de u et de v (couleur, hashé) et leur « situation » dans le graphe, 5 types de routage peuvent être utilisés.

2.1.1 Routage a : $v \in B(u)$ (v est un B-voisin de u)

Si v est un B-voisin de u, le routage de u à v se fait à l'intérieur de la boule de voisinage de u par plus court chemin. Il est néanmoins nécessaire pour cela que pour tout sommet u' de $B(u)$ tel que $d(u',v) < d(u,v)$ on a $v \in B(u')$.

2.1.2 Routage b : v est un landmark

Si v est un landmark, le routage de u à v se fait sur un plus court chemin dans l'arbre de v. Ce chemin est également un plus court chemin dans le graphe car l'arbre de v est un arbre de plus court chemin.

2.1.3 Routage c : $h(v) = c(u)$; passage par le landmark le plus proche de v

On note l_v le landmark le plus proche de v. Le routage dans ce cas se fait sur un plus court chemin **dans l'arbre de** l_v comme décrit dans [FG01]. Ce chemin n'est pas nécessairement un plus court chemin dans le graphe.

2.1.4 Routage d : $h(v) = c(u)$; passage par des boules de voisinage contigüe

Le routage de u à v dans ce cas se fait en 4 étapes.
- le nœud u route le message vers un nœud w tel que $u \in B(w)$. Cette étape se fait dans l'arbre de u suivant un plus court chemin dans l'arbre et dans le graphe.
- le message est envoyé de w vers x qui est un B-voisin de w ($x \in B(w)$)
- le routage de x vers y qui est un voisin direct de x
- la dernière étape du routage se fait entre y et v dans l'arbre de y suivant un plus court chemin.

Les trois dernières étapes de ce routage doivent se faire suivant un plus court chemin de w à v.

2.1.5 Routage e : cas général ($h(v) \neq c(u)$)

Ce type de routage se fait en deux étapes.
- router le message de u à z qui est un B-voisin de u tel que $h(v) = c(z)$.
- z route le message vers v suivant le routage c (passage par le landmark le plus proche de v) ou suivant le routage d (passage par des boules de voisinage contigüe).

2.2 Structure de données

Les différents types de routage nécessitent la présence de diverses informations de routage contenues dans les différentes tables de routage et également l'ajout d'en-têtes dans les paquets pour certains types de routage.

2.2.1 Routage a

Pour assurer la routage a, il suffit que chaque nœud stocke le lien permettant d'atteindre par un plus court chemin chacun de ces B-voisins. Cette information est stockée dans la table 1. Cette seule information est nécessaire du fait de la construction des boules de voisinage.

2.2.2 Routage b

Pour assurer le routage b, il suffit que chaque nœud stocke le lien vers son père dans les arbres de chaque landmark. Cette information est stockée dans la table 2.

2.2.3 Routage c

Pour assurer le routage c, il suffit que :

- chaque nœud u stocke les informations de routage décrites dans [FG01] le concernant pour les arbres de chaque landmark. Ces informations sont stockées dans la table 2.
- chaque nœud u stocke pour tous les nœuds v tel que $h(v) = c(u)$ le landmark le plus proche du nœud v noté l_v et les informations de routage de v dans l'arbre de l_v. Ces informations sont stockées dans la table 3a.

En plus de ces informations de routage, il est nécessaire de rajouter un en-tête au message envoyé de u vers v. Cet en-tête doit contenir l'identifiant du landmark l_v et « l'adresse » de v dans l'arbre de l_v (contenant son identifiant dans l'arbre de l_v et son capth [3] dans cet arbre).

2.2.4 Routage d

Pour assurer le routage d, il suffit que :

- chaque nœud stocke les informations de routage décrites dans [FG01] le concernant pour les arbres de tous ses B-voisins. Ces informations sont stockées dans la table 1. Ces informations vont permettre d'assurer la première et la quatrième étape du routage d.
- chaque nœud u stocke pour chaque nœud v tel que $h(v) = c(u)$ « l'adresse » de w dans l'arbre de u, le nœud x, le nœud y et « l'adresse » de v dans l'arbre de y. Ces informations sont stockées dans la table 3b.

En plus de ces informations de routage, il est nécessaire de rajouter un en-tête au message contenant l'étape du routage, « l'adresse » de w dans l'arbre de u, l'identifiant du nœud x, celui du nœud y et « l'adresse » de v

2.2.5 Routage e

La seconde étape du routage e est assurée grâce aux informations stockées pour les routage c ou d. Pour assurer la première étape du routage, il suffit que chaque nœud stocke, pour chaque couleur, un nœud de cette couleur présent dans sa boule de voisinage.

Un en-tête est nécessaire pour assurer la première étape du routage. Cet en-tête contient l'identifiant de z, qui est un B-voisin de u.

2.2.6 Remarques

Les informations stockées dans la table 3a et dans la table 3b sont redondantes. Pour un nœud u et un nœud v donné qui vérifie la propriété $h(v) = c(u)$, le nœud u ne va stocker que la meilleure entrée (celle qui minimise la distance $d(u,v)$) concernant le nœud v soit dans la table 3a soit dans la table 3b.

La construction des tables de routage nécessitent tout d'abord la coloration des nœuds du graphe et à partir de cette coloration la construction des boules de voisinage.

2.2.7 Tables de routage

En fonction de la description faite des types de routage, les tableaux 2, 3, 4 et 5 présentent en résumé le contenu des différentes tables de routage.

3. cpath = suite des idenfiant des fils ($1 \leq identifiant \leq$ nombre de fils) ou pères (identifiant particulier, ex. -1) pour atteindre le sommet destination

id B-voisin	lien	id DFS	poids	poids du 1er fils	longueur du cpath	lien vers la racine	lien vers le 1er fils

TABLE 2 – Résumé d'une entrée de la table de routage 1

Une entrée de la table 1 (voir *tableau 2*) contient un lien sur un plus court chemin vers le B-voisin identifié pour permettre le routage a. Le reste de l'entrée correspond aux informations de routage du nœud considéré dans l'arbre du B-voisin identifié.

id du landmark	id DFS	poids	poids du 1er fils	longueur du cpath	lien vers la racine	lien vers le 1er fils

TABLE 3 – Résumé d'une entrée de la table de routage 2

Une entrée de la table 2 (voir *tableau 3*) contient les informations de routage du nœud considéré dans l'arbre du landmark identifié par l'entrée.

id du landmark	id DFS	cpath	id de la destination

TABLE 4 – Résumé d'une entrée de la table de routage 3a

Une entrée de la table 3a (voir *tableau 4*) contient l'identifiant du landmark le plus proche de la

destination et son « adresse » (id DFS et cpath) dans l'arbre du landmark. Ces informations doivent être reportées dans l'en-tête du message à envoyer pour permettre le routage.

Une entrée de la table 3b (voir *tableau 5*) contient de nombreuses informations sur le routage à effectuer. Ces informations seront reportées dans l'en-tête du message à envoyer pour permettre le routage.

Remarque : Il faut également que chaque nœud puisse router un message vers un nœud de la « bonne » couleur dans le cas du routage e. Ceci peut être fait de différentes manières. La plus simple est qu'un nœud stocke une liste contenant un nœud de chaque couleur présent dans sa boule de voisinage.

2.2.8 Description schématique du routage

id DFS de w	cpath de w	id de x	id de y	id DFS de v	cpath de v	id de la destination

TABLE 5 – Résumé d'une entrée de la table de routage 3b

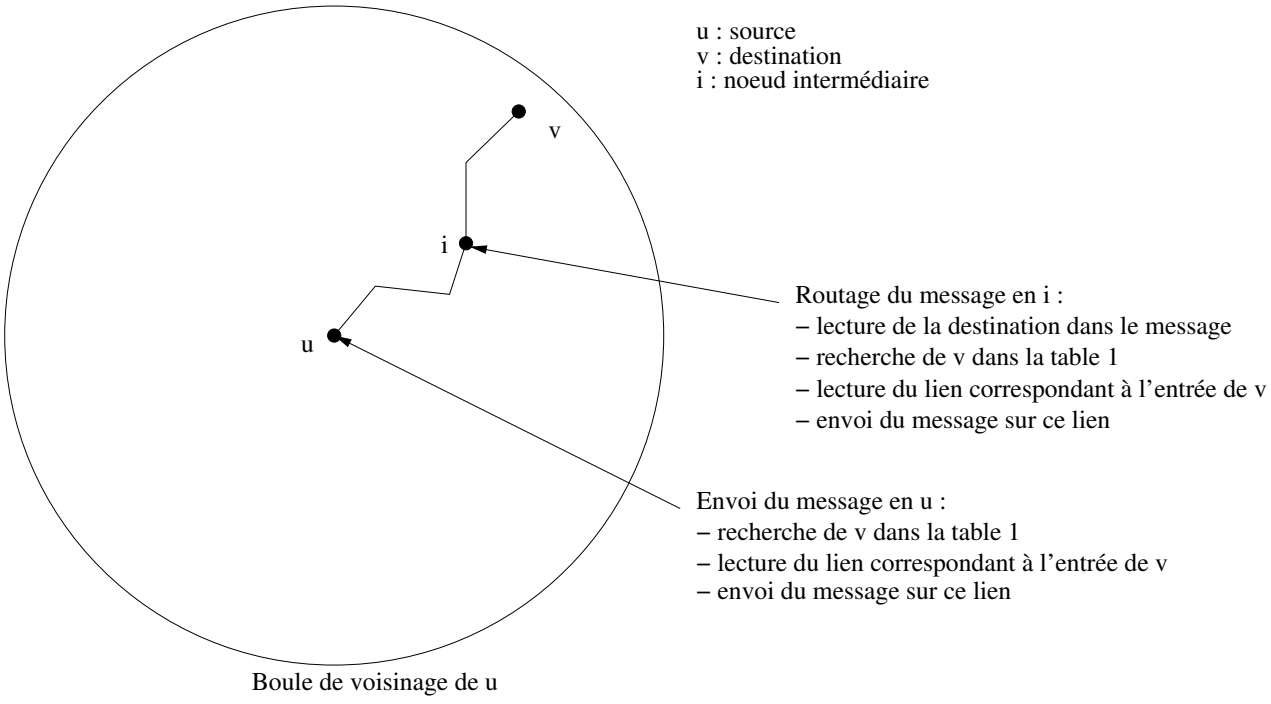
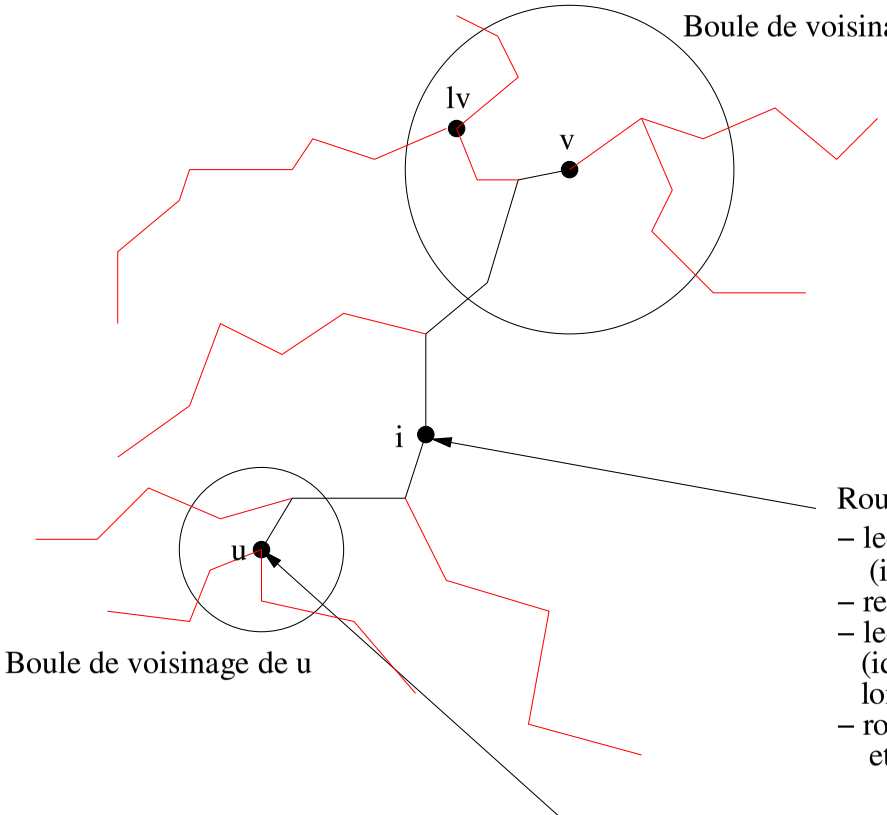


FIGURE 1 – Schéma du routage a

FIGURE 2 – Schéma du routage b



u : source
v : destination (h(v)=c(u))
i : noeud intermédiaire
lv : landmark le plus proche de v
——— : arbre de lv
——— : chemin du message

Routage du message en i :
– lecture de l'en–tete dans le message
(identifiant de lv, identifiant DFS de v et son cpath dans l'arbre de lv)
– recherche de lv dans la table 2
– lecture de cette entrée
(identifiant DFS de i, poids de i, poids du premier fils de i,
longueur du cpath de i, lien vers la racine, lien vers le premier fils)
– routage dans l'arbre de lv en utilisant l'entrée lue précédemment
et l'identifiant DFS de v et son cpath dans l'arbre de lv

Envoi du message en u :
– recherche de v dans la table 3a
– lecture de lv, de l'identifiant DFS de v dans l'arbre de lv et du cpath de v dans cet arbre
– création du message avec un en–tete contenant l'identifiant de lv,
l'identifiant DFS de v et son cpath dans l'arbre de lv
– recherche de l'entrée de lv dans la table 2
– lecture de cette entrée (id DFS de u, poids de u, poids du premier fils de u,
longueur du cpath de u, lien vers la racine, lien vers le premier fils)
– routage dans l'arbre de lv en utilisant l'entrée lue précédemment
et l'identifiant DFS de v et son cpath dans l'arbre de lv

FIGURE 3 – Schéma du routage c

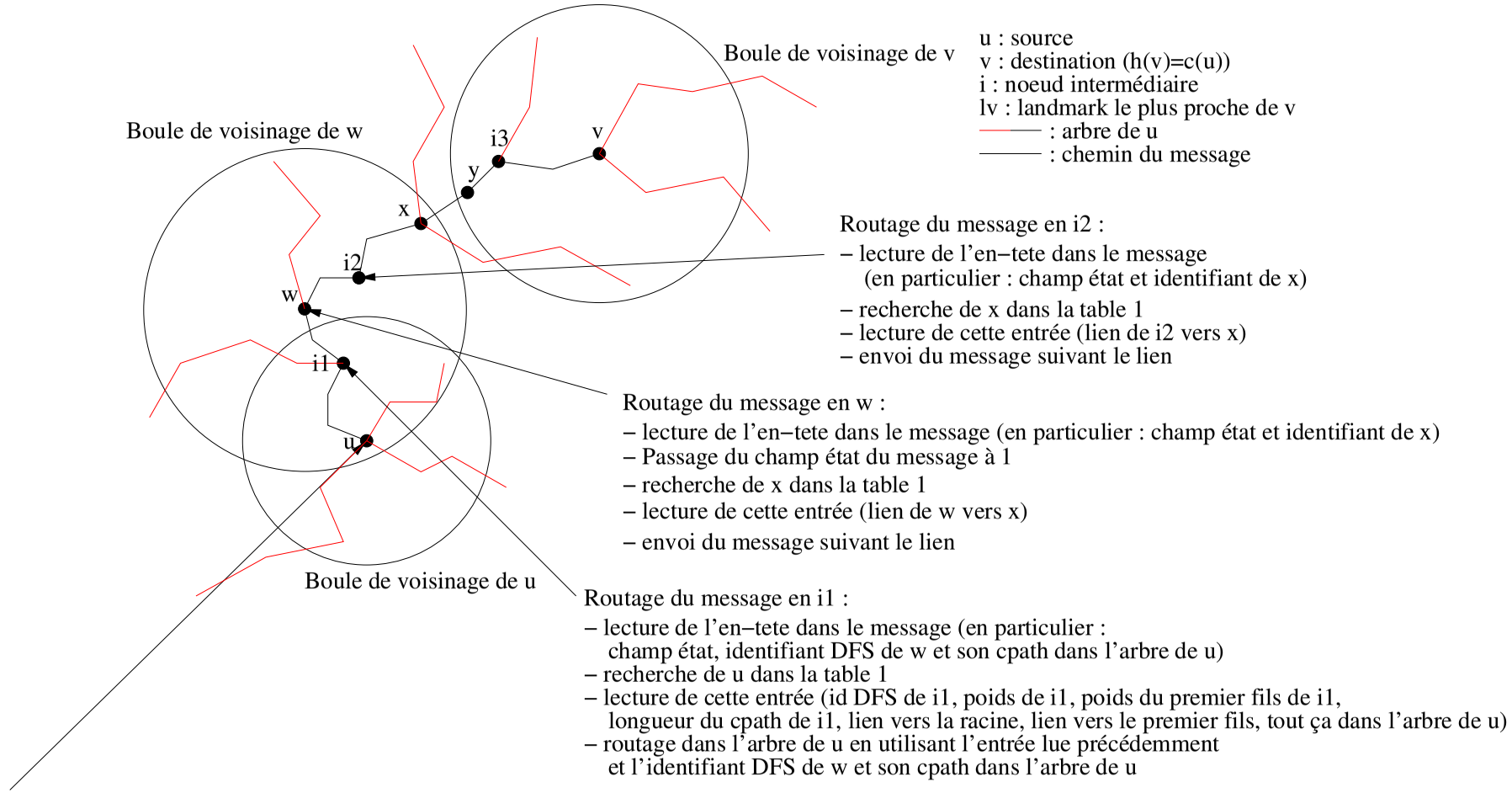
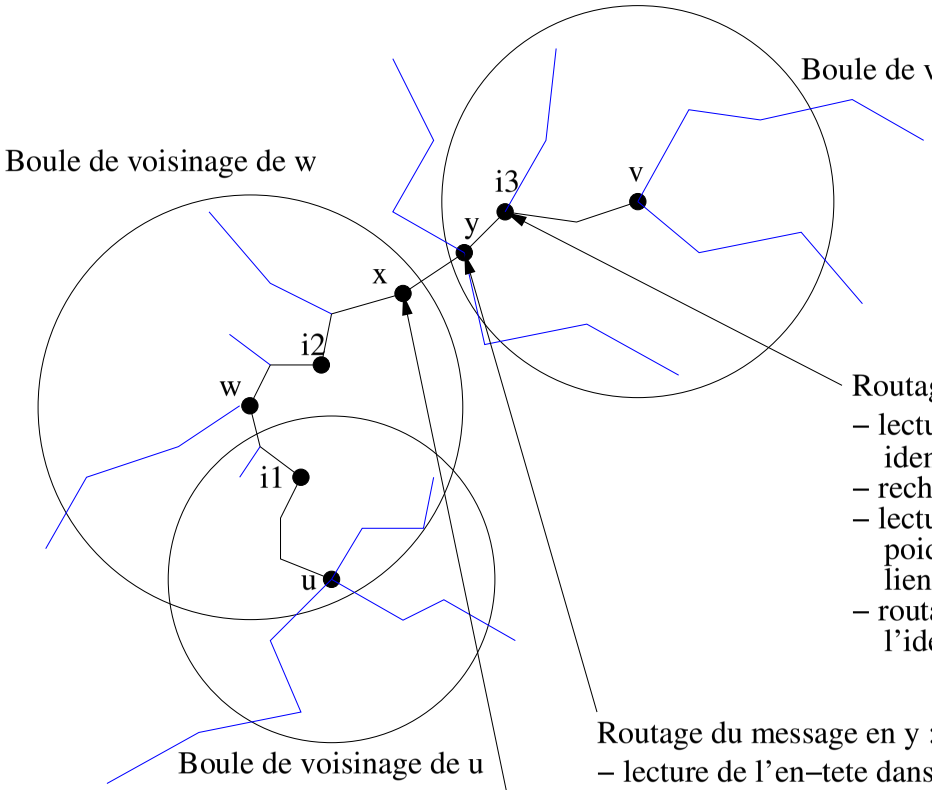


FIGURE 4 – Schéma du routage d (première partie)



u : source
v : destination (h(v)=c(u))
i : noeud intermédiaire
lv : landmark le plus proche de v
—— : arbre de y
—— : chemin du message

Routage du message en i3 :
– lecture de l'en–tete dans le message (en particulier : champ état, identifiant DFS de v et son cpath dans l'arbre de y)
– recherche de y dans la table 1
– lecture de cette entrée (identifiant DFS de i3, poids de i3, poids du premier fils de i3, longueur du cpath de i3, lien vers la racine, lien vers le premier fils, tout ça dans l'arbre de y)
– routage dans l'arbre de y en utilisant l'entrée lue, l'identifiant DFS de v et son cpath dans l'arbre de y

Routage du message en y :
– lecture de l'en–tete dans le message (en particulier : champ état, identifiant DFS de v et son cpath dans l'arbre de y)
– passage du champ état du message à 3
– recherche de y dans la table 1
– lecture de cette entrée (identifiant DFS de y, poids de y, poids du premier fils de y, longueur du cpath de y, lien vers la racine, lien vers le premier fils, tout ça dans l'arbre de y)
– routage dans l'arbre de y en utilisant l'entrée lue l'identifiant DFS de v et son cpath dans l'arbre de y

Routage du message en x :
– lecture de l'en–tete dans le message (en particulier : champ état et identifiant de y)
– passage du champ état du message à 2
– y est un voisin de x, le lien de x à y est connu
– envoi du message suivant ce lien

FIGURE 5 – Schéma du routage d (seconde partie)

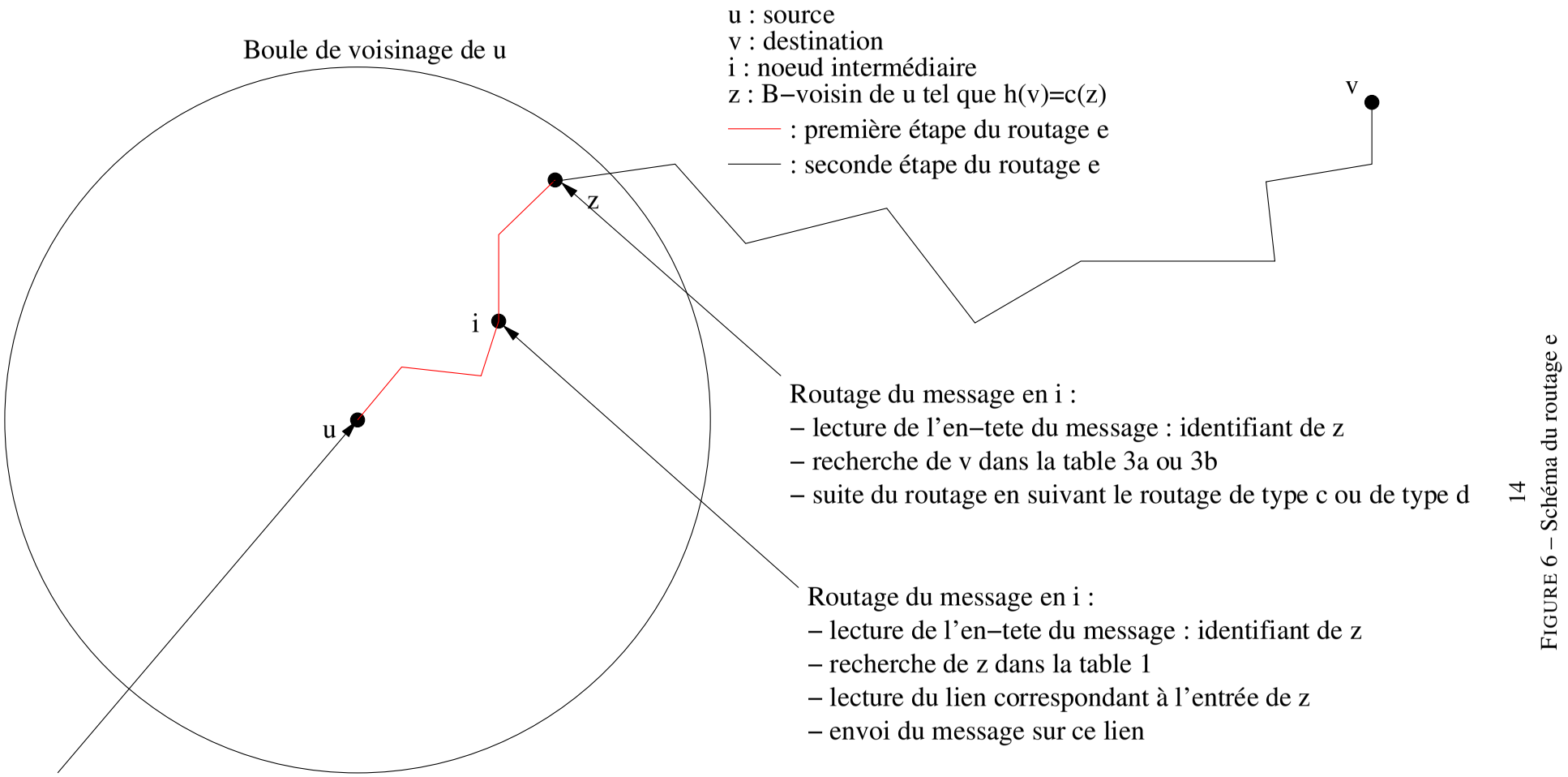


FIGURE 6 – Schéma du routage e

3 Implémentation

3.1 Calculs préliminaires nécessaires à la construction des tables

Cette partie va décrire tout ce qu'il est nécessaire de faire pour construire les tables du schéma AGMaNT. Le seul élément que l'on possède au départ est le graphe. Ces calculs préliminaires nécessitent une connaissance globale du réseau.

3.1.1 Affectation de couleurs des nœuds

La première chose que l'on va faire est de colorier les différents nœuds du graphe G. Pour cela, on peut considérer deux algorithmes :

- **Affectation aléatoire uniforme des couleurs** : Cet algorithme a l'avantage de pouvoir être réalisé de manière distribué et garantit, avec grande probabilité, des boules de voisinage de taille $K \log K$.
- **En ordonnant selon les degrés** : cette seconde méthode consiste tout d'abord à trier les nœuds suivant leur degré et ensuite de choisir la couleur suivant la position du nœud dans la liste triée. La couleur du nœud en position i est égale à $(i-1) mod K$. Cette seconde méthode, conçu pour des graphes sans échelle, permet de réduire la taille de tables en moyenne de 3% (580 entrées en moyenne au lieu de 600) pour un graphe de type GLP de 10000 nœuds. L'inconvénient de cette méthode est qu'elle n'apporte aucune garantie théorique sur la taille des boules de voisinage et nécessite une connaissance globale. Ceci est cependant un premier pas dans la direction de l'optimisation des graphes sans échelle.

3.1.2 Calcul des boules de voisinage

Une fois le coloriage des nœuds terminé, on calcule les boules de voisinage de chaque nœud. Pour cela, on va rechercher les voisins du nœud considéré à distance de plus en plus grande jusqu'à ce que la boule de voisinage de ce nœud respecte la condition énoncée dans l'article, c'est-à-dire qu'au moins un nœud de chaque couleur soit présent dans la boule de voisinage. Pour garantir l'inclusion des chemins intra-boule, on utilise un ordre total sur les nœuds du graphe et lorsequ'on étend la boule de voisinage, on parcoure les nouveaux sommets dans cet ordre. Chaque boule de voisinage est stockée sous la forme d'une *ArrayList*. L'ensemble des boules de voisinage est stocké dans un *Vector*. La boule de voisinage du nœud i est stocké à la $i-1$ème position du Vector. De la même manière, pendant le calcul des boules de voisinage, on crée une structure contenant « l'inverse » des boules de voisinage. La boule inverse du nœud u contient tous les nœuds qui ont u dans leur boule de voisinage. Cette dernière structure de données servira lors du calcul des entrées dans la table 3b.

3.1.3 Structures annexes

On stocke également dans des listes les nœuds de chaque couleur et les nœuds dont le hashé est le même.

On calcule également les landmarks les plus proches de chaque nœud. On stocke les identifiants dans le graphe de ces landmarks dans un tableau de `short`.

3.1.4 Calcul des plus courts chemins

Ensuite, on effectue un Floyd-Warshall pour calculer les plus courts chemins et on calcule la matrice des *next*. Cette matrice contient en (i, j) l'identifiant d'un voisin de i sur un plus court chemin vers j. Cette matrice permet donc de calculer très rapidement les plus courts chemins de tous les nœuds à tous les nœuds et leur longueur (en au plus le diamètre du graphe, on peut obtenir la longueur d'un plus court chemin et ce plus court chemin avec des opérations très simples).

3.1.5 Calcul des arbres de plus courts chemins et des informations de routage dans ces arbres

Ensuite pour chaque nœud, on va calculer un arbre de plus court chemin dont le nœud considéré est la racine couvrant le graphe et on va également calculer le DFS dans l'arbre comme spécifié dans [FG01]. Dans ce DFS particulier, on

numérote en premier le fils dont le sous-arbre de l'arbre de plus court chemin contient le plus de nœuds. Le résultat du DFS est stocké dans un tableau (liste de `short`) et les arbres sont stockés sous forme de plusieurs tableaux (tableaux de N `short` : tableau du nombre de fils, tableau des fils, tableau d'index). On calcule également les poids des nœuds dans chacun des arbres (stockés dans des tableaux de taille N de `short`).

3.2 Construction des tables de routage

La construction des tables à ce stade peut se faire localement en utilisant uniquement les informations calculées précédemment. En préambule, notre implémentation suppose que le nombre de nœuds $N < 2^{16}$. Cela nous permet d'utiliser le type `short` qui prend 2 octets. D'autre part, les pointeurs évoqués nécessitent 8 octets.

3.2.1 Table 1

Par rapport à l'entrée décrite dans le *tableau 2*, l'entrée présente dans l'implémentation contient en plus l'information sur la couleur du B-voisin. Cela permet éventuellement de jouer sur un paramètre du schéma de routage AGMNT.

À partir de ces informations calculées lors de la première étape de construction des tables, il est possible de remplir une première partie de la table 1. Pour chaque B-voisin d'un nœud, on peut calculer le lien qui relie ce B-voisin au nœud considéré. Pour chaque B-voisin, on remplit dans la table 1 le lien qui permet d'atteindre ce B-voisin et la couleur du B-voisin. Cette dernière information n'est plus utilisée par la fonction de routage mais peut éventuellement servir si l'on considère des pannes.

Une fois cette première partie remplie, il est possible de remplir les autres informations contenues dans la table 1. Ces informations concernent le routage dans l'arbre de plus court chemin de ces B-voisins. Pour chaque nœud u, on va considérer les nœuds de l'inverse de sa boule de voisinage, c'est-à-dire tous les nœuds dont u est un B-voisin. Pour chacun de ces nœuds, on va remplir les informations de ce nœud dans l'arbre de routage enraciné en u. Il faut ainsi trouver l'identifiant du nœud considéré dans l'arbre enraciné en u, son poids, le poids de son fils le plus lourd, la longueur de son cpath (voir [FG01]) et également le lien vers la racine et le lien vers le fils le plus lourd. Ces liens sont stockés sous forme de pointeurs (8 octets) actuellement : il est donc possible de réduire l'empreinte mémoire de la table 1 en remplaçant ces pointeurs par des indices dans un tableau de liens stockés dans le graphe. On peut retrouver ces informations à partir de l'arbre enraciné en u, du tableau des poids de cet arbre et du tableau contenant les indices du DFS de cet arbre. La longueur du cpath est l'élément le plus complexe à obtenir : il faut parcourir le chemin de la racine de l'arbre jusqu'au nœud cible et compter le nombre de fois où l'on ne passe pas par le fils le plus lourd.

Le *tableau 6* représente de manière synthétique les informations contenues dans une entrée de la table 1.

id B-voisin	lien	couleur	id DFS	poids	poids du 1er fils	longueur du cpath	lien vers la racine	lien vers le 1er fils
`short`	pointeur	`short`	`short`	`short`	`short`	`short`	pointeur	pointeur

TABLE 6 – Résumé d'une entrée de la table 1

À l'heure actuelle, cette table prend en mémoire un total de 36 octets.

3.2.2 Table 2

La construction de la table 2 est très similaire à la construction de la seconde partie de la table 1. Pour tous les nœuds, on calcule les informations de routage dans les arbres des landmarks : identifiant dans le dfs, poids, poids du premier fils, cpath, lien vers la racine et lien vers le premier fils.

La table des landmarks prend en mémoire un total de 26 octets.

id du landmark	id DFS	poids	poids du 1er fils	longueur du cpath	lien vers la racine	lien vers le 1er fils
short	short	short	short	short	pointeur	pointeur

TABLE 7 – Résumé d'une entrée de la table 2

3.3 Table 3a

Considérons tout d'abord la construction de la table 3a (routage c : passage par le landmark le plus proche). Dans cette table d'un nœud u, on stocke les informations nécessaires pour router vers un nœud v dont le hashé est égal à la couleur du nœud u. On va stocker dans cette table l'identifiant dans le graphe du landmark le plus proche de v, l'identifiant DFS du nœud v dans l'arbre de ce landmark, le cpath du nœud v dans cet arbre. Ces informations sont nécessaires pour le routage. On va stocker une information supplémentaire qui n'est pas nécessaire pour le routage : une borne max sur la longueur du chemin pris en utilisant cette entrée (distance de u au landmark + distance du landmark à v). Cette information sera utile lorsque l'on va comparer les entrées de la table 3a à celle de la table 3b pour qu'il n'y ait pas de redondance dans les entrées de ces tables (il est ainsi possible d'enlever cette information de la table 3a après la comparaison des entrées entre les deux tables ; ceci n'est pas fait dans l'implémentation actuelle). Le *tableau 8* résume le contenu d'une entrée de la table 3a.

id du landmark	id DFS	cpath	id de la destination	longueur du chemin
short	short	liste de liens	short	short

TABLE 8 – Résumé d'une entrée de la table 3a

Pour remplir les informations dans la table 3a, on va donc, pour chaque nœud u, parcourir la liste des nœuds dont le hashé correspond à la couleur du nœud u. Pour chacun des nœuds, on va rechercher son identifiant dans le DFS et le cpath. Le cpath est l'élément le plus complexe à obtenir : il s'obtient de la même manière que la longueur du cpath.

Cette table permet le routage vers les sommet de même couleur en utilisant les landmarks prend en mémoire un total de $(20 + distance(l_v, v) * 8 < 20 + 8D)$ *octets au maximum.*

3.3.1 Table 3b

La table 3b contient également des informations de routage vers des nœuds dont le hashé est égal à la couleur de la du nœud source. C'est la table la plus complexe à construire. D'après [AGM+08], le chemin suivi par le message de source u à destination de v est le suivant :

1. de la source u à w, dont u est un B-voisin, suivant un plus court chemin ;
2. de w à x, x étant un B-voisin de w ;
3. de x à y où y est un voisin de x ;
4. de y à v, y étant un B-voisin de v ;

Le chemin de w à v doit également être un plus court chemin d'après [AGM+08].

L'étape 1 du routage s'effectue dans l'arbre enraciné en u. L'étape 2 s'effectue dans la boule de voisinage de w. L'étape 3 est un routage vers un voisin. L'étape 4 du routage s'effectue dans l'arbre enraciné en y.

Le *tableau 9* résume les informations contenues dans une entrée de la table 3b. Pour des facilités d'implémentation, l'entrée implémentée contient en plus de l'identifiant de y le lien de x à y qui n'est pas nécessaire.

L' *algorithme 1* présente de manière détaillée en pseudo-code la construction de cette table.

Après avoir trouvé w, x et y, il reste à calculer les deux cpaths nécessaires. Cela se fait comme décrit précédemment.

Cette table permet le routage vers les sommet de même couleur en utilisant les boules de voisinage prend en mémoire un total de $(46 + (distance(u, w) + distance(l_v, v)) * 8 < 46 + 16D)$ *octets au maximum.*

Algorithme 1 : Construction de la table 3b

Data : *nodes* l'ensemble des nœuds, *landmarkNodes* l'ensemble des landmarks, $vicinityBall(u)$ l'ensemble des B-voisins du nœud u, $invVicinityBall(u)$ l'ensemble des nœuds qui ont u pour B-voisin, $neighbors(u)$ l'ensemble des voisins du nœud u, $color(u)$ la couleur du nœud u, $coloredNodes(color)$ l'ensemble des nœuds de la couleur $color$, $hash(u)$ le hashé du nœud u, $radius(u)$ la valeur du rayon de la boule de voisinage de u, $dist(u,v)$ la distance de u à v, $intersection(nodes1, nodes2)$ l'intersection des ensembles $nodes1$ et $nodes2$, $shortestPath(u,v)$ la liste des nœuds sur un plus court chemin de u à v, $nextOnShortestPath(u,v,w)$ donne le nœud qui suit w sur un plus court chemin de u à v

```
for v in nodes do
    if v in landmarkNodes then
        next
    hash = hash(v)
    largerVicinityBall = vicinityBall(v)
    for neighbor in vicinityBall(v) do
        Ajouter neighbors(neighbor) à largerVicinityBall
    /* largerVicinityBall contient la boule de voisinage de v et les voisins de cette boule */
    for u in coloredNodes(hash) do
        /* On vérifie que u et v ne sont pas les mêmes nœuds et que v n'est pas dans la boule de
        voisinage de u */
        if u != v and v not in vicinityBall(u) then
            minDist = MaxInteger
            for w in invVicinityBall(u) do
                if radius(w) + radius(v) + 1 < dist(w,v) or dist(u,w) + dist(w,v) >= minDist then
                    /* On ne trouvera pas de x et de y qui vont convenir ou on a déjà trouvé mieux */
                    next
                intersectionNodes = intersection(largerVicinityBall, vicinityBall(w))
                if w == v then
                    /* On ne pourra pas trouver mieux comme chemin */
                    break
                else
                    potentialXNodes = intersection(intersectionNodes, shortestPath(w,v)) for x in
                    potentialXNodes do
                        if x != v then
                            y = nextOnShortestPath(u,v,x)
                            if y in VicinitBall(v) then
                                minDist = dist(u,w) + dist(w,v)
            /* Pour u et v, on a trouvé w, x, et y qui donnent une longueur de chemin minimale */
```

id de w	id DFS de w	cpath de w	id de x	id de y	lien de x à y	id DFS de v	cpath de v	id de la destination	longueur du chemin
`short`	`short`	liste de liens	`short`	`short`	pointeur	`short`	liste de liens	`short`	`short`

TABLE 9 – Résumé d'une entrée de la table 3b

3.3.2 Réduction des tables 3a et 3b

La table 3b contient des entrées pour des nœuds destination qui sont également présent dans la table 3a. Il faut enlever les entrées (qui correspondent à un chemin) qui sont les moins performantes. On compare donc pour chaque entrée de la table 3b si le chemin qui a la même destination dans la table 3a est plus performant ou non. Pour cela, on utilise le champ « longueur du chemin » des deux tables. Ce qu'il faut noter est que ce champ ne donne pas forcément la longueur du chemin qui sera réellement utilisé lors d'un routage du paquet mais seulement une borne supérieure. On enlève donc des deux tables *pour chaque destination l'entrée la moins performante a priori.*

3.4 Étude de la complexité de création des tables

On va s'intéresser dans cette partie à la complexité des calculs prenant la majeure partie du temps de la construction des tables : le calcul des plus couts chemins, la construction des arbres de plus courts chemins et la construction de la table 3b.

Le calcul des plus courts chemins (matrice des *nexts*) se fait grâce à un Floyd-Warshall et est donc de complexité $O(N^3)$.

La construction des arbres de plus courts chemins permet de construire une structure d'arbre prenant peu de place en mémoire. Cette construction se fait à partir de la matrice des *nexts* construite à partir du calcul des plus courts chemins. Cette construction se fait en $O(N^2)$ dans le pire des cas et en $O(N)$ dans le meilleur. Au total, la complexité de cette partie est $O(N^3)$.

Ces deux étapes peuvent éventuellement grandement améliorées par l'utilisation de BFS dont la complexité est moindre que le Floyd-Warshall et permettrait éventuellement la construction de la structure d'arbres en même temps que le calcul des plus courts chemins.

La construction de la table 3a se fait suivant une complexité en moyenne $O(N^2 * K \ln^2(K))$, en effet, si l'on considère que les boules inverses ont en moyenne la même taille que les boules de voisinage, ce qui se vérifie de manière expérimentale sur les graphes GLP, pour chaque noeud destination, il est nécessaire de parcourir les noeuds dont la couleur est égale au hash de la destination. Ces noeuds sont en moyenne N/K. Pour chacun de ces noeuds, il faut parcourir tous les noeuds qui sont dans leur boule inverse. Ces noeuds sont au nombre de $K * \ln(K)$. Il est nécessaire ensuite de calculer une intersection entre deux ensembles qui peut être faite en $K * ln(K)$.

3.5 Tableau des nœuds de chaque couleur

Dans chaque nœud, on stocke un tableau de taille K de `short` contenant l'identifiant d'un B-voisin. Ce tableau contient le B-voisin à qui envoyer un message lorsque le nœud considéré ne possède pas l'information nécessaire pour router le message vers la destination. D'après [AGM+08], n'importe quel nœud de la boule de voisinage dont la couleur est égal au hashé de la destination peut router le message. Le choix est donc libre dans l'implémentation : dans notre cas, nous avons choisi de router le message vers un nœud de la « bonne » couleur qui a le degré le plus élevé. L'idée ici est de ne pas envoyer un message à un nœud de faible degré car il se trouve à la « périphérie » du graphe GLP.

3.6 Description de la fonction de routage

Étant donné la description du schéma de routageAGMaNT, il est nécessaire d'utiliser différents en-têtes pour les messages suivant les cas de routage dans lesquels on se situe :

a Le nœud destination est un voisin du nœud source : aucun en-tête n'est nécessaire.

b Le nœud destination est un landmark : aucun en-tête n'est nécessaire.

c Le nœud destination a une entrée dans la table 3a du nœud source : il faut un en-tête pour indiquer que le routage s'effectue dans l'arbre du landmark le plus proche du nœud destination. Cet en-tête doit contenir l'identifiant du landmark, l'identifiant DFS de la destination et son cpath.

d Le nœud destination a une entrée dans la table 3b du nœud source : il faut un en-tête pour indiquer qu'il faut passer par w, x, y et les informations de routage dans les arbres de u et de y.

e' Le nœud destination n'a aucune entrée dans les tables du nœud source. Il faut envoyer le message à un nœud qui possède ces informations. Il faut donc indiquer dans un en-tête quel nœud, qui possède ces informations et qui se trouve dans la boule de voisinage du nœud source, a été choisi.

- Les en-têtes des routages c et d sont également utilisés quand le message a atteint le nœud qui possède les informations de routage dans le routage e pour router le message vers la destination. On notera ces cas c' et d'.

Les cas a, b, c et d correspondent respectivement aux routages a, b, c et d. Le cas e', quant à lui, correspond à la première étape du routage e.

3.6.1 Cas a : routage a

Lorsqu'un nœud reçoit un message, il commence tout d'abord par vérifier si la destination n'est pas dans sa boule de voisinage. Si c'est le cas, le message peut être envoyé suivant un plus court chemin vers la destination.

3.6.2 Cas b : routage b

Si la destination n'est pas un de ces B-voisins, le nœud vérifie si la destination est un landmark ou non. Si c'est un landmark, de la même manière, le message peut être envoyé suivant un plus court chemin.

Si on ne se trouve pas dans ces cas, le message doit avoir un en-tête qui indique la manière dont le message doit être routé. Il faut alors examiner cet en-tête.

3.6.3 Cas e' : première étape du routage e

Si le message contient l'en-tête correspondant à la première partie du routage e, celui-ci a la forme donnée par le *tableau 10*. Deux cas se présentent : le nœud qui reçoit ce message est le nœud spécifié dans l'en-tête ou ce n'est pas le nœud spécifié dans l'en-tête. Dans le second cas, le nœud a pour B-voisin (voir [AGM+08]) le nœud spécifié dans l'en-tête : il utilise alors la table 1 pour transférer le message suivant un plus court chemin vers ce nœud. Si le nœud qui reçoit le message est celui qui est spécifié dans le

message, il a les informations de routage nécessaires pour envoyer le message au nœud destination soit dans sa table 3a soit dans sa table 3b. Il crée un nouvel en-tête pour le message (cas c' et d').

id du B-voisin à atteindre

TABLE 10 – En-tête utilisé dans la première étape du routage e

3.6.4 Cas c ou c' : routage c et seconde étape du routage e

Si le message contient l'en-tête correspondant au cas c ou c', celui-ci a la forme donnée par le *tableau 11*.

id du landmark	id DFS de la destination	cpath de la destination

TABLE 11 – En-tête utilisé dans les cas c et c'

Le nœud utilise le schéma de routage dans les arbres décrit dans [FG01] pour router le message. L'arbre qui doit être utilisé est l'arbre de plus court chemin dont la racine est le landmark désigné dans l'en-tête. Les autres informations nécessaires sont stockées dans l'en-tête (l'identifiant dans le DFS de la destination et le cpath de la destination) et dans l'entrée de la table 2 correspondant au landmark désigné dans l'en-tête.

3.6.5 Cas d ou d' : routage d et seconde étape du routage e

Si le message contient l'en-tête correspondant au cas d ou d', celui-ci a la forme donnée par le *tableau 12* (en reprenant les notations de l'article et utilisées dans la construction de la table 3b).

id de la source intermédiaire	id DFS de w	cpath de w	id de x	id de y	lien de x à y	id DFS de v	cpath de v	step

TABLE 12 – En-tête utilisé dans les cas d et d'

À part l'identifiant de la source intermédiaire et le step, les autres données proviennent de l'entrée de la table 3b du nœud qui a créé cet en-tête. Le nœud qui a créé cet en-tête peut soit être le nœud source (cas c) soit un nœud de la « bonne » couleur qui a été choisi par le nœud source (cas c'). Ce que l'on appelle identifiant de la source intermédiaire correspond donc soit à l'identifiant de la source (cas c) soit à l'identifiant du nœud de la « bonne » couleur (cas c'). Le champ step sert à indiquer à quelle étape du routage le message se trouve actuellement (routage vers w dans l'arbre de la source intermédiaire, routage de w à x, routage de x à y et routage dans l'arbre de y vers la destination v).

Quand un nœud reçoit un message avec un en-tête de ce type-là, il identifie tout d'abord grâce au champ step à quelle étape du routage le message se trouve. Si le message doit être routé vers w dans l'arbre du nœud source intermédiaire, il utilise les informations de l'en-tête (id DFS de w et cpath de w) et les informations de sa table 1 concernant la source intermédiaire. Si le message doit être routé de w à x, le nœud utilise sa table 1 pour envoyer le message vers x (qui se trouve dans sa boule de voisinage). Si le message doit être routé de x à y (qui sont voisins), le nœud x envoie le message au nœud y directement. Si le message doit être routé de y à v dans l'arbre de y, le message est routé grâce aux informations de l'en-tête (id DFS de v et cpath de v) et des informations présentes dans la table 1 correspondant à l'entrée concernant y.

4 Expérimentations

Plusieurs paramètres sont laissés libres dans le schéma décrit dans l'article [AGM+08] et d'autres peuvent être modifiées sans modifier les propriétés du schéma de routage.

4.1 Construction des boules de voisinage

La construction des boules de voisinage présentée dans l'article propose des boules de voisinage de taille fixe. Si le tirage des couleurs est aléatoire, la taille des boules de voisinage proposée dans l'article assure que ces boules de voisinage contiennent un nœud de chaque couleur avec une très forte probabilité. Cependant la taille des boules de voisinage proposée est assez importante et, par conséquence, le nombre d'entrées dans la table 1 pour chaque nœud est assez conséquente. Il est possible de réduire la taille de ses boules de voisinage en les construisant de manière à s'arrêter dès que la propriété de couleurs est vérifiée. C'est le choix qui a été fait dans notre implémentation. Il y a ici un compromis entre le nombre d'entrées et les performances en terme d'étirement de l'algorithme : plus les boules de voisinage sont grandes, plus le nombre d'entrées dans la table 1 est grand et également plus le nombre de routage en plus court chemins est important.

4.2 Choix de l'algorithme de coloration

Le choix de la coloration des nœuds est également un paramètre qui peut avoir une influence sur certains paramètres. L'article [AGM+08] propose de colorier les nœuds de manière aléatoire. En choisissant un algorithme différent de coloration comme décrit précédemment (tri par degré), il est possible de réduire la taille des boules de voisinage de 3%. Cet algorithme de coloration n'est cependant pas réaliste à mettre en place dans un cadre réel. Vu son influence sur la taille des boules de voisinage, son influence sur l'étirement doit être assez minime. Il est éventuellement possible de colorier différemment les nœuds évidemment : une idée pourrait être de limiter le nombre de pairs de voisins ayant la même couleur par exemple, ce qui pourrait avoir un effet bénéfique à la fois sur la taille des tables de routage (réduction de la taille de la table 1) et sur l'étirement (les nœuds de couleur différente pourraient être plus proches ce qui limiterait le détour causé par la première étape du routage e).

4.3 Choix de la couleur des landmarks et du placement des landmarks

Le choix de la couleur des landmarks est également un paramètre libre qui est fortement relié au choix de l'algorithme de coloration. Pour l'algorithme de coloration aléatoire ou celui utilisé dans l'implémentation actuelle, le choix de la couleur des landmarks ne doit pas avoir d'influence sur l'étirement.

Par contre, il serait intéressant d'évaluer si le choix de la position des landmarks dans le graphe ne pourrait pas avoir une influence sur l'étirement. Dans le cadre d'un graphe GLP, il serait par exemple intéressant de placer un ou deux landmarks dans le tier-1 et le reste des landmarks dans le tier-2 et aucun landmark dans le tier-3 et d'évaluer l'influence sur les tables de routage de ce choix (nombre d'entrées dans la table 3a par rapport au nombre d'entrées dans la table 3b) et sur l'étirement.

4.4 Choix du nombre de couleurs

Le schéma de routage est paramétré par le nombre de couleurs possibles. Ce paramètre va jouer sur la taille des boules de voisinage et ainsi influencer le nombre d'entrées dans les tables et également l'étirement. Le choix qui a été fait est de choisir ce paramètre de telle manière à ce que le nombre d'entrées soit minimal.

Une détermination théorique approximative de *l'espérance du nombre d'entrées* en fonction du nombre de couleurs K permet d'établir la formule : $KH_K + \frac{2N}{K}$ où $H_K = K\ln K + \gamma + o(1)$ avec $\gamma = 0.57..$, H_K étant le K-ième nombre harmonique. Ce résultat est valide pour pour une affectation aléatoire uniforme des couleurs.

Ainsi, le minimum théorique approximatif pour $N = 10000$ est pour $K = 64$. Nous avons également cherché à évaluer cette valeur de K de manière pratique en considérant l'affectation des couleurs en fonction des degrés.

Pour cela, nous avons utilisé 50 graphes GLP de 10000 nœuds avec les paramètres suivants :
– number of edges per step = 1,15

– beta = 0,6753
– number of initial nodes = 6
– step probability = 0,4669

Pour chacun de ces graphes, nous avons évalué le nombre moyen d'entrées par noeud pour des valeurs de K variant de 30 à 95 puis nous avons fait la moyenne sur tous les graphes pour chacune des valeurs de K.

La coloration des nœuds des graphes n'est pas aléatoire mais effectuée après avoir trié les nœuds par degré.

On obtient la courbe présentée *figure 7*. La courbe obtenue grâce aux simulations est très proche de la courbe théorique, les valeurs étant très légèrement supérieures dans le cadre des simulations.

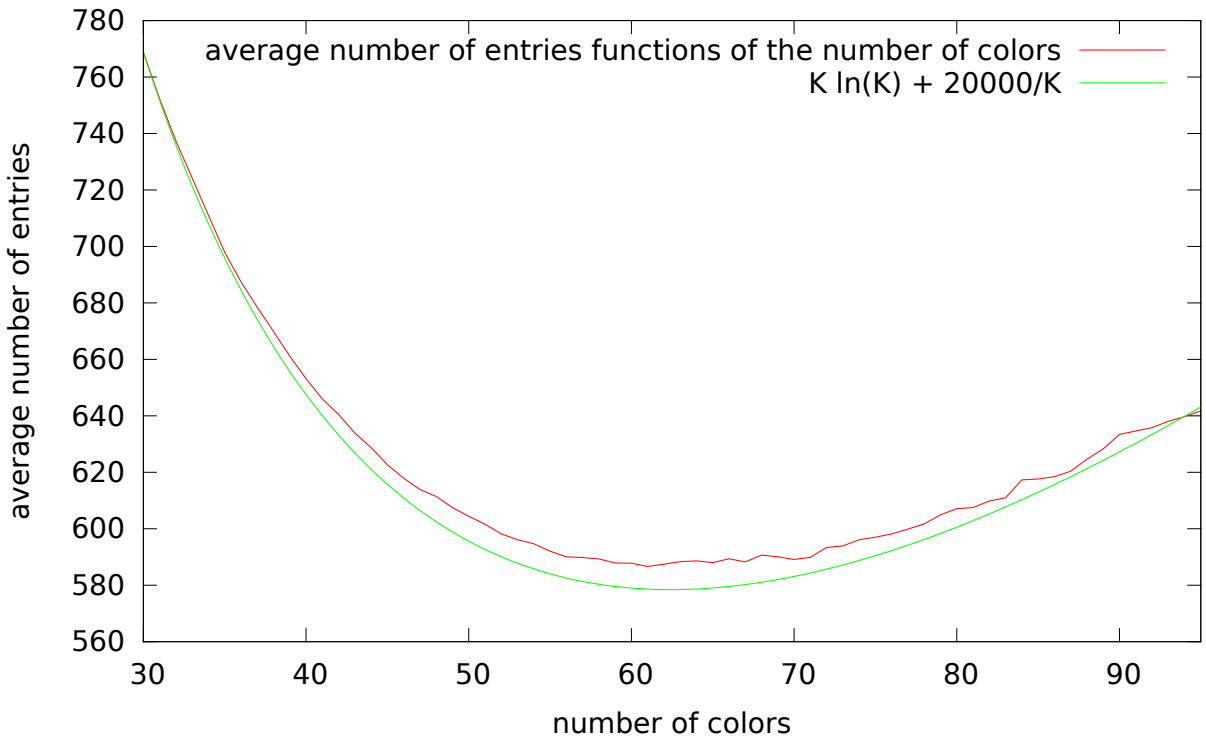


FIGURE 7 – Nombre d'entrées moyen par nœud en fonction du nombre de couleurs K

Le minimum obtenu par les simulations est pour $K = 61$, ce qui est proche de la valeur théorique. La courbe présente de plus un plateau entre $K = 60$ et $K = 65$ donc nous avons choisi de mener les expérimentations avec la valeur théorique de $K = 64$.

Influence du choix de k sur le stretch ? Nous avons mesuré sur des graphes de taille moins importante (GLP 5000 sommets) l'impact du choix du nombre de couleurs sur le stretch. Il en ressort que le stretch est bien moins influencé par ce choix que ne l'est la taille des tables (voir figures 8 et 9).

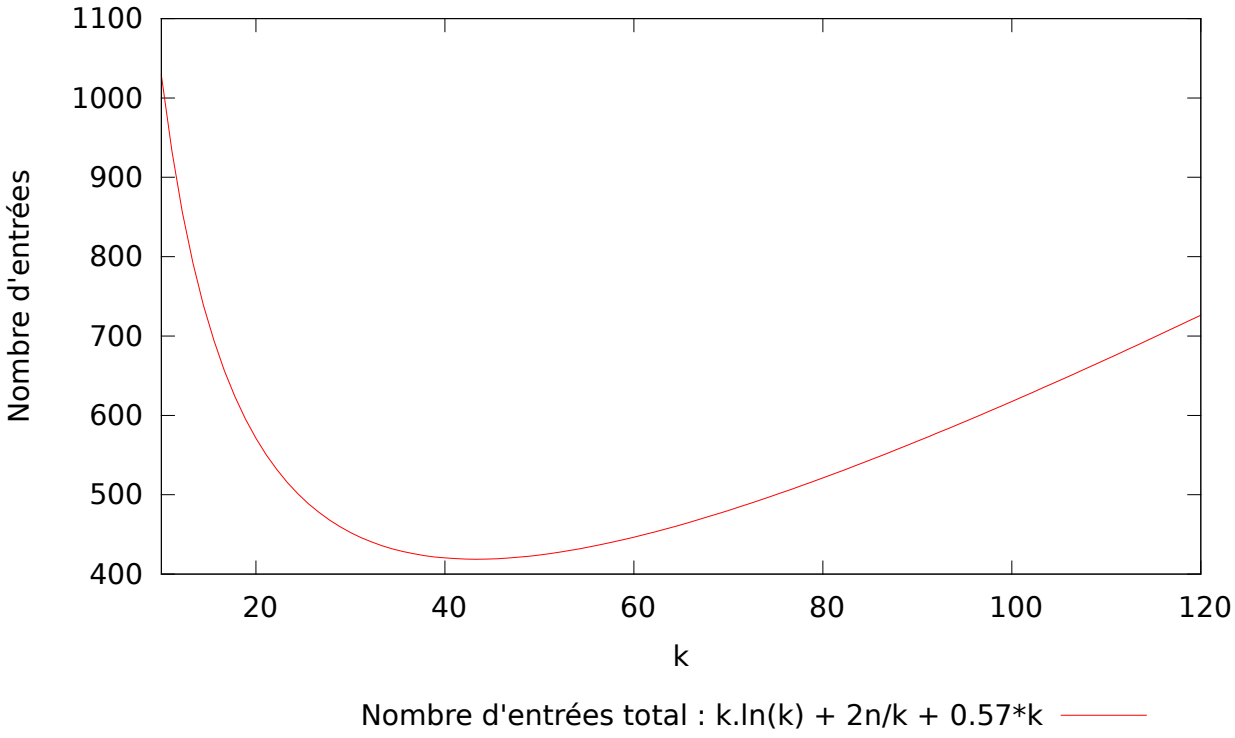


FIGURE 8 – mémoire en fonction de k

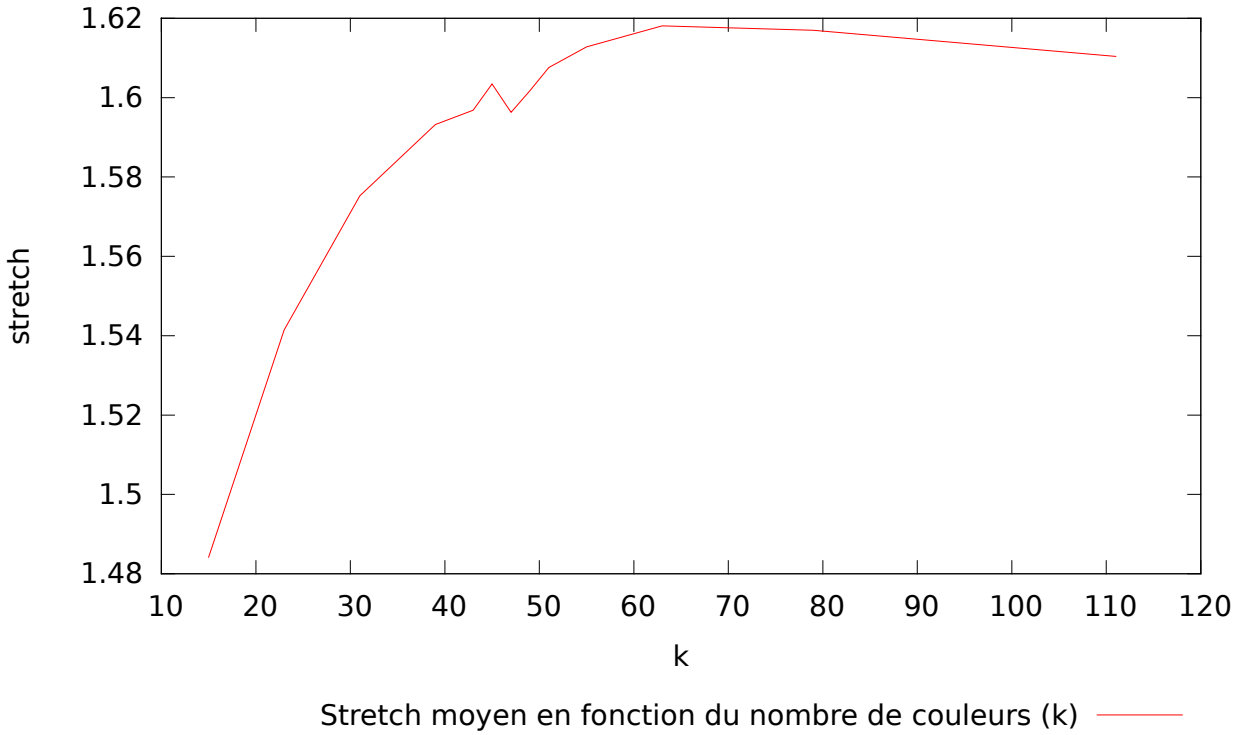


FIGURE 9 – stretch en fonction de k

Pour CAIDA ? Le même calcul théorique est utilisé pour les expériences sur les graphes de type CAIDA (cartographie des AS). Cependant pour des limitations de temps, le calcul expérimental du k optimal n'a pas été fait pour CAIDA. Il est néanmoins réaliste de supposer que les valeurs théorique et expérimentales de k seront également proches car les graphe suivant le modèle GLP ont des propriétés proches du graphe des AS. Pour le réseau des AS datant de 2004 on obtient ainsi $k = 79$

4.5 Choix du nœud de la « bonne » couleur pour le routage e

Le dernier paramètre laissé libre par le schéma décrit dans l'article est le choix du nœud de la « bonne » couleur lors de la première étape du routage e. Sachant qu'en sélectionnant une paire de nœuds différents dans le graphe, la probabilité que le choix de cette paire entraîne un routage e entre le premier nœud et le second est d'environ $\frac{(K-2)\times n - K^2 \times ln(K)}{K\times(n-1)}$. Pour les paramètres utilisés dans nos expérimentations et notre implémentation, cela donne une probabilité d'environ 0,9422. Ce qui signifie que dans un scénario de routage de type « all to all », environ 94,22% des routes sont des routages e. Ceci montre que le choix de ce nœud **peut** avoir une influence importante sur les performances de routage.

Il est possible d'envisager de multiples algorithmes pour déterminer ce choix. L'algorithme utilisé dans l'implémentation actuelle consiste en la sélection du nœud de plus fort degré de la boule de voisinage pour chaque couleur, c'est-à-dire : considérons un nœud u du graphe, pour chaque couleur, u va stocker l'identifiant d'un de ses B-voisins de cette couleur qui sera le relai lors de la première étape du routage e ; le B-voisin choisi sera le B-voisin de cette couleur dont le degré est le plus grand (en cas, d'égalité, le choix est aléatoire entre les B-voisins de plus fort degré).

L'idée derrière cet algorithme est de choisir comme relai lors de la première étape d'un routage e un nœud proche du centre d'un graphe de type GLP et espérer ainsi « partir dans la bonne direction ». De nombreuses autre variante peuvent être proposées : choisir le nœud le plus proche, choisir aléatoirement. Le choix pourrait même être éventuellement dynamique après l'envoi de messages sondes à tous ses B-voisins de la couleur donnée.

5 Analyse des résultats

5.1 Tables

Ayant fixé le paramètre K (nombre de couleurs), il est possible maintenant d'évaluer plus précisément le nombre d'entrées dans les tables de chaque nœud et regarder la répartition des entrées notamment dans les tables 3a et 3b.

Pour cela, nous avons utilisé le même protocole de simulations que précédemment : nous avons utilisé 50 graphes GLP de 10000 nœuds avec les mêmes paramètres que dans la section précédente. Pour chaque graphe, nous avons construit de manière statique toutes les tables dans chaque nœud.

5.1.1 Nombre d'entrées globales

La première chose que nous souhaitons évaluer est le nombre d'entrées globales, c'est-à-dire la somme du nombre d'entrées pour les 4 tables (tables 1, 2, 3a et 3b). La *figure 25* présente le pourcentage du nombre de nœuds qui ont un nombre d'entrées donné.

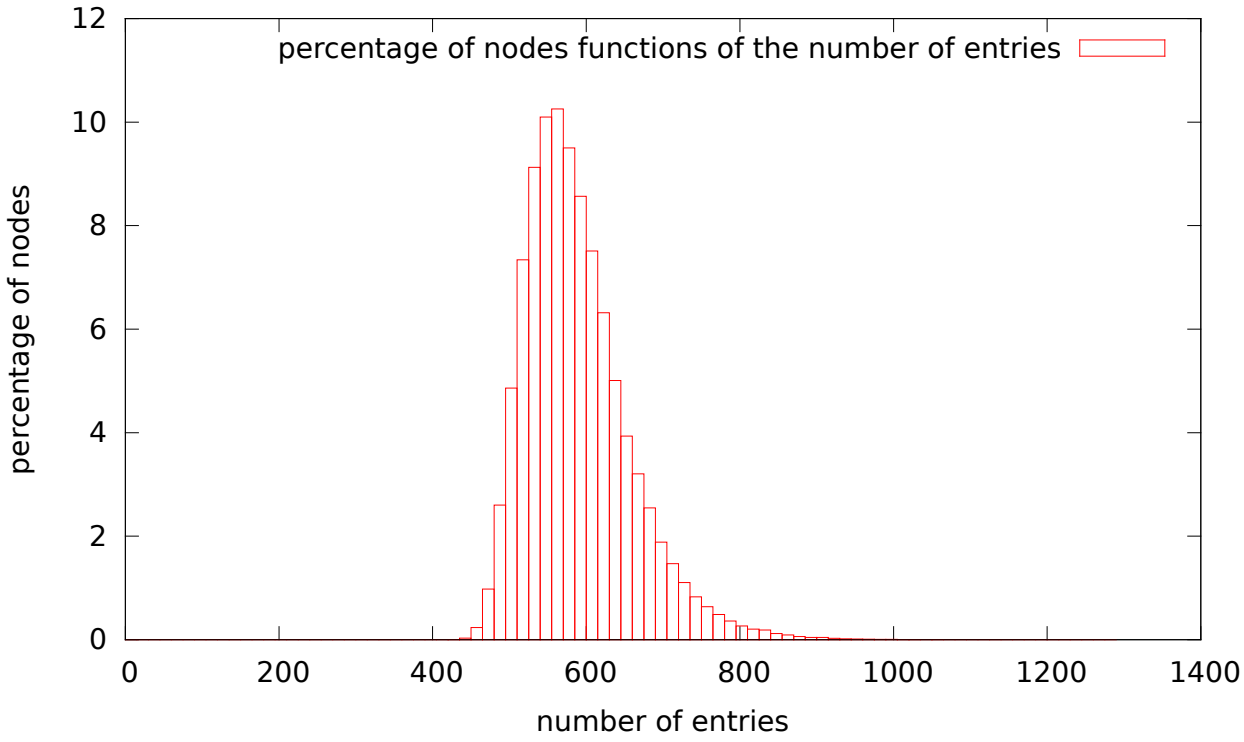


FIGURE 10 – Pourcentage de nœuds qui ont un nombre d'entrées donné

La grande majorité des nœuds ont un nombre d'entrées proches de la valeur théorique qui est environ 580. Cependant la variabilité est assez importante car certains nœuds ont relativement moins d'entrées, autour de 450 entrées et surtout d'autres nœuds ont beaucoup plus d'entrées : jusqu'à 1,5 fois la valeur moyenne. On peut alors se poser la question de la variabilité de nombre d'entrées. Le nombre d'entrées dans la table 2 est exactement $\frac{N}{K}$ et la somme du nombre d'entrées dans les tables 3a et 3b est majoré également par $\frac{N}{K}$. La partie variable est en fait lié aux nombre d'entrées dans la table 1.

On a également voulu vérifier l'influence du degré sur le nombre d'entrées de chaque nœud. Pour cela, on a fait la moyenne pour chaque simulation du nombre d'entrées de tous les nœuds ayant un degré donné, puis la moyenne de ces moyennes pour les 50 graphes GLP. La *figure 28* présente les résultats obtenus. On peut observer qu'il n'y a pas de corrélation entre le degré d'un nœud et son nombre d'entrées. Les nœuds de petit degré qui sont les plus nombreux dans

les graphes GLP ont des valeurs moyennes du nombre d'entrées proches de la moyenne attendue. La variabilité est plus grande pour les nœuds de fort degré car ces derniers sont moins fréquents.

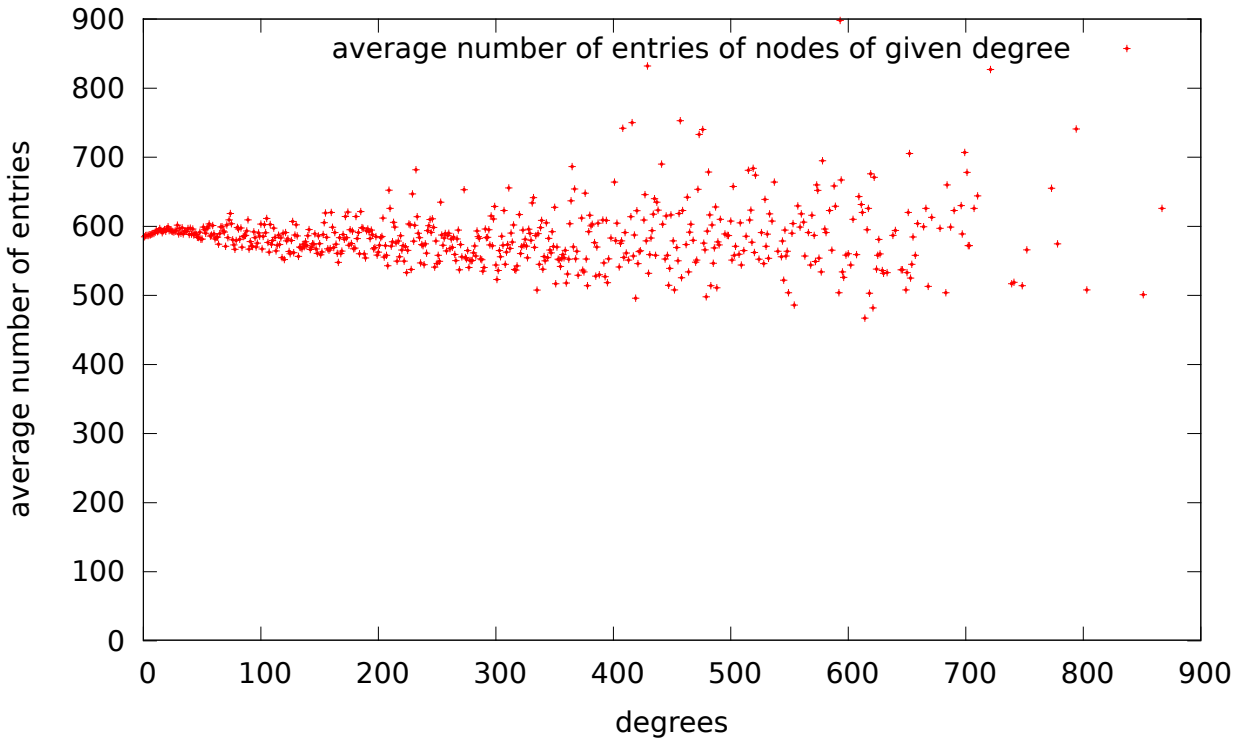


FIGURE 11 – Nombre moyen d'entrées des noeuds d'un degré fixé

5.1.2 Nombre d'entrées dans la table 1 : taille des boules de voisinage

Dans cette partie, on a cherché à évaluer le nombre d'entrées dans la table 1 pour chaque nœud, ce qui revient à évaluer la taille des boules de voisinage. La *figure 26* montre donc le pourcentage de nœuds ayant un nombre donné d'entrées dans la table 1 en moyenne sur les 50 graphes GLP. Cette courbe est très semblable à la courbe du nombre global d'entrées, ce qui correspond bien aux attentes car le nombre d'entrées dans les autres tables doit être pratiquement constant.

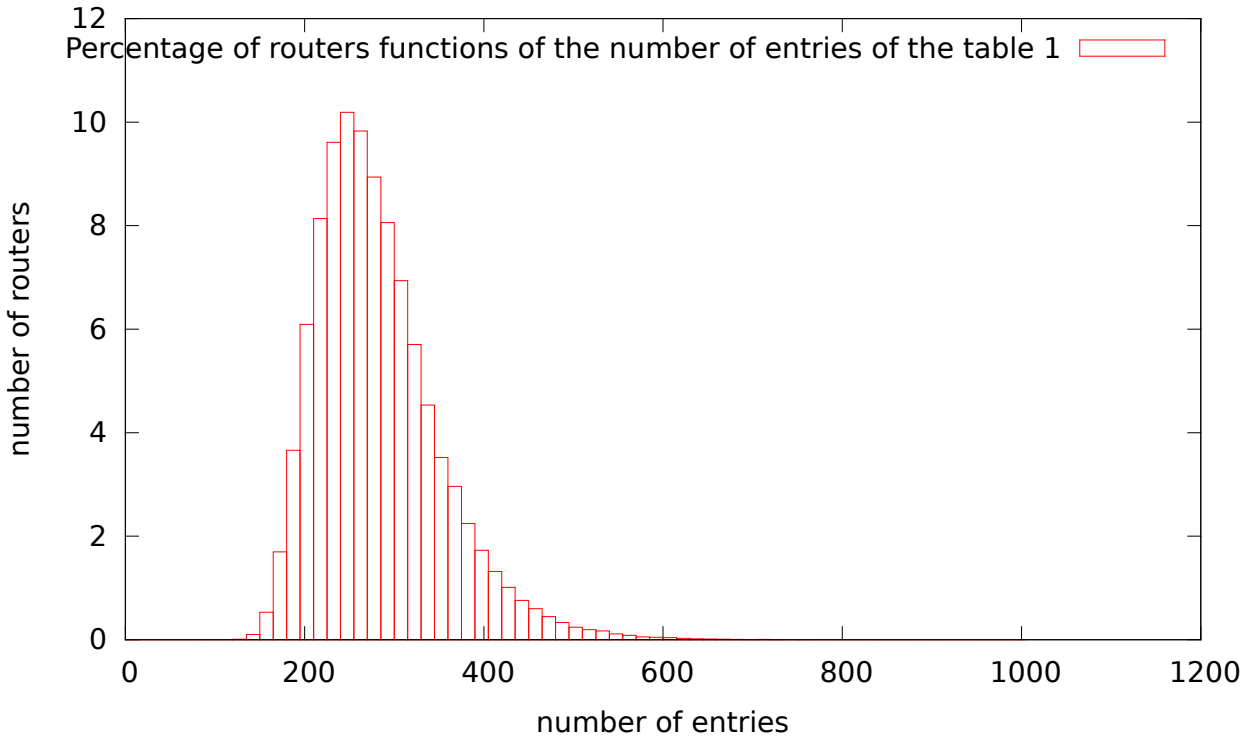


FIGURE 12 – Pourcentage de nœuds qui ont un nombre d'entrées donné

De la même manière que pour le nombre d'entrées globales, on peut étudier le nombre d'entrées moyen en fonction du degré des nœuds pour la table 1. La *figure 13* montre cette répartition. On peut faire les mêmes conclusions que dans le cas précédent. Le degré n'a pas d'influence sur le nombre d'entrées dans la table 1. La variabilité est plus importante pour les nœuds qui ont un degré élevé car ces nœuds sont peu nombreux malgré les 50 graphes utilisés.

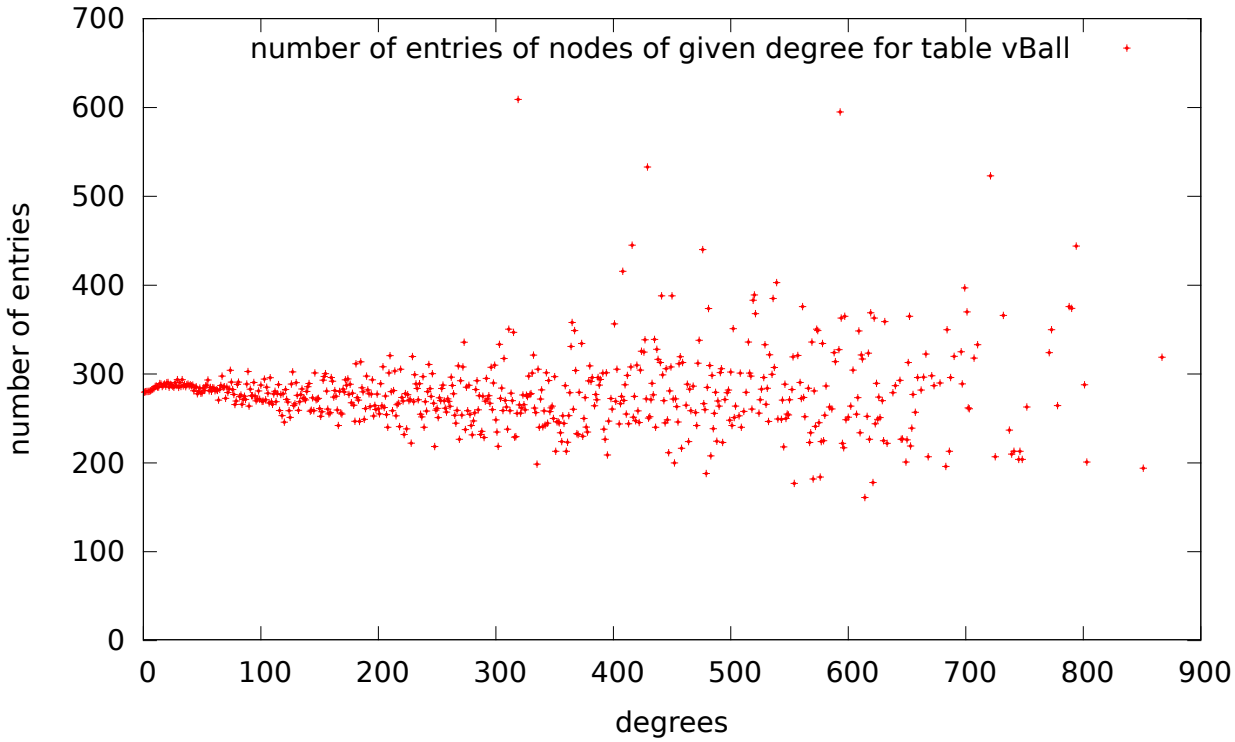


FIGURE 13 – Nombre moyen d'entrées dans la table 1 des nœuds d'un degré fixé

5.1.3 Nombre d'entrées dans la table 2

La table 2 contient les entrées servant à router dans les arbres relatifs au landmark. Le nombre de landmarks étant fixé (il est égal à $\frac{N}{K}$), le nombre d'entrées dans ces tables est constant pour tous les nœuds, c'est ce que l'on peut vérifier sur la *figure 14*.

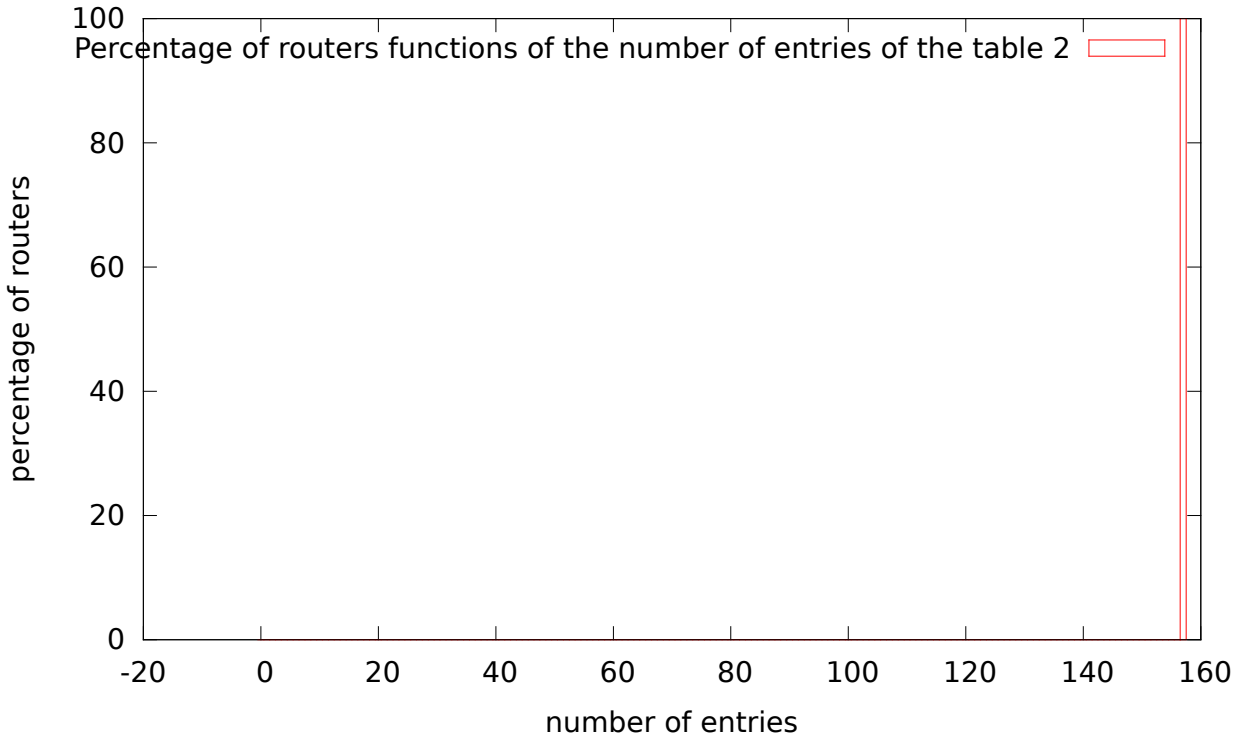


FIGURE 14 – Pourcentage de nœuds qui ont un nombre d'entrées donné dans la table 2

5.1.4 Nombre d'entrées dans les tables 3a et 3b

Il est plus intéressant d'étudier la répartition des entrées entre ces deux tables. Pour chaque nœud, la somme des entrées dans les tables 3a et 3b doit être de l'ordre de $\frac{N}{K}$ et a priori légèrement inférieur, étant donné la manière dont sont construites les tables.

Les figures 15 et 29 présentent le pourcentage de nœuds qui ont le nombre d'entrées données respectivement dans la table 3a et la table 3b. La remarque la plus importante est que la table 3a qui correspond aux routes passant par le landmark le plus proche de la destination est très peu remplie, c'est-à-dire qu'il est très souvent possible de trouver des routes plus courtes passant par les boules de voisinage contigue dans les graphes de type GLP. Certains nœuds n'ont même aucune entrée dans la table 3a. Ceci s'explique par le fait que les graphes GLP sont des graphes très connectés avec un diamètre petit. On peut également obtenir des courbes qui montrent qu'il n'y a pas de corrélation entre le degré et le nombre d'entrées dans ces tables (ce qui est logique étant donné la construction de ces tables).

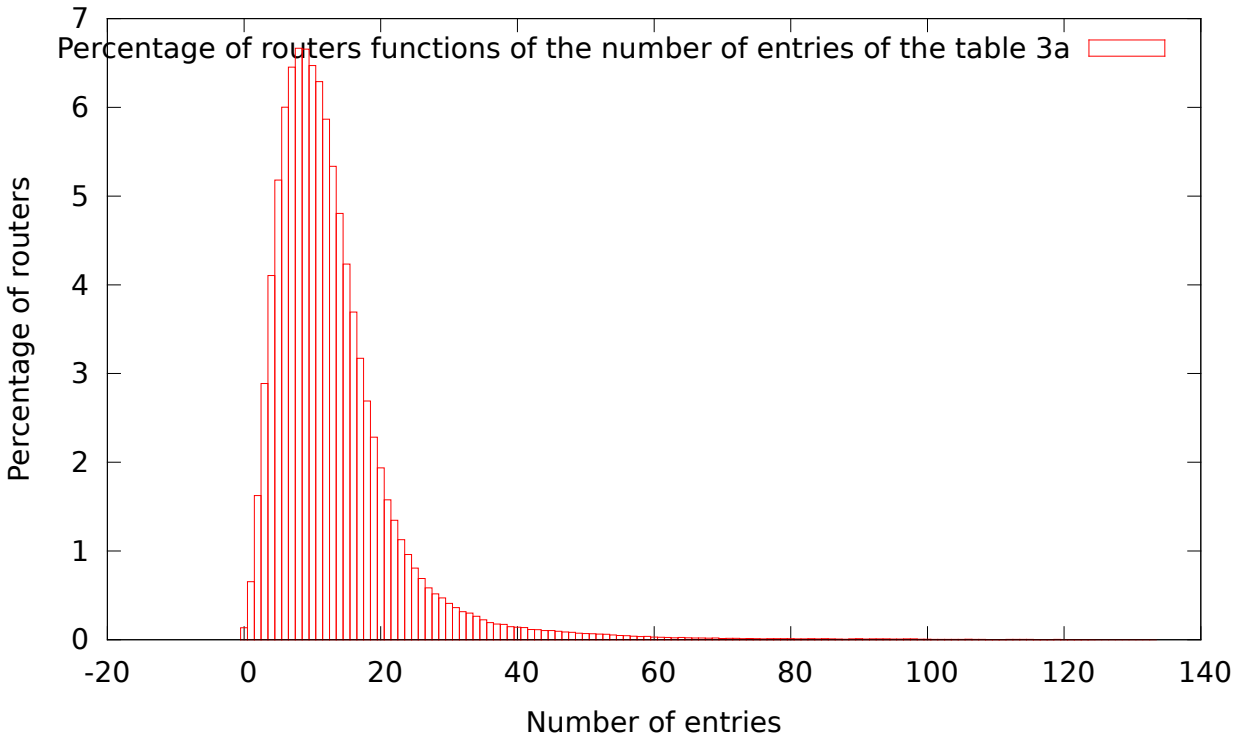


FIGURE 15 – Pourcentage de nœuds qui ont un nombre d'entrées donné dans la table 3a

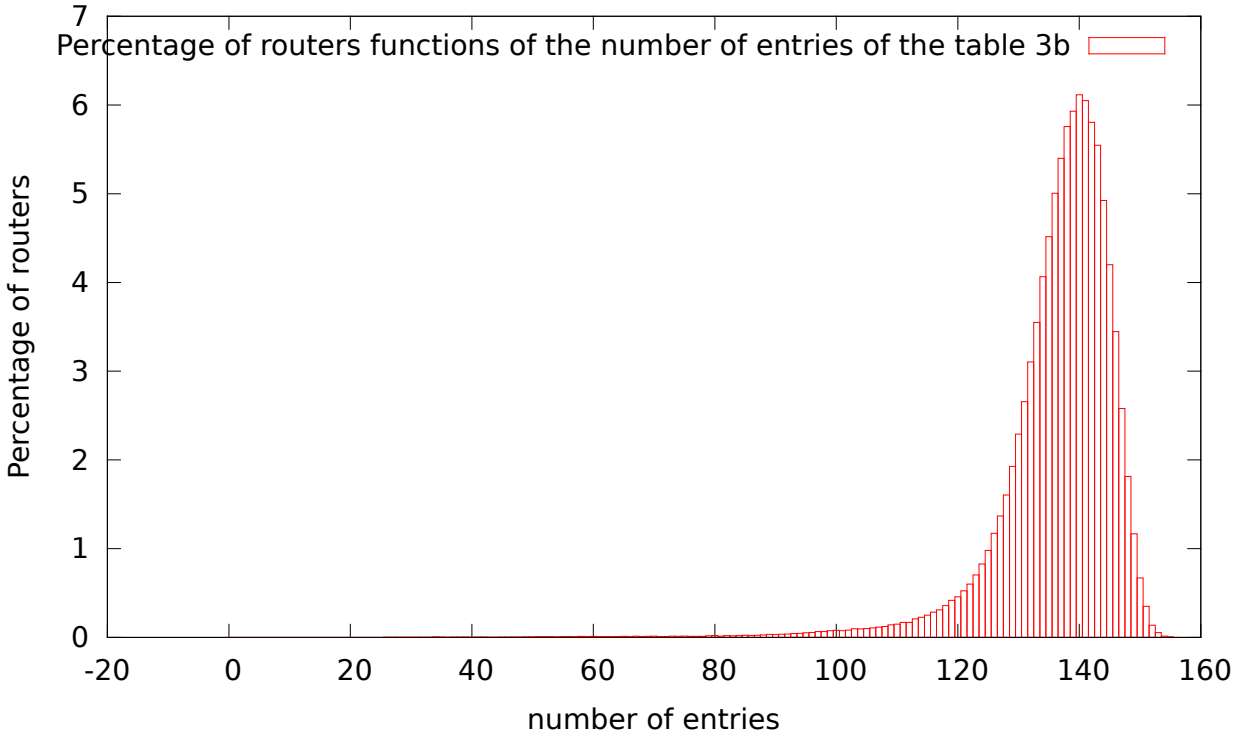


FIGURE 16 – Pourcentage de nœuds qui ont un nombre d'entrées donné dans la table 3b

5.2 Plus courts chemins et routes

Pour évaluer les performances de l'algorithme en terme de routage (longueur des routes, étirement multiplicatif et additif), nous avons utilisé 50 graphes GLP avec les mêmes paramètres que ceux définis précédemment et effectué un routage de tous les nœuds à tous les nœuds soit $n \times (n-1)$ paquets envoyés.

La *figure 35* présente la répartition des plus courts chemins en moyenne sur les 50 graphes (en vert) et la répartition des routes engendrées par le schéma de routage AGMNT (en rouge). Les lignes verticales représentent la moyenne des plus courts chemins (en vert) et la moyenne des routes (en rouge). Les valeurs sont respectivement 3,59 et 5,39. Ce que l'on peut remarquer sur cette figure, c'est que les courbes de répartition sont très semblables avec un décalage vers la droite pour les routes. Approximativement, on peut penser que le schéma de routage AGMNT ajoute 2 sauts supplémentaires par rapport au plus court chemin dans les graphes GLP utilisés.

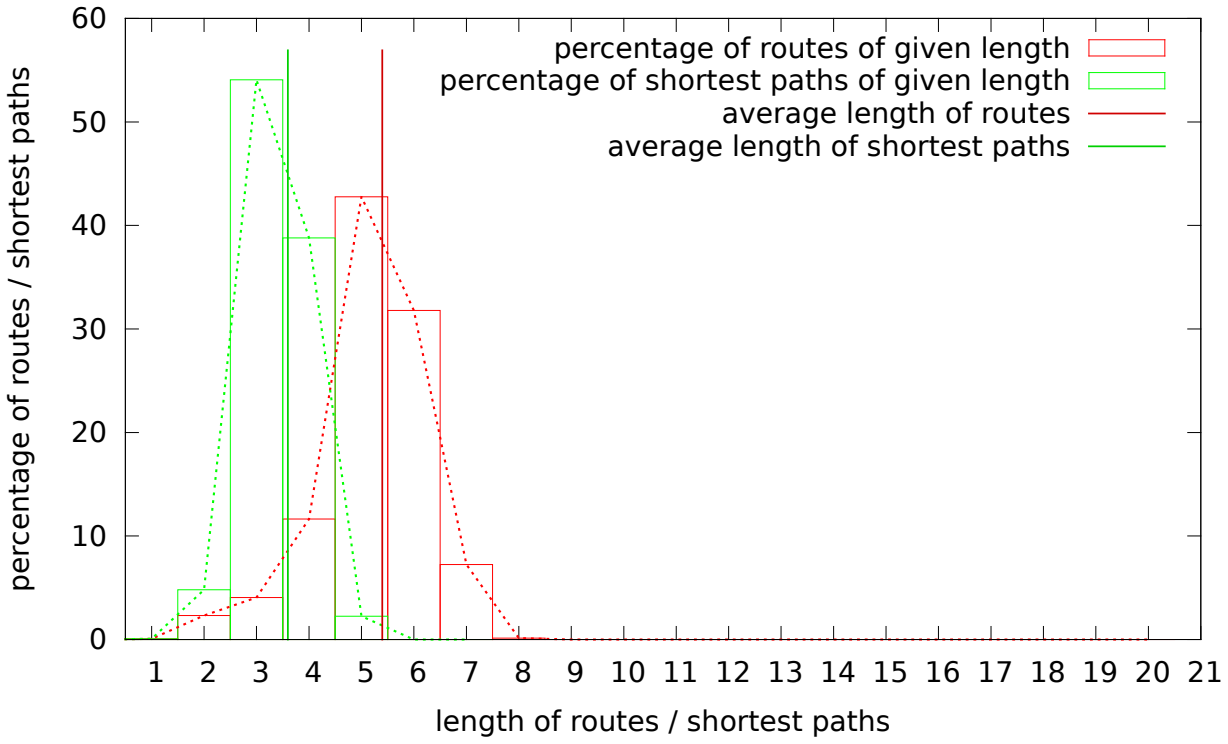


FIGURE 17 – Longueur des plus courts chemins et des route

La figure suivante 18 présente la longueur des routes utilisées en fonction de la longueur du plus court chemin. Par exemple, pour les plus courts chemins de longueur 1, environ 95% des routes sont réalisées sur le plus court chemin et environ 5% des routes sont réalisées avec un étirement additif de 2 (donc sont de longueur 3). Pour les plus courts chemins de longueur 2, 48% des routes utilisées sont optimales, 2% ont un étirement additif de 1, 24% un étirement additif de 2, 24% un étirement additif de 3 et 2% un étirement additif de 4. Pour les plus courts chemins de longueur 3, 7% des routes sont optimales, 14% ont un étirement de 1, 60% un étirement de 2, 17% un étirement de 3, 1% un étirement de 4 et des pourcentages très faibles pour les étirements de 5 et 6. On retrouve le même phénomère pour les plus courts chemins de longueur 4 et 5 : entre 55% et 60% des routes ont un étirement additif de 2. Ces courbes expliquent la ressemblance très forte des répartitions de la *figure 35*.

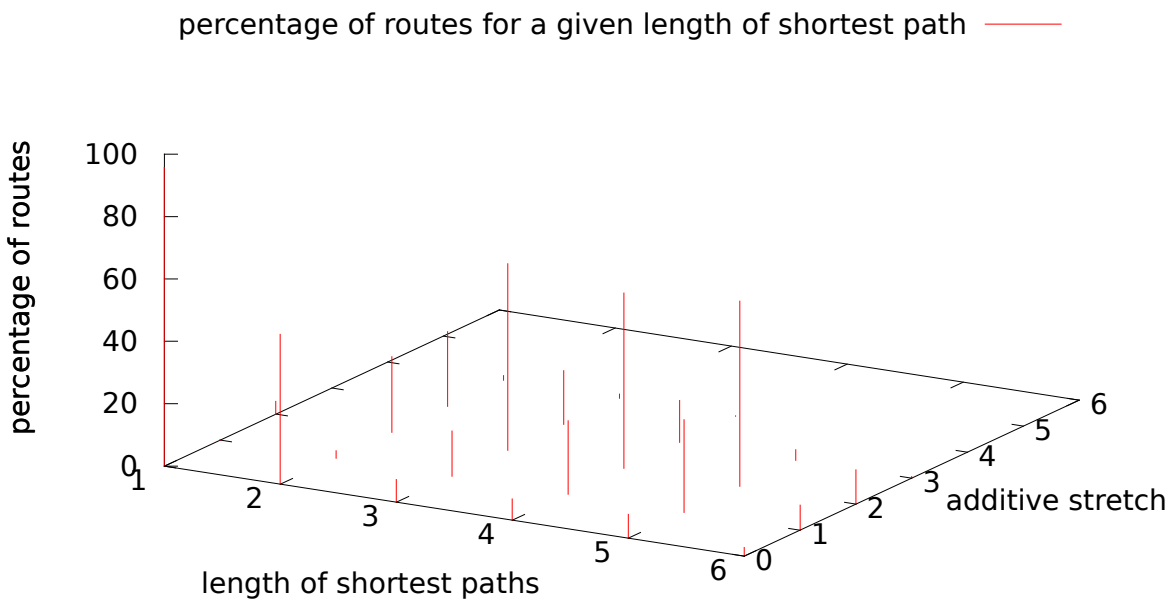


FIGURE 18 – Longueur des plus courts chemins et des routes

Les figures 19, 20 et 21 présentent un résultat synthétique concernant l'étirement additif et l'étirement multiplicatif. La première figure confirme bien qu'environ 60% des routes ont un étirement additif de 2. La figure suivante montre que 30% des routes ont un étirement multiplicatif autour de 1,6, ce qui correspond bien aux routes de longueur 5 pour lesquelles existent un plus court chemin de longueur 3. La dernière figure est intéressante car elle montre qu'environ 10% des routes sont optimales, 30% ont un étirement multiplicatif inférieur à 1,5. 85% des routes ont un étirement multiplicatif inférieur à 1.7 et 95% des routes ont un étirement multiplicatif inférieur à 2. Les résultats semblent inférieurs à ceux présenté par Krioukov en 2004 mais le schéma de routage AGMNT est *name-independent* et peut être encore certainement optimisé pour les graphes de type GLP.

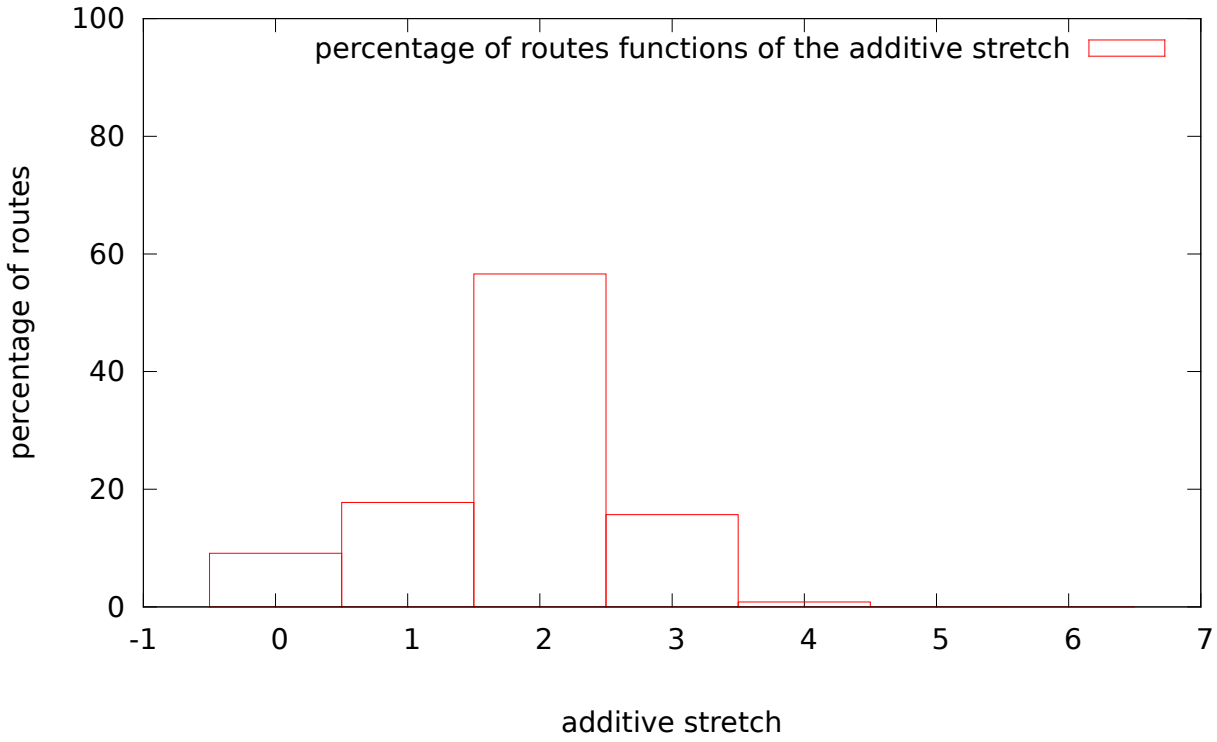


FIGURE 19 – Etirement additif

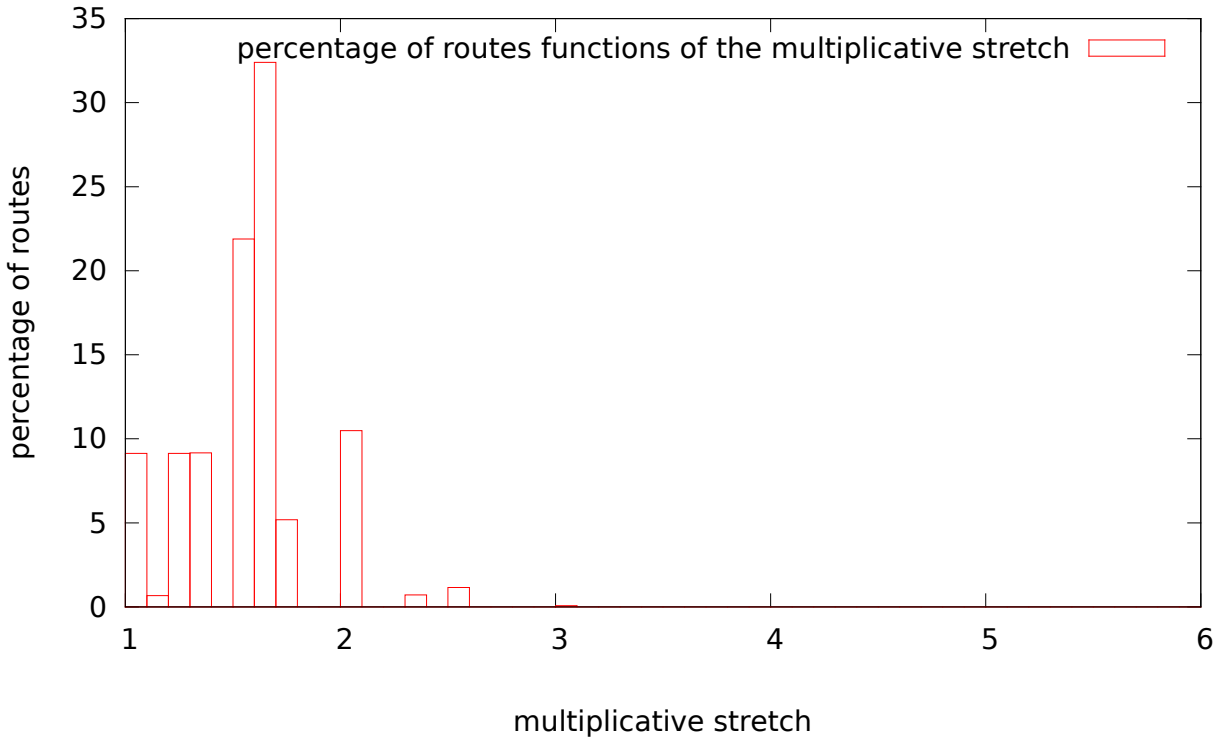


FIGURE 20 – Etirement multiplicatif

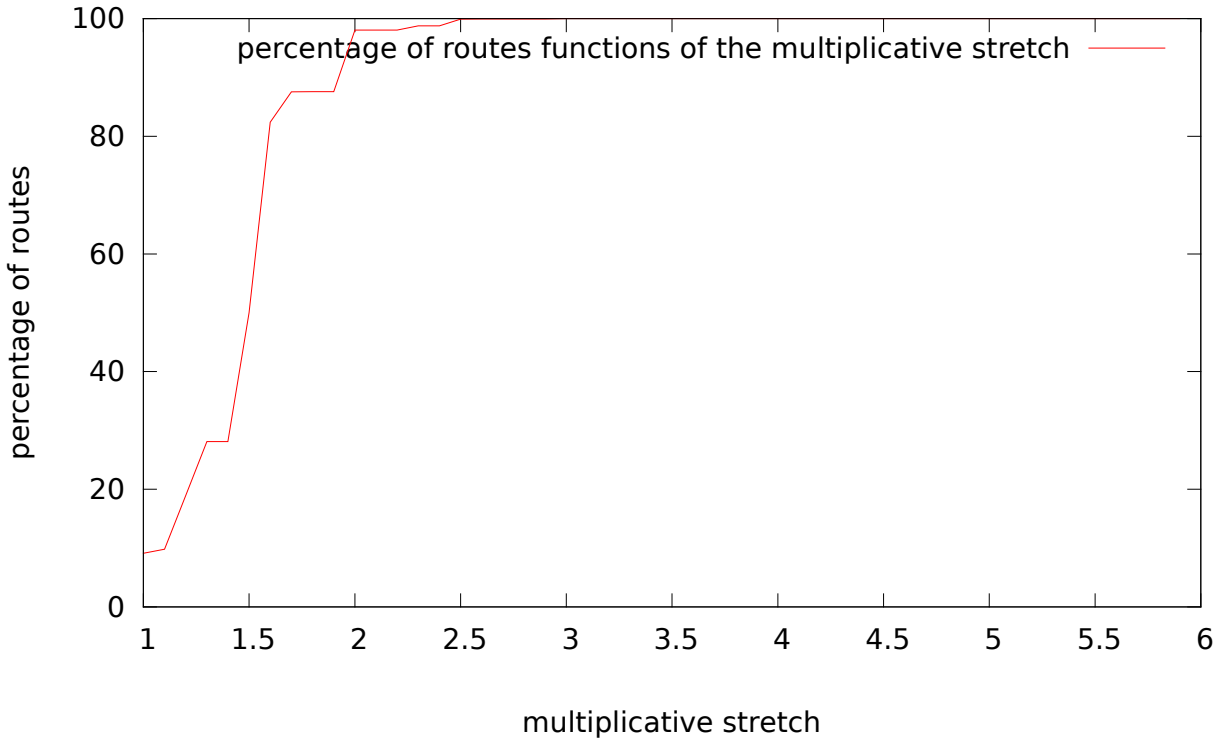


FIGURE 21 – Etirement multiplicatif

5.3 Charge des liens et des nœuds

À partir des mêmes simulations, nous avons évalué également la charge des nœuds et des liens de manière synthétique.

5.3.1 Charge des liens

La *figure 22* présente la moyenne des charges des liens sur les 50 graphes GLP. Pour chaque simulation, on mesure le nombre de paquets qui sont passés par chaque lien et ensuite on trie ces liens en fonction de ce nombre de paquets. On fait ensuite la moyenne pour chaque position dans le tri. Ce qu'il faut remarquer sur cette figure, c'est qu'une grande majorité des liens transmettent entre 1000 et 10000 paquets, qu'environ 15% des liens transmettent moins de 1000 paquets et qu'un très petit nombre transmettent beaucoup plus de paquets pour les autres (plus de 100000 paquets). Ces résultats sont un peu bruts et nécessiteraient une double comparaison :

- une comparaison avec d'autres graphes plus réguliers comme une grille par exemple. Cela permettrait de voir si le petit nombre de liens qui transmettent beaucoup de paquets est lié aux graphes GLP ou au schéma de routage AGMNT.
- une comparaison avec BGP ou d'autres schémas de routage sur des graphes GLP pour observer également si ce phénomène est du à AGMNT ou au graphe GLP.

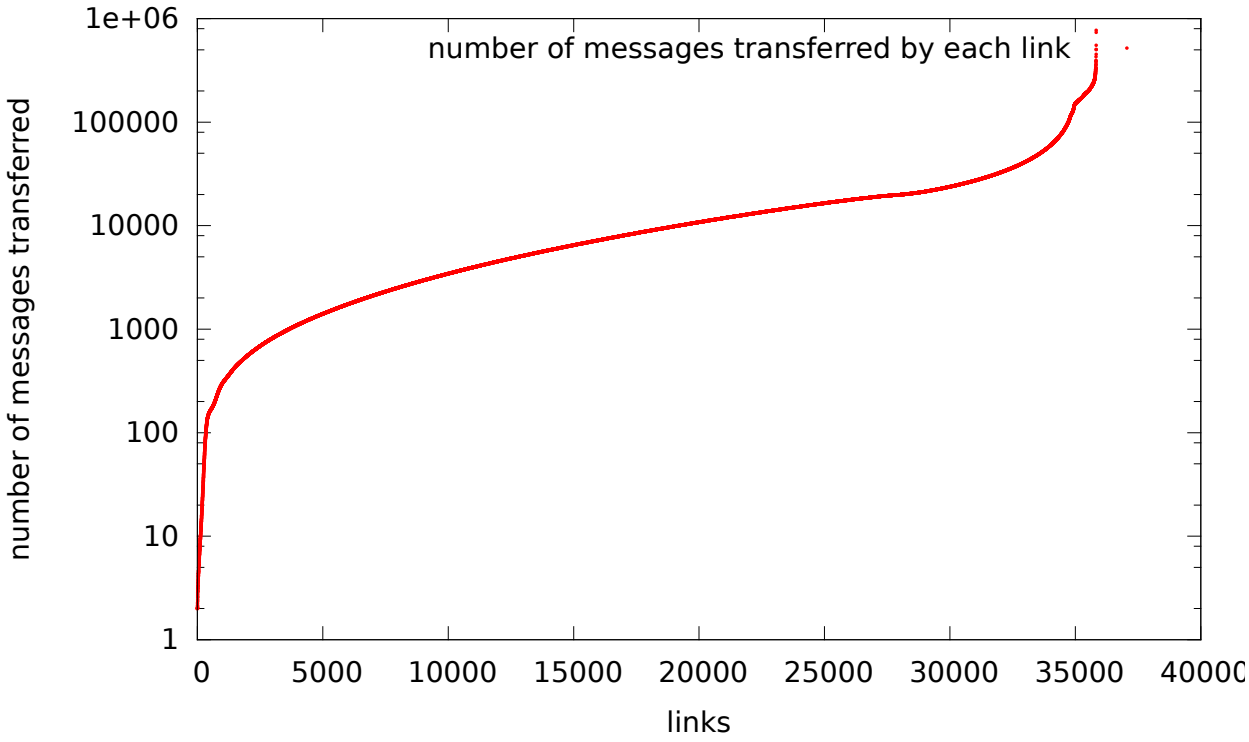


FIGURE 22 – Charge des liens

5.3.2 Charge des nœuds

La *figure 30* présente la charge des nœuds. Ces résultats ont été calculés de la même manière que dans le cadre de la charge des liens. On peut remarquer le même phénomène de pic pour les nœuds les plus chargés. Ce phénomène est certainement lié à celui que l'on peut observer pour les liens. Encore une fois, il est nécessaire de faire d'autres simulations en comparant AGMNT avec d'autres schémas de routage et en comparant AGMNT avec d'autres graphes plus réguliers.

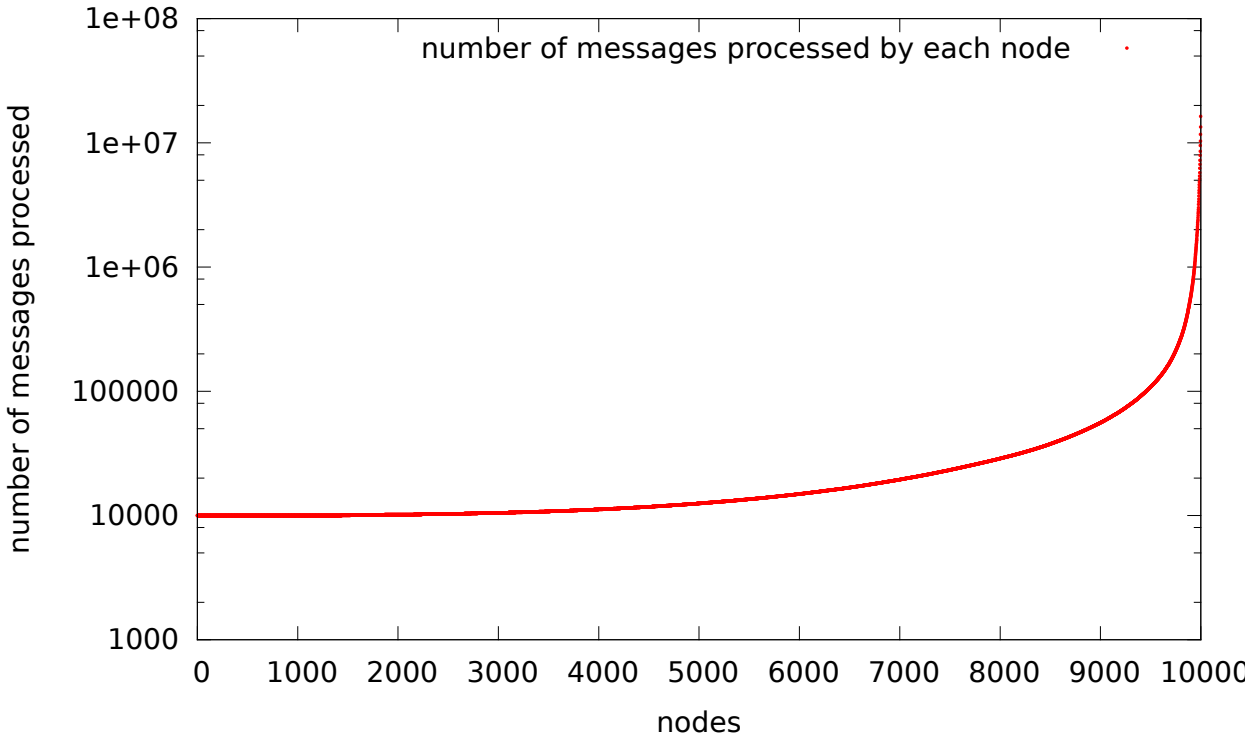


FIGURE 23 – Charge des nœuds

Nous avons également évalué la charge des nœuds en fonction de leur degré. Les résultats sont présentés sur la *figure 32*. Cette figure montre qu'il y a une corrélation entre le degré et la charge des nœuds en moyenne. De la même manière, il serait intéressant de comparer AGMNT avec d'autres protocoles sur des graphes GLP.

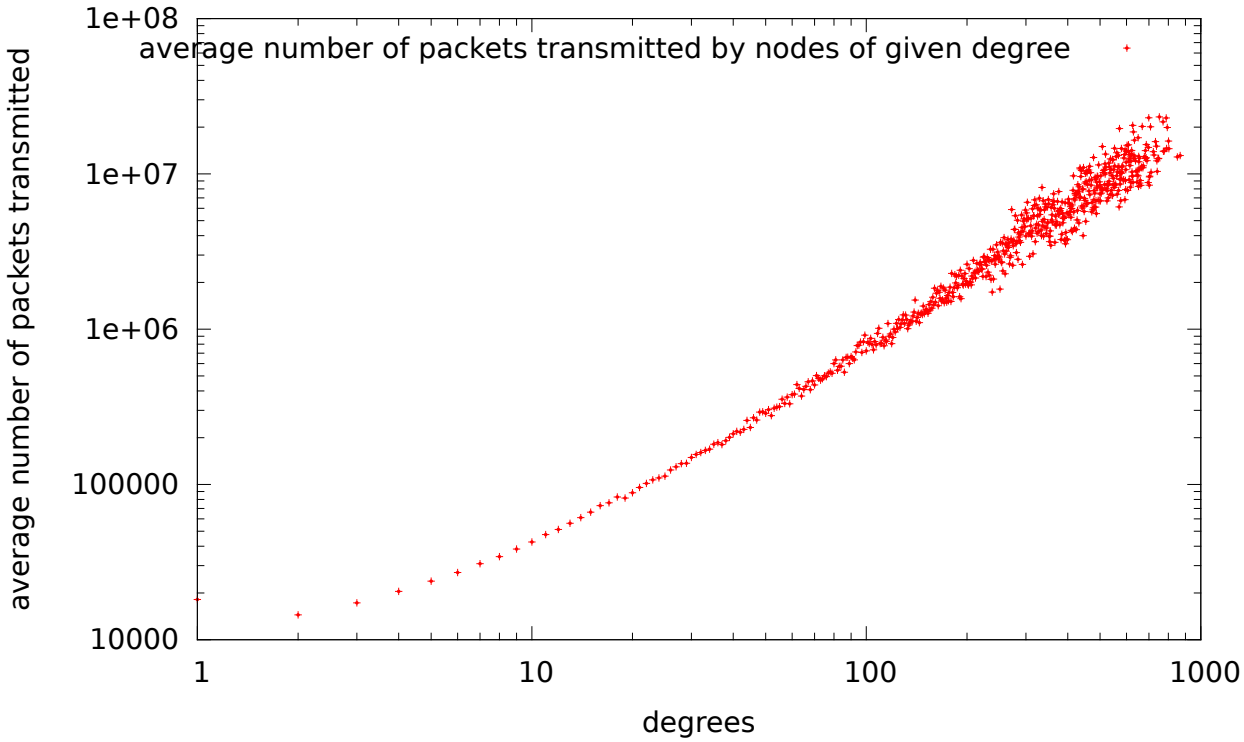


FIGURE 24 – Moyenne de la charge des nœuds par degré

6 Analyse des résultats sur une carte CAIDA

6.1 Données

La carte du réseau des AS datant de 2004, récupérée sur le site de l'association CAIDA [4] est utilisée pour cette expérience. Des cartes plus récentes existent, mais pour des raisons de limitation physique, nous avons du nous limiter à la plus petite map CAIDA existante (16301 nœuds et 36465 liens).

6.2 Machine utilisée pour les expérimentations

À cause de la taille du graphe, nous avons du pour cette simulation utiliser une machine avec une grosse capacité mémoire ($50Go$). La machine en question possède également 8 cœurs, mais notre algorithme n'avait pas été, à la base, fait pour fonctionner en parallèle (processeur : Intel(R) Xeon(R) CPU X5570 @2.93GHz, taille du cache : 8192 Ko). Durant les simulations, la consommation de RAM atteint un maximum de $36Go$ à la fin du processus de construction des tables.

6.3 Résultats - observations

Toutes les courbes sont présentées pour deux versions de l'algorithme, une avec coloration aléatoire et l'autre avec une coloration plus maline (cf. *sous-section 3.1.1*), dans les courbes, les deux versions seront nommées respectivement par les raccourcis "aléatoire" et "degré". Nous essayons pour ce graphe réaliste de mettre en avant les points qui restent à améliorer.

6.3.1 Tables de routage - Répartition de l'information et temps de construction

Globalement la répartition de la charge entre les nœuds en terme de mémoire (*figure 25*) est relativement acceptable comparativement à la taille du graphe mais reste trop importante comparativement à la taille des tables optimales. De plus, cette charge ne dépend pas du degré (*figure 28*), il est donc impossible de prévoir à l'avance quels nœuds du réseau auront de plus grandes quantités de donnés à stocker.

On observe que cette variation est due à la taille des tables de voisinage (*figure 26*), la taille des autres table ne variant que très peu (voir par exemple la *figure 29*). En superposant sur la *figure 27* pour la version de coloration "degré" le :

- nombre total d'entrés dans les différentes tables
- et le nombre de B-voisins auquel on ajoute le nombre d'entrées moyen dans les 2 autres tables ($|table2| = N/K$, $|table3| = N/K$)

,on peut voir assez clairement que l'unique problème est due à la table 1 et donc à la taille trop variable des boules de voisinage (le problème est le même pour la coloration aléatoire).

Il serait donc intéressant d'essayer de réduire la taille des tables de voisinage en vue de réduire l'utilisation de mémoire pour certains routeur du réseau. Pour cela il faudrait réussir à obtenir une coloration permettant d'avoir pour tout nœud ($u \in V, |V| = N$) un représentant de chaque couleur (k) dans une boule de taille proche de $B(u) = N/K$. Ainsi, le nombre total d'entrées par nœud se rapprocherait de la valeur optimale :

$$K.p(K,G) + \frac{2N}{K}, \text{ avec } p(K,G) \simeq 1$$

pour le moment, $p(K,G) = H_K = K\ln K + \gamma + o(1)$ avec $\gamma = 0.57..$, H_k étant le $k$ième nombre harmonique . Nous n'avons pas mis en place un tel système de coloration pour le moment, néanmoins l'idée principale se résumerait à créer des partitions du graphe en boules de taille de taille proche de N/K puis de colorier chacune de ces partitions avec K couleurs disposées aléatoirement. On construirait ensuite les boules de voisinage de la même manière que pour les deux autres méthodes. Cette méthode de coloration alternative est décrite plus en détails dans la *sous-section 7.1.2*.

4. **C**ooperative **A**ssociation for **I**nternet **D**ata **A**nalysis

Comparatif des deux versions de coloriage , on voit en *figure 25* que le nombre d'entrées est toujours plus petit pour l'algoritme de coloration suivant un ordre sur les degrés (gain de $817-721=96$ entrées en moyenne). Cela est due en partie à la réduction de la taille des tables de voisinages (gain de $387-314=73$ entrées en moyenne - *figure 26*), mais également au fait que dans ce cas précis de simulation, il y a plus de n/k sommets landmarks (coloration aléatoire et choix d'une couleur pour désigner l'ensemble des landmarks). Cependant, ce dernier point donnerait sur plusieurs expérimentation un nombre de landmarks moyen égal à N/K tout comme pour la coloration suivant l'ordre des degrés.

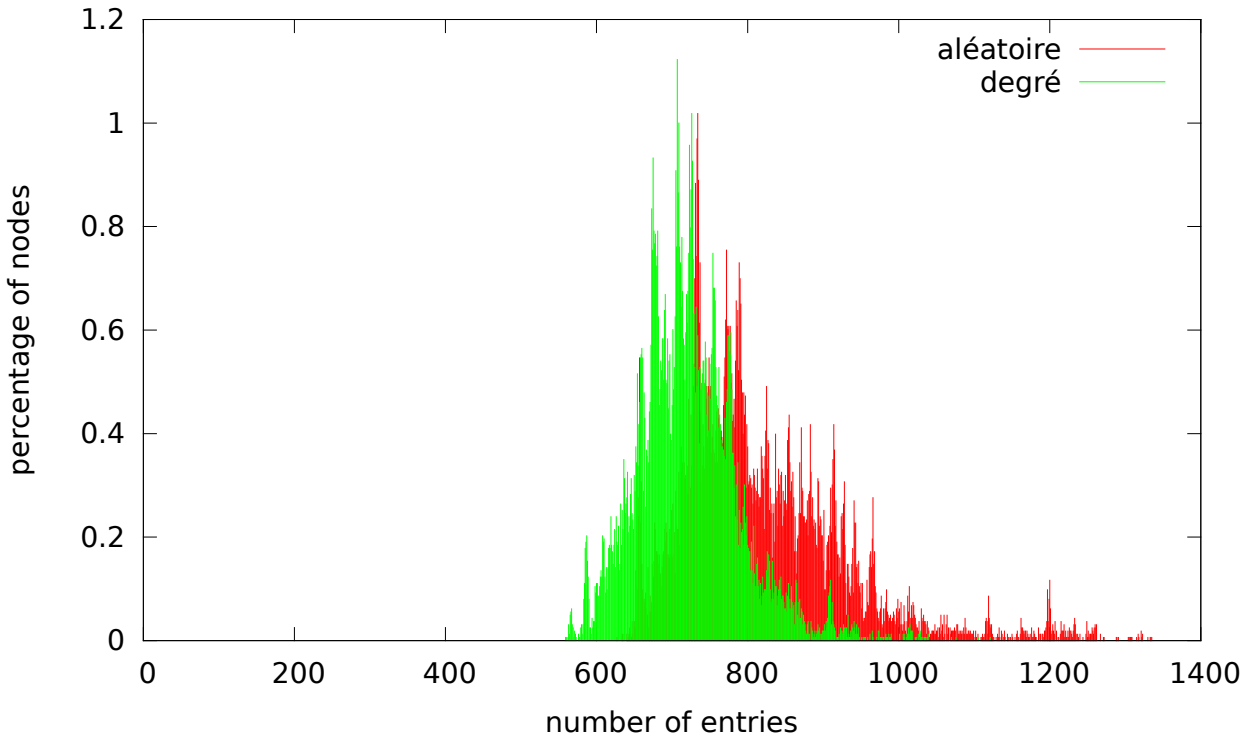


FIGURE 25 – Répartition du pourcentage du nombre d'entrées par routeur

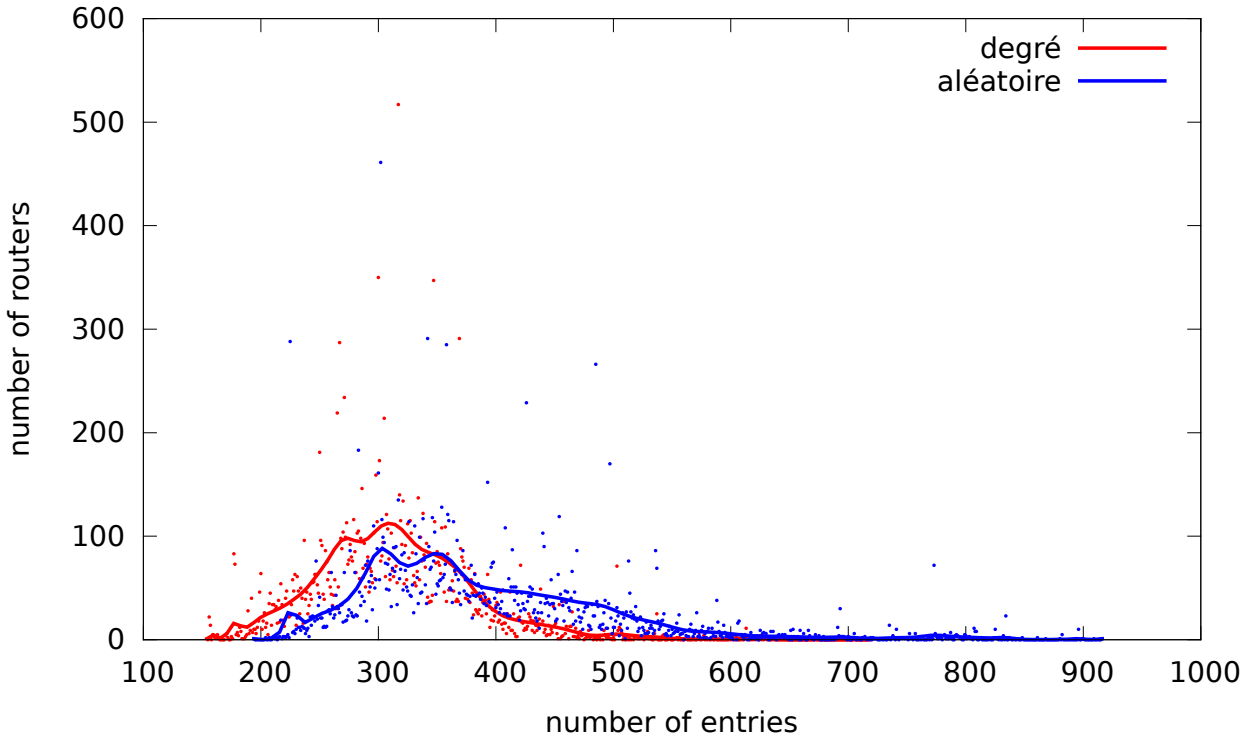


FIGURE 26 – Nombre d'entrées dans la table de voisinage

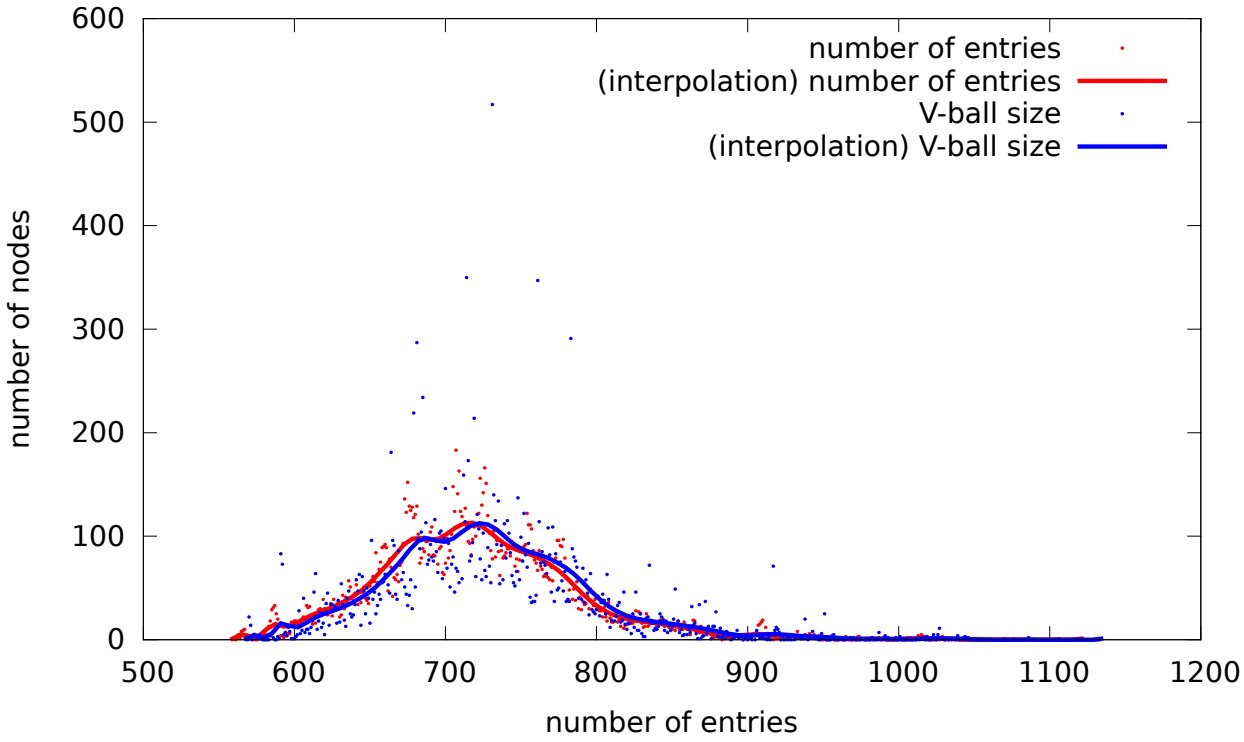


FIGURE 27 – Nombre d'entrées dans la table de voisinage + $2N/K$ VS nombre d'entrées totale pour la version degré uniquement

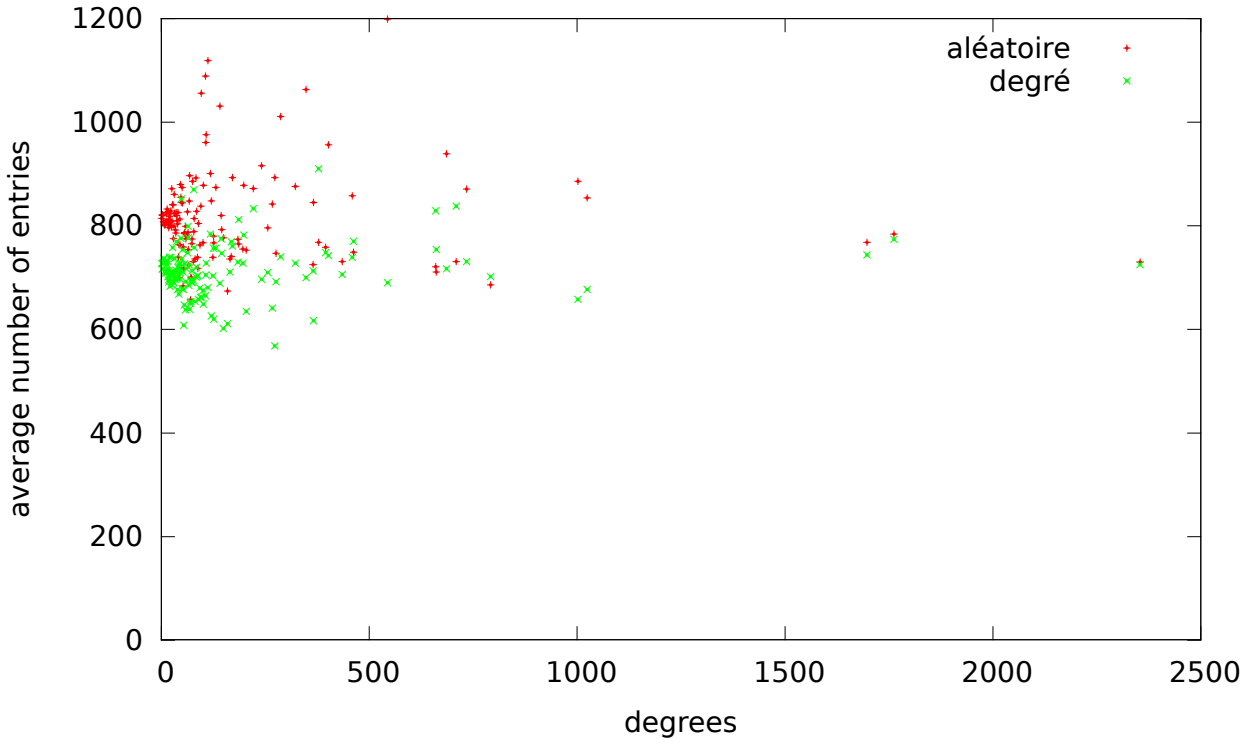


FIGURE 28 – Nombre d'entrées en fonction du degré

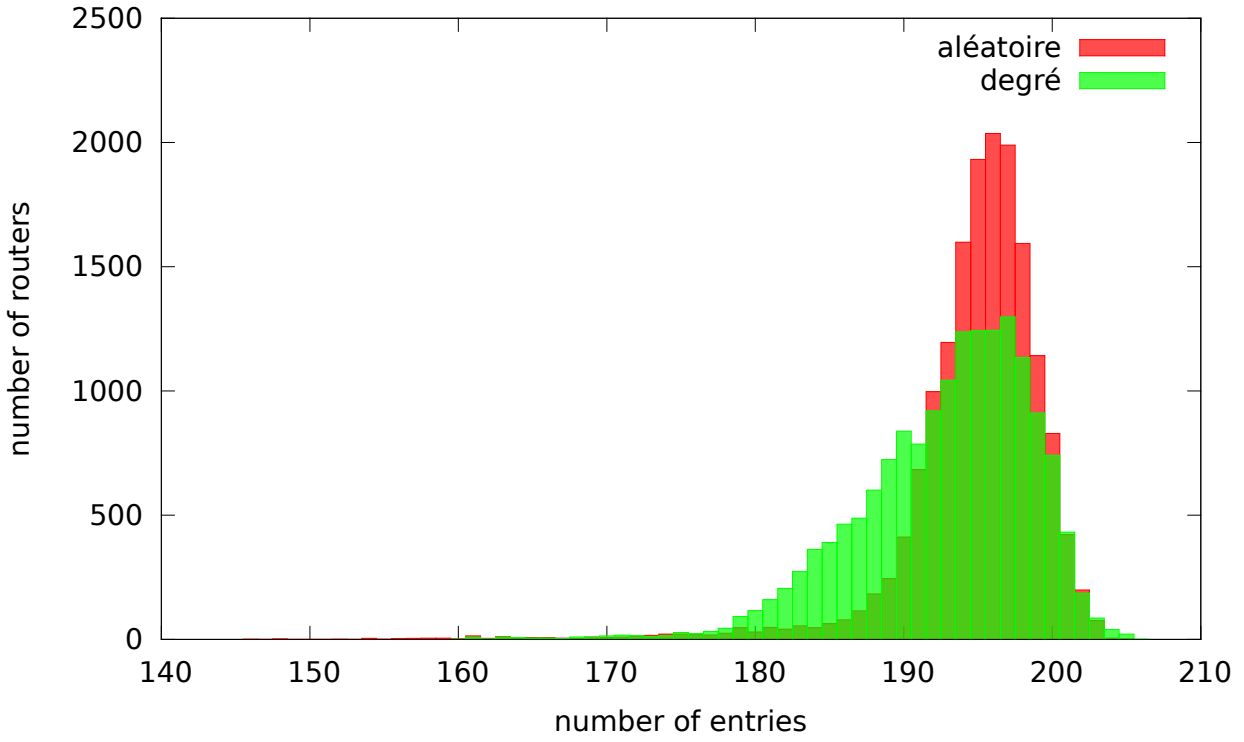


FIGURE 29 – Nombre d'entrées dans la table 3*b*

6.3.2 Proportion de messages transmis par nœud - Répartition de la charge sur le réseau

On observe qu'une faible proportion des nœuds transmet beaucoup plus de paquets que les autres (cf. *figure 30*). Ces nœuds en question sont ceux de fort degré, on peut voir sur la *figure 32* que le le nombre de paquets transmis est en moyenne proportionnel au degré. Comme le montre la courbe de répartition des degrés (*figure 31*), les nœuds de fort degrés sont très peu nombreux [5] et la répartition des degrés et de la charge en fonction du nombre de nœuds suivent approximativement la même loi de répartition. On peut donc en conclure que la charge d'un nœud est directement lié à son degré dans un graphe CAIDA.

Comparatif des deux versions de coloration La coloration n'influe pas de façon remarquable la charge des nœuds.

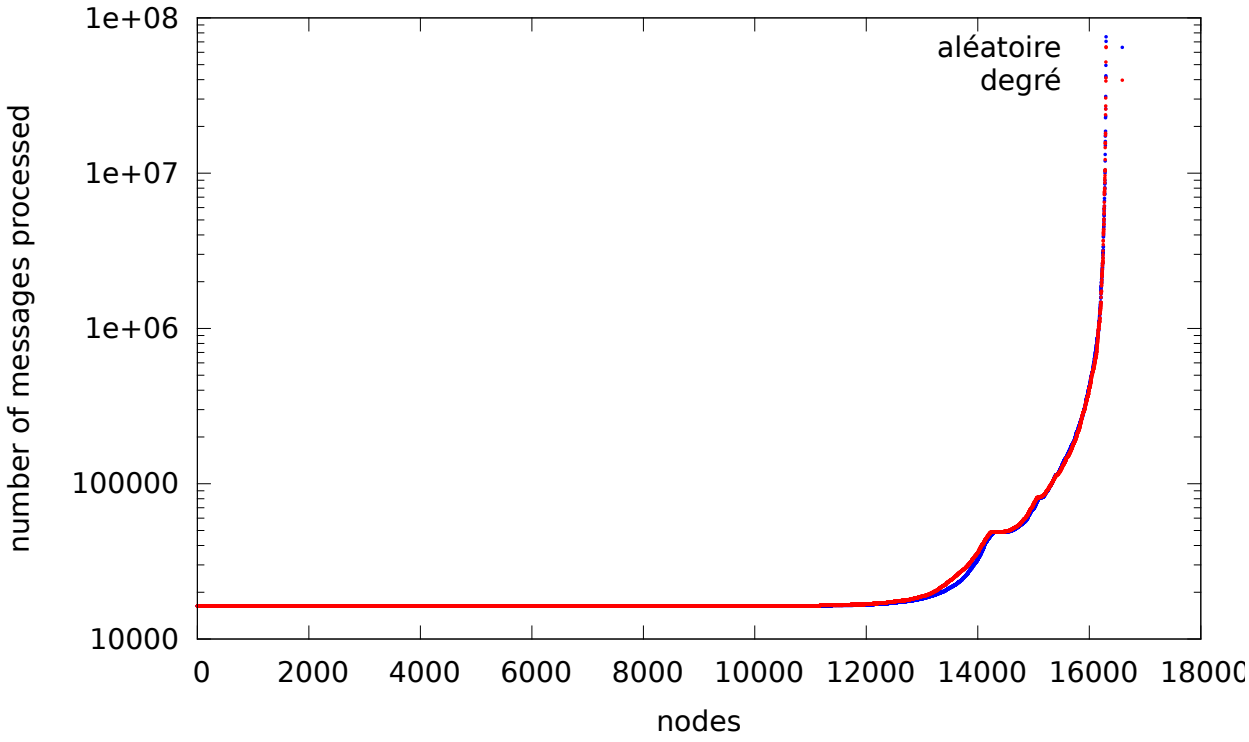


FIGURE 30 – charge en fonction du nombre cumulé de nœuds

5. le réseau des AS (map CAIDA) fait parti de la famille de graphes dont les degrés suivent une loi de répartition en puissance

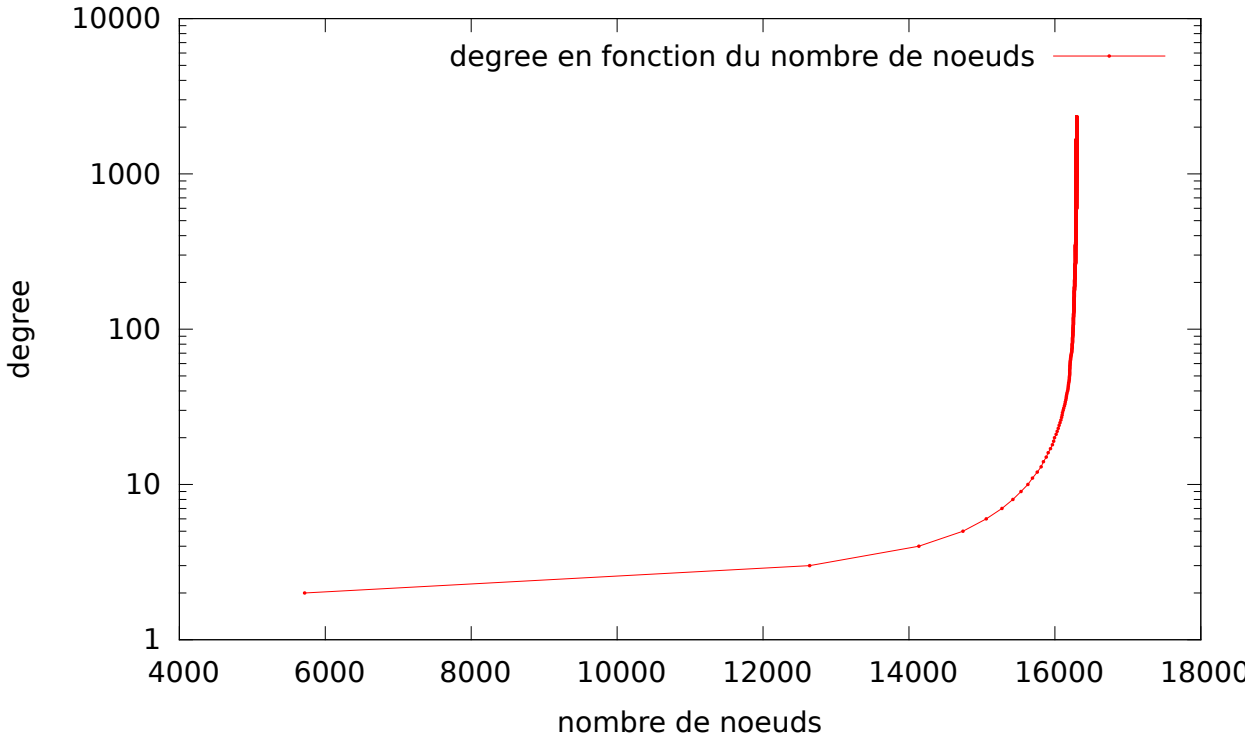


FIGURE 31 – Répartition des degrés dans le graphe CAIDA

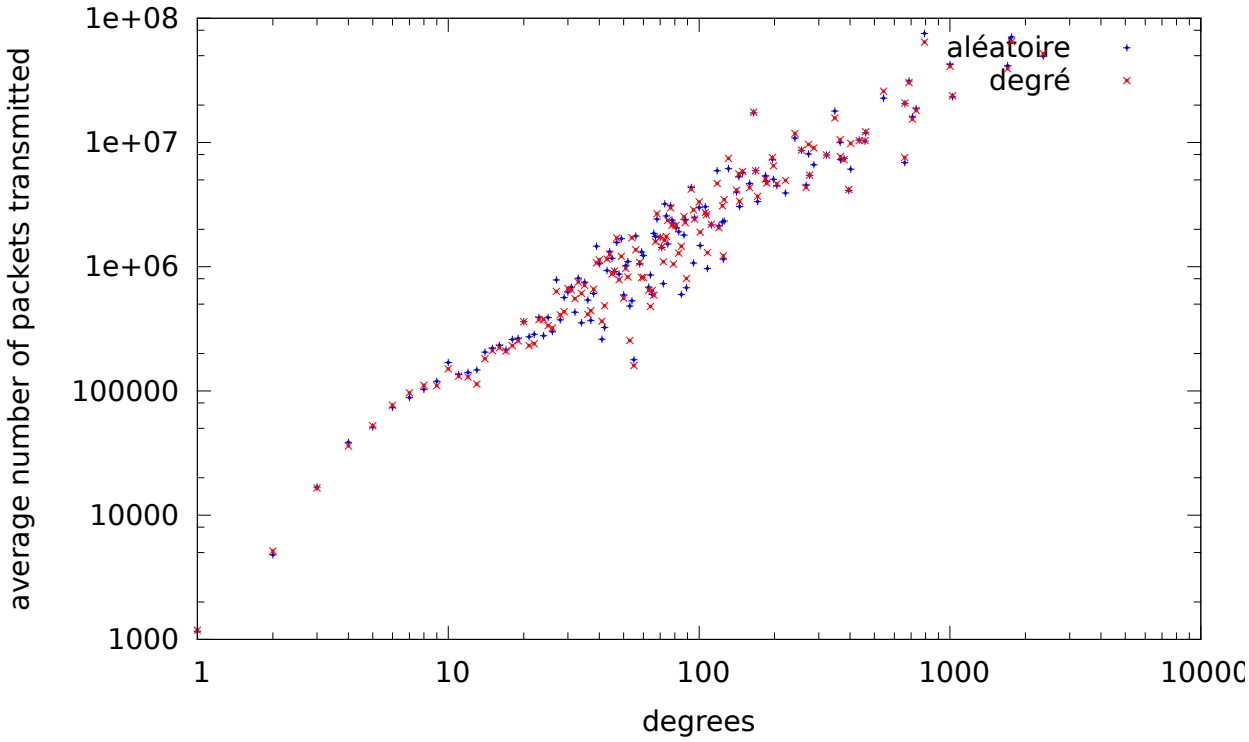


FIGURE 32 – Charge en fonction du degré

6.3.3 Stretch et latence

Stretch Pour cette expérience tous les routages pair à pair ont été effectués soit N^2 routages. Le stretch moyen est de 1.4234 pour la coloration aléatoire et 1.4318 de pour la coloration intelligente.La répartition du stretch est présenté en *figure 33* sous forme cumulative et 34 en terme additif,

- plus de 11% des routes se font sans détour ($stretch = 1$)
- plus de 50% des routes ont un $stretch < 1,3$
- 99% des routes ont un $stretch \leq 2$
- 93% des paquets font un détour d'un ou deux sauts
- 0.3% des paquets font un détour de plus de trois sauts (stretch additif max $= 7(3,7.10^{-6}$% des routes))

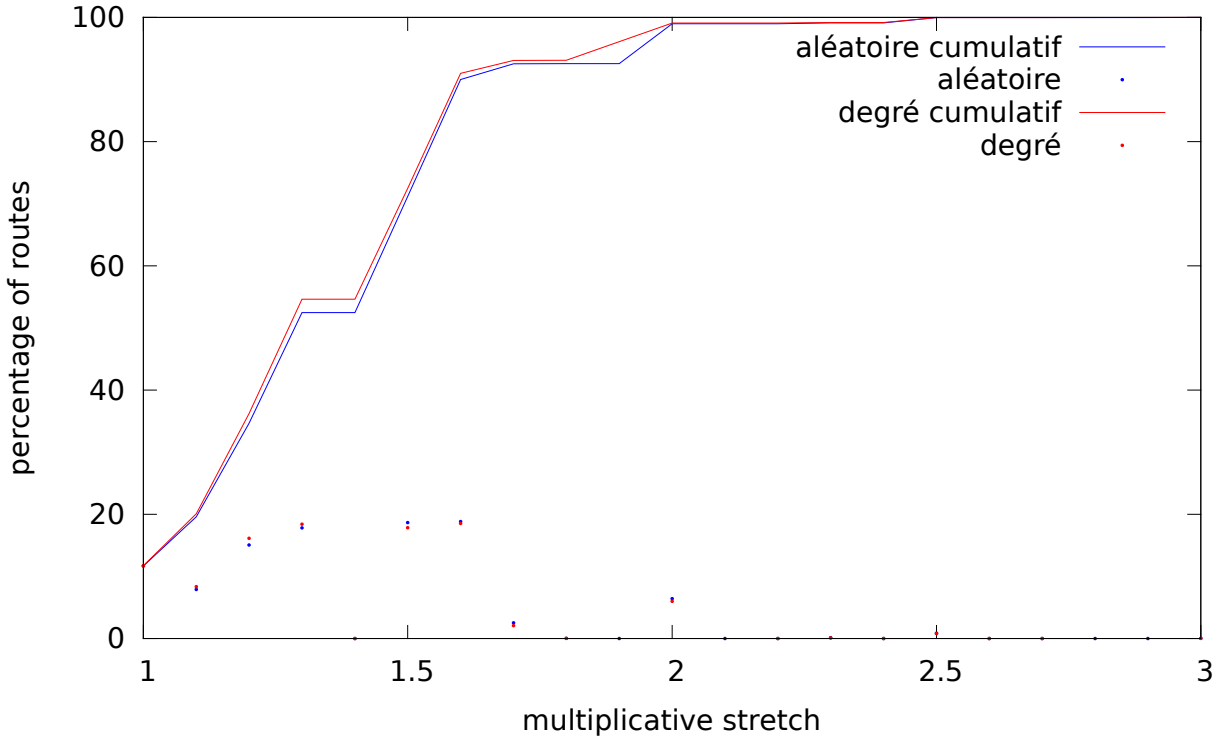


FIGURE 33 – Stretch relatif cumulé et non cumulé

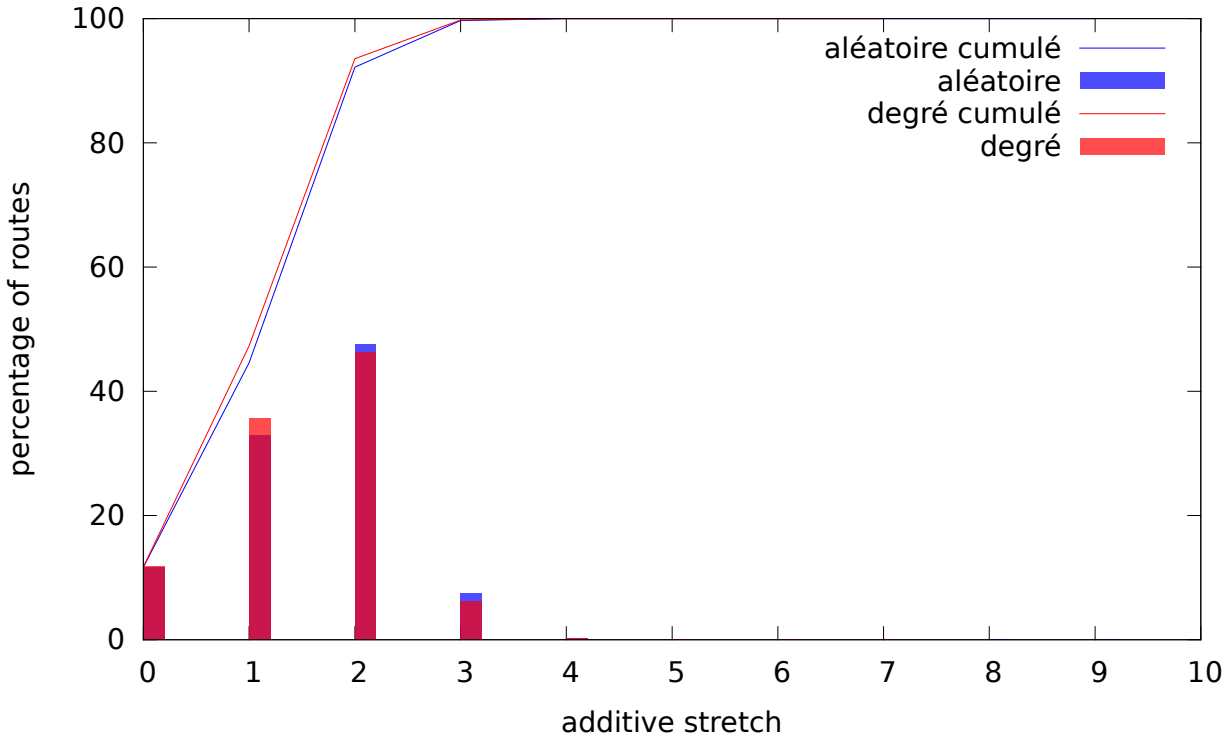


FIGURE 34 – Détour relatif cumulé et non cumulé

On remarque de plus que le détour, ou stretch additif ne dépend pas de la distance entre la source et la destination, voir *figure 35* pour la version de coloration suivant l'ordre des degrés. Le détour est donc constant et quasiment réparti en deux parts égale entre 1 et 2 comme on peut le voir en *figure 36* pour la même version de coloration.

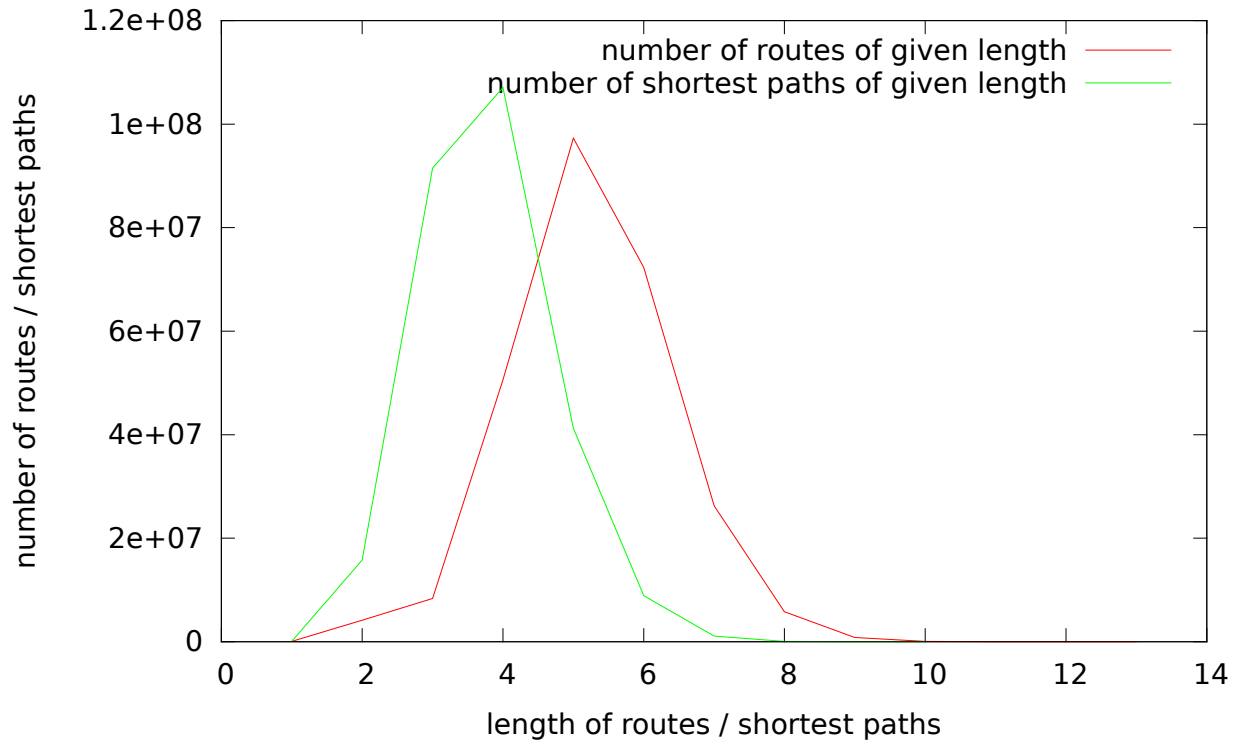


FIGURE 35 – Comparatif entre détour et distance $(source, destination)$

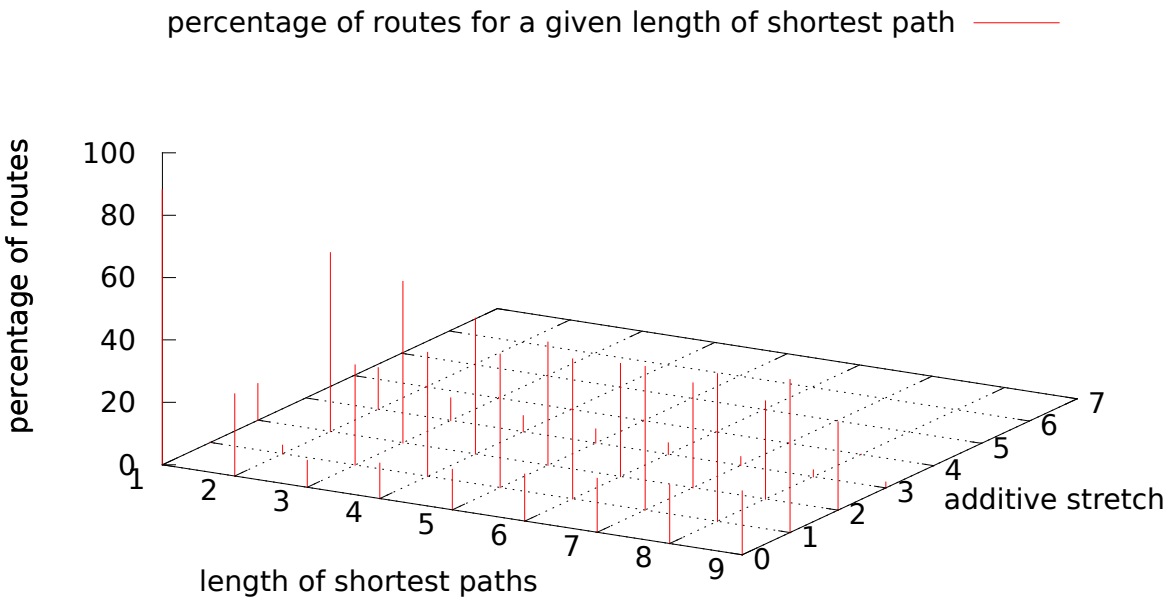


FIGURE 36 – Comparatif relatif entre détour et distance $(source, destination)$ – vue 3D

Latence La totalité des routages prend un peu moins d'une heure et vingt minutes et tous les routages ont étés effectués en série. *Il est à noter que ceci est fait sans utiliser le simulateur DRMSim qui bloque aprés 3/4 jours de calculs*. La latence liée au recherche des entrées et à la création/lecture des en-têtes des messages protocolaires pour un routage est donc au maximum de 18 μs :

$$\frac{1h19min13s146ms}{n^2} = \frac{4753146 \text{ (ms)}}{16301^2 \text{ routages}} = 0.017887624ms \text{ par routage}$$

La distance moyenne entre 2 sommets est de plus de 4, on a donc un temps de calcul d'au plus 4,5 μs

7 Améliorations/Modifications possibles du schéma de routage

7.1 Modification des paramètres décrits dans la partie Expérimentations

Les modifications proposées dans cette partie ne vont pas avoir de conséquences sur les propriétés de l'algorithme : étirement maximal et taille des tables. En effet, ces modifications restent dans les degrés de liberté du schéma proposé dans [AGM+08].

7.1.1 Modification du nombre de couleurs K

Le nombre de couleurs K va jouer sur la taille des tables comme montré dans la section 4.4 et également sur l'étirement. Augmenter K va augmenter la taille des boules de voisinage, ce qui signifie qu'un plus grand nombre de routages dans le cadre d'un scénario « all to all » se feront à l'intérieur des boules de voisinage donc en plus court chemin. Cependant d'un autre côté, d'une part, il y aura moins d'entrées dans les tables 3a et 3b et d'autre part, la première étape du routage e pourrait être plus longue en moyenne. Il serait intéressant d'évaluer sur un graphe GLP donné, l'influence de K sur l'étirement pour quelques valeurs.

7.1.2 Réduction de la taille des boules de voisinage - Coloration par parts

Problème La taille totale des tables de routage en moyenne est représentée par la fonction qui suit :

$$k.p(k,G) + \frac{2n}{k}, \text{ avec } 1 \leq p(k,G) \leq \frac{n}{k}$$

$k.p(k,G)$ étant le facteur représentant la taille des boules de voisinage (table 1). En changeant la coloration du graphe il est possible d'influer sur la fonction $p(k,G)$, en effet, si on arrivait à colorier le graphe de la façon optimale on pourrait avoir pour chaque nœud du graphe une boule de taille k comportant un représentant de chaque couleur. Cependant une telle coloration est très couteuse à calculer, de plus la solution optimale n'existe pas forcément pour tout couple (k,G). Par exemple, en prenant $k=4$ et G un graphe de Petersen ($n=10$), alors il est impossible d'avoir uniquement des boules de taille $k=4$. Nous proposons donc une technique de coloration qui permet réduire la valeur de la fonction $p(k,G)$.

Idée d'amélioration

1. On choisit un ensemble $X = \{x_i \in G\}$ de n/k sommets, choisi aléatoirement uniforme.
2. On crée une partition $\{P_i\}$ des sommets du graphe comme suit : Au départ les P_i sont vides, puis dans un ordre arbitraire on considère chaque sommet $u \in G$
   - on ajoute u à P_i s'il est plus proche de x_i que de tous les autres x_j,
   - ou bien s'il a le choix on l'ajoute à la part P_i dont x_i est le plus proche de u et telle que $|P_i|$ est le plus petit.
   - en cas d'égalité, on choisit aléatoirement entre les deux parts.
3. Dans chaque part P_i, on colorie les sommets comme suit : $|P_i| \% k = r$, on prend successivement $|P_i| - r$ nœuds de P_i aléatoirement qu'on colorie avec les couleurs $\{1_1, 2_1, ..., k_1, ..., 1_j, ..., k_j, ..., 1_{\frac{|P_i|}{k}}, ...k_{\frac{|P_i|}{k}}\}$ ($couleur_x = couleur_y \forall (x,y) \in \left[1, \frac{|P_i|}{k}\right]^2$) puis on choisit aléatoirement un ensemble de r couleurs pris parmi $\{1, ..., k\}$ pour colorier les r sommets restants.

7.1.3 Inclusion de tous les voisins directs dans les boules de voisinage

Ce paramètre ne devrait pas avoir beaucoup d'influence sur la taille des tables 1 puisque seuls quelques nœuds ont un nombre de voisins supérieurs à la taille des boules de voisinage (la taille des boules de voisinage pour un graphe GLP de 10000 nœuds est d'environ 600). Cela va améliorer les performances en terme d'étirement mais non de manière significative puisque le nombre de routes concernées sera faible. L'avantage est que tous les messages entre voisins directs se feront suivant le lien direct entre les deux voisins (pas de détour pour envoyer un message à un voisin direct ce qui est une propriété raisonnable).

7.1.4 Modification du choix du nœud de la bonne couleur

Le choix de cette couleur est pour l'instant déterminé statiquement. Le choix qui a été fait dans l'implémentation est de choisir le nœud qui a le plus grand degré. Il serait intéressant d'évaluer les performances en termes d'étirement lorsqu'un autre choix est fait. Un choix intéressant serait éventuellement de choisir le nœud de la bonne couleur le plus proche et en cas d'égalité de choisir le nœud de plus fort degré.

7.2 Modifications du schéma de routage

Ces modifications sont plus importantes et ne garantissent plus éventuellement les propriétés du schéma de routage (taille des tables et étirement maximal). Il faudrait éventuellement redémontrer ses propriétés (si elles sont toujours vérifiées).

7.2.1 Routage c

Il est éventuellement possible de modifier le routage c et le contenu de la table 3a. Considérons donc un nœud source u, un nœud destination v et l_v le landmark le plus proche de v. Le routage c consiste à router de u à v en plus court chemin **dans l'arbre** l_v. Le nœud u doit connaître « l'adresse » de v dans cet arbre (l'identifiant DFS de v dans l'arbre et son cpath) et l'identifiant de l_v. De plus, le nœud u doit insérer « l'adresse » de v dans l'arbre de l_v dans l'en-tête du message.

Il est éventuellement possible de ne stocker que l_v en u et de stocker en l_v « l'adresse » de v dans l'arbre de l_v. Le chemin suivi par le message est alors u, l_v, v, possiblement plus long que le chemin suivi dans le cadre originel du schéma AGMNT. Dans ce cas, la table 3a d'un nœud u ne contient plus que les identifiants de v et l_v pour tous les v tels que $h(v) = c(u)$. Les landmarks, par contre, doivent stocker davantage d'informations. Il leur faut stocker, pour chaque nœud dont ils sont le landmark le plus proche, stocker « l'adresse » de ce nœud dans leur arbre.

En moyenne, le nombre de nœuds dont un landmark est le plus proche est égal à $K-1$. On rajoute donc environ $\frac{n\times(K-1)}{K}$ entrées dans les tables de manière globale. En contrepartie, on simplifie les entrées dans les tables 3a.

Dans ce cas, le routage c est divisé en deux étapes qui ne nécessitent plus les mêmes en-têtes. La première étape est l'envoi du message de u à l_v. Le message nécessite alors un en-tête contenant seulement l'identifiant de l_v. La seconde étape est l'envoi du message de l_v à v. L'en-tête du message contient alors « l'adresse » de v dans l'arbre l_v.

7.2.2 « Symétrisation » des boules de voisinage

Ce que l'on entend par symétrisation est de rendre la relation d'appartenance à une boule de voisinage symétrique : $u \in B(v) \iff v \in B(u)$. Cette modification va certainement augmenter de manière considérable la taille des boules de voisinage au moins pour certains nœuds (nœuds de degré élevé). Elle permet par contre de simplifier certaines entrées dans les tables et certains en-têtes de messages.

Cas du routage c Il n'est plus nécessaire pour les landmarks de stocker une table supplémentaire comme décrit dans la section 7.2.1. En effet, comme $v \in B(l_v)$, l_v connaît grâce à sa table 1 un lien sur un plus court chemin de l_v à v. Il n'est doncplus nécessaire d'avoir d'en-têtes pour le message dans ce cas.

Cas du routage d Les tables 3b sont très simplifiées si on « symétrise » les boules de voisinage. Il n'est plus nécessaire de stocker les « adresses » de w dans l'arbre de u et de v dans l'arbre de y. En effet, $w \in B(u)$ et $v \in B(y)$ par « symétrisation ». Les tables 3b se contentent de contenir alors l'identifiant de v, de w, de x, de y. L'entête est simplifié car il ne doit contenir que les identifiants de w, x et y et un état pour indiquer à quelle étape se trouve le routage.

De la même manière, il n'est plus nécessaire de stocker dans les entrées de la table 1 les informations de routage dans les arbres des B-voisins et les entrées des tables 2 peuvent se limiter à l'identifiant du landmark et le lien vers le père dans l'arbre du landmark.

Ces modifications changent profondément le schéma de routage AGMNT et ne garantissent donc plus les bornes démontrées dans l'article.

Références

[AGM+08] Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, Noam Nisan, and Mikkel Thorup. Compact name-independent routing with minimum stretch. *ACM Transactions on Algorithms*, June 2008.

[BT02] T. Bu and D. Towsley. On distinguishing between Internet power law topology generators. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 638–647. IEEE, 2002.

[FG01] Pierre Fraigniaud and Cyril Gavoille. Routing in trees. In *Proceedings of the 28th International Colloqium on Automata, Languages and Programming (ICALP)*, pages 757–772, July 2001.

[ZM04] Shi Zhou and Raúl J. Mondragón. Accurately modeling the internet topology. *Phys. Rev. E*, 70(6) :066108, Dec 2004.