Simulations de routage du schéma AGMNT

1 Description du schéma de routage

On appelera B-voisin un nœud v présent dans la boule de voisinage d'un nœud u. Le schéma de routage AGMNT [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 noeud source considéré. La table 3a sert lorsque l'on passe par le landmark le plus proche de la destination et la table 3b lorsqu'on passe par les boules de voisinage contigues. 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).

1.1 Description de la construction des tables

Cette partie va décrire tout ce qu'il est nécessaire faire pour construire les tables du schéma AGMNT. Le seul élément que l'on possède au départ est le graphe $G = (V, E)$. On appelle K le nombre de couleurs du schéma de routage AGMNT.

La première chose que l'on va faire est de colorier les différents nœuds du graphe G. Pour cela, on peut soit les colorier de manière aléatoire uniforme soit les colorier d'une manière plus astucieuse. 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 permet de réduire la taille de tables en moyenne de 3% (580 entrées en moyenne au lieu de 600).

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. 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.

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`.

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).

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`).

1.1.1 Construction de la table 1

À partir de ces informations, 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. 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.

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 1 représente de manière synthétique les informations contenues dans 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 1 – Résumé de la table 1 (un `short` prend 2 octets et un pointeur 8 octets)

1.1.2 Construction de la 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.

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 2 – Résumé de la table 2 (un `short` prend 2 octets et un pointeur 8 octets)

1.1.3 Contruction de la table 3a et 3b

Construction de la table 3a Considérons tout d'abord la construction de la table 3a (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 deux informations supplémentaires qui ne sont pas nécessaires pour le routage : l'identifiant dans le graphe du nœud v et une borne max sur la longueur du chemin pris en utilisant cette entrée (distance de u au landmark + distance du landmark à v). Ces informations seront utiles lorsque l'on va comparer les entrées de la table 3a à celle de la table 3b (il est ainsi possible d'enlever ces informations de la table 3a après la comparaison des entrées entre les deux tables ; ceci n'est pas fait dans l'implémentation actuelle). Le tableau 3 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 3 – Résumé de la table 3a (un `short` prend 2 octets et la liste de liens occupe au moins (12 + longueur de la liste * 8) octets) sauf si elle est vide (8 octets)

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.

Construction de la 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 4 résume les informations contenues dans une entrée de la table 3b.

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

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`	liste de liens	`short`	liste de liens	`short`	`short`

TABLE 4 – Résumé de la table 3b (un `short` prend 2 octets et la liste de liens occupe au moins (12 + longueur de la liste * 8) octets) sauf si elle est vide (8 octets)

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

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 les entrées les moins performantes a priori.

1.2 Description de la fonction de routage

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

1. Le nœud destination est un voisin du nœud source : aucun en-tête n'est nécessaire.
2. Le nœud destination est un landmark : aucun en-tête n'est nécessaire.
3. 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.
4. 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.
5. 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 cas 3 et 4 sont également utilisés quand le message a atteint le nœud qui possède les informations de routage dans le cas 5 pour router le message vers la destination. On notera ces cas 3' et 4'.

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.

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.

Si le message contient l'en-tête correspondant au cas 5, celui-ci a la forme donnée par le tableau 5. 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 3' et 4').

id du B-voisin à atteindre

TABLE 5 – En-tête utilisé dans le cas de 5

Si le message contient l'en-tête correspondant au cas 3 ou 3', celui-ci a la forme donnée par le tableau 6.

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

TABLE 6 – En-tête utilisé dans les cas 3 et 3'

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.

Si le message contient l'en-tête correspondant au cas 4 ou 4', celui-ci a la forme donnée par le tableau 7 (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 7 – En-tête utilisé dans les cas 4 et 4'

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 3) soit un nœud de la « bonne » couleur qui a été choisi par le nœud source (cas 3'). Ce que l'on appelle identifiant de la source intermédiaire correspond donc soit à l'identifiant de la source (cas 3) soit à l'identifiant du nœud de la « bonne » couleur (cas 3'). 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.

2 Résultats

2.1 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 de manière opposée l'étirement et le nombre d'entrées dans les tables. 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 du nombre d'entrées en fonction du nombre de couleurs K permet d'établir la formule : $K \times ln(K) + \frac{2N}{K}$. Le minimum théorique approximatif pour $N = 10000$ est pour $K = 64$. Nous avons également cherché à évaluer cette valeur de K de manière pratique.

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 1. La courbe obtenue grâce aux simulations est très proche de la courbe théorique, les valeurs étant très légèrement supérieures dans le cadre des simulations.

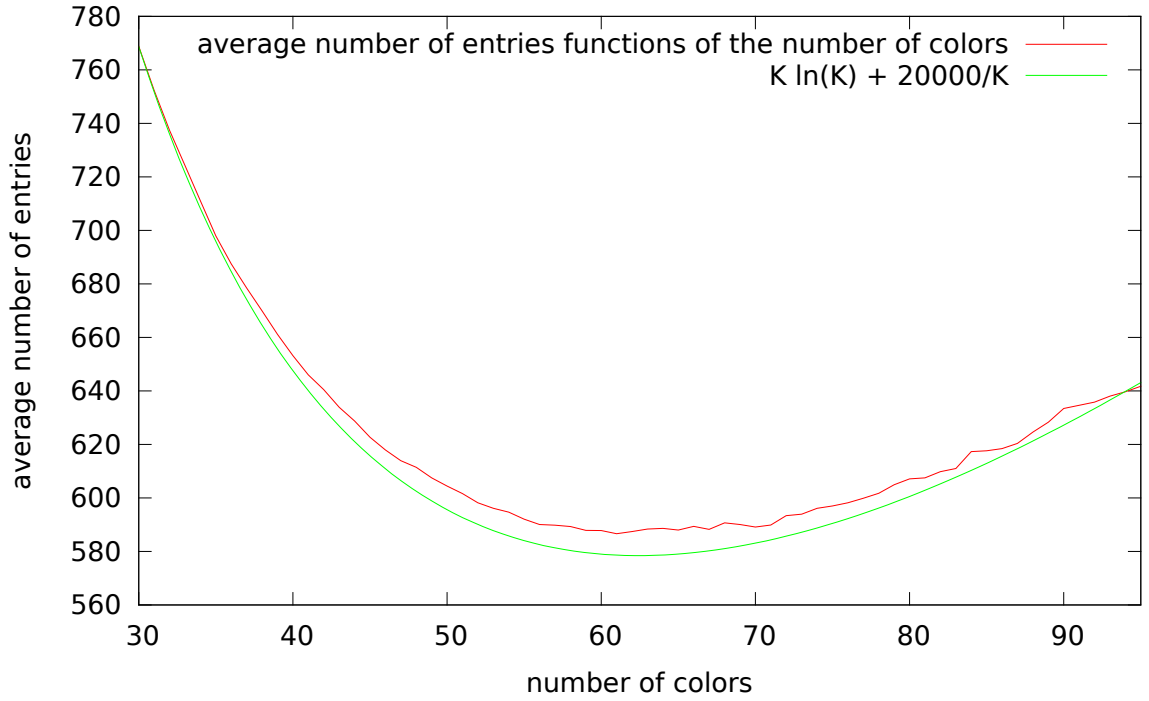

FIGURE 1 – Nombre d'entrées moyen par nœud en fonction du nombre de couleurs K

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 $K = 64$.

2.2 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.

2.2.1 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 2 présente le pourcentage du nombre de nœuds qui ont un nombre d'entrées donné.

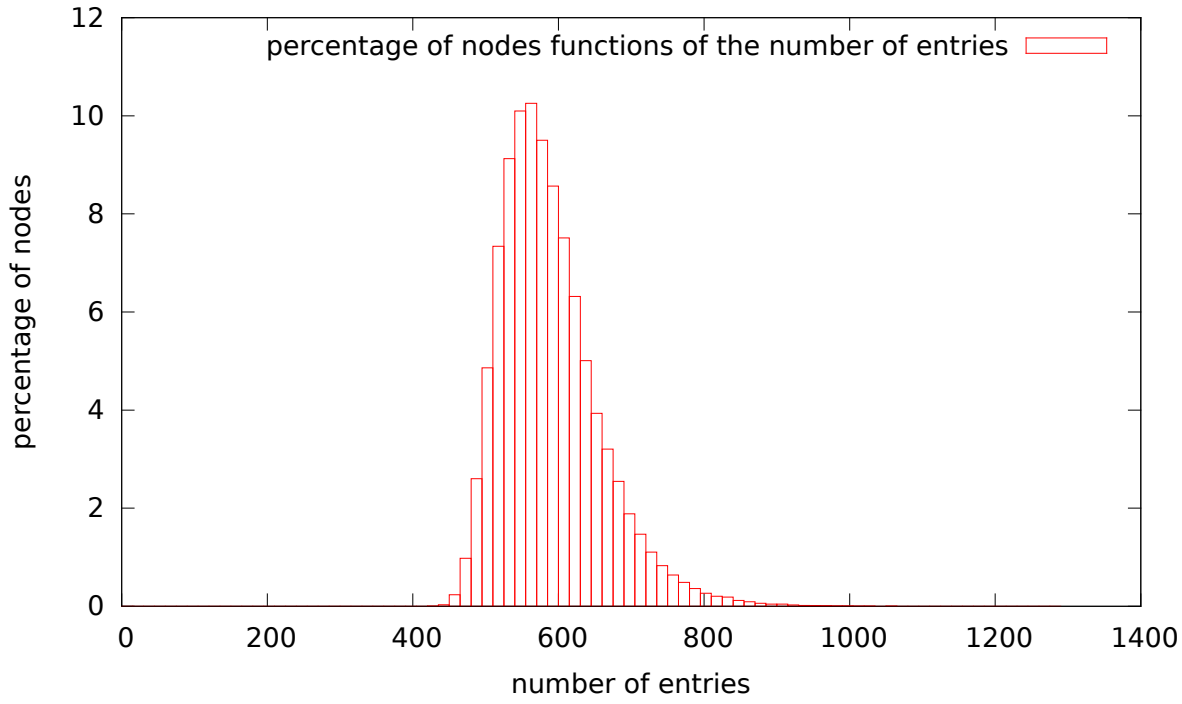

FIGURE 2 – Pourcentage de nœuds qui ont un nombre d'entrées donné

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 3 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.

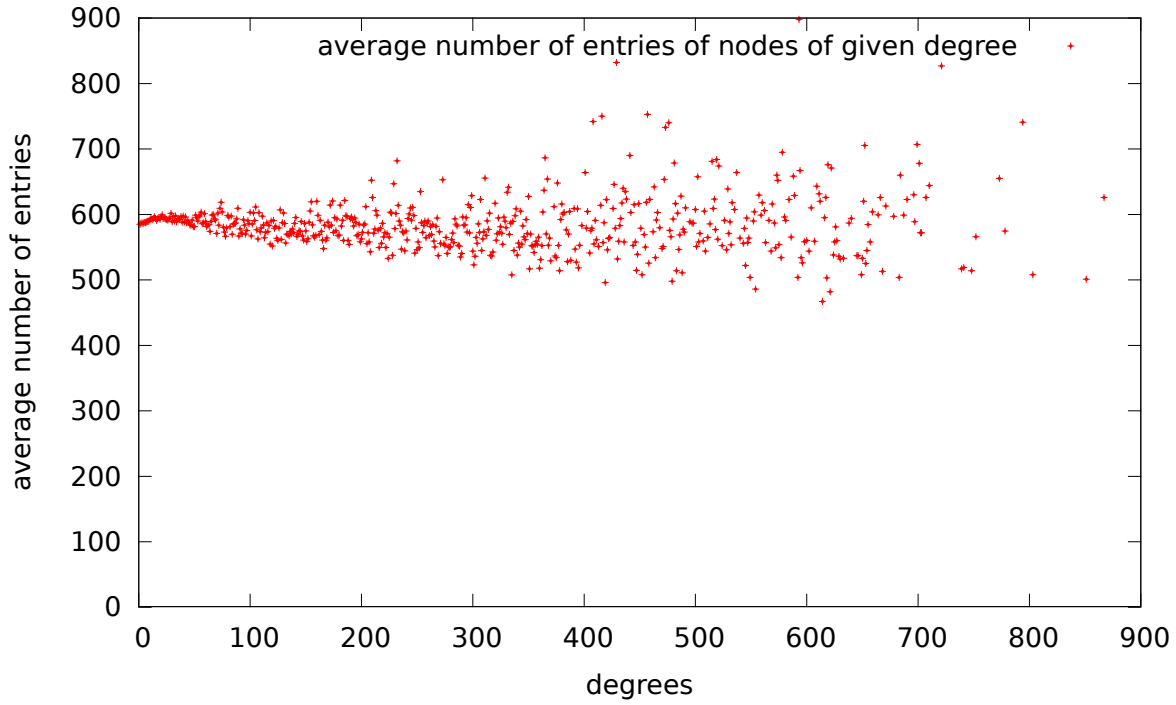

FIGURE 3 – Nombre moyen d'entrées des noeuds d'un degré fixé

2.2.2 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 4 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.

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 5 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.

2.2.3 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 6.

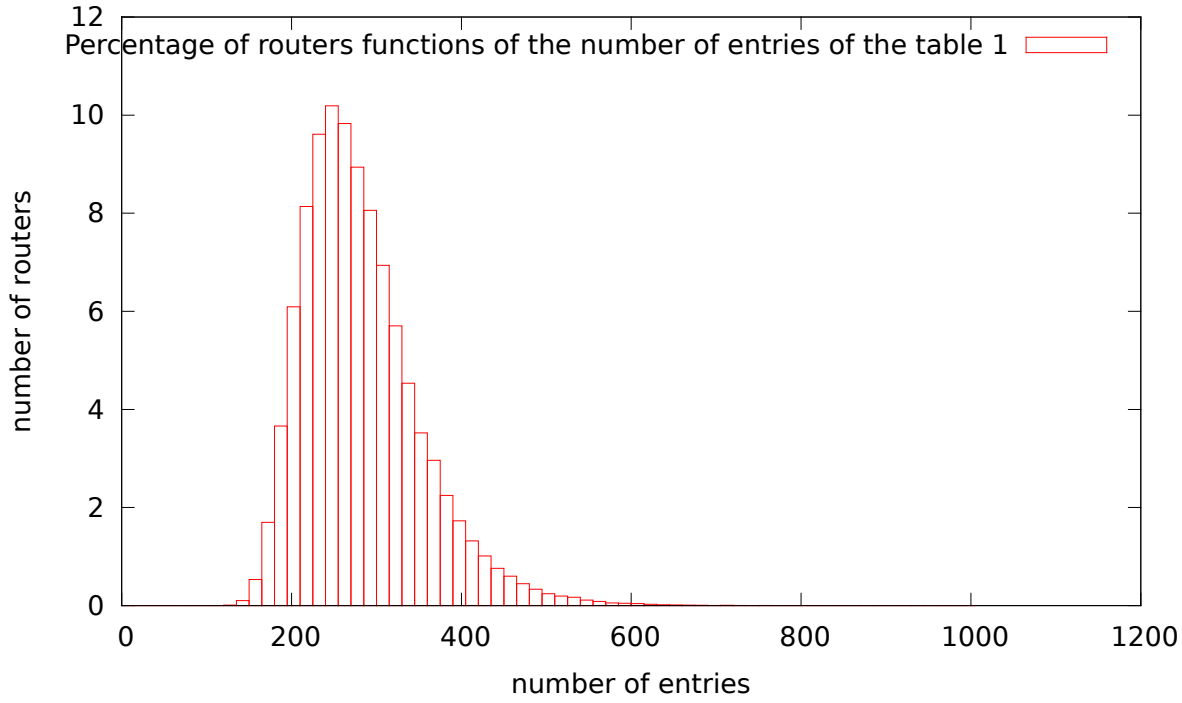

FIGURE 4 – Pourcentage de nœuds qui ont un nombre d'entrées donné

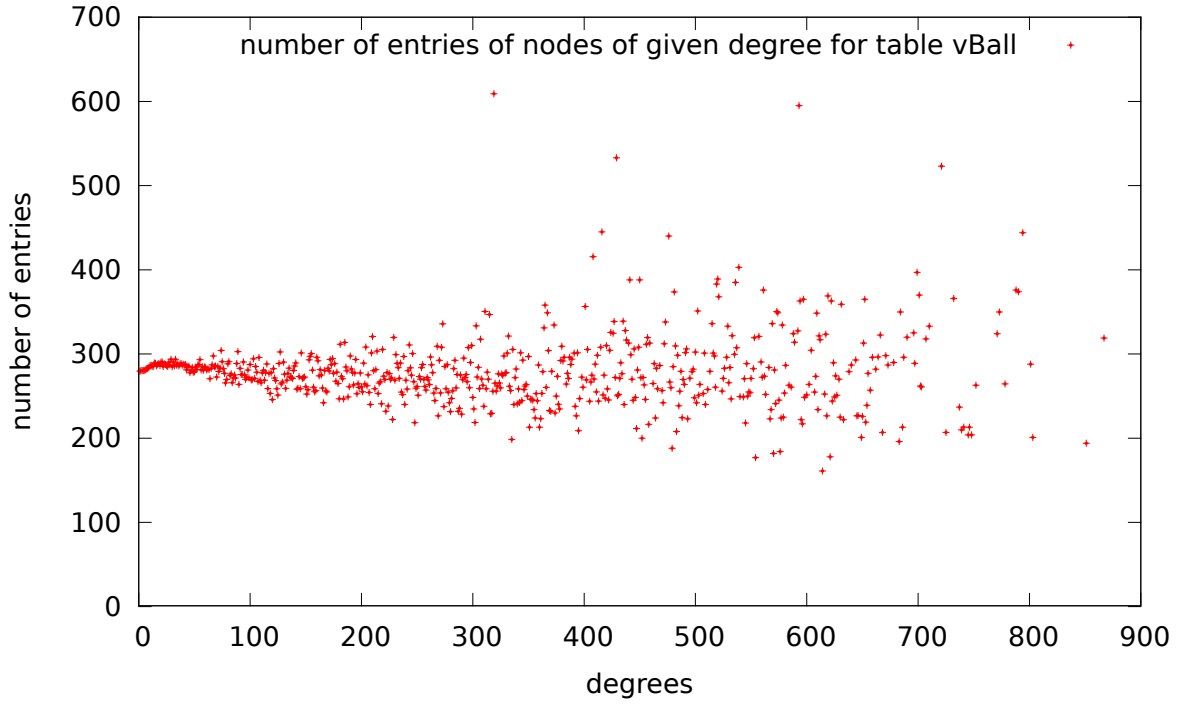

FIGURE 5 – Nombre moyen d'entrées dans la table 1 des nœuds d'un degré fixé

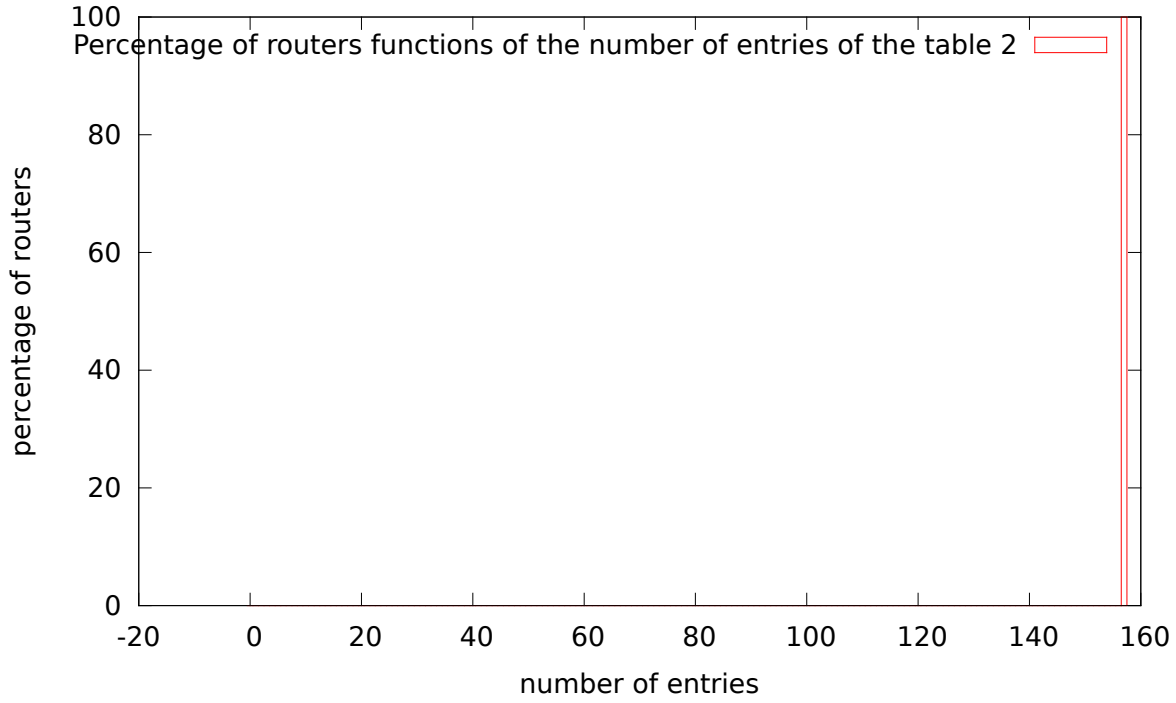

FIGURE 6 – Pourcentage de nœuds qui ont un nombre d'entrées donné dans la table 2

2.2.4 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 7 et 8 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).

2.3 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 9 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.

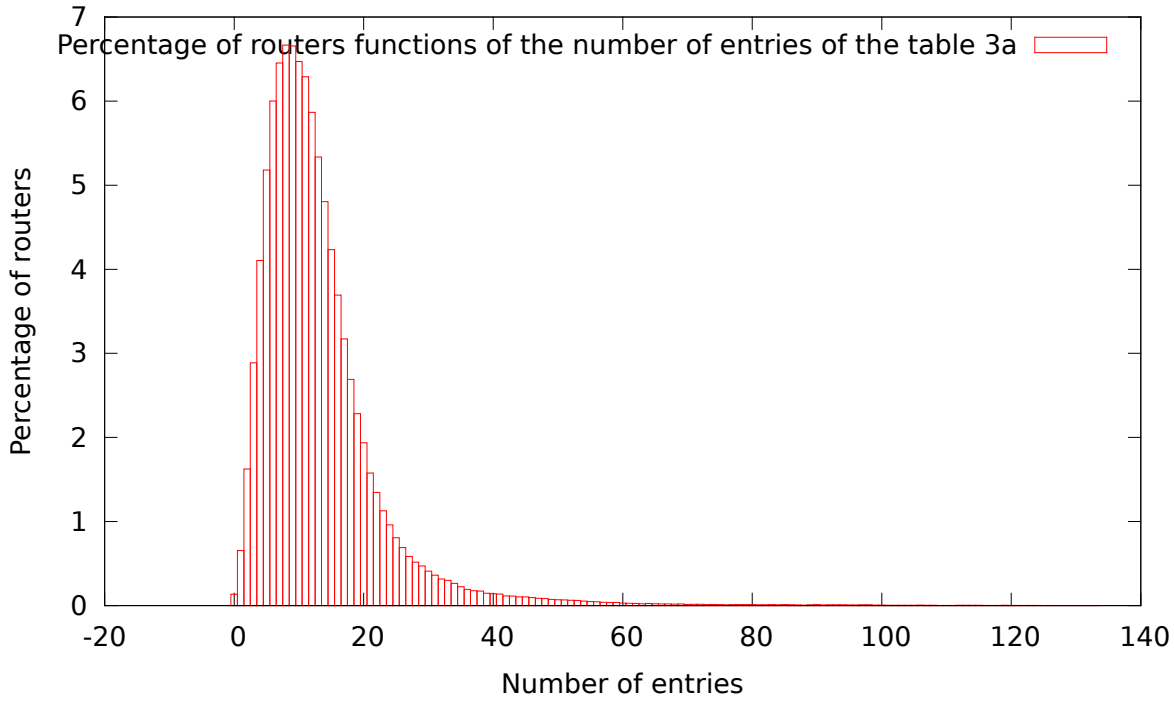

FIGURE 7 – Pourcentage de nœuds qui ont un nombre d'entrées donné dans la table 3a

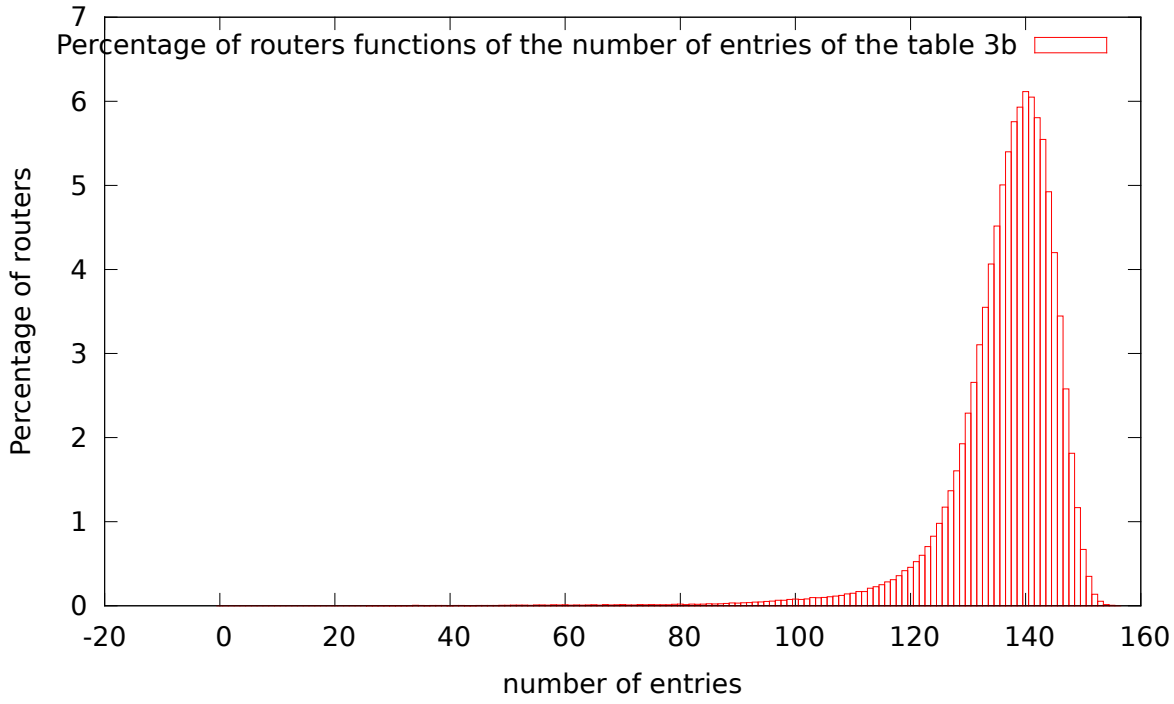

FIGURE 8 – Pourcentage de nœuds qui ont un nombre d'entrées donné dans la table 3b

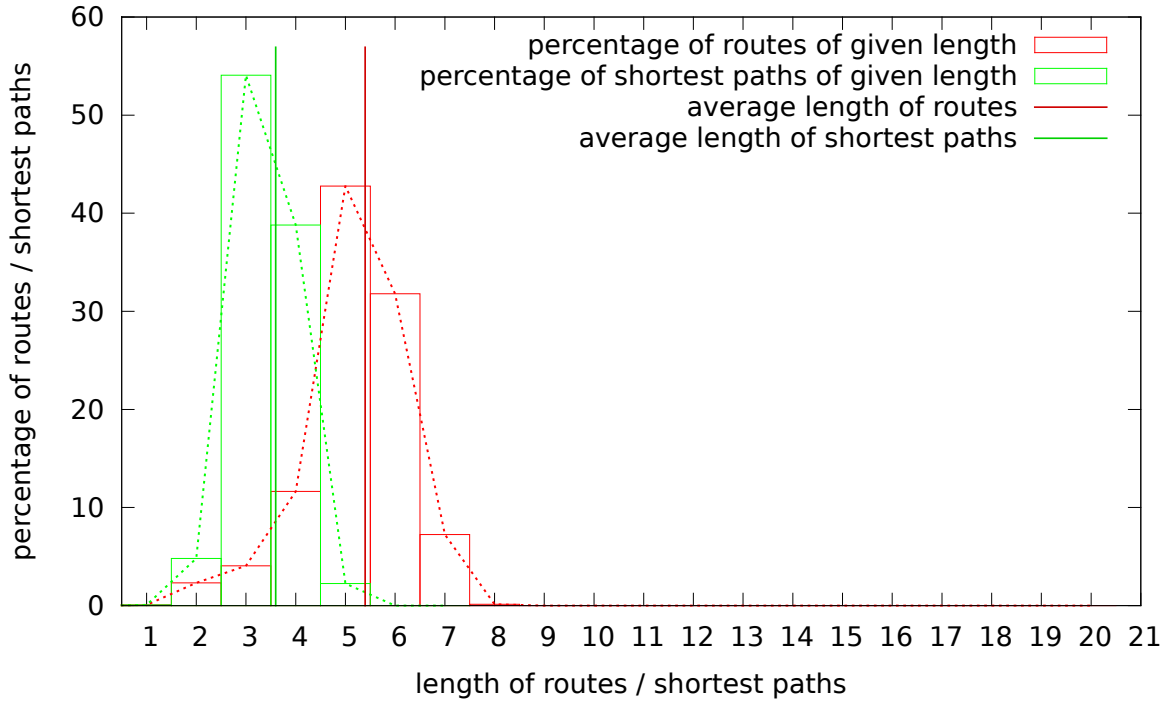

FIGURE 9 – Longueur des plus courts chemins et des routes

La figure suivante 10 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 9.

Les figures 11, 12 et 13 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.

2.4 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.

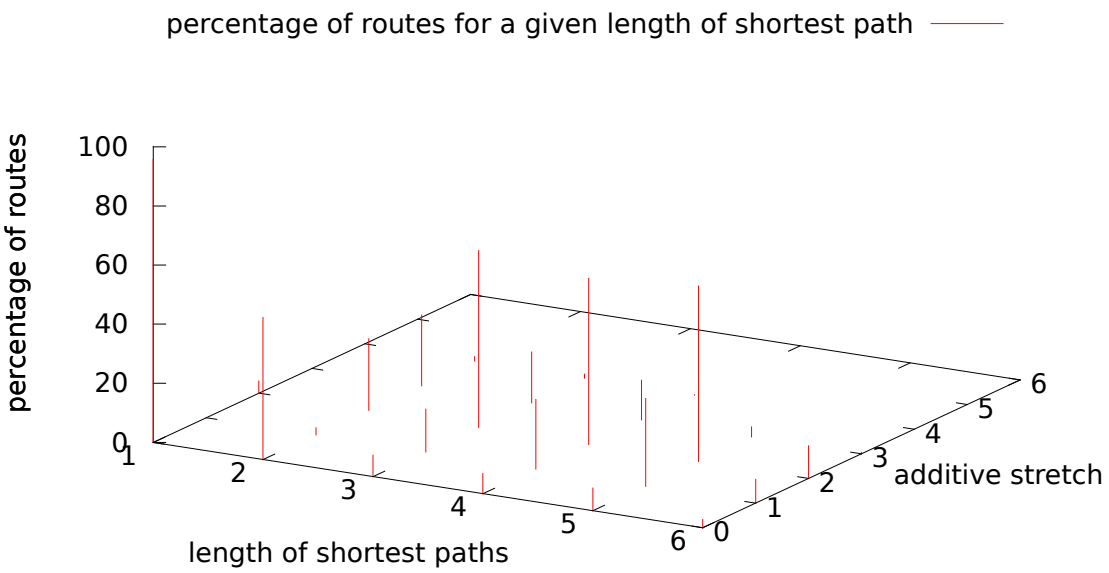

FIGURE 10 – Longueur des plus courts chemins et des routes

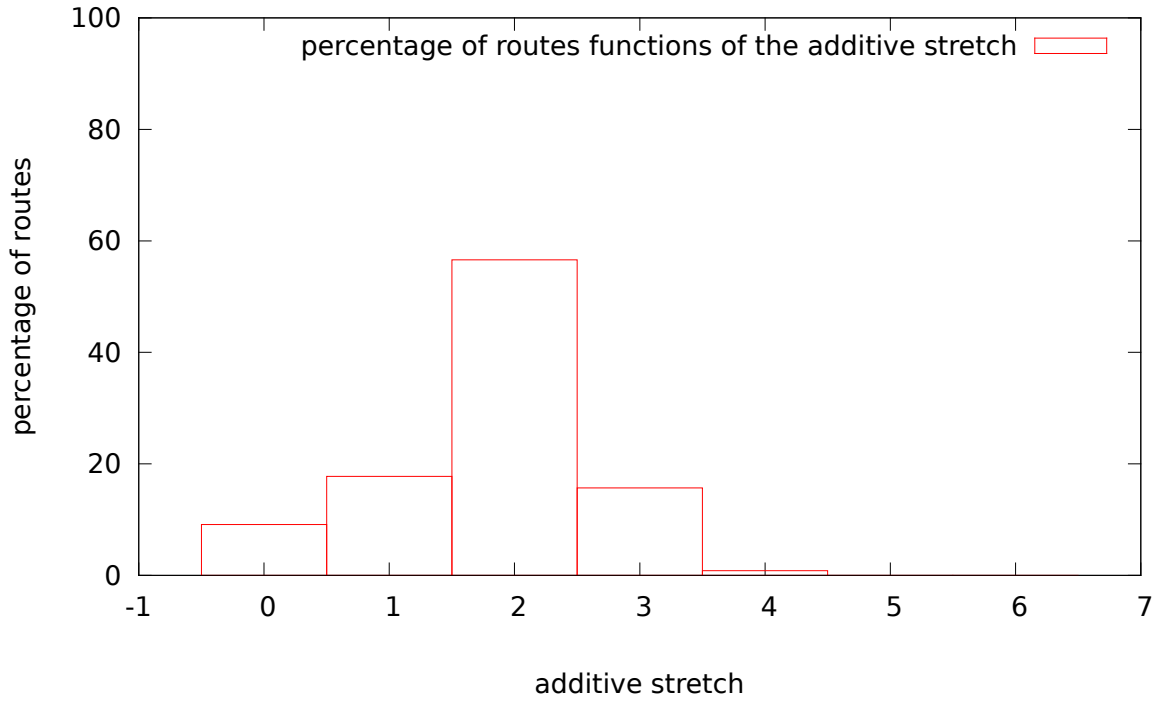

FIGURE 11 – Etirement additif

2.4.1 Charge des liens

La figure 14 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

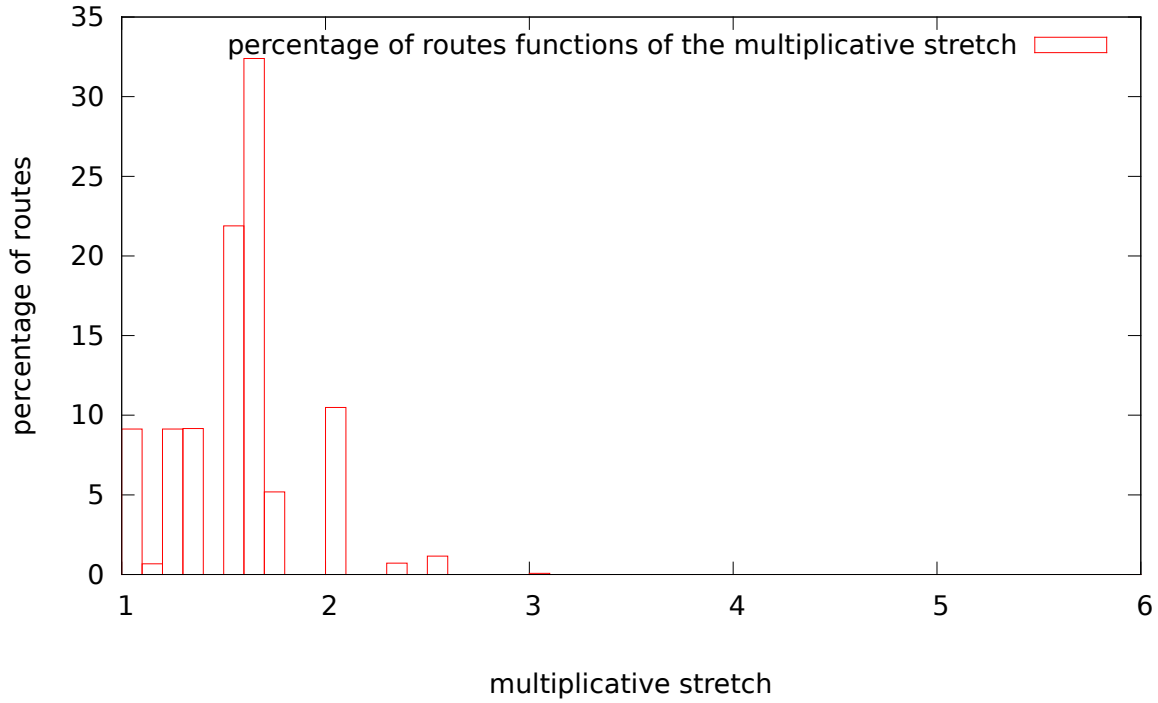

FIGURE 12 – Etirement multiplicatif

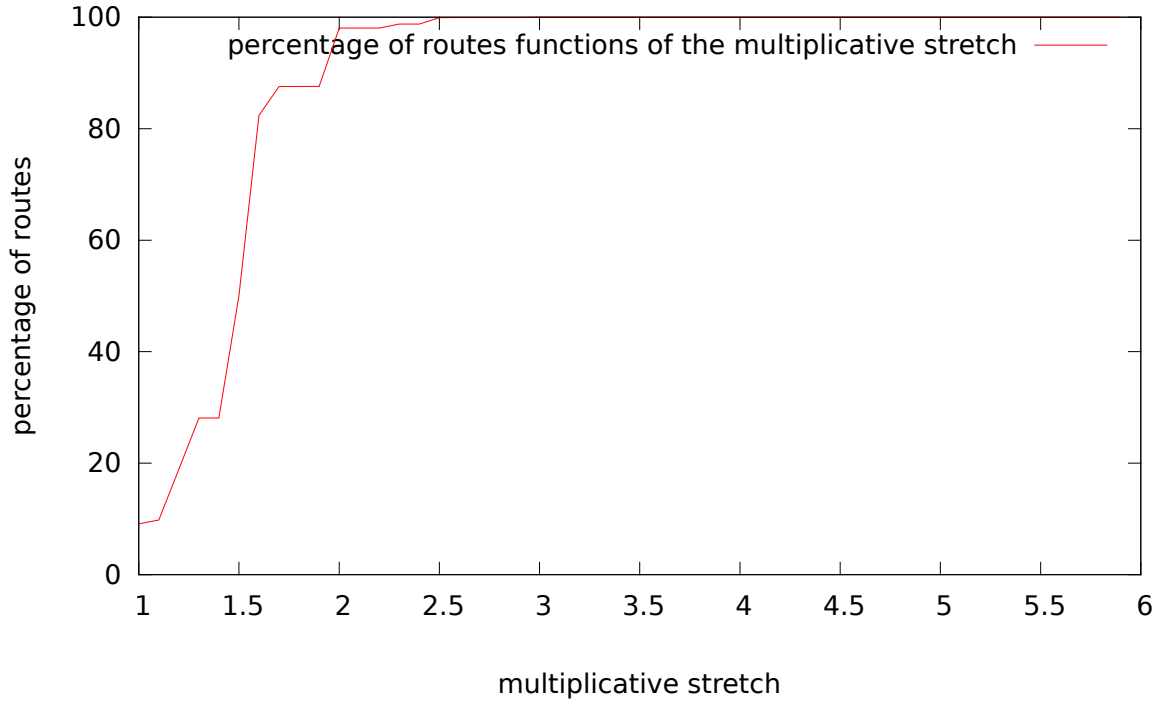

FIGURE 13 – Etirement multiplicatif

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.

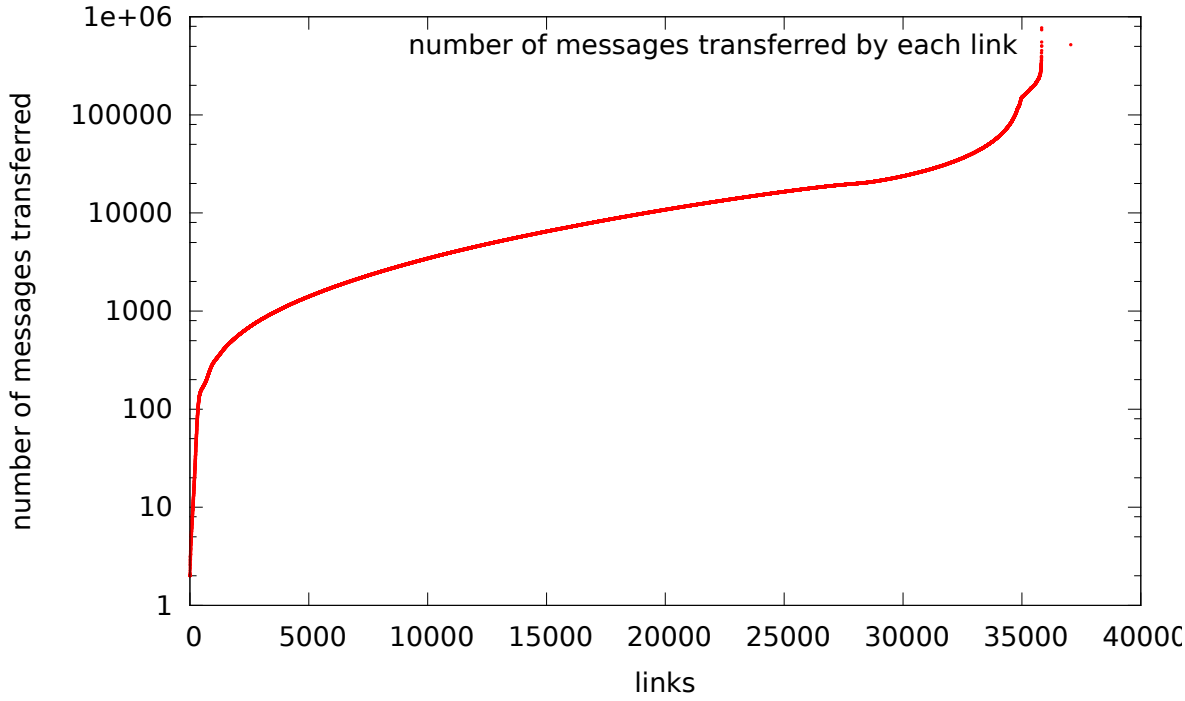

FIGURE 14 – Charge des liens

2.4.2 Charge des nœuds

La figure 15 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.

Nous avons également évalué la charge des nœuds en fonction de leur degré. Les résultats sont présentés sur la figure 16. Cette figure montre qu'il y a une corrélation de nature linéaire 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.

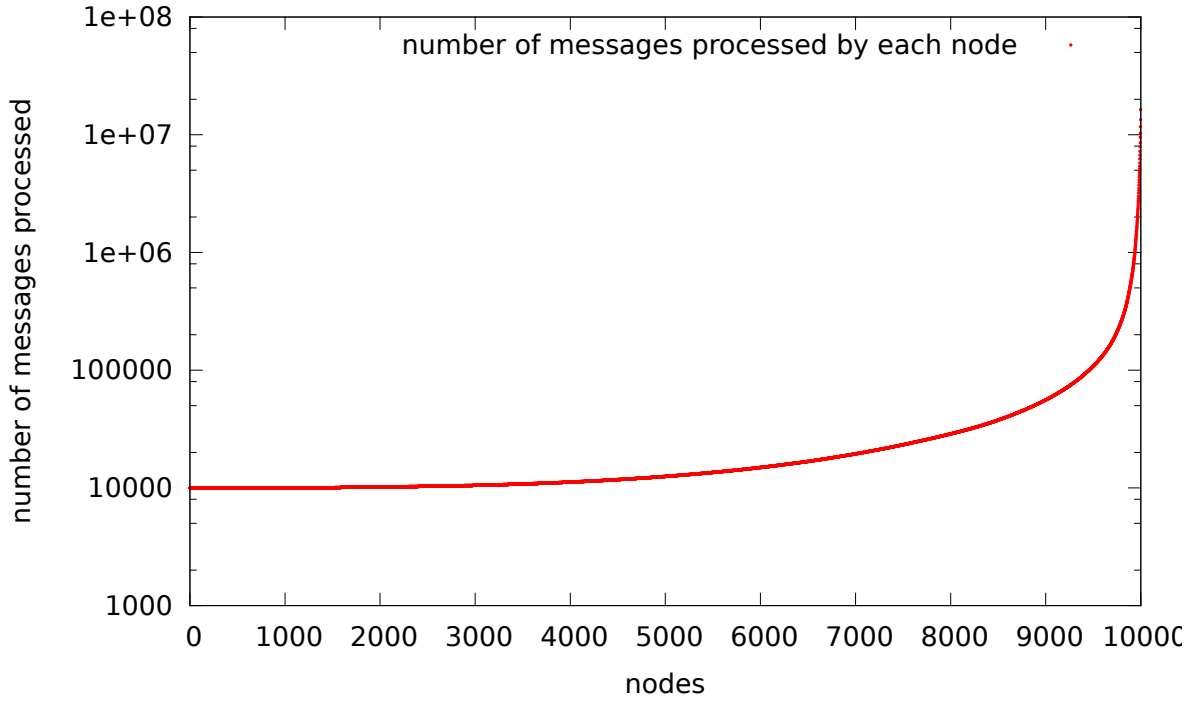

FIGURE 15 – Charge des nœuds

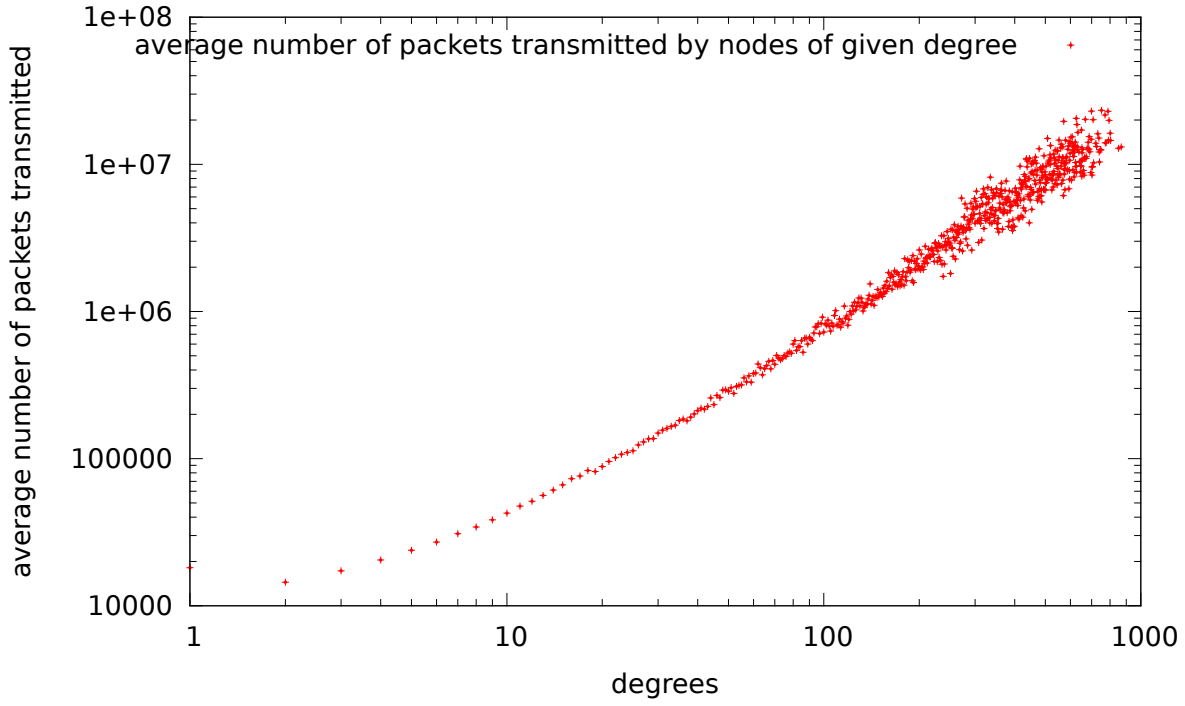

FIGURE 16 – Moyenne de la charge des nœuds par degré

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.

[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.