THESE DE DOCTORAT DE L'UNIVERSITE PARIS 6

Spécialité : Informatique

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITE PARIS 6

Présentée par

Frédéric Giroire

Sujet de la thèse :

Réseaux, algorithmique et analyse combinatoire de grands ensembles

soutenue le 29 novembre 2006

devant le jury composé de :

Philippe Flajolet	INRIA	directeur de these
Michèle Soria	Université Paris 6	directrice de these
Fabrice Guillemin	France Telecom R&D	rapporteur
Christian Lavault	Université Paris 13	rapporteur
Anne Doucet	Paris 6	examinateur
Jean-Claude Bermond	CNRS/INRIA	examinateur

Remerciements.

Je tiens à exprimer ma profonde reconnaissance à Philippe Flajolet qui m'a permis de découvrir un thème de recherche passionnant : l'analyse de petits algorithmes probabilistes et élégants. Je le remercie de m'avoir accueilli au sein de son équipe et d'avoir encadré mon travail pendant mes années de thèse avec pédagogie et gentillesse.

Je suis également très reconnaissant à Michèle Soria d'avoir accepté de diriger ma thèse.

J'aimerais remercier Christian Lavault et Fabrice Guillemin d'avoir accepté d'être les rapporteurs de ma thèse et pour leur lecture et commentaires.

Jean-Claude Bermond et Anne Doucet me font l'honneur d'être membres de mon jury et je les en remercie.

Mes plus chaleureux remerciements vont aux les membres du groupe ALGO de l'INRIA Rocquencourt : Bruno Salvy, Mireille Régnier, Philippe Dumas, mes homonymes Frédéric Chyzak et Frédéric Meunier, Alin Bostan, Carine Pivoteau, André Balsa, Vincent Puyhaubert, Julien Fayolle, Marianne Durand et Ludovic Meunier, ainsi que Virginie Collette pour sa gentillesse et son efficacité. Ils m'ont accueilli à bras ouverts dans leur labo pour effectuer mon travail de thèse et je les remercie pour leur disponibilité et leur aide précieuses.

J'ai une pensée particulière pour mes co-bureaux Antonio Vera, Pierre Nicodème et les canadiens du 923 Marni Mishna et Hâ Lè, ainsi que pour Éric Fusy.

J'adresse à nouveau un grand merci à Jean-Claude Bermond et également à Stéphane Pérennes de m'avoir accueilli il y a maintenant 5 ans pour un stage de maitrise qui m'a fait découvert la recherche et m'a amené à une collaboration fructueuse avec le projet MASCOTTE. Je remercie d'ailleurs tous les membres du groupe pour leur disponibilité et leur camaraderie, en particulier Frédéric Havet, Omid Amini et Florian Huc avec qui j'ai eu la chance de travailler.

Merci à Thomas Camara, avec qui j'ai partagé la condition de doctorant à Rocquencourt et sans qui ces trois années n'auraient pas été les mêmes.

Il me faut citer Aurélien Delpirou, géographe émérite, et Martin Garagnon, agent ou montagnard, qui m'ont forcé à les mettre dans ces remerciements.

J'ai une pensée émue pour celui qui fut mon collocataire pendant ces années, Philippe Nadeau.

J'embrasse mes parents et mes frères Christophe et Bruno, qui ont aussi été mes collocataires pendant pas mal d'années, et je pense très fort à mes quatre grands-parents.

Je tiens aussi à citer pêle-mêle Romuald, Georges, Vincent, Ronan, Mamadou, Bruce, Marc, Jessy et les membres des équipes de foot qui ont eu la malchance de voir s'exercer mes talents.

Et bien sûr, last but not least, j'envoie un petit bisou à Marion.

Résumé

Deux classes de problèmes algorithmiques motivés par les réseaux sont étudiés dans cette thèse.

Le premier problème étudié est d'estimer le nombre d'éléments distincts ou cardinalité de très grands multi-ensembles en utilisant une mémoire auxiliaire très petite. Le nombre d'applications de cette question très simple est étonnamment important. On peut citer en particulier la détection de certains types d'attaques dans les réseaux. Nous avons proposé de nouvelles familles d'algorithmes qui répondent à ce problème. Elles ont été validées par l'analyse mathématique et à l'aide de simulations sur du trafic réel.

Le second problème est la conception de réseaux embarqués dans les satellites efficaces. Ces réseaux doivent d'une part pouvoir tolérer un certain nombre de pannes de leurs composants et d'autre part être de petites tailles en raison de leur coût extrêmement élevé. Nous avons introduit un nouvelle classe de réseaux et proposé des constructions minimales pour de nombreux cas.

Abstract

Two classes of algorithmic problems motivated by network thematics are studied in this thesis.

The first problem is to estimate the number of distinct elements or cardinality of very large multisets while using a very small amount of auxiliary memory. The number of applications of this very simple question is surprisingly important. In particular, we may mention the detection of some kinds of attack against networks. We proposed new families of algorithms to answer this problem. They are validated by mathematical analysis as well as by simulations with real traffic.

The second problem is the design of efficient on-board networks in satellites. On one hand these networks must be able to tolerate a given number of mechanical failures of their components. On the other hand they should be of small sizes because of their extremely high cost. We introduced a new class of networks and proposed minimal constructions in lots of cases.

Table des matières

Chapitre 1

Introduction

Prologue.

PF - Comment connaître le nombre d'éléments distincts n d'un ensemble en utilisant une mémoire constante ?
FG (Il parait que les réponses à l'aide d'une question sont toujours les meilleures.) - Comment pourrait-on, quand on lit un élément, savoir qu'on l'a déjà vu s'il n'est pas stocké en mémoire —ce qui implique une mémoire linéaire en n ?
PF (Juste retour des choses) - Je te donne une petite aide. Peut-on compter le simple nombre d'éléments sans qu'une mémoire de $\ln n$ bits —mémoire nécessaire pour un simple compteur— ne soit disponible ?
FG - Super comme aide.
PF - Il s'avère qu'un un petit algorithme très simple et très élégant proposé par Morris [Mor77] et analysé par Flajolet [Fla85] peut résoudre ce problème. Le principe est le suivant : on initialise un compteur K à 0. Quand on lit un élément on tire à pile ou face de façon biaisée : on obtient pile avec probabilité 2^{-K} et face avec probabilité $1 - 2^{-K}$. On incrémente le compteur de un si on tombe sur pile, sinon on ne fait rien. Que se passe-t-il ? Intuitivement, il faut en moyenne 1 coup pour que le compteur aille de 0 à 1, 2 coups de 1 à 2, 4 coups de 2 à 3 et donc un état κ est atteint en $2^\kappa - 1$ coups. Un ensemble de n éléments ne fait donc avancer le compteur que d'en gros $\log_2 n$ coups. L'algorithme retourne donc 2^K comme estimée de la valeur de n. L'idée cruciale à retenir ici est que le *choix probabiliste* de l'incrément permet de *répondre de façon approchée* à la question en utilisant une *mémoire très faible* —La mémoire utilisée par l'algorithme n'est plus de $\log_2 n$, mais de $\log \log n$. Pour indication $\log \log 10^{25} \approx 4$ et ainsi, avec une toute petite mémoire de quelques kilo bits, on peut compter à peu près tout dans l'univers.
PF - Alors peut-on compter le nombre d'éléments distincts sans stocker les éléments ?

FG (émerveillé) - Et vous qu'en pensez-vous ?

Deux classes de problèmes algorithmiques motivés par les réseaux sont étudiées dans cette thèse. D'une part, la surveillance de réseaux met en jeu de grands ensembles de données liés aux connexions établies par les routeurs qu'il s'agit d'analyser quantitativement. Philippe Flajolet m'a introduit le problème et aidé au long de son étude. D'autre part, certains réseaux embarqués dans les satellites doivent pouvoir tolérer des pannes de leurs composants et il faut concevoir de tels réseaux de petites tailles pour optimiser leur coût. Cette thématique m'a été expliquée par Jean-Claude Bermond et Stéphane Pérennes et m'a mené à une collaboration avec le projet Mascotte de l'INRIA. Les deux problèmes ont une nature combinatoire et ils font appel, entre autres, à des traitements probabilistes. Ils font aussi une belle part à des études asymptotiques.

1.1 Algorithmes probabilistes de comptage

1.1.1 Aperçu du problème et contexte

Un *multi-ensemble* est un ensemble dans lequel les éléments peuvent être répétés. Sa *taille* N est le nombre de ses éléments et sa *cardinalité* n est le nombre de ses éléments *distincts*. Le problème considéré ici est de compter le nombre d'éléments distincts de très grands multi-ensembles. Ce problème est apparu dans les années 70 dans le contexte des bases de données, dans le but d'optimiser différentes requêtes (union, intersection, insertion, tri). Le volume gigantesque des données ne permet pas l'utilisation des algorithmes exacts. En effet, ces algorithmes utilisent une mémoire linéaire en n : ils lisent les mots un par un et pour chacun ils vérifient dans un dictionnaire —vide au départ— si le mot est présent. Si oui ils ne font rien et passent au mot suivant, si non ils rajoutent le mot dans le dictionnaire et incrémentent de un un compteur.

Dans les années 90, le problème a connu une seconde jeunesse avec le développement rapide de réseaux à très grandes capacités. Dans ce contexte, les éléments sont les paquets qui arrivent à un routeur ou qui passent sur un lien internet. Un paquet appartient à une connexion identifiée par un sous-ensemble des champs de son en-tête (habituellement par le couple adresse IP de la source, adresse IP de la destination). Compter le nombre d'éléments distincts revient ici à compter le nombre de connexions actives sur le lien. Or il s'avère que connaître ce nombre de connexions à un nombre d'applications étonnamment important dans les domaines de la surveillance et de la sécurité des réseaux (network monitoring[1] and security). C'est tout d'abord une information simple mais pertinente sur la nature du trafic. En effet, par exemple, un trafic généré essentiellement par des courriels et des requêtes web

[1] L'équivalent anglais de certains termes sera quelques fois indiqué entre parenthèses quand celui-ci est d'un usage international courant.

est constitué de connexions de seulement quelques paquets. A l'opposé, un trafic pair-à-pair (peer-to-peer) comportera beaucoup de paquets mais peu de connexions.

Ensuite, certains types d'attaques peuvent être détectés en étudiant l'évolution du nombre de connexions. Lors d'une attaque par déni de service (ou DoS attack pour Denial of Service attack), un virus, qui a infecté de nombreuses machines —jusqu'à plusieurs centaines de milliers— lors d'une phase initiale de propagation, lance tout à coup une pluie de requêtes à partir de ces machines en direction d'un serveur cible. Ce dernier ne peut pas faire face à cet afflux de demandes, se retrouve saturé et ne peut plus assurer son service normal. Il ne reste souvent plus qu'à l'éteindre. Pour détecter ce type d'attaques, il ne suffit pas de regarder les statistiques sur les paquets. En effet l'augmentation du nombre de connexions, même brusque, pourrait être masqué par un fort trafic. Il est plus efficace d'utiliser des statistiques sur le nombre de connexions. Estan et Varghese, dans [EVF03], recensent ainsi cinq utilisations majeures de ces statistiques : détecter des recherches d'intrusion (port scan), détecter les attaques par déni de service, effectuer des mesures générales, estimer la vitesse de propagation d'un ver et faire de la gestion de paquets (packet scheduling) à haute vitesse.

Les réseaux cœurs (backbone networks) sont aujourd'hui pour la plupart des réseaux optiques synchrones décrits par le standard international SDH[2] (pour Synchronous Digital Hierarchy). Dans ces réseaux, les liens entre routeurs sont des faisceaux de fibres optiques classifiés selon leur capacité, des STM-0 (51,84 Mbps[3]) aux STM-256 (40 Gbps), qui commencent à être déployés, en passant par les STM-64 (10 Gbps)[4], qui sont utilisés abondamment. À ces vitesses, même stocker les données devient problématique. Un disque dur d'un Tera octet (To) est rempli en moins de 4 minutes et un paquet peut arriver toutes les 60 nano secondes (ns) ne laissant que 150 instructions machine à un processeur très rapide de 2,5 Giga Hertz (GHz) pour traiter chaque paquet.

*Dans ces conditions, les seuls algorithmes qui peuvent être utilisés ne doivent utiliser qu'*une mémoire constante *et ne faire qu'*une passe très simple *sur les données. De plus, on ne fait ici* pas d'hypothèses sur le trafic *et en particulier sur la structure et le nombre des répétitions*, contrairement à beaucoup de travaux où, par exemple, un trafic poissonnien est supposé.

L'idée cruciale, développée en premier par Flajolet et Martin pour leur algorithme pionner Probabilistic Counting [FM83], est de relaxer le problème : pour de nombreuses utilisations algorithmiques, il n'est pas nécessaire de connaître le nombre exact d'éléments distincts, mais une estimation avec une bonne précision suffit. L'idée est ensuite d'utiliser des *estimées probabilistes* qui ne dépendent que de n et pas du nombre de répétitions.

De nombreux travaux ont été faits sur la gestion de requêtes approchées (approximate query processing) dans la communauté des bases de données (voir [Gib01], [GTK01],

[2]Un autre standard est utilisé aux États-Unis et au Canada :SONET pour Synchronous Optical NETworks.

[3]Unité de mesure du débit : Mbps pour Mega bit par seconde et Gbps pour Giga bit par seconde.

[4]Les équivalents SONET sont OC-1, OC-768 et OC-192.

[CLKB04]). Dans [WZT90] Whang, Zanden et Taylor ont introduit le *Comptage linéaire (Linear Counting)*. Le principe est de répartir des valeurs hachées dans des boîtes (buckets) et d'utiliser le nombre de boîtes remplies pour obtenir une estimée du nombre de valeurs lues. Mais la mémoire utilisée est ici linéaire. Pour pouvoir étudier de très grands ensembles de données, Estan, Varghese et Fisk ont étendu ce principe en proposant une version multi-échelles dans leur algorithme MULTIRESOLUTION BITMAP [EVF03]. L'idée est de ne garder qu'un petit nombre de fenêtres sur le bitmap entier. Leur estimée utilise m mots de mémoire et a une erreur standard de $4.4/\sqrt{m}$. L'échantillonnage est une autre façon d'estimer des cardinalités. L'idée est ici de ne garder qu'une fraction des valeurs déjà lues. Cette fraction peut être choisie dynamiquement de manière élégante comme dans l'algorithme d'*Échantillonnage adaptatif (Adaptive Sampling)* de Wegner, qui a été décrit et analysé par Flajolet dans [Fla97]. La précision de cette méthode est $1.20/\sqrt{m}$. Gibbons propose dans [Gib01] une autre approche, l'*échantillonnage distinct (Distinct Sampling)*, pour pouvoir donner des réponses approchées à des questions variées en utilisant une fraction des données d'une base de données. L'algorithme de *comptage probabiliste (Probabilistic Counting)* de Flajolet et Martin (voir [FM83]) utilise des motifs sur la représentation binaire de nombres. Il a d'excellentes propriétés statistiques avec une erreur proche de $0.78/\sqrt{m}$. L'algorithme de Durand et Flajolet, *LogLog Counting* (voir [DF03]), a la même idée de départ mais utilise une observée différente. L'erreur standard est de $1.30/\sqrt{m}$ pour la première version et de $1.05/\sqrt{m}$ pour la version Super-LogLog, mais les m 'mots' mémoires ont ici une taille en $\log\log n$ et non en $\log n$. Enfin dans [BYJK$^+$02] les auteurs présentent trois algorithmes pour compter les éléments distincts. Le premier utilise le $k-$ième minimum. Les auteurs prouvent que cet algorithme $(\epsilon, \delta)-$approxime n en utilisant une mémoire de $O(1/\epsilon^2 \log m \log(1/\delta))$ bits et un temps processeur de $O(\log(1/\epsilon) \log m \log(1/\delta))$ par élément. Cet algorithme est le point de départ de notre étude. Notre approche généralise cette idée en introduisant des familles d'estimateurs nouvelles et plus efficaces et donne une analyse précise de celles-ci.

1.1.2 Notre approche.

Contrairement à la plupart des algorithmes existants [WZT90, FM83, DF03, EVF03] qui s'intéressent à des propriétés de la représentation binaire de nombres, nous allons utiliser des statistiques d'ordre.

L'idée de départ est que le minimum M de n variables aléatoires tirées uniformément dans l'intervalle réel [0,1] a pour espérance $\mathbb{E}[M] = 1/(n+1)$ (voir la Section 2.1 pour le calcul et la Figure 2.1 pour une intuition géométrique). Le minimum donne donc une idée du nombre de valeurs qui ont été vues. Intuitivement, il a plus de chance d'être petit si n est grand.

Un multi-ensemble idéal de cardinalité n est un multi-ensemble constitué de n réels tirés uniformément dans l'intervalle [0,1], répliqués arbitrairement et mélangés par une permutation arbitraire. Le deuxième point crucial est que le minimum d'un multi-ensemble

idéal peut être déterminé en une passe sur les éléments et qu'il ne dépend ni de leur ordre, ni du nombre des répétitions. C'est le minimum de l'ensemble sous-jacent.

Le troisième point décisif est l'existence d'une fonction de hachage h qui simule un tirage uniforme sur [0,1]. Les valeurs hachées sont bien réparties sur l'intervalle et "ont l'air" indépendantes. Ce pseudo-aléa est largement suffisant en pratique (voir les études de Knuth [Knu98], et de la Section 3.2.1 p. 66).

Ces trois points donnent le principe d'un premier algorithme pour déterminer la cardinalité inconnue n d'un multi-ensemble. On lit les éléments de notre multi-ensemble et, grâce à la fonction de hachage, on se ramène à un multi-ensemble idéal. Au cours de la lecture, on actualise le minimum des valeurs hachées déjà vues. À la fin de la lecture, on obtient une estimée de n en utilisant le minimum. Le point remarquable à retenir ici est que l'utilisation d'une mémoire d'*un seul nombre réel* donne de l'information sur les n que l'on a vus.

Quand le multi-ensemble a été lu en entier, nous disposons du minimum des valeurs hachées. Nous voulons maintenant une estimée de n et non de $\frac{1}{n+1}$. Le plus naturel serait de prendre l'inverse de ce minimum, mais il s'avère que cet inverse a une espérance infinie. Notre solution est d'obtenir une estimée indirectement à partir de ce minimum en combinant deux principes : au lieu de prendre le premier minimum, on utilise le deuxième, le troisième ou le $k-$ième et, au lieu de prendre la seule fonction inverse, on la combine avec des fonctions sous-linéaires comme logarithme ou racine carrée. On obtient ainsi, en Section 2.1 p. 32, *trois familles d'estimateurs* de n : l'inverse du $k-$ième minimum, son logarithme et sa racine carrée.

1.1.3 Contribution

La première partie de cette thèse, constituée des trois premiers chapitres, porte sur les algorithmes probabilistes de comptage de cardinalité. C'est une *étude complète d'algorithmes* : de la conception, en passant par l'analyse mathématique et l'implémentation optimisée, pour aboutir à la validation expérimentale et l'étude d'applications pratiques dans divers domaines. Si ces différentes étapes ont un ordre naturel, il existe aussi un dialogue fécond entre elles. Ainsi, par exemple, comme nous l'avons évoqué, c'est l'analyse mathématique de l'inverse du premier minimum qui nous a amené à introduire de nouvelles familles d'estimateurs. De même, c'est le besoin pour certaines applications pratiques d'avoir aussi une bonne précision pour des ensembles pas trop grands (problème des petites cardinalités) qui amène à regarder le comptage par collisions (Section 3.1).

Dans le premier chapitre nous introduisons de nouvelles familles d'algorithmes pour estimer le nombre d'éléments distincts de très grands multi-ensembles (voir Figure 2.1.2 p. 34). Ces estimateurs utilisent des statistiques d'ordres alors que la plupart des algorithmes connus se basent sur des motifs de tableaux de bits. Ces algorithmes sont analysés mathématiquement. Il est prouvé (Théorème 1 p. 42) que leurs estimateurs ξ sont des estimateurs asymptotiquement non biaisés de n et leur précision est exprimée par leur erreur

standard définie comme $\sqrt{\xi}/n$. Ils atteignent une précision de $1/\sqrt{m}$ en utilisant une mémoire de m nombres flottants, ce qui les placent dans la classe des meilleurs algorithmes connus. Leur boucle interne très simple leur donne un avantage en termes de temps d'exécution Ces estimateurs ont été validés sur des fichiers de différents types et de différentes tailles.

Dans le Chapitre 3, nous étudions pour l'un de ces algorithmes, MINCOUNT*, le passage de la description sur papier à une implémentation efficace. Ce passage pose des problèmes non seulement pratiques (choix de la fonction de hachage, écrire un programme rapide,...), mais aussi théoriques. En effet pour résoudre le problème des petites cardinalités, nous avons introduit en Section 3.1 p. 52 de nouveaux estimateurs (EXTENDED HIT COUNTING) qui sont analysés dans le Théorème 6 p. 59. À la fin de l'étude de la Section 3.1.4, on dispose d'un algorithme qui pour chaque taille de fichier, de quelques unités à plusieurs centaines de millions, choisit un estimateur approprié qui donne le nombre d'éléments distincts *n à quelques pourcents près.* (2% quand $m = 2^{10}$, en utilisant $3m$ flottants de 32−bit, ce qui correspond à $96kB$, voir la Figure 3.4 p. 61.) L'introduction du EXTENDED HIT COUNTING pour certains taux $\lambda := n/m$, améliore la précision d'un facteur proche de 4 —en comparaison du HIT COUNTING classique.

Du point de vue pratique, nous expliquons en Section 3.2 les différentes étapes qui nous ont amené à une implémentation optimisée de l'algorithme. Ces étapes (choix de la structure du programme, gestion des entrées sorties,...) ont permis de faire diminuer d'un facteur supérieur à 10 le temps d'exécution. L'implémentation a été chronométrée et validée en Section 3.2.3 sur des fichiers de différents types. Elle n'est que 3 à 4 fois plus lente que la commande unix `cat -t` qui ne lit que les caractères de son entrée.

On utilise cette implémentation pour simuler deux types d'applications réseaux et bioinformatiques. En Section 3.3, on regarde comment mettre en place un système d'alerte automatique d'attaques par déni de service. En Section 3.4, on a montré que les algorithmes probabilistes de comptage peuvent être employés pour mesurer la corrélation des bases de différentes zones du génome humain, voir la Figure 3.4 p. 79. Cela pourrait être utilisé, par exemple, pour déterminer en une passe très rapide certaines localisations possibles de zones codantes.

Le Chapitre 4 est le fruit d'une collaboration avec Éric Fusy du projet ALGO de l'INRIA. On s'y intéresse au comptage de cardinalité dans le contexte des flux de données. Les ensembles étudiés ne sont plus fixes. Une requête typique est "combien d'éléments distincts du flux ont été vus dans la dernière fenêtre d'une heure ? " Or sur un lien de capacité 40 Gbps, en supposant une taille moyenne de paquets de 300 octets, 60 milliards de paquets peuvent arriver en une heure. La difficulté pour adapter les méthodes avec statistiques d'ordre est que si on veut disposer du minimum pour chaque fenêtre, il semble impossible de stocker les valeurs qui peuvent devenir un minimum dans le futur. On présente un algorithme SLIDING MINCOUNT. C'est une nouvelle méthode pour donner une estimation du nombre n de flots actifs parmi les paquets d'un flux de données dans une fenêtre coulissante. Il atteint une précision relative de l'ordre de $1/\sqrt{m}$ en utilisant en gros une mémoire d'ordre

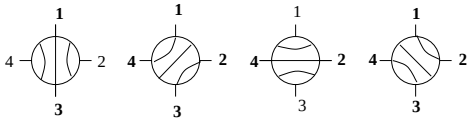


FIG. 1.1 – Un switch peut être dans quatre états différents.

$m \ln(n/m)$ mots, où n est une borne supérieure du nombre de flots vus sur la fenêtre coulissante (Théorème 8 p. 88 Proposition 4 p. 91). Par exemple, une mémoire de seulement $64kB$ est suffisante pour maintenir une estimée d'une précision de l'ordre de 4 pourcents pour un flux de plusieurs millions de flots. Cet algorithme est validé par l'étude du trafic d'un routeur passerelle de l'INRIA (Figures 4.7 p. 96 et Figures 4.8 p. 97). Mentionnons qu'il est aussi possible d'adapter les algorithmes probabilistes basés sur les motifs de bits (bitmap patterns) [DF03, FM83] aux fenêtre coulissantes. Le procédé est expliqué rapidement dans l'article de Datar et al [DGIM02]. Ils obtiennent le même ordre de complexité que notre algorithme, bien qu'ils ne donnent pas d'analyse détaillée. Par contre, nous donnons une analyse poussée de la mémoire requise. Cela inclut l'analyse de la mémoire maximale nécessaire sur une très longue période de temps (Théorème 9 p. 92), une question importante si l'on veut être sûr qu'il n'y ait pas de dépassement des capacités de la mémoire.

1.2 Conception de réseaux embarqués tolérants aux pannes minimaux

1.2.1 Aperçu du Problème

Motivation. Le second problème a été posé originellement par la branche spatiale d'Alcatel. Les signaux arrivant à leurs satellites de télécommunication au travers de ports doivent être amplifiés avant de pouvoir être renvoyés en direction de la terre. Les ports sont donc reliés à des amplificateurs. Mais ces derniers s'usent et peuvent tomber en panne, alors que le service de réparation Darty n'est pas assuré dans l'espace. Il faut donc mettre un surplus d'amplificateurs dans le satellite. Les ingénieurs estiment que, si le satellite est prévu pour une durée de vie de x années, les pannes ne dépasseront pas un nombre fixé noté k. Les ports sont reliés aux amplificateurs au travers d'un réseau composé de commutateurs, que nous nommerons switchs dans la suite, connectés entre eux par des liens. Les switchs sont de gros objets circulaires auxquels on peut connecter quatre liens et dont une partie peut effectuer une rotation permettant les différentes configurations de connexions montrées en Figure 5.1. D'autre part le satellite tourne sur lui-même et à chaque instant tous les ports ne sont pas orientés de façon à pouvoir capter un signal. Un surplus de ports est donc aussi nécessaire. Le problème est donc de router p signaux d'un sous-ensemble des entrées vers un sous-ensemble des sorties. Les chemins doivent être disjoints en arêtes. Construire des

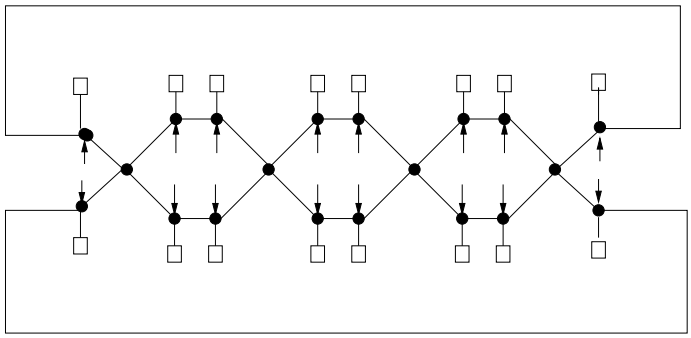


FIG. 1.2 – Réseau-$(12, 4, 4)$ valide et minimum.

réseaux satisfaisant ces propriétés n'est pas très difficile. Un réseau de permutation fait par exemple très bien l'affaire. Mais les switchs, en plus d'être de gros bidules tournants, sont très coûteux à construire et forment une machinerie très lourde. Or le poids du réseau est un élément déterminant du coût de lancement du satellite. Les ingénieurs et commerciaux d'Alcatel Space estiment que le coût induit par un switch se chiffre en centaines de milliers d'euros. Dans ces conditions le problème devient de concevoir des réseaux tolérants aux pannes *minimaux* en nombre de switchs et économiser ne serait-ce qu'un seul commutateur vaut la peine. Les industries spatiales sont intéressées par la conception de tels réseaux pour des valeurs spécifiques des paramètres. "On [Alcatel Space] a un réseau-$(20, 8, 5)$ avec 50 switchs. Est-ce que vous avez mieux ?" Cependant la théorie générale est intéressante en elle-même.

Problème Nous considérons ici des *réseaux*, c'est-à-dire des graphes reliant des entrées à des sorties. On définit en Section 5.1 p. 104 un *réseau*-(p, λ, k) comme un graphe non orienté 4-régulier ayant $p + \lambda$ entrées et $p + k$ sorties. Un réseau-(p, λ, k) est dit *valide* si, pour tout sous-ensemble E des entrées de taille p et pour tout sous-ensemble S des sorties de même taille, il existe p chemins disjoints en arêtes reliant E à S. Le problème est de déterminer $N(p, \lambda, k)$ le nombre *minimal* de switchs d'un réseau-(p, λ, k) valide et de donner des *constructions* de tels réseaux. Pour des raisons de symétrie, on suppose dans la suite que $\lambda \leq k$ et on note $n := p + k$.

Les entrées du réseau correspondent aux ports des satellites, les sorties aux amplificateurs et les nœuds aux switchs. Les entrées sont représentées par des $\rightarrow$ dans les figures, alors que les sorties le sont par des $\square$. La Figure 1.2 montre un réseau-$(12, 4, 4)$ valide. Ce réseau a 16 entrées et 16 sorties. Il tolère 4 pannes. Quels que soient les 4 sorties en panne, on peut relier n'importe quel sous-ensemble de 12 entrées à des sorties valides. Il a 20 switchs et est minimum.

1.2.2 Position de notre étude

Si le problème a un côté réjouissant de casse tête chinois ou de recette de cuisine — Quelles sont les bonnes proportions d'entrées et de sorties ? Et si je rajoutais quelques doublons à un cycle hamiltonien ?—, il fait aussi appel à des notions importantes de théorie des graphes —expansion, maille,...— et s'insère aussi dans une tradition riche d'analyse de familles de réseaux classiques. Cette étude doit ainsi être considérée dans l'esprit de la théorie des *superconcentrateurs (superconcentrators)*. Un *concentrateur*-$(p+\lambda, p)$ *(concentrator)* [Val75] est un graphe orienté acyclique $G = (V, E)$ avec $p+\lambda$ nœuds entrées désignés et p nœuds sorties désignés ($\lambda \geq 0$), tel que pour chaque sous-ensemble de p nœuds entrées, il existe p chemins disjoints en arêtes (ou, selon le contexte, disjoints en sommets) qui relient les entrées aux sorties. Un *sélecteur (selector)*, introduit pour la première fois dans [BDD02], est un réseau-$(p, \lambda, 0)$ ($k = 0$). Une théorie générale des sélecteurs peut être trouvée dans [BPT], où plusieurs résultats sont obtenus pour de petites valeurs de λ. Un sélecteur peut être vu comme une version non orientée d'un concentrateur-$(p+\lambda, p)$. Un *n-superconcentrateur* [Pip77] est un graphe orienté acyclique $G = (V, E)$ avec n nœuds entrées désignés et n nœuds sorties désignés, tel que, pour tout ensemble S de $p \leq n$ entrées et tout ensemble T de p sorties, il existe p chemins disjoints en arêtes (ou disjoints en sommets, selon le contexte) reliant S à T. Il existe une vaste théorie des superconcentrateurs (Voir le travail pionner de Pippenger [Pip77] pour une introduction, [AM84, AGM87, AC03] et Schöning [Sch06] pour des constructions, et [BKP$^+$81] pour les problèmes de complexité).

En ce sens, les réseaux-(p, λ, k) généralisent le concept de sélecteurs de la même manière que les superconcentrateurs généralisent celui de concentrateurs. Il est aussi clair que, prendre un superconcentrateur (resp. un concentrateur-$(p+\lambda, p)$) en oubliant les orientations, procure un réseau-(p, λ, k) valide pour n'importe quelle valeur de $k = \lambda$ (resp. n'importe quel λ et $k = 0$). On peut ainsi essayer d'utiliser des superconcentrateurs connus pour construire des réseaux embarqués efficaces, mais, et ce n'est pas étonnant, cela ne donne généralement pas des réseaux minimaux. La différence majeure entre les superconcentrateurs et les réseaux valides généraux est que dans un réseau valide le nombre d'entrées et de sorties qui peuvent tomber en panne est borné (dans notre cas, étant donné une probabilité de panne, pas plus de k pannes n'arriveront au cours de la durée de vie du satellite).

Trouver un réseau minimal (un superconcentrateur de faible densité ou ayant un nombre minimal de sommets) est un problème difficile et un domaine actif de recherche (voir les papiers [AM84], [AGM87], [AC03] et la construction récente de [Sch06]). D'un point de vue algorithmique, on peut montrer que le problème de tester si un graphe est un concentrateur-$(p+\lambda, p)$ ou un superconcentrateur est co-NP complet. En fait, ce sont des problèmes complets même quand ils sont réduits au cas spécial important des graphes de taille linéaire par rapport au nombre d'entrées (voir [BKP$^+$81]). On peut montrer, avec le même type d'arguments, que le problème de décider si un réseau-(p, λ, k) est valide ou non est coNP-complet, même réduit aux réseaux 4-réguliers.

1.2.3 Notre contribution

Nous avons effectué une *formalisation d'un problème industriel pratique*. Celui-ci a des *applications majeures*, comme la conception des séries de satellites Eutelsat et Astra destinées aux transmissions télévisuelles et de vidéos et il conduit à des questions régulières destinées au groupe Mascotte de l'INRIA. Cela nous a conduit a définir une *nouvelle famille de réseaux*, les *réseaux-*(p, λ, k) (Section 5.1 p. 104). Les techniques de recherches de bornes pour des réseaux valides minimaux incluent, entre autres, l'utilisation de réseaux classiques (réseaux de permutations, expandeurs,...) et des techniques d'analyse probabilistes.

Dans le Chapitre 5, en collaboration avec J.-C. Bermond et S. Pérennes, nous avons introduit un critère simple qui caractérise la validité des réseaux, le *critère de coupe* (Proposition 5 p. 105). Ce critère est essentiel pour les preuves de bornes inférieures et pour prouver la validité des constructions qui donnent des bornes supérieures. En effet, il permet de montrer que certains motifs sont interdits. Par exemple, on définit un *doublon* comme étant un switch avec une entrée et une sortie. Dès que $k \geq 3$, trois doublons ne peuvent être reliés directement puisque, en cas de panne des trois sorties, les trois entrées ne pourraient être évacuées par les trois arêtes sortantes. On étudie ainsi les motifs possibles entre différentes sortes de switchs et on en déduit des bornes inférieures pour $N(p, \lambda, k)$. On prouve des bornes inférieures sur le nombre de switchs pour de petites valeurs du nombre de pannes k. On donne aussi des constructions, et donc des bornes supérieures proches d'être minimales pour $k \leq 8$, voir la Figure 5.5 p. 109. Enfin on donne une construction générale de réseaux valides pour n'importe quel λ et k.

Le Chapitre 6 est le fruit d'une collaboration avec Omid Amini, Florian Huc, et Stéphane Pérennes du projet Mascotte de l'INRIA. On s'y intéresse aux grands réseaux pour lesquels $n := p + k$ tend vers l'infini et k est assez grand. Nous donnons des bornes asymptotiques linéaires pour $N(p, \lambda, k)$ pour de nombreux cas. On rappelle que pour des raisons de symétrie on suppose $\lambda \leq k$.

Le problème s'avère très sensible à l'ordre de λ et k. Quand λ et k sont petits devant p, le problème se réduit à éviter certaines configurations locales interdites (voir Théorème 5 p. 136). La construction de réseaux valides optimaux repose très fortement sur les expandeurs. En utilisant ces derniers, nous sommes capables de construire pour une classe particulière de réseaux, les réseaux simplifiés (pour la définition voir la Section 5.1 p. 104), des réseaux avec $2n$ switchs dès que n est assez grand et $k \leq c_1 \log n$ pour une constante c_1 (Section 6.1.4). En Section 6.3.3 nous donnons aussi une borne inférieure d'ordre $2n(1-\epsilon(k))$ où $\epsilon(k)$ tend vers zéro quand k tend vers l'infini (mais $k \leq c_1 \log n$ n'est pas nécessaire). Ainsi, le problème est résolu dans le cas asymptotique pour les réseaux simplifiés pour $k \leq c_1 \log n$.

Pour les réseaux généraux, en utilisant des expandeurs bi-partie, nous obtenons une borne supérieure en $n + \frac{3}{4}n$ quand $k \leq c_2 \log n$ pour une constante c_2. La borne inférieure obtenue est $(n + \frac{2}{3}n)(1 - \epsilon(\lambda, k))$, et nous conjecturons que $n + \frac{3}{4}n$ doit être la vraie valeur.

Nous donnons aussi une construction de sélecteurs (cas $\lambda = 0$) de taille $n + \frac{n}{2}$, ce qui

nous donne dans ce cas aussi une borne inférieure serrée.

Pour étendre ces résultats pour des valeurs plus larges de k, nous définissons en Section 6.2 p. 128 une propriété d'expansion locale que nous appelons $\alpha-r$obustesse (voir [CRVW02, AHK99] pour des notions proches). Intuitivement, l'α-robustesse d'un graphe G est une version locale du facteur d'expansion. C'est l'entier maximal r_α tel que pour tous les petits sous-ensembles le facteur d'expansion est α, mais pour les sous-ensembles plus-grands leur nombre d'arêtes sortantes est au moins r_α. Pour nos constructions, il est nécessaire que le graphe soit un expandeurs pour les petits sous-ensembles, mais pour les plus grands on a juste besoin d'une connexité minimale constante avec le reste du graphe. Il s'agit d'une approche perpendiculaire à l'approche classique. On fixe un facteur d'expansion et on se demande jusqu'à quelle taille de sous-ensembles cette expansion est respectée. Cette notion est aussi intéressante en elle-même et contient comme cas particulier plusieurs invariants d'expansion comme la longueur de bi-section et la constante de Cheeger (voir [Chu97]). Comme les graphes aléatoires sont de très bons expandeurs, on étudie leur α-robustesse. De cette façon, on généralise le résultat de Bollobàs sur le facteur d'expansion des graphes aléatoires 4-réguliers. À notre connaissance, c'est la première fois qu'est défini et étudié ce nouveau concept d'expansion locale. On calcule la robustesse des graphes aléatoires 4-réguliers (Théorème 23 p. 130) qui mène à la construction d'un réseau$-(p, \lambda, k)$ valide avec $3n$ switchs pour $\lambda \leq k \leq \frac{n}{7}$ (Théorème 4 p. 129).

1.3 Travaux et publications

Liste des travaux publiés ou en cours

Le Chapitre 2 présente la construction, l'analyse mathématique et la validation de nouvelles familles d'estimateurs probabilistes de cardinalité. Les résultats qui y sont exposés ont fait l'objet de la publication

> [Gir05] F. Giroire, Order Statistics and Estimating Cardinalities of Massive Datasets. In *Proceedings of Discrete Mathematics and Theoretical Computer Science, 2005.*

Le chapitre 3 traite du passage d'un algorithme écrit sur papier à l'obtention d'une véritable implémentation. Différents thèmes et applications y sont traités. En particulier, le traitement des petites cardinalités en Section 3.1 p. 52 a été publié dans

> [Gir06b] F. Giroire. Extended Hit Counting to Estimate Cardinality. To appear in *Proceedings of WWW/Internet 2006, Murcia, Spain, october 2006.*

Les applications des algorithmes probabilistes de comptage de cardinalités au domaine biologique, présentées dans la Section 3.4 p. 75, ont fait l'objet d'une courte publication présentée sous forme de poster

> [Gir06a] F. Giroire. Directions to use probabilistic algorithms for cardinality for dna analysis. In *Proceedings of JOBIM, Bordeaux, France, july 2006.*

L'adaptation de nos algorithmes au contexte des flux de données est étudié dans le Chapitre 4. Ces travaux vont faire l'objet d'un publication

> [FG07] E. Fusy and F. Giroire. Estimating the number of Active Flows in a Data Stream over a Sliding Window. In *Proceedings of ANALCO, New Orleans, january 2007.*

Un résumé des résultats sur la conception de réseaux embarqués tolérants aux pannes peut être trouvé dans la publication

> [ABG+06] O. Amini, J-C. Bermond, F. Giroire, F. Huc, and S. Pérennes. Design of minimal fault tolerant networks : Asymptotic bounds. In *Proceedings of AlgoTel*, Trégastel, France, 2006.

Les constructions présentées en Chapitre 5 font l'objet d'un article en préparation

> [BGP06] J-C. Bermond, F. Giroire, et S. Pérennes. Design of minimal fault tolerant networks : Constructions.

Les résultats du Chapitre 6 sur le cas asymptotique seront publiés dans

> [AGHP06a] and [AGHP06b] O. Amini, F. Giroire, F. Huc, et S. Pérennes. Minimal selectors and fault tolerant networks.

En 2002, à l'occasion d'un stage de 6 mois au sein du groupe IP des laboratoires de recherche de Sprint[5] (Burlingame, Californie), j'ai travaillé sur la protection des réseaux backbones au niveau IP. Nous avons proposé un nouvel algorithme, qui utilise une méta-heuristique de recherche tabou, pour construire une topologie IP sur un réseau de fibres optiques. Cette construction est optimale dans le sens où elle maximise la résistance aux pannes du réseau en minimisant le partage des fibres, tout en respectant les impératifs de délai des opérateurs. En plus, l'algorithme prend en considération des contraintes pratiques comme le manque de longueurs d'onde disponibles ou les priorités entre différentes routes. Cette étude a conduit à la publication suivante et à deux brevets pour Sprint dont je suis un co-auteur.

> [GNTD03] F. Giroire, A. Nucci, N. Taft, and C. Diot. Increasing the robustness of ip backbones in the absence of optical level protection. In *Proceedings of IEEE INFOCOM, San Francisco, USA, april 2003.*
>
> Method and Systems for Idenditying Optimal Mapping in a Network F. Giroire, A. Nucci, N. Taft, and C. Diot. Filed on July 10, 2003. Sprint Docket Number #2315/SPRI.104359.
>
> Method and Systems for Correlating Practical Constraints in a Network. F. Giroire, A. Nucci, N. Taft, and C. Diot. Filed on July 10, 2003. Sprint Docket Number #2315/SPRI.104360.

[5] `http://www.sprintlabs.com/`

En 2003, pendant mon master, j'ai travaillé avec Mireille Régnier sur des sujets de bioinformatique. Il y a quatre étapes principales pour séquencer un génome : copier l'ADN plusieurs fois, la découper en petits morceaux, lire chacun de ces morceaux et reconstituer la séquence de base. Une des difficultés majeure rencontrée au cours de la quatrième étape est de distinguer les morceaux provenant des nombreuses zones répétées du génome. Nous avons proposé un nouvel algorithme pour séparer les répétitions, en utilisant une analyse à base de séries génératrices. Les mêmes techniques peuvent être utilisées pour repérer les polymorphismes d'une seule base (SNPs) qui sont déterminants, par exemple, pour la compréhension des maladies génétiques.

[Gir03] F. Giroire. Étude de problèmes combinatoires liés à l'analyse du génome : Séquençage et polymorphisme. In *Master Thesis*, 2003.

Première partie

Algorithmes probabilistes de Comptage de Cardinalité

Chapitre 2

Étude de familles d'estimateurs

mots-clés : *cardinalité, estimées, très grands multi-ensembles, statistiques d'ordre.*

Nous introduisons ici une nouvelle classe d'algorithmes pour estimer la cardinalité de très grands multi-ensembles qui utilisent une mémoire constante et ne font qu'une passe sur les données. Cette classe est construite à partir de *statistiques d'ordre* —et non pas à partir de motifs de la représentation binaire de nombres comme la plupart des algorithmes existants (Probabilistic Counting [FM83], LogLog Counting [DF03]). Nous analysons trois familles d'estimateurs. Ils atteignent une erreur standard de $\frac{1}{\sqrt{M}}$ en utilisant M unités de mémoire, ce qui les place dans la même classe que les meilleurs estimateurs connus. Ils ont une boucle interne très simple qui leur procure un avantage en terme de vitesse d'exécution. Les algorithmes sont validés par l'analyse probabiliste ainsi que par simulations et exécutions sur des traces du trafic internet.

Introduction

Problème et contexte. Un multi-ensemble est un ensemble où un élément peut apparaître plusieurs fois. La *cardinalité* n d'un multi-ensemble est son nombre d'éléments distincts, alors que sa *taille* N est son nombre total d'éléments en incluant les répétitions. Un problème important en informatique est d'estimer la cardinalité de multi-ensembles de très grande tailles : par exemple, estimer le nombre de valeurs distinctes prises par un attribut dans les entrées d'une base de données. Ce problème est apparu dans les années 80 motivé par l'optimisation d'opérations algorithmiques classiques sur les bases de données (union, intersection, tri...). Comme les ensembles de données considérés ont en général une taille N gigantesque, qui dépasse de beaucoup les capacités de la RAM, une condition naturelle est de traiter les données en une seule passe avec une boucle très simple et avec une mémoire auxiliaire très petite (constante ou logarithmique en N). Plus récemment, au début des années 2000, le problème a pris une seconde jeunesse avec le développement très

rapide des réseaux internet, et avec, de la nécessité d'en observer et surveiller le trafic. En particulier il est très utile de connaître le nombre de connexions distinctes auxquelles appartiennent les paquets IP qui transitent sur un lien. Mais les liens des réseaux cœur (backbone networks) ont pour certains une capacité de 40 Gbps —STM-256 des réseaux optiques SDH. À ces vitesses, le simple fait de stocker les données devient alors problématique, un disque dur de 1 TB étant rempli en moins de 4 minutes. Une étude sur les difficultés pour surveiller des liens à très hautes vitesses peut être trouvée dans [IDGM01]. Quelle est la première idée qui vient à l'esprit pour compter le nombre de mots distincts d'un texte par exemple ? On peut garder en mémoire les mots déjà observés et vérifier, pour chaque nouveau mot lu, s'il a été rencontré auparavant. C'est le principe de tous les algorithmes de comptage exact qui utilisent des dictionnaires plus ou moins performants pour stocker et rechercher les mots. Ils utilisent une mémoire en $O(n)$ et ont un temps d'exécution en $O(N \log n)$ —ou en $O(N)$ en utilisant du hachage. Le point crucial, développé en premier par Flajolet et Martin [FM83] dans leur algorithme Probabilistic Counting, est de relâcher la contrainte de donner le nombre exact n de valeurs distinctes. En effet, pour les problèmes algorithmiques, seule une estimée approchée de n avec une bonne précision est souvent nécessaire. L'idée est ensuite de construire des *estimées probabilistes* qui ne dépendent que du nombre n de valeurs distinctes et non de la multiplicité des valeurs du multi-ensemble. L'algorithme pionnier Probabilistic Counting et le récent LogLog Counting [DF03] construisent cette estimée en maintenant des tableaux de bits (bitmap patterns).

De nombreux travaux ont été faits sur la gestion de requêtes approchées (approximate query processing) dans la communauté des bases de données (voir [Gib01], [GTK01], [CLKB04]). Dans [WZT90] Whang, Zanden et Taylor ont introduit le *Comptage linéaire (Linear Counting)*. Le principe est de répartir des valeurs hachées dans des boîtes (buckets) et d'utiliser le nombre de boîtes remplies pour obtenir une estimée du nombre de valeurs lues. Mais la mémoire utilisée est ici linéaire. Pour pouvoir étudier de très grands ensembles de données, Estan, Varghese et Fisk ont étendu ce principe en proposant une version multi-échelles dans leur algorithme Multiresolution Bitmap [EVF03]. L'idée est de ne garder qu'un petit nombre de fenêtres sur le bitmap entier. Leur estimée utilise m mots de mémoire et a une erreur standard de $4.4/\sqrt{m}$. L'échantillonnage est une autre façon d'estimer des cardinalités. L'idée est ici de ne garder qu'une fraction des valeurs déjà lues. Cette fraction peut être choisie dynamiquement de manière élégante comme dans l'algorithme d'*Échantillonnage adaptatif (Adaptive Sampling)* de Wegner, qui a été décrit et analysé par Flajolet dans [Fla97]. La précision de cette méthode est $1.20/\sqrt{m}$. Gibbons propose dans [Gib01] une autre approche, l'*échantillonnage distinct (Distinct Sampling)*, pour pouvoir donner des réponses approchées à des questions variées en utilisant une fraction des données d'une base de données. L'algorithme de *comptage probabiliste (Probabilistic Counting)* de Flajolet et Martin (voir [FM83]) utilise des motifs sur la représentation binaire de nombres. Il a d'excellentes propriétés statistiques avec une erreur proche de $0.78/\sqrt{m}$. L'algorithme de Durand et Flajolet, *LogLog Counting* (voir [DF03]), a la même idée de départ mais utilise une observable différente. L'erreur standard est de $1.30/\sqrt{m}$ pour la première

version et de $1.05/\sqrt{m}$ pour la version Super-LogLog, mais les m 'mots' mémoires ont ici une taille en $\log\log n$ et non en $\log n$. Enfin dans [BYJK$^+$02] les auteurs présentent trois algorithmes pour compter les éléments distincts. Le premier utilise le $k-$ième minimum et correspond en gros à la Famille Inverse d'estimateurs. Les auteurs prouvent que cet algorithme $(\epsilon, \delta)-$approxime n en utilisant une mémoire de $O(1/\epsilon^2 \log m \log(1/\delta))$ bits et un temps processeur de $O(\log(1/\epsilon) \log m \log(1/\delta))$ par élément. Cet algorithme est le point de départ de notre étude. Le présent chapitre généralise cette idée en introduisant des familles d'estimateurs nouvelles et plus efficaces et donne une analyse précise de celles-ci.

Notre contribution. Dans ce chapitre, nous introduisons *trois nouvelles familles d'algorithmes* qui résolvent le problème d'estimer la cardinalité n d'un multi-ensemble en ne faisant qu'une passe sur les données en utilisant une mémoire constante et en ne faisant aucune hypothèse sur la structure des répétitions. La remarque de départ est que le minimum de n variables aléatoires uniformes prenant leur valeur sur $[0, 1]$ est en moyenne égal à $\frac{1}{n+1}$ (voir la Section 2.1 pour le calcul et la Figure 2.1 pour une intuition géométrique). On est donc capable de se faire une idée de n à partir de cette unique valeur. De plus, un point crucial à observer est que le minimum n'est pas sensible à la structure des répétitions, c'est-à-dire que le minimum des valeurs d'un multi-ensemble est celui de son ensemble sous-jacent. Nos algorithmes ont une boucle interne très simple —chaque élément du multi-ensemble est lu, haché uniformément vers $[0, 1]$ et comparé au minimum des valeurs hachées— et utilisent une mémoire constante —un seul nombre flottant étant nécessaire pour garder le minimum en mémoire. Quand le multi-ensemble a été lu en entier, nous disposons du minimum des valeurs hachées. Nous voulons maintenant une estimée de n et non de $\frac{1}{n+1}$. Le plus naturel serait de prendre l'inverse de ce minimum, mais il s'avère que cet inverse a une espérance infinie. Notre solution est d'obtenir une estimée indirectement à partir de ce minimum en combinant deux principes : au lieu de prendre le premier minimum, on utilise le deuxième, le troisième ou le $k-$ième et, au lieu de prendre la seule fonction inverse, on la combine avec des fonctions sous-linéaires comme logarithme ou racine carrée. On obtient ainsi, en Section 2.1, *trois familles d'estimateurs* de n : l'inverse du $k-$ième minimum, son logarithme et sa racine carrée. Pour obtenir des estimées précises, on couple ces algorithmes avec un processus de *moyennage stochastique* (*stochastic averaging*) introduit dans [FM83]. Le moyennage stochastique consiste à simuler l'effet de $m = 2^b$ expériences sur le multi-ensemble et ensuite à renvoyer la moyenne des observées sur les m expériences – en effet la moyenne arithmétique de m variables aléatoires indépendantes et de même loi a même espérance mais son erreur standard est divisée par $\sqrt{m}$. Nous prouvons dans le Théorème 1 de la Section 2.3 que les estimateurs sont bien *asymptotiquement non biaisés* —leur espérance est n— et nous donnons leur *erreur standard*. Un exemple typique d'analyse est donné en Section 2.2 —les autres calculs peuvent être trouvés en annexe. Les estimateurs sont ensuite comparés les uns avec les autres selon leur compromis entre mémoire et précision dans Théorème 2. Nous montrons que l'on obtient de meilleures estimées en appliquant la fonction logarithme, que la précision est meilleure en utilisant le $k-$ième minimum avec k grand et qu'il existe, pour chaque famille, un même compromis optimal :

une erreur standard de $\frac{1}{\sqrt{M}}$ en utilisant une mémoire de M nombres flottants. On choisit, à la fin de ce chapitre, un algorithme MINCOUNT qui peut être considéré comme le plus efficace des trois familles. Il sera étudié plus en détail dans le Chapitre 3. De plus nous proposons des validations expérimentales variées de nos algorithmes en Section 2.4. Les simulations montrent l'adéquation entre leur comportement et les prévisions de l'analyse.

2.1 Trois familles d'estimateurs

2.1.1 Définitions et notations.

Soit $\mathcal{D}$ notre *domaine de données*. Il représente, par exemple, l'ensemble des entiers naturels, l'ensemble des mots de la langue française ou l'ensemble des connexions possibles entre deux ordinateurs sur internet. Étant donné un multi-ensemble $\mathcal{M}$ d'éléments de $\mathcal{D}$ de taille N, le problème est d'estimer sa *cardinalité,* n, c'est-à-dire son nombre d'éléments *distincts*.

Pour ce faire, on suppose à partir de maintenant que l'on dispose d'une *fonction de hachage* h qui envoie un élément de $\mathcal{D}$ sur un nombre réel qui ressemble à un tirage aléatoire uniforme dans l'intervalle $[0, 1]$. Une discussion sur cette fonction de hachage peut être trouvé dans la Section 3.2.1.

Un *multi-ensemble idéal* de cardinalité n est une séquence de n valeurs distinctes choisies uniformément dans $[0, 1]$, répliquées de façon arbitraire et mélangées en appliquant une permutation aléatoire. Les familles d'algorithmes présentées ici prennent en *entrée (input)* un multi-ensemble idéal, $\mathcal{I}$, construit en appliquant h à $\mathcal{M}$. Leur *sortie (output)* est une *estimée* de sa cardinalité.

2.1.2 Construction d'estimateurs basée sur le minimum M.

Pour estimer le nombre d'éléments distincts d'un multi-ensemble idéal, $\mathcal{I}$, nous considérons son minimum, M. La remarque importante ici est que le minimum d'une séquence de nombres est trouvé en une passe unique sur les éléments et qu'il n'est pas sensible aux répétitions. Il est suffisant pour chaque nombre de le comparer au minimum sur les valeurs précédentes. Voir un élément dix fois n'a aucun effet sur ce minimum. La densité du minimum de n variables aléatoires uniformes sur $[0, 1]$ est $\mathbb{P}(M \in [x, x + dx]) = n(1 - x)^{n-1}dx$. Ainsi son espérance pour $(n \geq 1)$ est

$$\mathbb{E}[M] = \int_0^1 x \cdot n(1 - x)^{n-1}dx = \frac{1}{n + 1}.$$

Frédéric Meunier m'a donné une explication géométrique de ce résultat. Pour $n = 2$, on tire deux réels h_0 et h_1 entre 0 et 1. Le couple (h_0, h_1) correspond à un point du carré unité (voir la Figure 2.1). Si $h_0 \geq h_1$, le point est dans le triangle délimité par la diagonale. En moyenne le point se trouvera au barycentre du triangle. Le minimum est h_1 et il correspond

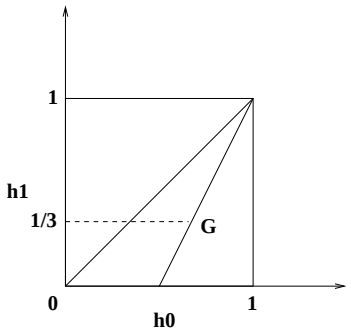


FIG. 2.1 – Intuition du fait que l'espérance du minimum de deux nombres réels tirés aléatoirement de façon uniforme sur $[0,1]$ est $1/3$.

à l'ordonnée du barycentre qui est à un tiers de la hauteur. On a donc en moyenne $M = 1/3$. Si $h_0 \leq h_1$, le raisonnement est symétrique. Cette approche se généralise à n'importe quelle dimension (si $n = 3$, le barycentre de la pyramide se situe à $1/4$ de la hauteur).

M est en gros un estimateur de $1/n$. A ce stade, notre espoir est de pouvoir prendre $1/M$ comme estimateur de n.

$$\mathbb{E}\left[\frac{1}{M}\right] = \int_0^1 \frac{1}{x} \cdot n(1-x)^{n-1} dx = +\infty$$

Malheureusement l'intégrale est divergente en 0. Pour obtenir une estimée de n, nous utilisons *indirectement* le minimum M selon deux principes différents qui peuvent être combinés :

1. Au lieu de prendre $1/M$ comme estimateur, nous prenons $f(1/M)$, avec f une fonction sous-linéaire de n. Le caractère sous-linéaire de la fonction aplatit les valeurs à l'origine et donne une espérance finie. Les fonctions utilisées ici sont le *logarithme (népérien)* et la *racine carrée*.
2. Au lieu d'utiliser le premier minimum, nous utilisons le second, troisième ou plus généralement le $k-$ième minimum. Ces minimums sont moins proches de zéro, ce qui donne aussi des espérances finies.

On obtient ainsi trois familles d'estimées, à savoir la *Famille Inverse*, la *Famille Racine carrée* et la *Famille Logarithme*. On parle de famille parce qu'il existe un estimateur pour chaque valeur de k. La Figure 2.1.2 donne le pseudo-code d'algorithmes utilisant ces familles d'estimateurs.

2.1.3 Simuler m expériences.

La précision des ces algorithmes est donnée par l'*erreur standard* de leur estimateur ξ, notée $\mathbb{SE}[\xi]$ et définie comme l'écart type divisé par n. Dans le but d'avoir des estimateurs

Algorithme de la Famille Inverse (F : multi-ensemble des valeurs hachées ; m)
pour $x \in F$ faire
    si $\frac{i-1}{m} \leq x \leq \frac{i}{m}$ faire
        actualiser les k minimums de la boîte i avec x
renvoyer $\xi := (k-1)\sum_{i=1}^{m} \frac{1}{M_i^{(k)}}$ comme estimée de la cardinalité.

Algorithme de la famille Racine carrée (F : multi-ensemble des valeurs hachées ; m)
pour $x \in F$ faire
    si $\frac{i-1}{m} \leq x \leq \frac{i}{m}$ faire
        actualiser les k minimums de la boîte i avec x
renvoyer $\xi := \frac{1}{\left(\frac{1}{(k-1)} + \frac{m-1}{(k-1)!^2}\Gamma(k-\frac{1}{2})^2\right)} \left(\sum_{i=1}^{m} \frac{1}{\sqrt{M_i^{(k)}}}\right)^2$ comme estimée de la cardinalité.

Algorithme de la Famille Logarithme (F : multi-ensemble des valeurs hachées ; m)
pour $x \in F$ faire
    si $\frac{i-1}{m} \leq x \leq \frac{i}{m}$ faire
        actualiser les k minimums de la boîte i avec x
renvoyer $\xi := m \cdot \left(\frac{\Gamma(k-\frac{1}{m})}{\Gamma(k)}\right)^{-m} \cdot e^{-\frac{1}{m}\sum_{i=1}^{m} \ln M_i^{(k)}}$ comme estimée de la cardinalité.

FIG. 2.2 – Pseudo-code des trois Familles d'Estimateurs.

plus précis, nous utilisons le procédé de *moyennage stochastique* introduit par P. Flajolet et Martin dans [FM83]. L'idée de départ est que la *moyenne arithmétique* de m variables aléatoires indépendantes de même loi, d'espérance μ et d'écart type σ, a même espérance μ mais son écart type est $\sigma/\sqrt{m}$. Ainsi, utiliser la moyenne arithmétique, notée $\mathcal{M}$, sur plusieurs expériences similaires permet d'augmenter fortement la précision des algorithmes.

Faire m expériences indépendantes nécessite d'utiliser m fonctions de hachage différentes. Hacher tous les éléments m fois prend beaucoup de temps et construire m fonctions de hachage vraiment indépendantes nécessite un peu de travail. Pour contourner ces difficultés, nous simulons ces m expériences. Le principe est de répartir les valeurs hachées entre m *boîtes (buckets)* différentes. Il suffit de diviser l'intervalle $[0,1]$ en m sous-intervalles de taille $1/m$. Une valeur hachée x tombe dans la $i-$ème boîte si $\frac{i-1}{m} \leq x < \frac{i}{m}$. Pour chacune des boîtes (i allant de 1 à m), les algorithmes gardent en mémoire pendant l'exécution le $k-$ième minimum, noté $M_i^{(k)}$ dans l'analyse. A la fin, ils les utilisent pour construire m estimées. Ensuite il s'agit de prendre leur moyenne arithmétique $\mathcal{M} := 1/m \sum_{i=1}^{m} f(1/M_i^{(k)})$. Et enfin d'inverser cette dernière pour obtenir un estimateur asymptotiquement non biaisé de n. L'analyse d'un des estimateurs est donnée en Section 2.2 comme exemple.

2.2 Le logarithme du second minimum

Nous présentons ici l'analyse complète de l'estimateur de la Famille Logarithme construit à partir du second minimum. Celui-ci a été choisi parce qu'il montre les difficultés typiques qui sont rencontrées et en particulier l'apparition d'un biais qui doit être corrigé pour obtenir une estimée asymptotiquement non biaisée. Cette section présente donc les étapes de construction d'un estimateur et la preuve de la proposition suivante (qui est une partie du Théorème 1 p. 42)

Proposition 1 *1. L'estimée renvoyée par l'algorithme de la famille logarithme construit avec le second minimum défini par*

$$\xi := m \cdot \Gamma\left(2 - \frac{1}{m}\right)^{-m} \cdot e^{\frac{1}{m}(\ln(1/M_1^{(2)}) + \ldots + \ln(1/M_m^{(2)}))} \tag{2.1}$$

est asymptotiquement non biaisée, au sens où

$$\mathbb{E}[\xi] \underset{n\to\infty}{\sim} n. \tag{2.2}$$

2. Son erreur standard, définie comme $\frac{1}{n}\sqrt{\mathbb{V}(\xi)}$, satisfait

$$\mathbb{SE}[\xi] \sim \sqrt{\Gamma\left(2 - \frac{1}{m}\right)^{-2m} \Gamma\left(2 - \frac{2}{m}\right)^{m} - 1}, m \geq 2. \tag{2.3}$$

2.2.1 Petit préliminaire : les fonctions spéciales.

Les calculs d'espérance et de variance utilisent fortement certaines fonctions spéciales comme, en particulier, la fonction Gamma définie par

$$\Gamma(z) := \int_0^\infty t^{z-1} e^{-t} dt$$

et la fonction Beta d'Euler définie par

$$\mathcal{B}(\alpha, \beta) := \int_0^1 t^{\alpha-1}(1-t)^{\beta-1} dt.$$

Ces deux fonctions et leurs dérivées sont intimement liées. Ainsi

$$\mathcal{B}(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)},$$

et

$$\frac{d}{d\alpha}\mathcal{B}(\alpha, \beta) = \mathcal{B}(\alpha, \beta)(\psi(\alpha) - \psi(\alpha, \beta)),$$

où ψ est la dérivée logarithmique de la fonction Gamma appelée aussi fonction digamma et définie par

$$\psi(z) = \frac{d}{dz} \ln \Gamma(z) = \frac{\Gamma'(z)}{\Gamma(z)}.$$

ψ est liée à la somme harmonique $H_n := \sum_1^n 1/n$ par

$$\psi(n) = H_{n-1} - \gamma,$$

où γ est la constante d'Euler définie comme $\lim_{n\to\infty}(H_n - \ln n)$.

La dérivée de la fonction ψ notée ψ' et appelée fonction trigamma est aussi utilisée. Une de ses propriétés intéressantes est que

$$\psi'(z) = \sum_0^\infty \frac{1}{(z+n)^2}.$$

On peut en particulier calculer $\psi'(1) = \pi^2/6$.

2.2.2 Calculs de l'espérance et de la variance

Nous avons vu en Section 2.1 que l'inverse du premier minimum a une espérance infinie. L'inverse du second minimum a aussi une espérance infinie. Mais, en prenant le logarithme de l'inverse, on obtient des intégrales convergentes. Les calculs d'espérance et de variance sont un préliminaire à la construction de l'estimateur. En effet il donne une indication que

l'estimateur construit avec cette fonction a une espérance et une variance finie et il donne une fonction inverse à utiliser pour construire l'estimée ξ.

Les calculs utilisent les fonctions spéciales définies plus haut. Plus précisément, on identifie ici la forme intégrale de la dérivée de la fonction Beta d'Euler avec son expression avec des fonctions digamma. La densité du deuxième minimum est $\frac{1}{n(n-1)}x(1-x)^{n-2}dx$. Donc nous avons :

$$\begin{aligned}
\mathbb{E}[\ln \tfrac{1}{M^{(2)}}] &= n(n-1)\int_0^1 (\ln \tfrac{1}{x})x(1-x)^{n-2}dx \\
&= n(n-1)\int_0^1 (\ln x)(1-x)^{n-1}dx - n(n-1)\int_0^1 \ln x(1-x)^{n-2}dx \\
&= n(n-1)\tfrac{d\mathcal{B}}{d\alpha}(1,n) - n(n-1)\tfrac{d\mathcal{B}}{d\alpha}(1,n-1) \\
&= n(n-1)\ (-\tfrac{\gamma}{n} - \tfrac{\psi(n+1)}{n}) - n(n-1)(-\tfrac{\gamma}{n-1} - \tfrac{\psi(n)}{n-1}) \\
&= H_n - 1
\end{aligned}$$

Les calculs pour obtenir la variance sont similaires à ceci près que c'est la seconde dérivée de l'intégrale $\mathcal{B}$ qui est utilisée. On obtient $\mathbb{V}[\ln \frac{1}{M^{(2)}}] = \frac{\pi^2}{6} - 1 - \psi'(n+1)$.

2.2.3 Lemmes de déterminisation

Les lemmes de déterminisation de cette section permettent de se ramener au cadre de l'*hypothèse simplificatrice* où un même nombre de valeurs hachées tombe dans chaque boîte. En effet ils montrent que l'on y obtient les mêmes résultats asymptotiques que quand les nombres de valeurs hachées suivent des multinomiales. La preuve de ces lemmes est une application de la méthode de Laplace exposée en Annexe (Section .1 p. 144). L'idée est de montrer que les termes qui contribuent asymptotiquement à la somme S_n se trouvent tous dans un domaine central que l'on sait étudier.

Les différents estimateurs prennent la forme d'une fonction g de f des m minimums M_i, avec f pouvant être le logarithme ou la racine carrée par exemple. L'obtention d'estimateurs non biaisés et de leur erreur standard implique des calculs d'espérances au cours desquels on est confronté à des expressions de la forme :

$$\mathbb{E}[g(f(M_1),\cdots,g(f(M_t))] = \sum_{reps}\mathbb{P}_{rep} \times \underset{rep\ donne}{\mathbb{E}}[g(f(M_1),\cdots,g(f(M_t))]$$

Quand la répartition est fixée les M_i deviennent indépendants et on a

$$\underset{\{rep\ donne\}}{\mathbb{E}}[g(f(M_1),\cdots,g(f(M_t))] = g(\underset{N_1=n_1}{\mathbb{E}}[f(M_1)],\cdots,\underset{N_t=n_t}{\mathbb{E}}[f(M_t)])$$

Remarquez que les $\underset{N_i=n_i}{\mathbb{E}}[f(M_1)]$ ont déjà été calculés. D'autre part

$$\mathbb{P}_{rep} = \mathbb{P}_{\{N_1=n_1,\cdots,N_m=n_m\}} = \frac{1}{m^n}\binom{n}{n_1,\cdots,n_m},$$

avec $\binom{n}{n_1,\cdots,n_m} := \frac{n!}{n_1!\cdots n_m!}$ le coefficient multinomial qui correspond aux nombres de façons de répartir n objets distincts dans m boîtes avec n_1 objets dans la première, n_2 dans la deuxième et ainsi de suite. On se retrouve donc dans tous nos calculs avec des formules de la forme

$$\sum_{n_1+\cdots+n_m=n} \frac{1}{m^n}\binom{n}{n_1,\cdots,n_m} f_{n_1}\cdots f_{n_m}.$$

Les lemmes de déterminisation montrent que sous certaines hypothèses pour les f_i, cette somme est équivalente à $(f_{n/m})^m$.

Remarquez que les f_i proviennent des espérances d'estimateurs ou d'estimateurs au carrée. Ce sont donc des fonctions proches souvent d'être linéaires ou polynomiales en n. Elles sont donc à variation lente au sens de la définition suivante.

Définition 1 (Fonctions à variation lente) *Une fonction f de n est à variation lente s'il existe une fonction ϵ telle que $\epsilon(n) \to 0$ quand $n \to \infty$ et si, pour tous $j \leq \sqrt{n}\ln n$,*

$$f_{n+j} = f_n(1 + O(\epsilon(n))). \tag{2.4}$$

Lemme 1 (Déterminisation des répartitions aléatoires) *Soit f une fonction de n*

1. *à croissance au plus polynomiale et*
2. *à variation lente (au sens de la Définition 1).*

Alors, pour m fixé, si

$$S_n := \sum_{n_1+\cdots+n_m=n} \frac{1}{m^n}\binom{n}{n_1,\cdots,n_m} f_{n_1}\cdots f_{n_m}, \tag{2.5}$$

on obtient

$$S_n = (f_{n/m})^m(1 + O(\epsilon(n))). \tag{2.6}$$

Preuve Dans la suite, on note de façon abbrégée $M_{\{n_i\}}$ le multinomial $M_{(n_1,\ldots,n_m)} := \binom{n}{n_1,\cdots,n_m}$.

Définition du Domaine Central. On définit tout d'abord un *domaine central (DC)* autour de $(n/m,\cdots,n/m)$ où le multinomial est maximum. Une répartition est à l'intérieur du DC si, pour tout i,

$$\frac{n}{m} - \sqrt{\frac{n}{m}}\ln\frac{n}{m} \leq n_i \leq \frac{n}{m} + \sqrt{\frac{n}{m}}\ln\frac{n}{m}.$$

Si une répartition n'appartient pas au DC, on dit qu'elle est dans la *périphérie (P)*.

Étude préliminaire de la somme des multinomiaux ($M_{\{n_i\}}$). On montre que

$$\sum_P \frac{1}{m^n}\binom{n}{n_1,\cdots,n_m} \underset{n\to\infty}{\to} 0$$
$$\sum_{DC} \frac{1}{m^n}\binom{n}{n_1,\cdots,n_m} \underset{n\to\infty}{\to} 1$$

Il suffit de prouver la première formule, vu que l'on passe de l'une à l'autre en utilisant

$$\sum_{n_1+\cdots+n_m=n} \frac{1}{m^n}\binom{n}{n_1,\cdots,n_m} = 1.$$

Pour ce faire, on majore d'abord $M_{\{n_i\}}$ pour une répartition $(n_1,\cdots,n_m)$ quelconque de P, puis on obtient une majoration de la somme recherchée en multipliant par le nombre total de répartitions.

Dans la suite, on note $n_i := n/m + \alpha_i$. Remarquez que $\sum \alpha_i = 0$. On a

$$\begin{array}{rcl} \frac{1}{m^n}M_{\{n_i\}} & = & \frac{1}{m^n}\frac{n!}{\prod_{i=0}^m (\frac{n}{m}+\alpha_i)!} \\ & \underset{n\to\infty}{\sim} & \frac{1}{m^n}\frac{n^n}{\prod_{i=1}^m (\frac{n}{m}+\alpha_i)^{(\frac{n}{m}+\alpha_i)}} \cdot \frac{\sqrt{2\pi n}}{\prod_{i=1}^m \sqrt{2\pi(\frac{n}{m}+\alpha_i)}} \\ & \underset{n\to\infty}{\sim} & \frac{n^n}{m^n(\frac{n}{m})^{mn/m+\sum a_i}} \frac{1}{\prod_{i=1}^m (1+\frac{m}{n}\alpha_i)^{(\frac{n}{m}+\alpha_i)}} \cdot \frac{\sqrt{2\pi}\sqrt{n}}{\sqrt{2\pi}^m \sqrt{\frac{n}{m}}^m \prod_{i=1}^m \sqrt{1+\frac{m}{n}\alpha_i}}. \end{array}$$

Soient $A := \prod_{i=1}^m (1+\frac{m}{n}\alpha_i)^{n_i}$ et $B := \prod_{i=1}^m \sqrt{1+\frac{m}{n}\alpha_i}$. Comme $\alpha_i \le n$, B tend vers une constante de n. On a

$$\begin{array}{rcl} A & \underset{n\to\infty}{\sim} & \prod \exp[\frac{m}{n}\alpha_i(\frac{n}{m}+\alpha_i)] \\ & \underset{n\to\infty}{\sim} & \exp[\sum \alpha_i + \sum \frac{m}{n}\alpha_i^2] = \exp[\sum \frac{m}{n}\alpha_i^2]. \end{array}$$

Pour une répartition de la périphérie, il existe au moins un α_i tel que $\alpha_i^2 \ge \frac{n}{m}(\ln\frac{n}{m})^2$. D'où

$$A \ge \exp[(\ln\frac{n}{m})^2].$$

Et

$$\frac{1}{m^n}M_{\{n_i\}} \le C\cdot\sqrt{n}^{-(m-1)}\cdot\exp[-(\ln\tfrac{n}{m})^2].$$

Le nombre total de répartitions est $n(n-1)\cdots(n-m+1) \sim n^m$. D'où

$$\sum_P \frac{1}{m^n}M_{\{n_i\}} \le C_1 n^{\frac{m+1}{2}} \times e^{-C_2(\ln n)^2} \underset{n\to\infty}{\to} 0.$$

Cela conclut notre étude préliminaire.

Étude de S_n sur le domaine central. On cherche à trouver un équivalent à S_n avec

$$S_n := \sum_{n_1+\cdots+n_m=n} \frac{1}{m^n}\binom{n}{n_1,\cdots,n_m} f_{n_1}\cdots f_{n_m}.$$

On a maintenant

$$S_n = \sum_{DC} + \sum_{P}.$$

Comme f est à variation lente, il existe $\epsilon(n/m)$ tel que $\epsilon(n/m) \underset{n\to\infty}{\to} 0$ et

$$\forall j \leq \sqrt{\frac{n}{m}} \ln \frac{n}{m}, f_{n/m+j} = f_{n/m}(1 + O(\epsilon(n/m))).$$

D'où

$$\begin{array}{rcl}
\sum\limits_{DC} & = & \sum\limits_{\sum l_i=0, |l_i| \leq \sqrt{n/m}\ln(n/m)} \frac{1}{m^n}\binom{n}{n/m+l_1,\cdots,n/m+l_m}(f_{n/m})^m(1+O(\epsilon(n/m)))^m \\
& = & (f_{n/m})^m(1+O(\epsilon(n/m)))^m \sum\limits_{DC} \frac{1}{m^n}\binom{n}{n/m+l_1,\cdots,n/m+l_m} \\
& = & (f_{n/m})^m(1+O(\epsilon(n/m)))\sum\limits_{DC} M_{\{n_i\}} \\
& \underset{n\to\infty}{\sim} & (f_{n/m})^m(1+O(\epsilon(n/m)))
\end{array}$$

Étude de S_n sur la Périphérie. Comme f est à croissance au plus polynomiale, il existe a tel que $f_n \leq n^a$, d'où

$$\begin{array}{rcl}
\sum\limits_{P} & = & \sum\limits_{P} \frac{1}{m^n}\binom{n}{n_1,\cdots,n_m} f_{n_1}\cdots f_{n_m} \\
& \leq & (n^a)^m \sum\limits_{P} \frac{1}{m^n}\binom{n}{n_1,\cdots,n_m} \\
& \leq & (n^a)^m \sum\limits_{P} M_{\{n_i\}} \\
& \leq & n^b \exp^{C(\ln n)^2} \\
& \underset{n\to\infty}{\to} & 0
\end{array}$$

On a donc

$$S_n = \sum_{DC} + \sum_{P} = (f_{n/m})^m(1 + O(\epsilon(n/m))).$$

Ce lemme peut être facilement étendu aux cas où on a une expression polynomiale (de polynôme g) de t (quelconque et plus de m exactement) fonctions différentes $(f_{n_1}^{(1)} \cdots f_{n_t}^{(t)})$ (et non plus identiques). Ces généralisations sont utilisées pour la construction d'estimateurs non biaisés des autres familles et en particulier de ceux de la famille racine carrée, en Section 2.3.2. La variante la plus générale est

Lemme 2 (Variante du lemme de déterminisation) *Soient $f^{(1)}, \cdots, f^{(t)}$ t fonctions différentes à variation lente et à croissance au plus polynomiale et g un polynôme. Alors, pour m fixé, si*

$$S_n := \sum_{n_1+\cdots+n_m=n} \frac{1}{m^n}\binom{n}{n_1,\cdots,n_m} g(f_{n_1}^{(1)} \cdots f_{n_t}^{(t)}),$$

on a

$$S_n = g(f^{(1)}_{n/m} \cdots f^{(t)}_{n/m})(1 + O(\epsilon(n))),$$

avec $\varepsilon(n) \underset{n\to\infty}{\to} 0$.

2.2.4 Construction d'un estimateur non biaisé et évaluation de son erreur standard.

On utilise la moyenne arithmétique $\mathcal{M}$ de m expériences pour construire des estimées précises. $\mathcal{M}$ a même espérance que $\ln \frac{1}{M^{(2)}}$, c'est-à-dire $g(n) = H_n - 1$. La Proposition 1 montre que l'on peut l'inverser en utilisant $g^{-1}(x) = e^{x+1}$. On prouve ici la proposition.

La preuve suit deux étapes. Dans la première partie, nous considérons une simplification du problème. On suppose que le même nombre de valeurs hachées tombe dans chaque boîte. Cette hypothèse est appelée *hypothèse d'équirépartition*. Les variables aléatoires $M^{(k)}_i$ sont alors indépendantes et de même loi. Dans une seconde partie de la preuve, nous montrerons, à l'aide du Lemme 1 de la Section 2.2.3, que l'on obtient les mêmes résultats asymptotiques sans cette hypothèse.

Première étape de la preuve.

$$\begin{aligned}
\mathbb{E}[e^{\mathcal{M}}] &= \mathbb{E}[e^{\frac{1}{m}(\ln \frac{1}{M^{(2)}_1} + \cdots + \ln \frac{1}{M^{(2)}_m})}] = \mathbb{E}[(M^{(2)})^{-\frac{1}{m}}]^m \\
&= [\textstyle\int_0^1 x^{-\frac{1}{m}} \frac{n}{m}(\frac{n}{m} - 1)x\ (1-x)^{\frac{n}{m}-2} dx]^m \\
&= [\tfrac{n(n-m)}{m^2} \mathcal{B}(2 - \tfrac{1}{m}, \tfrac{n}{m} - 1)]^m \\
&= [\tfrac{n}{n-1} \Gamma(2 - \tfrac{1}{m}) \tfrac{\Gamma(\frac{n}{m})}{\Gamma(\frac{n}{m} - \frac{1}{m})}]^m
\end{aligned}$$

Pour n grand, on obtient $\mathbb{E}[e^{\mathcal{M}}] \underset{n\to\infty}{\sim} \Gamma(2 - \frac{1}{m})^m \cdot \frac{n}{m}$. L'expression $\frac{1}{m}\Gamma(2 - \frac{1}{m})^m$ apparaît comme biais. Nous avons ainsi $\mathbb{E}[\xi] \underset{n\to\infty}{\sim} n$. ξ est un estimateur non biaisé de n.

Des calculs similaires donnent

$$\mathbb{E}[e^{2\mathcal{M}}] \underset{n\to\infty}{\sim} n^2 \cdot \Gamma(2 - \frac{2}{m})^m. \tag{2.7}$$

Prouvant le second point de la proposition.

Seconde étape de la preuve. On retire maintenant l'*l'hypothèse d'équirépartition*. Les nombres N_i ne sont plus identiques mais suivent une loi multinomiale. Ainsi $\mathbb{P}_{\{N_1 = n_1, \cdots, N_m = n_m\}} = \frac{1}{m^n} \binom{n}{n_1, \cdots, n_m}$. Cela introduit deux difficultés : les $M^{(2)}_i$ ne sont plus indépendantes et n'ont plus la même espérance.

Le facteur principal de l'estimée ξ devient

$$\mathbb{E}[e^{\mathcal{M}}] = \mathbb{E}\left[e^{\frac{1}{m}(\ln \frac{1}{M^{(2)}_1} + \cdots + \ln \frac{1}{M^{(2)}_m})}\right] = \sum_{\text{répartitions}} \mathbb{P}_{\text{rép}} \times \mathop{\mathbb{E}}_{\text{rép fixee}}[(M^{(2)}_1)^{-1/m} \cdots (M^{(2)}_m)^{-1/m}] \tag{2.8}$$

L'espérance est la somme sur toutes les répartitions possibles des valeurs hachées dans les boîtes, c'est-à-dire sur toutes les valeurs de N_i. Quand une répartition est donnée, les $M_i^{(2)}$ sont indépendants et

$$\mathbb{E}[e^{\mathcal{M}}] = \sum_{n_1+\cdots+n_m=n} \frac{1}{m^n} \binom{n}{n_1, \cdots, n_m} \mathop{\mathbb{E}}_{N_1=n_1}[(M_1^{(2)})^{-1/m}] \cdots \mathop{\mathbb{E}}_{N_m=n_m}[(M_m^{(2)})^{-1/m}] \quad (2.9)$$

Or en Section 2.2 on a vu que $\mathbb{E}[(M_1^{(2)})^{-1/m}]_{N_1=n_1} = \Gamma(2-1/m)\frac{n_1}{n_1-1}\frac{\Gamma(n_1/m)}{\Gamma(n_1/m-1/m)}$. Sa croissance est en $O(n^{1+1/n})$. Elle est donc à variation lente au sens de la Définition 1 de la Section 2.2.3. Le Lemme 1 s'applique. Nous avons donc :

$$\mathbb{E}[e^{\mathcal{M}}] = \left(\mathop{\mathbb{E}}_{N_1=n/m}[(M_1^{(2)})^{-1/m}] \right)^m (1 + O(\epsilon(n))). \quad (2.10)$$

Cette formule donne le même équivalent asymptotique qu'avec l'hypothèse d'équirépartition. La même méthode permet le calcul de l'erreur standard. Ainsi, dans le cas général, on peut utiliser le même estimateur et il a asymptotiquement la même erreur standard. Cela finit la preuve de la Proposition 1. □

2.3 Analyse des trois familles d'estimateurs.

Les résultats pour les trois familles d'estimateurs sont présentés dans le Théorème 1, comparés dans le Théorème 2. On propose aussi un *meilleur estimé.*

2.3.1 Résumé.

On étudie ici les trois familles d'estimées inverse, racine carrée et logarithme du $k-$ième minimum. Les étapes de l'analyse sont les mêmes que pour le logarithme du second minimum dans la 2.2 et les preuves sont omises ici. Les résultats sont présentés dans le Théorème 1. Certains résultats sont donnés pour m grands (mais petit devant n).

Théorème 1 *Considérons les trois familles suivantes d'algorithmes*

1, la famille inverse,
2, la famille racine carrée,
3, la famille logarithme,

appliquée à un multi-ensemble idéal de cardinalité inconnue n.

1. *Les estimées renvoyées par les trois familles d'algorithmes, ξ_1, ξ_2, ξ_3, définis respectivement par*

$$\xi_1 := (k-1)\sum_{i=1}^{m}\frac{1}{M_i^{(k)}}, \tag{2.11}$$

$$\xi_2 := \frac{1}{\left(\frac{1}{(k-1)!}\Gamma(k-1)+\frac{m-1}{(k-1)!^2}\Gamma(k-\frac{1}{2})^2\right)}\left(\sum_{i=1}^{m}\frac{1}{\sqrt{M_i^{(k)}}}\right)^2 \text{ et} \tag{2.12}$$

$$\xi_3 := m\cdot\left(\frac{\Gamma(k-\frac{1}{m})}{\Gamma(k)}\right)^{-m}\cdot e^{-\frac{1}{m}\sum_{i=1}^{m}\ln M_i^{(k)}}, \tag{2.13}$$

sont asymptotiquement non biaisées dans le sens où, pour $i=1,2,3$

$$\mathbb{E}[\xi_i] \underset{n\to\infty}{\sim} n. \tag{2.14}$$

2. *Leur erreur standard, définie par $\frac{1}{n}\sqrt{\mathbb{V}(\xi_i)}$, satisfait*

$$\mathbb{SE}[\xi_1] \underset{n\to\infty}{\sim} \frac{1}{\sqrt{k-2}}\cdot\frac{1}{\sqrt{m}}, \tag{2.15}$$

$$\mathbb{SE}[\xi_2] \underset{n\to\infty}{\sim} \frac{2}{\sqrt{m}}\sqrt{\frac{1}{k-1}\left(\frac{\Gamma(k)}{\Gamma(k-\frac{1}{2})}\right)^2-1}, m \text{ grand}, \tag{2.16}$$

$$\mathbb{SE}[\xi_3] \underset{n\to\infty}{\sim} \sqrt{\psi'(k)}\cdot\frac{1}{\sqrt{m}}, m \text{ grand}, \tag{2.17}$$

où Γ et ψ' sont les fonctions Gamma et trigamma d'Euler définies en Section 2.2.1 p. 36.

2.3.2 Analyse de la famille racine carrée

On donne ici la preuve du Théorème 1 pour la famille racine carrée. Elle suit les étapes de la construction de l'estimateur présentée en exemple en Section 2.2, mais est ici donnée de façon plus raccourcie. On peut étendre les preuves de cette section pour n'importe quelle puissance entière.

Calculs de l'espérance et de la variance

On effectue tout d'abord le petit calcul préliminaire de $\mathbb{E}[\frac{1}{(M^{(k)})^\alpha}]$. La densité du $k-$ième minimum est

$$\mathbb{P}(M^{(k)}\in[x,x+dx]) = k\binom{n}{k}x^{k-1}(1-x)^{n-k}dx.$$

Donc nous avons :

$$\begin{array}{rcl} \mathbb{E}[\frac{1}{(M^{(k)})^\alpha}] & = & k\binom{n}{k}\int_0^1 \frac{1}{x^\alpha}x^{k-1}(1-x)^{n-k}dx = k\binom{n}{k}\mathcal{B}(k-\alpha,n-k+1) \\ & = & k\binom{n}{k}\frac{\Gamma(k-\alpha)\Gamma(n-k+1)}{\Gamma(n+1-\alpha)} = \frac{\Gamma(k-\alpha)}{(k-1)!}\frac{\Gamma(n+1)}{\Gamma(n+1-\alpha)} \\ & \underset{n\to\infty}{\sim} & \frac{\Gamma(k-\alpha)}{(k-1)!}n^\alpha. \end{array}$$

L'espérance et la variance découlent directement :

$$\begin{array}{rcl} \mathbb{E}[\frac{1}{\sqrt{M^{(k)}}}] & \underset{n\to\infty}{\sim} & \frac{\Gamma(k-\frac{1}{2})}{(k-1)!}\sqrt{n} \\ \mathbb{V}[\frac{1}{\sqrt{M^{(k)}}}] & \underset{n\to\infty}{\sim} & (\frac{\Gamma(k-1)}{(k-1)!} - (\frac{\Gamma(k-\frac{1}{2})}{(k-1)!})^2)n \end{array}$$

Construction d'une estimée non biaisée et évaluation de son erreur standard.

On utilise la moyenne arithmétique $\mathcal{M}$ de m expériences pour construire des estimées précises. $\mathcal{M}$ a même espérance que $\frac{1}{\sqrt{M^{(k)}}}$, c'est-à-dire $g(n) = \frac{\Gamma(k-\frac{1}{2})}{(k-1)!}\sqrt{n}$. La Proposition 2 montre que l'on peut l'inverser en utilisant $g^{-1}(x) = x^2$.

Proposition 2 *1. L'estimée renvoyée par l'algorithme de la famille racine carrée défini par*

$$\xi := \frac{1}{\left(\frac{1}{(k-1)!}\Gamma(k-1) + \frac{m-1}{(k-1)!^2}\Gamma(k-\frac{1}{2})^2\right)}\left(\sum_{i=1}^{m}\frac{1}{\sqrt{M_i^{(k)}}}\right)^2 \tag{2.18}$$

est asymptotiquement non biaisé au sens où

$$\mathbb{E}[\xi] \underset{n\to\infty}{\sim} n. \tag{2.19}$$

2. Son erreur standard, définie comme $\frac{1}{n}\sqrt{\mathbb{V}(\xi)}$, satisfait

$$\mathbb{SE}[\xi] \underset{n\to\infty}{\sim} \frac{2}{\sqrt{m}}\sqrt{\frac{1}{k-1}\left(\frac{\Gamma(k)}{\Gamma(k-\frac{1}{2})}\right)^2 - 1}, m \text{ grand.} \tag{2.20}$$

Preuve. La preuve suit deux étapes. Dans la première partie, nous considérons une simplification du problème. On suppose que le même nombre de valeurs hachées tombe dans chaque boîte. Cette hypothèse est appelée *hypothèse d'équirépartition*. Les variables aléatoires $M_i^{(k)}$ sont alors indépendantes et de même lois. Dans une seconde partie de la preuve, nous montrerons, à l'aide du Lemme 1 de la Section 2.2.3, que l'on obtient les mêmes résultats asymptotiques sans cette hypothèse.

Première étape de la preuve. Par linéarité et parce que les $M_i^{(k)}$ sont iid, on a :

$$\begin{aligned}
\mathbb{E}[\mathcal{M}^2] &= \mathbb{E}[(\tfrac{1}{\sqrt{M_1^{(k)}}} + \ldots + \tfrac{1}{\sqrt{M_m^{(k)}}})^2] \\
&= \mathbb{E}[\textstyle\sum_{i=1}^m \frac{1}{M_i^{(k)}} + \sum_{1\le i<j\le m} \frac{2}{\sqrt{(M_i^{(k)})}\sqrt{(M_j^{(k)})}}] \\
&= m\mathbb{E}\left[\tfrac{1}{M^{(k)}}\right] + 2\tbinom{m}{2}\mathbb{E}\left[\tfrac{1}{\sqrt{M^{(k)}}}\right]^2 \\
&\underset{n\to\infty}{\sim} n\left(m\tfrac{1}{(k-1)!}\Gamma(k-1) + 2\tbinom{m}{2}\tfrac{1}{(k-1)!}^2\Gamma(k-\tfrac{1}{2})^2\right)
\end{aligned}$$

L'expression $\left(m\frac{1}{(k-1)!}\Gamma(k-1) + 2\binom{m}{2}\frac{1}{(k-1)!}^2\Gamma(k-\frac{1}{2})^2\right)$ apparaît comme biais. Nous avons ainsi $\mathbb{E}[\xi] \underset{n\to\infty}{\sim} n$. ξ est un estimateur non biaisé de n. Remarquez que pour m grand

$$\xi \underset{m\to\infty}{\sim} \tfrac{1}{m^2}\left(\tfrac{\Gamma(k)}{\Gamma(k-\frac{1}{2})}\right)^2 \mathcal{M}^2.$$

On s'attaque maintenant à l'erreur standard de ξ et donc tout d'abord à $\mathbb{E}[\mathcal{M}^4]$.

$$\begin{aligned}
\mathcal{M}^4 &= (\tfrac{1}{\sqrt{M_1^{(k)}}} + \ldots + \tfrac{1}{\sqrt{M_m^{(k)}}})^4 \\
&= \textstyle\sum_{i=1}^m \frac{1}{(M_i^{(k)})^2} + \sum_{1\le i<j\le m} \frac{8}{\sqrt{(M_i^{(k)})^3}\sqrt{(M_j^{(k)})}} + \sum_{1\le i<j\le m} \frac{6}{M_i^{(k)}M_j^{(k)}} \\
&\quad + \textstyle\sum_{1\le i<j<l\le m} \frac{36}{M_i^{(k)}\sqrt{(M_j^{(k)})}\sqrt{(M_l^{(k)})}} + \sum_{1\le i<j<l<o\le m} \frac{24}{\sqrt{(M_i^{(k)})}\sqrt{(M_j^{(k)})}\sqrt{(M_l^{(k)})}\sqrt{(M_o^{(k)})}}
\end{aligned}$$

D'où vient le 36 par exemple ? Quand on a choisit les i, j, l, il faut choisir le minimum répété (3 possibilités) et son emplacement ($\binom{4}{2}$ possibilités). Le nombre de façons de choisir i, j, l par exemple sera ensuite $\binom{m}{3}$. Par linéarité et parce que les $M_i^{(k)}$ sont iid, on a (on rappelle que $\mathbb{E}[\frac{1}{(M^{(k)})^\alpha}] \underset{n\to\infty}{\sim} \frac{\Gamma(k-\alpha)}{(k-1)!}n^\alpha$)

$$\begin{aligned}
\mathbb{E}[\mathcal{M}^4] &= m\mathbb{E}\left[\tfrac{1}{(M^{(k)})^2}\right] + 8\tbinom{m}{2}\mathbb{E}\left[\tfrac{1}{\sqrt{(M^{(k)})^3}}\right]\mathbb{E}\left[\tfrac{1}{\sqrt{(M^{(k)})}}\right] + 6\tbinom{m}{2}\mathbb{E}\left[\tfrac{1}{M^{(k)}}\right]^2 \\
&\quad + 36\tbinom{m}{3}\mathbb{E}\left[\tfrac{1}{M^{(k)}}\right]\mathbb{E}\left[\tfrac{1}{\sqrt{(M^{(k)})}}\right]^2 + 24\tbinom{m}{4}\mathbb{E}\left[\tfrac{1}{\sqrt{(M^{(k)})}}\right]^4 \\
&\underset{n\to\infty}{\sim} n^2\left(m\tfrac{1}{(k-1)!}\Gamma(k-2) + 8\tbinom{m}{2}\tfrac{1}{(k-1)!}^2\Gamma(k-\tfrac{3}{2})\Gamma(k-\tfrac{1}{2}) + 6\tbinom{m}{2}\tfrac{1}{(k-1)!}^2\Gamma(k-1)^2\right. \\
&\quad \left. + 36\tbinom{m}{3}\tfrac{1}{(k-1)!}^3\Gamma(k-1)\Gamma(k-\tfrac{1}{2})^2 + 24\tbinom{m}{4}\tfrac{1}{(k-1)!}^4\Gamma(k-\tfrac{1}{2})^4\right).
\end{aligned}$$

Quand m est grand (mais plus petit que n),

$$\begin{aligned}
\mathbb{E}[\mathcal{M}^4] &\underset{m,n\to\infty}{\sim} \left(m(m-1)(m-2)(m-3)\left(\tfrac{\Gamma(k-\frac{1}{2})}{\Gamma(k)}\right)^4\right. \\
&\quad \left. + 6m(m-1)(m-2)\left(\tfrac{\Gamma(k-\frac{1}{2})}{\Gamma(k)}\right)^3 + o(m^3)\right)\cdot n^2 \\
&\underset{m,n\to\infty}{\sim} \left(m^4\left(\tfrac{\Gamma(k-\frac{1}{2})}{\Gamma(k)}\right)^4 + \tfrac{6}{k-1}m^3\left(\tfrac{\Gamma(k-\frac{1}{2})}{\Gamma(k)}\right)^2 - 6m^3\left(\tfrac{\Gamma(k-\frac{1}{2})}{\Gamma(k)}\right)^4 + o(m^3)\right)\cdot n^2.
\end{aligned}$$

D'autre part

$$\begin{aligned}
\mathbb{E}[\mathcal{M}^2]^2 &\underset{n\to\infty}{\sim} n^2\left(\tfrac{m}{k-1}+m(m-1)\left(\tfrac{\Gamma(k-\frac12)}{\Gamma(k)}\right)^2\right)^2\\
&\underset{m,n\to\infty}{\sim} n^2\left(m^2(m-1)^2\left(\tfrac{\Gamma(k-\frac12)}{\Gamma(k)}\right)^4+2m^2\tfrac{m-1}{k-1}\left(\tfrac{\Gamma(k-\frac12)}{\Gamma(k)}\right)^2+o(m^3)\right)\\
&\underset{m,n\to\infty}{\sim} n^2\left(m^4\left(\tfrac{\Gamma(k-\frac12)}{\Gamma(k)}\right)^4-2m^3\left(\tfrac{\Gamma(k-\frac12)}{\Gamma(k)}\right)^4+2\tfrac{m^3}{k-1}\left(\tfrac{\Gamma(k-\frac12)}{\Gamma(k)}\right)^2+o(m^3)\right).
\end{aligned}$$

D'où

$$\begin{aligned}
\tfrac{\mathbb{E}[\mathcal{M}^4]-\mathbb{E}[\mathcal{M}^2]^2}{n^2} &\underset{m,n\to\infty}{\sim} \left(m^4\left(\tfrac{\Gamma(k-\frac12)}{\Gamma(k)}\right)^4+\tfrac{6}{k-1}m^3\left(\tfrac{\Gamma(k-\frac12)}{\Gamma(k)}\right)^2-6m^3\left(\tfrac{\Gamma(k-\frac12)}{\Gamma(k)}\right)^4+o(m^3)\right)\\
&\quad-\left(m^4\left(\tfrac{\Gamma(k-\frac12)}{\Gamma(k)}\right)^4-2m^3\left(\tfrac{\Gamma(k-\frac12)}{\Gamma(k)}\right)^4+2\tfrac{m^3}{k-1}\left(\tfrac{\Gamma(k-\frac12)}{\Gamma(k)}\right)^2+o(m^3)\right)\\
&\underset{m,n\to\infty}{\sim} \tfrac{4}{k-1}m^3\left(\tfrac{\Gamma(k-\frac12)}{\Gamma(k)}\right)^2-4m^3\left(\tfrac{\Gamma(k-\frac12)}{\Gamma(k)}\right)^4\\
&\underset{m,n\to\infty}{\sim} 4m^3\left(\tfrac{\Gamma(k-\frac12)}{\Gamma(k)}\right)^2\left(\tfrac{1}{k-1}-\left(\tfrac{\Gamma(k-\frac12)}{\Gamma(k)}\right)^2\right).
\end{aligned}$$

En tenant compte du biais, $\frac{1}{m^2}(\frac{\Gamma(k)}{\Gamma(k-\frac12)})^2$ pour m et n grands, on obtient

$$\mathbb{SE}[\xi] \underset{m,n\to\infty}{\sim} \tfrac{2}{\sqrt{m}}\sqrt{\tfrac{1}{k-1}\left(\tfrac{\Gamma(k)}{\Gamma(k-\frac12)}\right)^2-1}.$$

Si on développe maintenant en $1/k$ quand k grand (mais plus petit que m), on obtient

$$\begin{aligned}
\mathbb{SE}[\xi]^2 &\underset{m,n\to\infty}{\sim} \tfrac{4}{m}\left((\tfrac1k+\tfrac{1}{k^2}+o(\tfrac{1}{k^2}))(k-\tfrac34+o(1))-1\right)\\
&\underset{m,n\to\infty}{\sim} \tfrac{4}{m}\left(1-\tfrac{3}{4k}+\tfrac1k+o(\tfrac1k)-1\right)\\
\mathbb{SE}[\xi] &\underset{k,m,n\to\infty}{\sim} \tfrac{1}{\sqrt{km}}.
\end{aligned}$$

Seconde étape de la preuve. On retire maintenant l'*hypothèse d'équirépartition*. Les nombres N_i ne sont plus identiques mais suivent une loi multinomiale. Ainsi $\mathbb{P}_{\{N_1=n_1,\cdots,N_m=n_m\}} = \frac{1}{m^n}\binom{n}{n_1,\cdots,n_m}$. Cela introduit deux difficultés : les $M_i^{(k)}$ ne sont plus indépendantes et n'ont plus la même espérance. Le facteur principal de l'estimée ξ devient

$$\begin{aligned}
\mathbb{E}[\mathcal{M}^2] &= \mathbb{E}[(\tfrac{1}{\sqrt{M_1^{(k)}}}+\ldots+\tfrac{1}{\sqrt{M_m^{(k)}}})^2]\\
&= \textstyle\sum_{i=1}^m \mathbb{E}[\frac{1}{M_i^{(k)}}]+2\sum_{1\le i<j\le m}\mathbb{E}[\frac{1}{\sqrt{(M_i^{(k)})}\sqrt{(M_j^{(k)})}}]
\end{aligned}$$

avec

$$\mathbb{E}[\tfrac{1}{\sqrt{(M_i^{(k)})}\sqrt{(M_j^{(k)})}}] = \sum_{reps}\mathbb{P}_{rep}\times\underset{rep\ fixee}{\mathbb{E}}[\tfrac{1}{\sqrt{(M_i^{(k)})}\sqrt{(M_j^{(k)})}}].$$

L'espérance est la somme sur toutes les répartitions possibles des valeurs hachées dans les boîtes, c'est-à-dire sur toutes les valeurs de N_i. Quand une répartion des données est fixée, les M_i sont indépendants et

$$\mathbb{E}[\frac{1}{\sqrt{(M_i^{(k)})}\sqrt{(M_j^{(k)})}}] = \sum_{n_1+\cdots+n_m=n} \frac{1}{m^n}\binom{n}{n_1,\cdots,n_m} \mathop{\mathbb{E}}_{N_i=n_i}[\frac{1}{\sqrt{(M_i^{(k)})}}] \mathop{\mathbb{E}}_{N_j=n_j}[\frac{1}{\sqrt{(M_j^{(k)})}}].$$

Or on a vu précédemment que $\mathbb{E}[\frac{1}{\sqrt{M_i^{(k)}}}]_{N_i=n_i} \underset{n_i\to\infty}{\sim} \frac{\Gamma(k-\frac{1}{2})}{(k-1)!}\sqrt{n_i}$. Sa croissance est en $O(\sqrt{n})$. Elle est donc à variation lente au sens de la Définition 1 de la Section 2.2.3. La variante du lemme de déterminisation (Lemme 2) s'applique. Nous avons donc :

$$\begin{aligned}\mathbb{E}[\frac{1}{\sqrt{(M_i^{(k)})}\sqrt{(M_j^{(k)})}}] &= \frac{1}{(k-1)!}^2\Gamma(k-\tfrac{1}{2})^2(1+O(\epsilon(n)))\\ \mathbb{E}[\mathcal{M}^2] &= n\left(m\frac{1}{(k-1)!}\Gamma(k-1)+2\binom{m}{2}\frac{1}{(k-1)!}^2\Gamma(k-\tfrac{1}{2})^2\right)(1+O(\epsilon(n))).\end{aligned}$$

Cette formule donne le même équivalent asymptotique qu'avec l'hypothèse d'équirépartition. La même méthode permet le calcul de l'erreur standard. On y utilise les autres variantes du lemme de déterminisation, typiquement pour obtenir des équivalent de formules comme

$$\begin{aligned}\mathbb{E}[\frac{1}{M_i^{(k)}\sqrt{(M_j^{(k)})}\sqrt{(M_l^{(k)})}}] &= \sum_{n_1+\cdots+n_m=n} \frac{1}{m^n}\binom{n}{n_1,\cdots,n_m} \mathop{\mathbb{E}}_{N_i=n_i}[\frac{1}{(M_i^{(k)})}] \mathop{\mathbb{E}}_{N_j=n_j}[\frac{1}{\sqrt{(M_j^{(k)})}}] \mathop{\mathbb{E}}_{N_l=n_l}[\frac{1}{\sqrt{(M_l^{(k)})}}]\\ &= \mathbb{E}[\frac{1}{(M_i^{(k)})}]\mathbb{E}[\frac{1}{\sqrt{(M_j^{(k)})}}]^2(1+O(\epsilon(n))).\end{aligned}$$

Ainsi, dans le cas général, on peut utiliser le même estimateur et il a asymptotiquement la même erreur standard. Cela finit la preuve de la Proposition 2. □

2.3.3 Comparaison.

On détermine ici le degré de performance des algorithmes les uns par rapport aux autres. Comme les estimateurs ne font tous qu'une seule passe sur les données, on ne considère pas ici leur temps d'exécution mais leur précision et la mémoire M qu'ils utilisent. Celle-ci correspond principalement aux nombres flottants utilisés par les algorithmes pour stocker les minimums (la mémoire auxiliaire, pour les boucles ou les variables temporaires est considérée comme négligeable). On a donc pour un algorithme utilisant le k^e minimum $M = km$. De manière à faire une comparaison juste, nos regardons pour chaque algorithme son *compromis entre la mémoire et la précision*. Ainsi pour chaque famille d'algorithmes, on se demande quelle précision il peut atteindre si on se permet une mémoire fixe à utiliser.

Définition 2 (Précision à mémoire constante) *La précision à mémoire constante d'une estimée ξ, $\mathcal{P}_{CM}(\xi)$, est son erreur standard exprimée en fonction de la mémoire qu'il utilise, $M = km$.*

Théorème 2 *1. Les* précisions à mémoire constante *des trois familles d'estimées sont*

$$\mathcal{P}_{CM}(\xi_1) := \sqrt{\frac{k}{k-2}\frac{1}{M}} \tag{2.21}$$

$$\mathcal{P}_{CM}(\xi_2) := \frac{2}{\sqrt{M}}\sqrt{\frac{k}{k-1}\left(\frac{\Gamma(k)}{\Gamma(k-\frac{1}{2})}\right)^2 - 1} \tag{2.22}$$

$$\mathcal{P}_{CM}(\xi_3) := \sqrt{k\psi'(k)\frac{1}{M}}, \tag{2.23}$$

où Γ et ψ' sont les fonctions Gamma et trigamma d'Euler définies en Section 2.2.1 p. 36.

2. Pour tout k,

$$\mathcal{P}_{CM}(\xi_3) \leq \mathcal{P}_{CM}(\xi_2) \leq \mathcal{P}_{CM}(\xi_1). \tag{2.24}$$

3. La Précision à mémoire constante *est une fonction décroissante de k pour les trois familles.*

4. Quand k tend vers l'infini, on a

$$\mathcal{P}_{CM}(\xi_i)_{i=1,2,3} \underset{k\to\infty}{\to} \frac{1}{\sqrt{M}} \tag{2.25}$$

Théorème 2 présente la *précision à mémoire constante* des trois familles d'estimateurs. Quatre principaux résultats peuvent en être extraits.

Premier résultat. Comme l'exprime le point 2, on obtient de meilleures estimées en utilisant des fonctions sous-linéaires, comme la racine carrée ou le logarithme. Par exemple, dans le cas du troisième minimum ($k = 3$), $\mathcal{P}_{CM}(\xi_1) = 1.73/\sqrt{M}$, $\mathcal{P}_{CM}(\xi_2) = 1.26/\sqrt{M}$ and $\mathcal{P}_{CM}(\xi_3) = 1.09/\sqrt{M}$.

Deuxième résultat. Comme l'exprime le point 3, la précision à mémoire constante quand on utilise le $k-$ième minimum est croissante avec k pour chacune des trois familles. Par exemple, dans le cas de la Famille Logarithme, les précisions pour les premier, deuxième et troisième minimums sont respectivement 1.28, 1.13 and 1.09. Notez que, si l'on s'attendait bien à ce que l'erreur standard diminue quand k augmente vu que plus de mémoire est utilisé, le fait que ce soit aussi vrai à mémoire constante est plus inattendu.

Troisième résultat. Comme l'exprime le point 4, il existe pour chacune des familles un compromis optimale entre la précision et la mémoire : un erreur standard de $1/\sqrt{M}$ pour une mémoire de M nombres flottants. Remarquez que cela implique, que si la précision gagnée en utilisant la fonction logarithme est très importante pour des petits k, elle diminue quand k augmente.

Quatrième résultat : meilleure estimée. L'efficacité optimale est atteinte rapidement. En effet la constante pour l'estimée de la Famille Logarithme construite avec le troisième minimum est 1.09. Dans la suite, nous considérons cette estimée comme la meilleure

	m	4	8	16	32	64	128	256	512	1024
$\frac{1}{M^{(3)}}$	$\%th$	50	35.4	25	17.7	12.5	8.8	6.25	4.4	3.1
	$Random$	53.2	33.5	25.9	17.9	14.3	9.4	6.1	4.3	3.0
	$Ind-230m$	31.0	32.9	10.4	1.9	10.5	9.4	1.4	0.3	0.05
	$Auck-9M$	10.3	4.0	5.1	10.0	2.8	3.9	1.7	1.8	3.0
$\frac{1}{\sqrt{M^{(3)}}}$	$\%th$	36.3	25.7	18.1	12.8	9.1	6.4	4.5	3.2	2.3
	$Random$	38.4	26.5	18.0	13.4	9.0	6.2	4.6	3.1	2.1
	$Ind-230m$	27.9	18.0	5.9	1.9	10.7	11.4	2.3	0.5	0.15
	$Auck-9M$	10.6	4.7	2.1	4.7	5.2	0.08	0.3	2.1	2.9
$\ln \frac{1}{M^{(3)}}$	$\%th$	34.0	23.1	16.0	11.2	7.9	5.6	3.9	2.8	2.0
	$Random$	34.9	22.3	16.0	11.8	8.0	5.5	4.0	2.6	1.9
	$Ind-230m$	25.8	3.5	1.3	4.9	10.6	12.2	3.2	1.2	0.3
	$Auck-9M$	10.7	6.1	0.2	0.6	6.2	2.7	0.6	2.7	2.9

FIG. 2.3 – Résultats des simulations

et nous l'utilisons pour construire un algorithme efficace et optimisé, MINCOUNT dans le Chapitre 3. Remarquez que le fait que l'efficacité optimale soit atteinte rapidement pour la famille logarithme implique que l'étude d'autres familles d'algorithmes construites avec des fonctions plus fortement sous-linéaires, comme par exemple $\ln^2$, ne procurerait pas de gains pratiques décisifs en terme de précision à mémoire constante.

2.4 Validation des estimateurs

2.4.1 Les données

Nous avons validé l'analyse des trois familles d'algorithmes par de très nombreuses simulations sur des fichiers de différentes natures et de différentes tailles. Un aperçu de leurs résultats est donné en Figure 4.6. Les fichiers utilisés ici sont `Random`, `Ind-230` and `Auck-9M`. Les deux derniers correspondent à des traces accessibles sur le site internet du groupe d'analyse de réseaux du NLANR (the NLANR Measurement and Network Analysis Group, `http://moat.nlanr.net`). Une trace est la séquence des en-têtes (anonymisés ou non) de tous les paquets qui passent sur un lien internet pendant une période de temps déterminée. Notre analyse consiste à estimer le nombre de connexions distinctes, identifiées par un couple d'adresses IP —adresse de la source - adresse de la destination— pour chaque trace. La plupart des réseaux cœur d'internet (backbone networks) sont des réseaux optiques synchrones (Synchronous Optical NETworks (SONET)). Dans ces réseaux utilisant des fibres optiques les liens sont classés selon leur capacité, des OC-1 (51,84 Mbps) aux OC-192 (10 Gbps) (utilisés extensivement) et OC-768 (40 Gbps) (premières mises en place). `Ind-230m` a été enregistrée sur un lien OC-3 d'un POP de l'Indiana (États-Unis) pendant

une fenêtre d'une minute trente ; `Auck-9M` a été enregistrée à l'université d'Auckland (Nouvelle Zélande) pendant une fenêtre de quinze minutes. 4 322 835 paquets correspondant à 230 292 connexions distinctes sont passés sur le lien de l'Indiana et 13 846 690 paquets pour 9 083 474 connexions pour le lien d'Auckland. Le troisième jeu de données, identifié par `Random`, correspond à des résultats moyens de simulations avec 500 fichiers. Chacun de ces fichiers est constitué de 10000 entiers tirés au hasard entre 0 et $2^{32} - 1$. Nous estimons ici le nombre d'entiers distincts de ces fichiers. Dans le tableau de la Figure 4.6 chaque ligne —à l'exception de $\%th$— correspond aux résultats sur la donnée indiquée.

2.4.2 Protocole et Résultats

Les estimateurs de chacune des trois familles ont été utilisés sur ces données. Le tableau de la Figure 4.6 montre des résultats typiques pour les estimées construits avec le troisième minimum. Les trois blocs horizontaux correspondent chacun aux résultats pour l'estimée d'une famille. Chaque estimateur a été exécuté sur chaque fichier plusieurs fois avec différents nombres d'expériences simulées (les nombres de boîtes varient de $m = 4$ à $m = 1024$). Ainsi les colonnes correspondent à des précisions différentes des algorithmes. *Un nombre du tableau* est la différence —exprimée en pourcentage— entre l'estimée donnée par l'algorithme et la valeur exacte. Ainsi, par exemple, l'estimée du nombre de connexions distinctes de `Ind-230` quand $m = 8$ est 306 058. La valeur exacte est 230 292. La précision est $((306058-230292)/230292)\times 100 = 32,9\%$. Les lignes référencées par $\%th$ indiquent l'erreur standard attendue selon la théorie. Par exemple, pour $m = 256$, le nombre de connexions a été évalué avec des précisions respectives de $1,4\%$, $2,3\%$ et $3,2\%$ quand l'erreur standard est $6,25\%$, $4,5\%$ et $3,9\%$. Le troisième jeu de données, `Random`, qui correspond à des résultats moyens sur 500 fichiers, valide les précisions données dans le Théorème 1. En effet, les résultats sont proches de ceux attendus selon la théorie. Par exemple pour $m = 32$, nous observons $17,9$, $13,1$ et $11,8$ pour les trois familles à comparer aux $17,7$, $12,8$ et $11,2$ attendus. Cela valide les algorithmes. Le régime asymptotique est atteint rapidement.

Chapitre 3

MINCOUNT : Ingénierie algorithmique, Optimisation, Applications

mots-clés : *implémentation, optimisation, applications, trafic internet, ADN, comptage par collision.*

Dans le chapitre précédent, ont été introduites trois familles d'estimateurs de la cardinalité de multi-ensembles. Les différents estimateurs ont été analysés et comparés entre eux. L'un deux a été choisi comme le "meilleur". Ce chapitre est le passage pour celui-ci de la théorie à un véritable algorithme implémenté et performant, MINCOUNT*.

Les problèmes rencontrés sont d'ordre pratique —choix de la fonction de hachage, optimisation de la gestion des entrées/sorties, structure du code—, mais aussi théorique. En effet pour résoudre le problème des petites cardinalités, nous avons introduit en Section 3.1 p. 52 de nouveaux estimateurs (EXTENDED HIT COUNTING) qui sont analysés dans le Théorème 6. zone de transition, choix entre eux À la fin de l'étude de la Section 3.1.4, on dispose d'un algorithme qui pour chaque taille de fichier, de quelques unités à plusieurs centaines de millions, choisit un estimateur approprié qui donne le nombre d'éléments distincts n *à quelques pourcents près.* (2% quand $m = 2^{10}$, en utilisant $3m$ flottants de 32−bit, ce qui correspond à $96kB$, voir la Figure 3.4 p. 61.) L'introduction du EXTENDED HIT COUNTING pour certains taux $\lambda := n/m$, améliore la précision d'un facteur proche de 4 —en comparaison du HIT COUNTING classique.

Du point de vue pratique, nous expliquons en Section 3.2 les différentes étapes qui nous ont amené à une implémentation optimisée de l'algorithme. Ces étapes (choix de la structure du programme, gestion des entrées sorties,...) ont permis de faire diminuer d'un facteur supérieur à 10 le temps d'exécution. L'implémentation a été chronométrée et validée en Section 3.2.3 sur des fichiers de différents types. Elle n'est que 3 à 4 fois plus lente que la commande unix `cat -t` qui ne lit juste que les caractères de son

entrée.

On utilise cette implémentation pour simuler deux types d'applications réseaux et bio-informatiques. En Section 3.3, on regarde comment mettre en place un système d'alerte automatique d'attaques par déni de service. En Section 3.4, on a montré que les algorithmes probabilistes de comptage peuvent être employés pour mesurer la corrélation des bases de différentes zones du génome humain, voir la Figure 3.4 p. 79. Cela pourrait être utilisé, par exemple, pour déterminer en une passe très rapide certaines localisations possibles de zones codantes.

3.1 Comptage par Collisions et Problème des petites cardinalités.

3.1.1 Introduction

Dans le chapitre précédent, une famille entière d'estimateurs du nombre d'éléments distincts d'un multi-ensemble —construits à partir de la valeur hachée minimale des éléments du multi-ensemble— a été introduite. L'idée est que le $k-$ième minimum des valeurs d'un multi-ensemble idéal ne dépend pas de la structure des répétitions des données, ni de leur ordre d'apparition. Il donne donc une indication sur le nombre n de valeurs distinctes du multi-ensemble (en gros, le minimum de n réels tirés uniformément et indépendamment dans $[0,1]$ a plus de chances d'être petit si n est grand). Le principe est donc de construire une observée à partir du $k-$ième minimum, et, pour obtenir une estimée précise de n, de la combiner avec un *procédé de moyennage stochastique*, comme introduit dans [FM83]. Le moyennage stochastique revient à simuler l'effet de de $m = 2^b$ expériences sur le multi-ensemble en répartissant les éléments entre m boites (buckets). L'algorithme construit alors une estimée précise du nombre de valeurs distinctes en moyennant les observées construites à partir du $k-$ième minimum de chaque boite. La meilleure estimée de cette famille, MINCOUNT*, est défini pour $k = 3$ et a un très bon compromis entre mémoire et précision. Néanmoins, un problème apparaît quand le nombre de valeurs distinctes du multi-ensemble est petit : certaines boites peuvent recevoir moins de trois valeurs, et donc ne pas avoir de troisième minimum. L'estimée de MINCOUNT* ne peut pas être calculée dans ce cas. Une solution, TRUNCATED MINCOUNT*, est de ne moyenner que sur les boites pleines. Néanmoins cette estimée n'est précise que quand peu de boites sont vides.

On introduit dans cette section, une nouvelle méthode pour gérer ce *problème des petites cardinalités*. C'est une extension du comptage par collisions, HIT COUNTING, introduit par Whang, Vander-Zandem et Taylor dans [WZT90]. L'idée est d'utiliser l'information contenue dans le nombre de boites vides. Pour chaque boite, on connaît le nombre exact de d'éléments tombés dedans si celui-ci est plus petit que k, un petit entier fixé, sinon on sait qu'il y est tombé au moins k éléments —k est 3 dans le cas du MINCOUNT*. Cette information supplémentaire nous permet de construire k estimées différentes, $\mathcal{H}_j$, avec $0 \leq j < k$, du nombre n d'éléments, en utilisant les nombres X_j de boites avec *au plus* j éléments.

Le cas $k = 0$ correspond au HIT COUNTING classique. En Section 3.1.3, nous construisons les estimées H_j du EXTENDED HIT COUNTING. Nous les analysons : nous montrons qu'elles sont asymptotiquement non biaisées et donnons leur précision (Théorème 5). En Section 3.1.4, nous combinons les estimées du EXTENDED HIT COUNTING et de TRUNCATED MINCOUNT* pour pouvoir gérer des multi-ensembles de toutes les cardinalités. Le problème ici est d'utiliser l'estimée la plus précise pour chaque cardinalité, c'est-à-dire, plus précisément, pour chaque taux de remplissage $\lambda := n/m$. L'analyse montre (voir la Section 3.1.4), que, *en choisissant l'estimée adaptée, l'algorithme atteint une erreur relative de quelques pourcents quel que soit λ, tout en n'utilisant que quelques dizaines de kilobits de mémoire auxiliaire (2% quand $m = 2^{10}$, en utilisant $3m$ flottants de $32-$bit, ce qui correspond à $96kB$, voir les Figure 3.4 et 3.8). L'introduction du* EXTENDED HIT COUNTING *pour certains taux λ, améliore la précision d'un facteur proche de 4 —en comparaison du* HIT COUNTING *classique.* De plus, nous donnons des validations expérimentales de nos estimées en Section 3.1.6. Les résultats des simulations sont en adéquation avec la théorie.

3.1.2 Préliminaires

L'algorithme MINCOUNT*

On rappelle ici brièvement les principes de l'algorithme MINCOUNT*, qui estime le nombre n de valeurs distinctes dans un multi-ensemble avec un bon compromis entre la mémoire utilisée et la précision, le résumé en pseudo-code ci-dessous. L'algorithme utilise une *fonction de hachage* h qui envoie un élément sur un réel qui "a l'air" uniformément distribué sur $[0, 1]$. Pour obtenir une estimée précise, les valeurs hachées sont distribuées entre m boites (procédé de moyennage stochastique). Les premiers, deuxième et troisièmes minimums de chaque boite sont gardés en mémoire. —ainsi, la mémoire utilisée par l'algorithme est de $3m$ nombres flottants. Quand tous les éléments ont été lus, il construit une estimée précise à partir des troisièmes minimums des boites.

Algorithme MINCOUNT* (avec $m = 2^b$ boites)
Input : $(p_1, \ldots, p_N)$ un ensemble d'éléments ;
Initialiser $(M_1^{(1)}, M_1^{(2)}, M_1^{(3)}), \ldots, (M_m^{(1)}, M_m^{(2)}, M_m^{(3)})$ à 0
pour i de 1 à N
  $u_i \leftarrow h(p_i)$; [hacher p_i vers une valeur réelle de $[0, 1]$]
  $j \leftarrow$ entier correspondant aux b premiers bits de u_i ;
  $\tilde{u_i} \leftarrow 2^b u_i - \lceil 2^b u_i \rceil$; [$\tilde{u_i}$ est u_i tronqué de ses b premiers bits]
  $(M_j^{(1)}, M_j^{(2)}, M_j^{(3)}) \leftarrow \min(M_j^{(1)}, M_j^{(2)}, M_j^{(3)}, \tilde{u_i})$; [actualiser les minimums]
renvoyer $\xi := m \left(\frac{\Gamma(3-1/m)}{2}\right)^{-m} \exp\left(\frac{1}{m}(\ln(M_1^{(3)}) + \ldots + \ln(M_m^{(3)})\right)$
  comme estimée du nombre d'éléments distincts.

Théorème 3 *Soit $\mathcal{I}$ un multi-ensemble idéal de cardinalité n. Soit $m = 2^b$ le nombre de boites utilisés par l'algorithme. Pour $1 \leq j \leq m$, soit $M_j^{(3)}$ le troisième minimum des valeurs dirigées vers la boite d'indice j. Alors, en notant Γ la fonction Gamma d'Euler, la quantité*

$$\xi := m \left(\frac{\Gamma(3 - 1/m)}{2} \right)^{-m} \exp\left(\frac{1}{m} (\ln(M_1^{(3)}) + \ldots + \ln(M_m^{(3)}) \right) \tag{3.1}$$

peut être calculée en utilisant une mémoire de $3m$ mots et est une estimée précise de la cardinalité de $\mathcal{I}$, dans le sens où :

- *il est asymptotiquement non biaisé, c'est-à-dire, $\mathbb{E}(\xi) \underset{n\to\infty}{\sim} n$.*
- *Sa précision relative, définie comme $\sqrt{\mathbb{V}(\xi)}/\mathbb{E}(\xi)$, est d'environ $0.63/\sqrt{m}$.*

Truncated Mincount*est utilisé quand certaines boites —en nombre V— n'ont pas de troisième minimum et l'erreur standard de cet estimée, ξ_t^*, satisfait

$$\frac{\sigma[\xi_t^*]}{n} \underset{n\to\infty}{\sim} \frac{0.63}{\sqrt{m - V}}.$$

Hit Counting classique

Ce paragraphe présente les grandes lignes de l'algorithme proposé par Whang, Vander-Zandem et Taylor dans [WZT90] basé sur le comptage par collisions. On dispose d'un tableau de m bits, noté T, et d'une fonction de hachage h qui envoie un élément sur un entier entre 0 et $m-1$ selon la distribution uniforme. Le principe est de hacher le multi-ensemble vers le tableau de bits. Au départ de l'algorithme, toutes les entrées sont initialisées à 0. Ensuite, pour chaque élément x de l'ensemble, le tableau de bit est mis à jour en donnant la valeur 1 à l'entrée $h(x)$. À la fin, une entrée i est égale à 1, s'il existe un x tel que $h(x) = i$, et sinon elle vaut 0. Remarquez que l'état du tableau à la fin de l'algorithme ne dépend que de l'ensemble sous-jacent des éléments distincts et non des répétitions possibles. Pour obtenir une estimée de n, la cardinalité (inconnue) de cet ensemble, l'idée est alors d'utiliser l'information donnée par le nombre d'entrées vides du tableau.

3.1.3 Extended Hit Counting

Le comptage par collisions généralisé se situe dans le même contexte que le comptage par collisions classique : n boules sont lancées au hasard dans m urnes. Mais l'information disponible est plus précise : pour chaque urne, on connaît le nombre exact de boules qui y ont atterri si celui-ci est plus petit que k, un petit entier fixé, et, sinon, on sait qu'il y en a eu au moins k Ce surplus d'information permet de construire plusieurs estimateurs différents, $\mathcal{H}_j$, avec $j < k$, du nombre de boules lancées n à partir des nombres, $\mathcal{X}_j$, de boites avec *au plus* j éléments —Une remarque : pour simplifier l'analyse, on ne définit pas $\mathcal{X}_j$ comme le nombre de boites avec exactement j éléments. On obtient ainsi des fonctions monotones de

n que l'on peut inverser—. Notez que le cas $j = 0$ correspond au comptage par collisions classique. Dans cette section, on analyse les H_j, on montre que ce sont des estimateurs asymptotiquement non biaisés de n et on donne leur précision (Théorème 5). Les résultats sont exprimés en fonction des espérances et variances des X_j calculées ci-dessous.

Analyse du nombre X_j de boites avec au plus j éléments

Séries Génératrices Exponentielles (SGE) des $\mathcal{X}_k$. La variable z marque les boules. Une urne remplie avec r boules se marque $\frac{z^r}{r!}$. La SGE pour une urne est donc $1 + z + z^2/2 + \cdots + z^r/r! + \cdots = e^z$. Les urnes remplies avec au plus k boules sont marquées par la variable u. La SGE devient $e^z - (1 + z + \cdots + z^k/k!) + u(1 + z + \cdots + z^k/k!)$. On obtient ensuite facilement la SGE pour m urnes :

$$F_k(z, u) := \sum_{\alpha \text{ alloc}, r=|\alpha|} u^{\mathcal{Y}(\alpha)} z^r = (e^z + (u-1)(1 + z + \cdots + z^k/k!))^m$$

Espérance de $\mathcal{X}_k$. On peut obtenir l'espérance d'une variable aléatoire à partir de sa Série Génératrice Exponentielle :

$$\mathbb{E}[\mathcal{X}_k] = \frac{n![z^n]\mathrm{d}_u F_k(z, u)|_{u=1}}{n![z^n]F_k(z, 1)},$$

avec $\mathrm{d}_u f(u)$ la dérivée partielle de f par rapport à u. Or le dénominateur $n![z^n]F_k(z, 1)$ vaut $(e^z)^m$. C'est le nombre d'allocations possibles de n boules dans m urnes, c'est-à-dire m^n. Le numérateur satisfait

$$\begin{array}{lcl} d_u F_k(z, u) & = & m(1 + z + \cdots + z^k/k!)(e^z + (u-1)(1 + z + \cdots + z^k/k!))^{m-1} \\ d_u F_k(z, u)|_{u=1} & = & m(1 + z + \cdots + z^k/k!)e^{(m-1)z} \\ n![z^n]d_u F_k(z, u)|_{u=1} & = & m\sum_{i=0}^{k}[z^{n-k}]\frac{n!}{k!}e^{(m-1)z} \end{array}$$

D'où

$$\mathbb{E}[\mathcal{X}_k] = m\frac{(m-1)^n}{m^n}\sum_{i=0}^{k}\binom{n}{k}\frac{1}{(m-1)^k}.$$

Quand n et m tendent vers l'infini avec $n/m = \lambda$, on a

$$\mathbb{E}[\mathcal{X}_k] \sim me^{-\lambda}\sum_{i=0}^{k}\frac{\lambda^i}{i!}.$$

Variance de $\mathcal{X}_k$. On obtient aussi la variance à partir de la SGE :

$$\mathbb{E}[\mathcal{X}_k^2] = \frac{n![z^n]d_u^2 F(z, u)|_{u=1}}{n![z^n]F(z, u)|_{u=1}} + \frac{n![z^n]d_u F(z, u)|_{u=1}}{n![z^n]F(z, u)|_{u=1}}$$

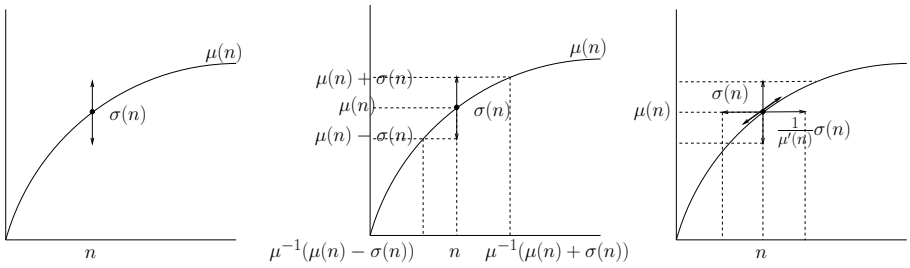


FIG. 3.1 – Inversion et erreur standard.

$$
\begin{array}{lcl}
d_u^2 F_k(z,u) & = & m(m-1)(1+z+\cdots+z^k/k!)^2 \\
 & & (e^z+(u-1)(1+z+\cdots+z^k/k!))^{m-2} \\
d_u^2 F_1(z,u)|_{u=1} & = & m(m-1)(1+z+\cdots+z^k/k!)^2 e^{(m-2)z} \\
(1+z+\cdots+z^k/k!)^2 & = & \sum_{r=0}^{k} \frac{z^r}{r!} \sum_{i=0}^{r} \binom{r}{i} + \sum_{r=k+1}^{2k} \frac{z^r}{r!} \sum_{i=r-k}^{k} \binom{r}{i} \\
n![z^n] d_u^2 F_1(z,u)|_{u=1} & = & m(m-1) \left[\sum_{r=0}^{k} \binom{n}{r} 2^r (m-2)^{n-r} + \right. \\
 & & \left. \sum_{r=k+1}^{2k} \binom{n}{r} (m-2)^{n-r} \sum_{i=r-k}^{k} \binom{r}{i} \right]
\end{array}
$$

La variance découle directement de $\mathbb{V}[\mathcal{X}_k] = \mathbb{E}[\mathcal{X}_k^2] - \mathbb{E}[\mathcal{X}_k]^2$.

Distribution limite gaussienne de $\mathcal{X}_k$. Pour obtenir la distribution limite de la variable X_k, il faut étudier l'asymptotique de sa fonction caractéristique, $n![z^n]F(z,u)/m^n$. On doit donc étudier l'asymptotique de $[z^n]F(z,u)$ pour tout u fixé. On suppose ici aussi que n et m sont grands avec n/m égal au rapport fixe λ. En utilisant la méthode du col (saddle point method) [FS, VIII.3] (la méthode peut être appliquée car $F(z,u)$ est exprimé sous la forme d'une puissance de m et satisfait donc de bonnes propriétés de concentration), on peut montrer que la fonction caractéristique de $\mathcal{X}_k$, centrée et normalisée, converge vers $e^{-x^2\frac{1}{2}}$. Cela assure que la distribution de $\mathcal{X}_k$ est asymptotiquement $\mathbb{E}[\mathcal{X}_k] + \sigma[\mathcal{X}_k]\dot{G}$, où G suit une loi standard gaussienne.

Construction des estimateurs $\mathcal{H}_j$

Les estimateurs $\mathcal{H}_j$ du nombre de boules lancées n construits à partir des nombres $\mathcal{X}_j$ de cases remplies avec au plus j éléments sont analysés dans le Théorème 5. L'idée pour la construction est que, pour certains types de variables aléatoires $\mathcal{X}$ avec $\mathbb{E}[\mathcal{X}] = f(n)$, on obtient très simplement une estimée de n en utilisant l'inverse fonctionnel de f, car on a $\mathbb{E}[f^{-1}(\mathcal{X})] = n$. Le théorème d'inversion de l'espérance (Théorème 4), illustré en figure 3.1, précise des conditions suffisantes pour de telles variables aléatoires.

Théorème 4 (Théorème d'inversion de l'espérance) *Soit $\mathcal{X}(n)$, notre observée, une variable aléatoire d'espérance $\mu(n)$ et d'écart type $\sigma(n)$. On suppose que*

1. $\mathcal{X}(n) \underset{n\to\infty}{\sim} \mu(n) + \sigma(n)G$, *avec* G *une gaussienne standard, et que*
2. $\sigma(n)$ *est petit devant* $\mu(n)$, *i. e.* $\sigma(n) = o(\mu(n))$ *quand* $n \to \infty$.

On a alors $\mu^{-1}(X(n))$ *est un estimateur non biaisé de* n, *c'est-à-dire* $\mathbb{E}[\mu^{-1}(X(n))] \underset{n\to\infty}{\sim} n$, *d'écart type donné par*

$$\boxed{\sigma[\mu^{-1}(X(n))] \underset{n\to\infty}{\sim} \frac{1}{\mu'(n)}\sigma(n).}$$

Preuve. On fixe $n = n_0$ et on s'intéresse à l'inverse de l'espérance μ en $\mathcal{X}(n_0)$:

$$\mu^{-1}(\mathcal{X}(n_0)) = \mu^{-1}(\mu(n_0) + \sigma(n_0)G)$$

Comme $\sigma(n_0)$ et donc $\sigma(n_0)G$ sont supposés petits devant $\mu(n_0)$, on peut effectuer un développement de Taylor en $\mu(n_0)$. On obtient

$$\mu^{-1}(\mathcal{X}(n_0)) = n_0 + (\mu^{-1})'(\mu(n_0))\sigma(n_0)G + O(\sigma(n_0)^2G^2) = n_0 + \frac{\sigma(n_0)G}{\mu'(n_0)} + O(\sigma(n_0)^2G^2).$$

□

Dans le cadre du Comptage par collision, les observées $\mathcal{X}_j$ suivent une distribution limite gaussienne. Leurs espérances sont en m, ainsi que leurs variances, ce qui donne des écarts-types en $\sqrt{m}$. On peut donc appliquer le théorème précédent pour obtenir les estimateurs $\mathcal{H}_j$.

Théorème 5 (Estimateurs du Comptage par Collisions généralisé) *Soit* $\mathcal{X}_j$ *le nombre d'urnes où sont tombées au plus* j *éléments. On définit* $f(n) := \mathbb{E}_n[\mathcal{X}_j]$ *et* $g(n) := \mathbb{V}_n[\mathcal{X}_j]$ *(les formules sont données précédemment). Soient* $\mathcal{H}_j$ $(j < k)$ *les variables aléatoires définies par*

$$\boxed{\mathcal{H}_j := f^{-1}(X_j).}$$

Les $\mathcal{H}_j$ *sont des estimateurs non biaisés de* n, *c'est-à-dire* $\mathbb{E}[\mathcal{H}_j] \underset{n\to\infty}{\sim} n$, *et leurs écarts-types sont donnés par*

$$\boxed{\sigma(\mathcal{H}_j) \underset{n\to\infty}{\sim} \frac{1}{f'(n)}\sqrt{g(n)}.}$$

3.1.4 Application à MINCOUNT*

Dans cette section, on introduit un algorithme qui est capable de traiter des multi-ensembles de toutes cardinalités. L'idée est de combiner les estimées de EXTENDED HIT COUNTING et de TRUNCATED MINCOUNT* et de choisir le plus précis pour chaque cardinalité. En lisant les éléments d'un fichier (de cardinalité inconnue n), l'algorithme MINCOUNT*

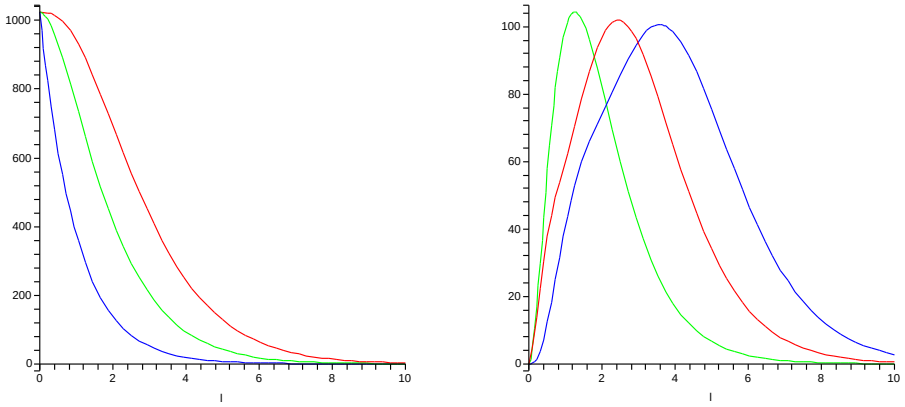


FIG. 3.2 – Espérances et Variances de $\mathcal{X}_0$, $\mathcal{X}_1$ et $\mathcal{X}_2$ ($m = 1024$).

garde en mémoire les trois premiers minimums des valeurs hachées pour pouvoir à la fin utiliser le troisième minimum. Le Théorème 6 construit les trois estimées du EXTENDED HIT COUNTING qui correspondent, $\mathcal{H}_0$, $\mathcal{H}_1$, $\mathcal{H}_2$, et donne leur précision en fonction du taux de remplissage $\lambda := n/m$. L'analyse montre (voir la Section 3.1.4) que, *en choisissant le meilleur estimateur –en fonction d'un λ_0 déterminé selon une heuristique très simple—, l'algorithme atteint une erreur relative de quelques pourcents pour tout λ (2% pour $m = 2^{10}$, voir la Figure 3.4). L'introduction du* EXTENDED HIT COUNTING *améliore, pour certains taux λ, la précision par un facteur proche de 4 —en comparaison du* HIT COUNTING *classique.*

Les estimateurs

Lemme 3 (Espérances et variances de $\mathcal{X}_0$, $\mathcal{X}_1$ et $\mathcal{X}_2$) *Quand n et m tendent vers l'infini avec $n/m = \lambda$,*

1. *les espérances des nombres de boites avec au plus 0, 1 ou 2 éléments sont données par*

$$\begin{aligned} \mathbb{E}[\mathcal{X}_0] &\underset{n\to\infty}{\sim} me^{-\lambda} \\ \mathbb{E}[\mathcal{X}_1] &\underset{n\to\infty}{\sim} me^{-\lambda}(1+\lambda) \\ \mathbb{E}[\mathcal{X}_2] &\underset{n\to\infty}{\sim} me^{-\lambda}(1+\lambda+\lambda^2/2); \end{aligned}$$

2. les variances des nombres de boites avec au plus 0, 1 ou 2 éléments sont données par

$$\begin{array}{lll}\mathbb{V}[\mathcal{X}_0] & \underset{n\to\infty}{\sim} & m(e^{-\lambda}-e^{-2\lambda}(1+\lambda)) \\ \mathbb{V}[\mathcal{X}_1] & \underset{n\to\infty}{\sim} & m(e^{-\lambda}(1+\lambda)-e^{-2\lambda}(1+2\lambda+\lambda^2+\lambda^3)) \\ \mathbb{V}[\mathcal{X}_2] & \underset{n\to\infty}{\sim} & m(e^{-\lambda}(1+\lambda+1/2\lambda^2)- \\ & & e^{-2\lambda}(1+2\lambda+2\lambda^2+\lambda^3+1/4\lambda^4+1/4\lambda^5)).\end{array}$$

On peut voir ces fonctions en figure 3.2.

Preuve Les espérances s'obtiennent directement. Pour les variances on a par exemple

$$\begin{array}{rcl}\mathbb{V}[\mathcal{Y}_1] & = & m(m-1)\frac{(m-2)^n}{m^n}\left(1+\frac{n}{m-2}+\frac{n(n-1)}{(m-2)^2}\right) \\ & + & m\frac{(m-1)^n}{m^n}\left(1+\frac{n}{m-1}\right) \\ & - & m^2\frac{(m-1)^{2n}}{m^{2n}}\left(1+\frac{n}{m-1}\right)^2\end{array}$$

Pour les calculs asymptotiques, on remplace n par λm et on développe tous les termes à l'ordre 2 $(1/m^2)$. Ainsi, par exemple

$$\begin{array}{rcl} m^2\frac{(m-1)^{2n}}{m^{2n}}\left(1+\frac{n}{m-1}\right)^2 & = & m^2(1-1/m)^{2\lambda m}\left(1+\frac{\lambda}{(1-1/m)}\right)^2 \\ & = & m^2\left(e^{-2\lambda}-\frac{\exp^{-2\lambda}\lambda^2}{\lambda m}+o\left(\frac{1}{m^2}\right)\right)\left(1+\lambda(1+1/m+o\left(\frac{1}{m^2}\right))\right)^2 \\ & = & m^2\exp^{-2\lambda}\left((1-\frac{\lambda}{m}+o\left(\frac{1}{m^2}\right))(1+2\lambda+\lambda^2+2\frac{\lambda}{m}+2\frac{\lambda^2}{m}+o\left(\frac{1}{m^2}\right))\right) \\ & = & m^2\exp^{-2\lambda}\left(1+2\lambda+\lambda^2+\frac{\lambda}{m}-\frac{\lambda^3}{m}+o\left(\frac{1}{m^2}\right)\right)\end{array}$$

On obtient les résultats avec des calculs similaires. □

Théorème 6 (Estimateurs du Extended Hit Counting pour Mincount*) *$\mathcal{H}_0$, $\mathcal{H}_1$ et $\mathcal{H}_2$, définis en suivant le Théorème 5, sont des estimateurs non biaisés de n et leurs écarts-types sont donnés par*

$$\begin{array}{lll}\sigma(\mathcal{H}_0) & \underset{n\to\infty}{\sim} & \frac{1}{e^{-\lambda}}\sqrt{m(e^{-\lambda}-e^{-2\lambda}(1+\lambda))} \underset{n\to\infty}{\sim} \sqrt{m(e^{\lambda}-\lambda-1)}, \\ \sigma(\mathcal{H}_1) & \underset{n\to\infty}{\sim} & \frac{1}{e^{-\lambda}(1+\lambda)}\sqrt{m(e^{-\lambda}(1+\lambda)-e^{-2\lambda}(1+2\lambda+\lambda^2+\lambda^3))}, \\ \sigma(\mathcal{H}_2) & \underset{n\to\infty}{\sim} & \frac{1}{e^{-\lambda}(1+\lambda+\lambda^2/2)}(m(e^{-\lambda}(1+\lambda+1/2\lambda^2)- \\ & & e^{-2\lambda}(1+2\lambda+2\lambda^2+\lambda^3+1/4\lambda^4+1/4\lambda^5)))^{1/2}.\end{array}$$

λ	2	4	6	8	10
$\sigma(\mathcal{H}_0)/n$	0.131	0.220	0.415	0.852	1.855
$\sigma(\mathcal{H}_1)/n$	0.071	0.106	0.176	0.316	0.614
$\sigma(\mathcal{H}_2)/n$	0.062	0.066	0.100	0.164	0.287

$m = 2^6$

λ	2	4	6	8	10
$\sigma(\mathcal{H}_0)/n$	0.065	0.110	0.207	0.426	0.927
$\sigma(\mathcal{H}_1)/n$	0.036	0.053	0.088	0.158	0.307
$\sigma(\mathcal{H}_2)/n$	0.031	0.033	0.050	0.082	0.143

$m = 2^8$

λ	2	4	6	8	10
$\sigma(\mathcal{H}_0)/n$	0.033	0.055	0.104	0.213	0.464
$\sigma(\mathcal{H}_1)/n$	0.018	0.026	0.044	0.079	0.153
$\sigma(\mathcal{H}_2)/n$	0.016	0.016	0.025	0.041	0.072

$m = 2^{10}$

λ	2	4	6	8	10
$\sigma(\mathcal{H}_0)/n$	0.016	0.028	0.052	0.106	0.232
$\sigma(\mathcal{H}_1)/n$	0.009	0.013	0.022	0.040	0.077
$\sigma(\mathcal{H}_2)/n$	0.008	0.008	0.013	0.020	0.036

$m = 2^{12}$

FIG. 3.3 – Erreur standard des estimées du EXTENDED HIT COUNTING pour $m = 6, 8, 10, 12$.

Plage d'utilisation des estimateurs

On a donc maintenant quatre estimateurs à notre disposition : les trois du comptage par collision généralisé, $\mathcal{H}_0$, $\mathcal{H}_1$, $\mathcal{H}_2$, et MINCOUNT*tronqué, ξ_t^*, qui se confond avec MINCOUNT*dès que toutes les boites sont remplies. Il s'agit maintenant, pour chaque taux de remplissage $\lambda = n/m$, d'utiliser l'estimateur le plus précis. Pour λ croissant, c'est-à-dire, à m fixé, pour un nombre d'éléments n croissant à estimer, les estimateurs les plus précis sont d'abord $\mathcal{H}_0$, puis $\mathcal{H}_1$, $\mathcal{H}_2$ et enfin ξ_t^*. Pour obtenir les valeurs de transition entre les différentes estimées, on résout des équations du type $\mathbb{V}[\mathcal{H}_0] = \mathbb{V}[\mathcal{H}_1]$. *L'algorithme doit utiliser*

- *l'estimée $\mathcal{H}_0$ pour λ entre 0 et $\lambda_{01} \approx 0.77$, avec λ_{01} solution de $\mathbb{V}[\mathcal{H}_0] = \mathbb{V}[\mathcal{H}_1]$;*
- *l'estimée $\mathcal{H}_1$ pour λ entre λ_{01} et $\lambda_{12} \approx 1.74$;*
- *l'estimée $\mathcal{H}_2$ entre λ_{12} et λ_{2*} avec λ_{2*} solution de $\mathbb{V}[\mathcal{H}_2] = \mathbb{V}[\xi_t^*]$ —celle-ci dépend de m, voir le tableau ci-dessous*

$\log_2 m$	0	2	4	6	8	10	20
λ_{2*}	7.1	5.5	5.2	5.2	5.2	5.2	5.2

- *et ensuite l'estimée ξ_t^* du* TRUNCATED MINCOUNT*.

La Figure 3.4 montre que pour $m = 2^{10}$, *en utilisant le meilleur estimateur, l'algorithme atteint une erreur standard d'environ 2% pour tout λ* (regarder l'enveloppe inférieure des courbes). L'introduction du EXTENDED HIT COUNTING donne une estimée plus précise pour de plus grands λ (voir la Figure 3.3). Par exemple, pour $\lambda = 6$, si nous avions utilisé le HIT COUNTING classique, nous aurions obtenu une erreur relative de 5.5% pour $m = 2^{10}$ ($\sigma(\mathcal{H}_0)/n = 0.055$). En comparaison, l'erreur relative de $\mathcal{H}_2$ est de seulement 1.4%. *Dans ce cas, l'introduction du* EXTENDED HIT COUNTING *a amélioré la précision par un facteur proche de 4.*

Quelques considérations d'ingénierie algorithmique

Choix de l'estimateur en pratique Le problème vient ici du fait que l'estimateur doit être choisi en fonction de $\lambda = n/m$ alors que celui-ci est inconnu (n doit être estimé). Le

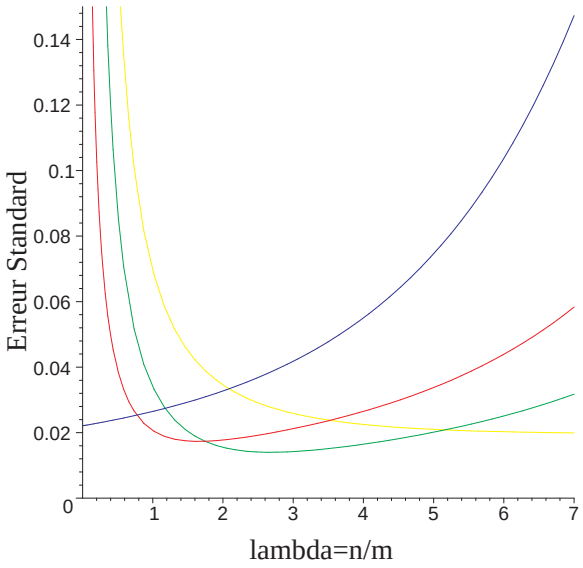


FIG. 3.4 – Erreurs standard données par la théorie des estimées du EXTENDED HIT COUNTING $\mathcal{H}_0$ (bleu) $\mathcal{H}_1$ (rouge) $\mathcal{H}_2$ (vert) et du TRUNCATED MINCOUNT* (jaune) en fonction du taux de remplissage $\lambda := n/m$ pour $m = 2^{10}$.

choix de l'estimée par l'algorithme, en pratique, est fait en suivant une heuristique très simple. Si toutes les boites ont un troisième minimum, on utilise MINCOUNT*. Sinon, le principe est de calculer un λ_0, qui donne un premier ordre de grandeur de la valeur réelle de λ. Ce λ_0 est ensuite utilisé pour choisir une estimée. La Figure 3.4 donne les erreurs standard des quatre estimées pour $m = 2^{10}$. On voit que $\mathcal{H}_2$ donne un bon λ_0, sauf pour les petites valeurs de λ. H_0, au contraire, donne une bon λ_0 pour les petites valeurs. L'idée est donc de calculer les estimées $\mathcal{H}_0$ et $\mathcal{H}_2$. Si elles ont des valeurs très différentes et que $\mathcal{H}_0$ donne un λ plus petit que 2, on utilise $\mathcal{H}_0$ pour obtenir un λ_0, sinon on utilise H_2. Cette heuristique marche très bien pour les applications pratiques.

Calcul des estimées en pratique $\mathcal{H}_0$, $\mathcal{H}_1$ et $\mathcal{H}_2$ sont définies implicitement comme inverses de fonctions connues. L'algorithme les calcule numériquement par dichotomie (Il s'arrête quand la précision de l'inversion dépasse les $1/1000e$).

Conditions d'application du théorème d'inversion de l'espérance pour différentes valeurs de m Le théorème de construction des estimateurs $\mathcal{H}$ ne s'applique que si $\sigma[\mathcal{X}]$ est petit devant $\mathbb{E}[\mathcal{X}]$. Comme $\sigma[\mathcal{X}]$ est en $\sqrt{m}$ et $\mathbb{E}[\mathcal{X}]$ en m, les estimateurs sont bons pour m asymptotique. Qu'en est-il pour différentes valeurs de m ? (Voir la Figure 3.5 pour $m = 2^{10}$.) Le tableau suivant donne le rapport $\mathbb{E}[\mathcal{X}]/\sigma[\mathcal{X}]$ limite de chaque estimateur sur sa plage d'utilisation pour m allant de 4 à 2^{12}.

$\log_2 m$	2	4	6	8	10	12
$\mathbb{E}[\mathcal{X}_0]/\sigma[\mathcal{X}_0](\lambda = 0.77)$	3.20	6.41	12.81	25.63	51.26	102.51
$\mathbb{E}[\mathcal{X}_1]/\sigma[\mathcal{X}_1](\lambda = 1.74)$	3.25	6.51	13.02	26.04	52.07	104.15
$\mathbb{E}[\mathcal{X}_2]/\sigma[\mathcal{X}_2](\lambda = 5.2)$	0.83	1.67	3.34	6.67	13.35	26.69

Les estimateurs sont rapidement bons : dès que m dépasse 256 les trois estimateurs peuvent être utilisés.

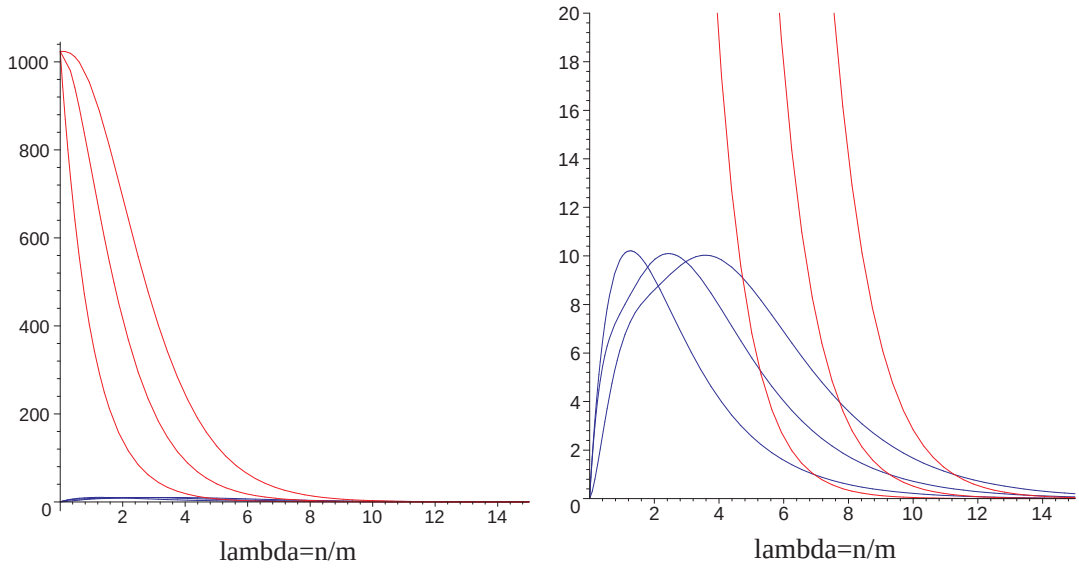


FIG. 3.5 – Comparaison de $\mathbb{E}[\mathcal{X}]$ et de $\sigma(\mathcal{X})$ pour $m = 2^{10}$.

3.1.5 Un petit détour par le problème des grandes cardinalités

L'algorithme de comptage, MINCOUNT*, utilise une fonction de hachage de l'espace des éléments du multi-ensemble étudié vers l'intervalle réel $[0, 1]$. Cette fonction de hachage est choisie de façon à "bien répartir" les valeurs hachées sur l'intervalle et à ne pas générer trop de collisions (sur le choix de la fonction de hachage, voir la Section 3.2.1). Néanmoins, l'algorithme de comptage est conçu pour pouvoir traiter des multi-ensembles de très grandes cardinalités, qui peuvent avoir jusqu'à quelques milliards d'éléments distincts. Ainsi, quand le nombre de valeurs distinctes n'est plus négligeable devant la taille de l'espace des valeurs hachées, aussi bonne soit la fonction de hachage et même pour une fonction idéale, les collisions deviennent inévitables. Pour un programme C, si les valeurs hachées sont stockées sur un `unsigned int` de taille 32 bits, le nombre de valeurs hachées possibles est $2^{32} \approx 4$ milliards. Dans ce cas précis les collisions seront nombreuses pour des fichiers de plusieurs

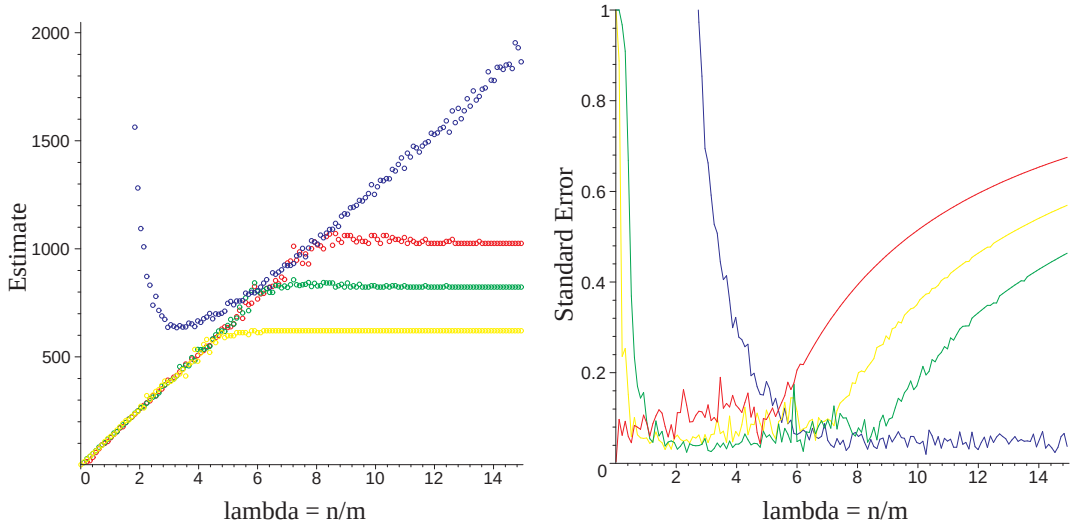


FIG. 3.6 – Estimées (gauche) du EXTENDED HIT COUNTING $\mathcal{H}_0$ (jaune), $\mathcal{H}_1$ (vert), $\mathcal{H}_2$ (rouge) et de TRUNCATED MINCOUNT* (bleu) et leur erreur standard (droite) pour des simulations sur des fichiers de différentes cardinalités correspondant à des taux de remplissage $\lambda := n/m$ de 0 à 15 pour $m = 128$.

centaines de millions d'éléments distincts. On appelle ceci le *Problème des grandes Cardinalités*. On a vu en Section 3.1.3 que l'on peut estimer le nombre de collisions en fonction de la cardinalité n des multi-ensembles. On se sert ensuite de cette information pour corriger les estimées. Si μ est le nombre de valeurs hachées possibles et X le nombre estimé de valeurs hachées atteintes à la fin de l'algorithme, on a

$$\mathbb{E}[X] \underset{n\to\infty}{\sim} \mu(1 - e^{-\lambda}),$$

avec $\lambda := n/\mu$. Remarquez que quand $n = \mu$, $\mathbb{E}[X] = (1 - 1/e)\mu \approx 0.65\mu$. On est à 35% d'erreur. On obtient un mécanisme de précision en inversant

$$n* := \mu \ln \frac{1}{1 - X/\mu}.$$

Durand montre dans sa thèse [Dur04] que l'on peut ainsi estimer précisément la cardinalité de fichiers ayant jusqu'à $\approx 10\mu$ éléments distincts.

3.1.6 Validation

Les estimateurs du comptage par collision, $\mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2$ et TRUNCATED MINCOUNT* ont été implémentés en Maple et en C (les codes sont disponibles en annexe...). De nombreuses simulations ont été effectuées pour différentes précisions des estimées —qui correspondent

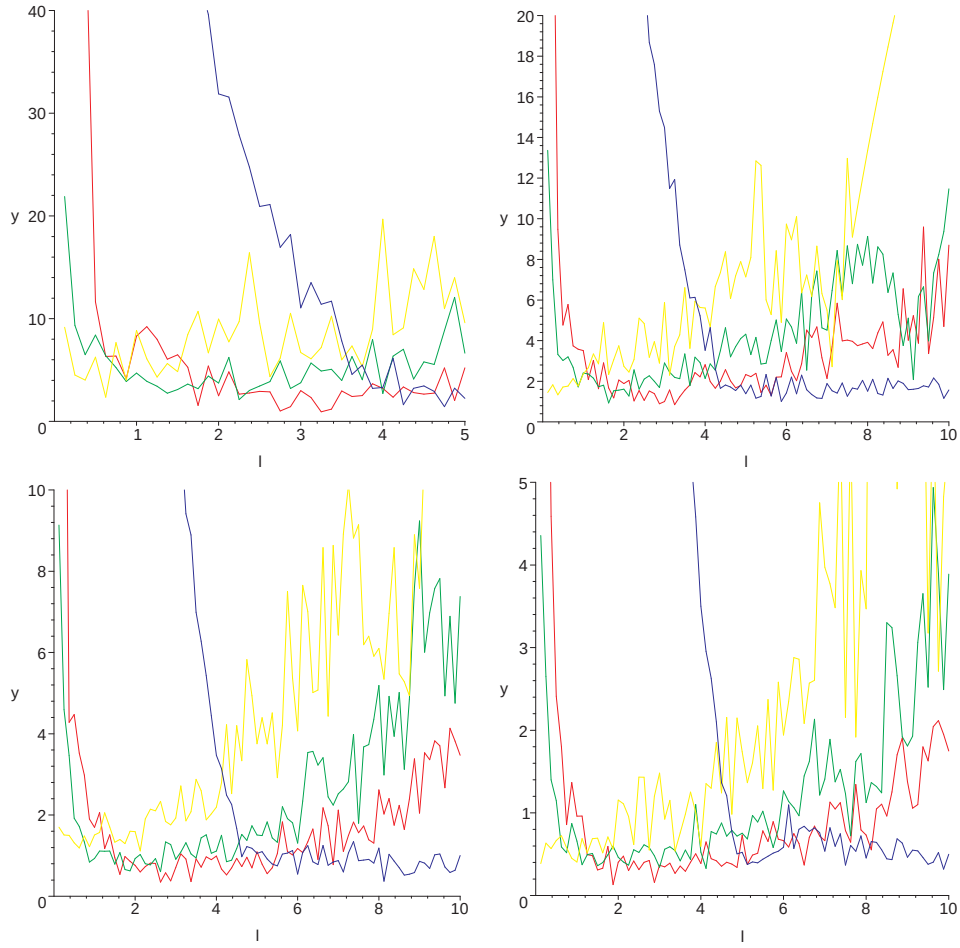


FIG. 3.7 – Erreur standard expérimentale (en pourcents) des estimées du EXTENDED HIT COUNTING $\mathcal{H}_0$ (jaune) $\mathcal{H}_1$ (vert) $\mathcal{H}_2$ (rouge) et de TRUNCATED MINCOUNT* (bleu) pour des simulations avec des fichiers de différentes cardinalités qui correspondent à des taux de remplissage $\lambda := n/m$ de 0 à 10. Des résultats sont présentés pour $m = 2^8$ (en haut à gauche), $m = 2^{10}$ (haut droite), $m = 2^{12}$ (bas gauche) and $m = 2^{14}$ (bas droite).

à des nombres d'expériences simulées variant de $m = 1$ à $m = 2^{14}$. Pour chaque valeur de m, on a utilisé des fichiers comptant de 0 à $15m$ éléments distincts —λ variant de 0 à 15. La Figure 3.6 montre un exemple de simulations Maple pour $m = 128$. Chaque point correspond à une moyenne de dix simulations indépendantes. La Figure 3.7 résume des simulations effectuées avec notre programme C pour différentes valeurs de m.

Le graphique de gauche de la Figure 3.6 présente les estimées données par les quatre estimateurs pour des fichiers de taille allant de 0 à $15m$. On constate que les estimateurs du Hit Counting $\mathcal{H}_0$, $\mathcal{H}_1$, $\mathcal{H}_2$ ont le même comportement avec un petit décalage : ils estiment bien n sur une première plage (jusqu'à environ $\lambda = 4, 6, 8$). Puis ils deviennent très mauvais. Par contre l'estimateur tronqué a un comportement inverse avec une transition vers $\lambda = 6$. *L'existence d'une diagonale continue* montre que l'on dispose, sur chaque plage d'utilisation de notre programme, d'un bon estimateur de n, quel que soit le nombre d'éléments distinct à estimer (i. e. pour tout λ).

Le graphique de droite de la Figure 3.6 et la Figure 3.7 montrent les erreurs standard des estimées. On observe bien que $\mathcal{H}_1$, $\mathcal{H}_2$ et ξ_t^* sont très mauvais pour les petites valeurs et que $\mathcal{H}_0$ est donc le bon estimateur pour les plus petits fichiers. Les *zones de transition* où un estimateur devient le meilleur sont proches de ce que la théorie prévoit : vers $\lambda = 1$, l'erreur standard de H_1 devient plus petite que celle de H_0 ; vers $\lambda = 2$, H_2 devient le plus précis ; enfin, quand $\lambda \geq 5$, il faut utiliser ξ_t^*. On constate aussi que l'on a bien pour chaque λ un estimateur d'une précision de quelques pourcents, d'autant plus précis que m est grand : environ 4% pour $m = 2^8$, 2% pour $m = 2^{10}$, 1% pour $m = 2^{12}$ et 0.5% pour $m = 2^{14}$.

Pour étudier les transitions de l'algorithme entre les différents estimateurs, on a modifié un peu l'implémentation pour qu'elle nous donne des estimées au cours de la lecture du fichier. On montre ainsi en Figure 3.8 Gauche l'erreur standard de l'estimée tous les 10 000 éléments lus pour un fichier de 500 millions d'entiers différents ($m = 2^{10}$). L'erreur standard reste bien autour de l'erreur théorique attendue de 2%. La Figure 3.8 Droite est une loupe sur le début de l'exécution. Les transitions entre les 4 estimateurs y sont visibles. Ainsi, autour de la m-ième valeur ($m = 1024$), on voit très clairement le passage de $\mathcal{H}_0$ à $\mathcal{H}_1$. $\mathcal{H}_2$ prend le relai vers la 2000-ième valeur puis TRUNCATED MINCOUNT*vers la 7000-ième. L'erreur standard reste autour de 2%.

À la fin de l'étude, on dispose d'un algorithme qui, pour chaque taille de fichier, peut choisir un estimateur approprié qui donne le nombre d'éléments distincts *n à quelques pourcents près.*

3.2 Implémentation de l'algorithme MINCOUNT*

Les algorithmes construits à partir de statistiques d'ordre sont parmi les algorithmes les plus rapides pour estimer la cardinalité. Non seulement ils ne font qu'une passe sur les données, mais cette passe est en plus très rapide (voir Section 3.2.2). C'est un avantage

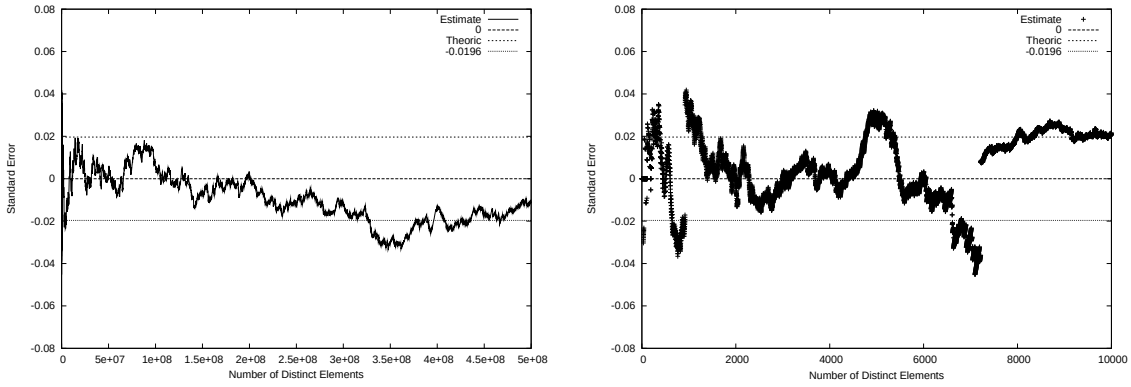


FIG. 3.8 – Le graphique de gauche donne l'erreur standard des estimées données par l'algorithme au cours de la lecture d'un fichier de 500 millions d'entiers distincts pour $m = 2^{10}$. Celui de droite est une loupe sur les 10 000 premiers éléments. On y voit les transitions de l'algorithme entre les 4 estimateurs.

crucial dans le contexte de l'analyse en-ligne du trafic internet où un routeur dispose de seulement quelques dizaines d'instructions machine pour traiter un paquet. Nous avons effectué une implémentation optimisée de l'algorithme MINCOUNT* dont les principes sont exposés succinctement en Section 3.2.3. Nous avons mesuré ses temps d'exécution sur des fichiers de différents types et différentes tailles (voir la Figure 3.11). L'implémentation s'avère très rapide : elle n'est que 3 à 4 fois plus lente que la commande `cat -t` qui lit juste les caractères de son entrée . Mais tout d'abord, la question de la fonction de hachage est discutée en Section 3.2.1.

3.2.1 Choix de la Fonction de Hachage

Contraintes pour la fonction de hachage

On a vu que l'algorithme MINCOUNT* repose très fortement sur l'existence d'une fonction de hachage. Nos analyses sont faites pour un multi-ensemble idéal construit de la manière suivante : tirer aléatoirement n réels uniformément sur l'intervalle $[0, 1]$. Les dupliquer et les mélanger de façon arbitraire. La fonction de hachage assure le passage du multi-ensemble "réel" au multi-ensemble idéal.

Comme un ordinateur ne peut traiter que des ensembles finis, le tirage d'un nombre sur l'intervalle réel $[0, 1]$ s'effectue en choisissant un nombre entier x entre 0 et H, avec H le nombre de valeurs hachées possibles. On divise ensuite x par H pour se ramener entre 0 et 1.

Le comportement de la fonction de hachage doit se rapprocher de celui d'un tirage

aléatoire uniforme sur $[0,1]$ et, en particulier, il faut :

- une *bonne répartition* sur tout l'ensemble des valeurs hachées possibles et
- *pas plus, pas moins de collisions* que lors d'un tirage uniforme.

La fonction de hachage doit être *rapide*. Des liens de capacité $40Gbps$ sont déployés dans les réseaux (liens OC-748 des réseaux SONET). En supposant une taille moyenne de paquets de 300 octets, un paquet peut arriver toutes les $300 \cdot 8/4 \cdot 10^{10} = 60$ nanosecondes. Pour un processeur de 4 GigaHertz, cela ne donne que $60 \cdot 10^{-9} \cdot 10^{9} = 240$ instructions machine pour traiter un paquet. Un routeur doit faire son travail classique (router les paquets...) en plus de la boucle interne de l'algorithme ce qui ne donne que quelques dizaines d'instructions machine pour la fonction de hachage.

Discussion sur les familles de fonctions de hachage

Fonctions de hachage congruentielles linéaires. On peut obtenir une fonction de hachage très simplement. Si X est le nombre à hacher, on renvoie $H(x)$ définit comme suit

$$h(X) := aX + b \mod m.$$

Pour un bon choix des coefficients a, c et m, le résultat aura de bonnes propriétés statistiques [Knu98].

LFSR. Un registre à décalage à rétroaction linéaire ou LFSR (pour Linear Feedback Shift Register) binaire de longueur L est un dispositif composé d'un registre à décalage contenant L bits $(s_t, \cdots, s_{t+L+1})$ et d'une fonction de rétroaction linéaire. Un chaîne binaire $(e_0, \cdots, e_t, \cdots)$ est connectée en entrée. À chaque top d'horloge, le bit de droite s_t constitue la sortie du LFSR et les autres bits sont décalés d'une position vers la droite. Le nouveau bit s_{t+2} placé dans la cellule de gauche du registre est donné par une fonction linéaire des bits du registre et du bit en entrée :

$$s_{t_L} = c_1 s_{t+L-1} + c_2 s_{t+L-2} + \cdots + c_L s_t + e_t$$

où les coefficients de rétroaction $(c_i)_{1 \leq i \leq L}$ sont des éléments de $\mathbb{F}_2$. Ils définissent un polynôme de rétroaction P :

$$P(X) := 1 + c_1 X + c_2 X^2 + \cdots + c_L X^L.$$

Pour un bon choix du polynôme de rétroaction (primitif, c'est-à-dire irréductible et d'ordre $2^L - 1$) les suites produites par le LFSR pour les différentes entrées ont de bonnes propriétés statistiques (grandes périodes, peu de collisions) [Gol82], [LN83, VIII].

Fonctions de hachage cryptographiques. Les fonctions de hachage cryptographiques sont utilisées très couramment parce qu'elles rendent possibles des mécanismes d'authentification par mot de passe sans stockage de ce dernier. Un algorithme de hachage est considéré comme cryptographique si les conditions suivantes sont remplies :

- il est très difficile de trouver le contenu du message à partir de la signature (attaque sur la première préimage) ;
- à partir d'un message donné et de sa signature, il est très difficile de générer un autre message qui donne la même signature (attaque sur la seconde préimage)
- il est très difficile de trouver deux messages aléatoires qui donnent la même signature (résistance aux collisions).

Par très difficile, on entend "techniquement impossible" que ce soit au niveau algorithmique ou matériel. Certaines fonctions de hachage cryptographiques classiques comme `sha1` ou `md5` sont implémentées dans les bibliothèques standard des systèmes d'exploitation.

Discussion. Les fonctions de hachage congruentielles sont simples, mais elles utilisent des opérations coûteuses en temps de calcul : des multiplications modulaires. Si la taille de l'entrée à hacher est plus grande que le modulo, il faut la hacher par petits bouts, ce qui va nécessiter plusieurs opérations.

Les fonctions de hachage cryptographiques ont tous les avantages (robustes, peu de collision,...) mais elles sont plus lentes. En effet, certaines de leurs propriétés ne sont pas requises dans le contexte des algorithmes probabilistes de comptage, comme, par exemple, les difficultés pour inverser le hachage.

Les LFSR permettent d'avoir directement des valeurs hachées sur 32 bits pour des tailles d'entrées quelconques. Elles ne font que des opérations binaires très simples (XOR) ce qui les rend très rapides. On peut de plus effectuer des décalages de plusieurs bits d'un coup en utilisant des masques. De plus elles sont modulables au sens suivant : en choisissant le nombre de décalages à vide que l'on fait après avoir lu tous les bits de l'entrée, on peut effectuer un choix entre le temps d'exécution et la "bonne" répartition des valeurs hachées. On a donc utilisé ce type de fonctions de hachage. En particulier pour l'étude des temps d'exécution de la Section 3.2.3 on s'est servi d'un LFSR de polynôme de rétroaction $P(X) = 1 + X^{13} + X^{33}$. Son implémentation optimisée a été par Cédric Lauradoux du Projet Codes de l'INRIA (`http://www-rocq.inria.fr/codes/LFSR/`).

Cependant pour des multi-ensembles très structurés et de cardinalités très grandes, elles peuvent rencontrer quelques problèmes de répartition et de collisions. Dans ces cas là, malgré le surcoût en temps d'exécution, il est conseillé d'avoir recours à une fonction de hachage cryptographique comme `md4` (qui est un peu plus rapide que `sha1`).

3.2.2 Boucle interne très simple

À la vitesse de 10 Gbps, si on suppose une taille moyenne de paquets de 300 bytes [FDL+01], un nouveau paquet arrive toutes les 240 ns —voir [IDGM01] pour une étude sur la surveillance des liens internet de très grandes capacités Même si l'on ignore le temps significant de transfert des données du liens vers le processeur, cela ne donne que 595 instructions machines à un processeur parmi les plus rapides de 2.5 GHz pour traiter un paquet. Ce nombre est même réduit à 150 instructions pour les nouvelles vitesses qui atteignent 40 Gbps. Le temps d'exécution des algorithmes est donc crucial non seulement pour leur efficacité mais

```
TANT QUE valeur hachée
    SI h < M^(2)
        SI h < M
            M^(2) = M
            M = h
        SINON M^(2) = h
```

FIG. 3.9 – Boucle interne pour $k = 2$.

aussi pour leur simple faisabilité. L'algorithme MINCOUNT est composé principalement d'une boucle interne très simple qui maintient les k premiers minimums des valeurs hachées du multi-ensemble, voir la Figure 3.2.2 pour $k = 2$. On appelle *coût d'une exécution d'un algorithme* le nombre d'instructions élémentaires (ici le nombre de comparaisons et d'affectations) effectuées. On parle de *coût moyen d'un algorithme* pour désigner l'espérance du coût sur toutes les exécutions possibles.

Proposition 3 *Soit k un entier fixé. Le coût moyen pour trouver le $k-$ième minimum parmi n variables aléatoires uniformes sur $[0,1]$ est*

$$\mathbb{E}_n[Cost] \underset{n\to\infty}{=} n + o(n). \tag{3.2}$$

En effet, quand on traite une valeur hachée de la séquence, il y a schématiquement deux cas. Soit cette valeur est plus grande que le $k-$ième minimum et on n'a besoin de faire qu'une seule comparaison. Soit cette valeur est plus petite et on doit alors faire quelques comparaisons de plus pour trouver son rang parmi les k premiers minimums et quelques affectations pour actualiser les minimums. On montre facilement qu'asymptotiquement le nombre de fois où cette actualisation est faite est négligeable —ainsi, pour $k = 1$, la probabilité que la t-ième valeur soit un nouveau minimum est $1/(t+1)$ ce qui donne un nombre moyen d'actualisation du minimum de $H_{n+1} \underset{n\to\infty}{\sim} \ln n$. Le coût moyen du traitement d'une valeur hachée est donc une comparaison.

Dans le cas d'un multi-ensemble de taille N et de cardinalité n, un résultat de complexité précis ne peut être énoncé en raison de l'absence d'hypothèses sur la nature et le nombre des répétitions qui peuvent être quelconques. Néanmoins, le coût moyen de l'algorithme MINCOUNT, dans l'esprit est de $\mathbb{E}_N[\text{Cost}] \underset{N\to\infty}{=} N + o(N)$. On peut le comparer avec celui d'autres algorithmes. Par exemple, la boucle interne des algorithmes PROBABILISTIC COUNTING [FM83] et LOGLOG COUNTING [DF03], au lieu d'effectuer une comparaison de la valeur hachée avec le minimum temporaire, doit trouver le premier 1 d'une séquence infinie de bits aléatoires. Cette opération ce fait en moyenne avec 3 instructions au lieu de seulement 1. Cela donne un avantage en terme de temps d'exécution à l'algorithme MINCOUNT.

Fichiers tests	Taille (Mo)	Nb d'éléments	Éléments distincts	Nature
hamlet.txt	0.164	32865	5044	Texte anglais
TAU.ip	9.9	677801	10569	Logs de routeurs
psd-superfamily.xml	11	186700	19615	Séq protéinique
access log cut	26	829224	101398	Logs d'accès
psd-author.xml	170	5668287	112226	Séq protéinique
psd-uid.xml	80	1949416	1195343	Séq protéinique
psd-cut.xml	681	23073127	4052131	Séq protéinique
aleatoire.dat	50	5000000	4993301	Nombres aléatoires
auck.ip	242	13856690	9083474	Logs de routeurs
100millions.dat	848	100000000	100000000	Texte structuré

FIG. 3.10 – Jeux de données de différentes natures et différentes tailles.

3.2.3 Temps d'exécution et Validation

Optimisation du programme

On a écrit une première implémentation en `C` de l'algorithme MINCOUNT*. Nous avons réussi à diviser par 10 le temps d'exécution de cette implémentation. Très succinctement, les étapes principales de l'optimisation ont été les suivantes.

- L'analyse des temps d'exécution des différentes parties du programme. L'idée est de regarder à l'aide de l'utilitaire... quelles sont les fonctions appelées un grand nombre de fois pendant l'exécution et celles qui utilisent un grand pourcentage du temps pour pouvoir concentrer ces efforts dessus.
- Le choix de la structure du programme. Dans les parties cruciales du programme, on a essayé de limiter au maximum les appels de fonctions et l'emploi de tests et d'effectuer ceux-ci en amont. Par exemple, la copie en plusieurs exemplaires du code, si elle alourdit le code, permet de limiter le nombre de tests.
- Un temps non négligeable est utilisé par le programme pour la gestion des entrées sorties. On s'est inspiré du code de la commande unix `cat` qui lit son entrée et l'affiche en sortie. Remplacer une lecture ligne par ligne à l'aide de fonctions haut niveau comme `scanf`, par la lecture à l'aide d'un grand buffer avec `write` a permis un grand gain de temps.

Jeux de données.

Pour valider l'algorithme MINCOUNT* de nombreuses simulations ont été effectuées en utilisant des fichiers de différents types et et de différentes tailles. Un aperçu en est donné dans la Figure 3.10. Les fichiers tests "psd" sont extraits d'une base de données de Séquence protéiniques. Elle est mise à disposition du public par le groupe base de donnée de

l'Université de Washington (`http://www.cs.washington.edu/research/xmldatasets/`). La base de donnée est sous format XML. On en a extrait trois attributs : l'identifiant des protéines concernées (<uid>), les auteurs qui les ont analysées (<author>) et une partie de leur classification (<superfamily>). "TAU.ip" et "Auck.ip" sont des traces de routeurs disponible sur le site internet du groupe NLANR (NLANR Measurement and Network Analysis `http://moat.nlanr.net`). Les traces sont des séquences d'en-têtes de paquets anonymisés. "TAU.ip" a été recueillie sur un lien OC-3 du MegaPoP de l'université de Tel Aviv pendant une période d'une minute trente ; "Auck.ip" à l'université d'Auckland pour une fenêtre de temps de 15 minutes. Notre analyse consiste à estimer le nombre de connexions distinctes identifiées par la paire adresse IP source, adresse IP destination. "access log cut" sont des logs d'accès de machines du campus de Rocquencourt de l'INRIA. "hamlet.txt" est le texte du livre de Shakespeare. "aleatoire.dat" est un multi-ensemble de 5 millions d'entiers tirés uniformément entre 0 et 2^{32}, dont 4 993 301 sont distincts. "100M.dat" est la liste de 100 millions d'entiers consécutifs. C'est un ensemble structuré dont l'intérêt est de pouvoir tester les bonnes propriétés d'aléa de l'algorithme.

Timings.

On étudie ici les temps d'exécution d'une implémentation optimisée de l'algorithme MINCOUNT* pour traiter les différents jeux de données. On utilise les commandes `shell` suivantes comme références de temps d'exécution : `cat` qui lit le fichier sans traiter les données ; `cat -T` qui lit le fichier et affiche ^I pour les tabulations ; `wc -l` qui affiche le nombre de lignes du fichier ; `wc -w` qui affiche le nombre de mots du fichier ; `sort -u | wc` qui donne le nombre de lignes différentes du fichier. Le tableau du haut de la Figure 3.11 montre les temps d'exécution en secondes sur les fichiers de notre jeux de données. L'algorithme ne met que quelques secondes pour donner une estimation de la cardinalité de fichiers de quelques millions d'éléments. Par exemple, il met 4.8 secondes pour la trace auck.ip qui a 13 millions d'éléments dont 9 millions sont distincts. Cela correspond à un débit de 53 Megabits par seconde ou de 3 millions d'éléments par seconde. Le tableau du bas de la Figure 3.11 donne ces débits et effectue les ratios entre le temps d'exécution de MINCOUNT* et les commandes `unix`. L'algorithme n'est que 3 à 4 fois plus lent que la commande `cat -T` qui remplace juste les tabulations de son entrée par ^I. Ces résultats sont validés par la Figure 3.12 qui montre que la précision de l'algorithme lors des exécutions chronométrées est proche de celle attendue par la théorie donnée par la seconde ligne du tableau.

3.3 Applications au Trafic Internet

L'introduction de notre algorithme est motivée par le traitement de multi-ensembles de données très grands comme, en particulier, l'analyse en ligne du trafic internet. Dans ce contexte, les éléments du multi-ensemble sont les paquets. Chaque paquet appartient à une connexion. Compter le nombre d'éléments distincts revient à donner le nombre de

Fichiers tests	cat	cat -T	wc -l	wc -w	../mincount	sort -u | wc
hamlet.txt	0.	0.	0.	0.	0.	0.23
TAU.ip	0.01	0.05	0.02	0.27	0.20	15.
psd-superfamily.xml	0.01	0.06	0.01	0.30	0.17	10.
access log cut	0.03	0.13	0.04	0.71	0.43	35.
psd-author.xml	0.17	0.90	0.28	4.59	2.94	251.
psd-uid.xml	0.07	0.41	0.12	2.16	1.29	80.
psd-cut.xml	0.59	3.90	1.27	18.7	11.7	821.
3aleatoire.dat	0.04	0.32	0.14	1.38	1.11	56.
auck.ip	0.21	1.41	0.53	6.9	4.6	303
100M.dat	1.2	6.0	3.22	29.3	21.7	684

Fichiers test	Débit (Mo/s)	Débit (Me/s)	Mc/cat -T	Mc/wc -l	wc -w/Mc
hamlet.txt	-	-	-	-	-
TAU.ip	50.	3.4	4.0	10.	1.4
psd-superfamily.xml	65.	3.4	4.0	10.	1.4
access log cut	60.	1.9	3.3	11.	1.7
psd-author.xml	58.	1.9	3.3	10.	1.6
psd-uid.xml	62.	1.5	3.1	11.	1.7
psd-cut.xml	58.	2.0	3.0	9.2	1.6
3aleatoire.dat	45.	4.5	3.5	7.9	1.2
auck.ip	53.	3.0	3.2	8.6	1.5
100M.txt	39.1	4.6	3.6	6.7	1.35

FIG. 3.11 – Le tableau du haut donne les temps d'exécution en seconde de l'implémentation de MINCOUNT* sur les fichiers du jeu de données. Le tableau du bas donne les débits correspondant en millions d'octets par seconde (Mo/s) et en millions d'éléments par sec (Me/s). Il donne aussi les ratios des timings entre MINCOUNT* et les commandes `unix`.

Nombre de boites	1	4	16	64	256	1024	4096
Err Std théorique (%)	-	34.	16.	7.9	3.9	1.9	1.0
hamlet.txt	48.	34.	16.	7.9	3.9	1.9	1.0
TAU.ip	38.	19.	0.9	0.9	0.6	0.80	0.7
psd-superfamily.xml	38	38	10	7.4	0.8	1.9	1.7
access log cut	26.	13.	14.	12.	6.5	0.9	0.1
psd-author.xml	70.	42.	22.	8.0	0.2	0.3	0.9
psd-uid.xml	22.	52.	1.1	1.	3.8	0.3	0.2
psd-cut.xml	61.0	10.	11.	11.	4.8	2.2	1.0
3aleatoire.dat	30.	12.	26.	3.2	2.3	1.8	1.1
auck.ip	36.	22.	0.9	4.4	2.6	2.7	1.5
100M.dat	64.8	3.8	20.8	5.2	4.3	3.4	2.3

FIG. 3.12 – Erreur standard de l'estimée de MINCOUNT* lors des exécutions chronométrées.

connexions. Pour diverses raisons (planification et conception du réseau, tarification du trafic, informations pour les clients, étude des protocoles de routage, détection de trafic anormal, étude de la composition du trafic (pier to pier ou courriels)), il est crucial pour les opérateurs de ces réseaux de connaître les caractéristiques du trafic dont, en particulier le nombre de connexions actives sur les liens. Cette information sert aussi à détecter des attaques sur les serveurs, comme les attaques par déni de service (DoS attacks) et par port scans, au cours desquelles un nombre anormal de connexions distinctes sont ouvertes au cours d'une période de temps très courte, voir l'étude détaillée d'Estan et Varghese [EVF03].

La plupart des réseaux cœurs (backbone networks) qui existent aujourd'hui sont des réseaux optiques de grande capacité (Synchronous Optical NETworks (SONET)). Leurs liens sont des fibres optiques classifiées selon leur capacité d'OC-1 (51.84 Mbps) en passant OC-192 (10 Gbps) (très largement répandu) à OC-768 (40 Gbps) (dont la mise en service commence). À ces vitesses, le simple fait de stocker les données devient alors problématique, un disque dur de 1 TB étant rempli en moins de 4 minutes. Une étude sur les difficultés pour surveiller des liens à très hautes vitesses peut être trouvée dans [IDGM01].

L'algorithme MINCOUNT permet de faire une analyse très rapide (voir 3.2.2) de tels ensembles de données. On a déjà vu en Section 2.4 une première étude de grandes traces de plusieurs millions de paquets pour valider les estimateurs qui montre que l'algorithme est efficace pour estimer le nombre de connexions actives de très grands multi-ensembles à quelques pourcents près. On peut voir aussi Section 4.4 une analyse du trafic internet d'un routeur passerelle de l'INRIA qui correspond au trafic agrégé de l'activité de 400 machines (Figures 4.7 et 4.8). Cette analyse montre comment utiliser l'algorithme pour faire de la détection d'attaques par déni de service (Denial of Service attack), au cours desquelles les machines infectées essaient d'ouvrir un très grand nombre de connexions vers certaines

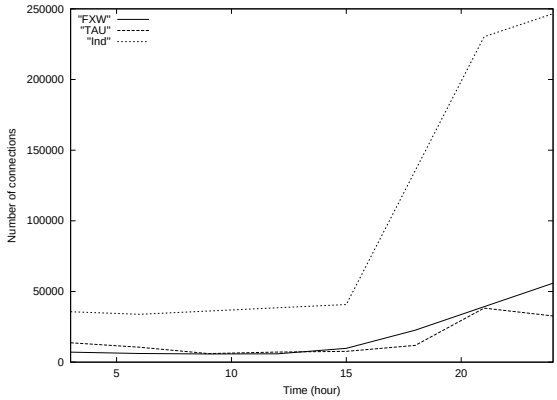


FIG. 3.13 – Pic de connexions du à la propagation du vers Code Rouge

destinations pour en perturber le fonctionnement. Dans la section suivante, on étudie des jeux de trace pour mesurer l'impact de la propagation du ver Code Rouge sur le trafic internet.

3.3.1 Code Rouge attaque

Un groupe d'analyse de réseaux, le NLANR Measurement and Network Analysis Group (`http://moat.nlanr.net`) a réalisé pendant quelques années des mesures quotidiennes de trafic sur certains liens internet. Nous avons analysé leurs traces de la journée du 19 juillet 2001 au cours de laquelle le vers Code Rouge a commencé à se répandre dans le réseau mondial. Ce vers n'avait pas été conçu pour infliger beaucoup de dégâts, mais pour se propager très rapidement. Plus de 359 000 ordinateurs ont été infectés en moins de 14 heures et, au plus fort de la propagation, plus de 2 000 nouvelles machines étaient touchées toutes les minutes. On a considéré trois ensembles de traces : une, référencée par `Ind`, recueillie dans le MegaPOP de l'université de l'Indiana, une autre, `FXW`, au centre FIX West de la NASA à Ames (EU) et une dernière, `TAU`, à l'université de Tel Aviv. Les traces donnent le trafic sur une période de 1 minute 30 toutes les trois heures. Nous avons utilisé l'algorithme MINCOUNT* pour estimer avec une précision de 4 % ($m = 256$) le nombre de connexions actives qui utilisent les trois liens pendant ces durées. Les résultats sont montrés en Figure 3.13. C'est bien sûr une première analyse un peu grossière ; il faudrait des traces plus longues, plus rapprochées et plus nombreuses (non disponibles sur le site internet) pour faire une analyse complète de la propagation du vers. Néanmoins nous sommes capables de détecter le changement d'activité du réseau causé par les machines touchées. On remarque une augmentation très nette du nombre de connexions actives à partir de 15 heures. La charge habituelle du lien `Ind` semble se situer autour de 35 000 connexions (33 800 à 6 heures). Vers minuit, le lien atteint un pic de charge sept fois supérieure de 246 500

connexions. Même observation pour `TAU` et `FXW` qui ont des trafics respectifs de 7,600 and 9,800 connexions à 15 heures et gèrent un maximum à minuit de 32 700 et 55 900 connexions. Ainsi, en utilisant l'algorithme MINCOUNT*, on est capable de détecter des augmentations inhabituelles du trafic qui peuvent être le signe d'une attaque lancées vers certains routeurs. On peut de plus donner des indications sur l'intensité et la propagation de l'attaque dans certaines parties du réseau. Le tout en utilisant une petite mémoire auxiliaire constante.

3.4 Applications bio-informatiques

Mots clés : algorithmes probabilistes, cardinalité, corrélation de l'ADN d'un génome, régions codantes.

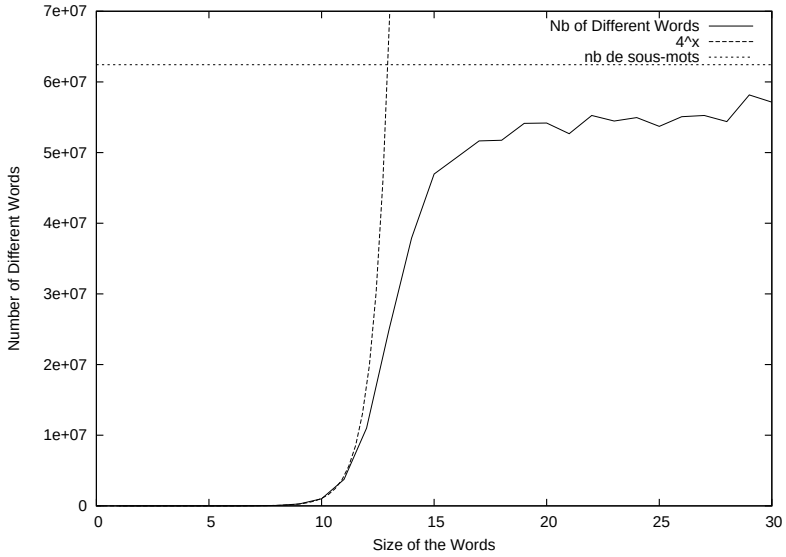


FIG. 3.14 – Nombre de sous-mots différents de taille 1 à 30 dans l'ADN du chromosome 20.

L'*analyse de séquences*, l'une des branches de la bio-informatique, consiste à étudier les *statistiques* de *motifs* présents dans des séquences biologiques comme l'ADN ou l'ARN d'organismes vivants. L'idée ici est d'utiliser les propriétés de vitesse des algorithmes probabilistes de comptage pour faire de l'analyse de séquences sur de très grandes masses de données biologiques. On indique ici quelques directions pour utiliser ces algorithmes pour analyser le génome.

3.4.1 Introduction

Les *algorithmes de comptage pour la cardinalité* permettent d'estimer le nombre de mots *distincts* de *très grands multi-ensembles*. Les meilleurs sont *très rapides* (seulement quelques

dizaines d'instructions machines par élément), précis, et utilisent une *mémoire constante* (erreur relative de $\frac{c}{\sqrt{M}}$ tout en utilisant M unités de mémoire) à comparer avec la mémoire linéaire utilisée par les algorithmes exacts. Ainsi, ils permettent de réaliser de nombreuses expériences en quelques minutes et en utilisant quelques Kilo Octets, qui seraient impossible à faire avec leurs équivalents exacts. Ils sont utiliser typiquement pour des applications dans les domaines des bases de données [CLKB04] ou des réseaux [EVF03] [IDGM01].

On utilise ici ces algorithmes pour analyser la *corrélation des bases* du *génome humain.* La corrélation est mesurée par le nombre de sous-mots distincts de taille fixée k (dix bases, par exemple) présents dans un bout d'ADN de taille N. L'idée est qu'une séquence avec peu de sous-mots est plus corrélée qu'une séquence de même longueur avec plus de sous-mots distincts. On introduit trois angles d'études :

- Tous les mots possibles (4^k sous-mots de taille k) sont-ils présents dans le génome, ou, au contraire, de nombreux motifs sont-ils interdits ?
- Le génome est-il homogène, ou certaines parties sont-elles plus corrélées que d'autres ? Dans ce dernier cas, est-il possible de reconnaître —ou du moins d'avoir des indications— , de façon rapide, des régions de différentes natures, comme des zones de répétitions, des régions codantes ou non codantes ?
- Quel est la fréquence d'apparition de sous-mots distinct dans le génome, quand on le considère comme une séquence lue à partir du "début" ? Comment se distingue-t-elle de celle de textes aléatoires générés par une source de Bernoulli ou de Markov ?

On a obtenu de *premiers résultats* sur un séquençage du génome humain. On a utilisé MINCOUNT* qui estime le nombre d'éléments distincts de textes de plusieurs milliards d'éléments avec une précision de 2% en utilisant une mémoire auxiliaire de seulement $12Ko$. Grâce à sa boucle interne très simple, l'algorithme est de l'ordre de seulement 5 fois plus lent que la commande unix `wc` qui ne compte que le nombre de mots distincts ou non. Ainsi, par exemple, estimer le nombre de sous-mots distincts de taille k parmi les 62 millions (resp. 3 milliards) de sous-mots du chromosome 20 (resp. du génome humain) ne lui prend que 12 secondes (resp. 10 minutes).

3.4.2 Motifs interdits ?

La Figure 3.14 donne les estimées du nombre de sous-mots distincts de taille k variant de 1 à 30 dans l'ADN du chromosome 20 (62 millions de paires de bases) comparées avec le nombre de sous-mots possibles de taille k, 4^k, sur un alphabet de quatre lettres (représentée en pointillés). La courbe se divise en trois phases. La première partie de la courbe suit exactement la la fonction 4^l. Ainsi *tous les sous-mots de taille inférieure ou égale à 11 sont présents dans le chromosome 20.* Ensuite, la courbe s'éloigne de 4^k. Pour des longueurs allant de 12 à 17, la croissance reste exponentielle, mais certains motifs ne sont pas présents. Dans la dernière phase, correspondant aux longueurs de 18 à 30, la croissance est très légère (presque constante) de 54 à 59 millions, à comparer avec les 62 millions de sous-mots du chromosome. La conclusion est qu'aucun motif ne semble interdit. Néanmoins, ils

n'apparaissent pas tous avec la même fréquence.

3.4.3 Régions plus ou moins corrélées

La Figure 3.4 présente une étude multi-échelles de la corrélation des 3 milliards de paires de bases d'un ADN de génome humain. Trois échelles sont introduites. Elles correspondent à trois découpes du génome en 10, 100 ou 1000 pièces de $300M$, $30M$ ou $3M$ de paires de bases. Pour chacune de ces pièces, on donne une estimée du nombre de sous-mots distincts de taille 13. La taille 13 a été choisie en fonction des différentes tailles de pièces ($4^{13} \approx 70$ millions). À chaque échelle, des régions avec différentes corrélations des bases apparaissent clairement.

3.4.4 Fréquence d'apparition des motifs

On étudie la fréquence d'apparition des motifs dans l'ADN du chromosome 20. Pour des tailles 6 (haut gauche), 8 (haut droite), 10 (bas gauche) et 12 (bas droite), la Figure 3.4 donne le nombre de sous-mots distincts vus après avoir lu x bases, x allant de 1 au chromosome entier. Ces nombres sont comparés à l'espérance théorique pour un texte aléatoire de même taille généré par une source de Bernoulli. Elle peut être approximée (voir [RR00]) par $\mathbb{E}[X^{(n)}] \approx \sigma^q \left(1 - \frac{1}{\sigma^q}\right)^{n-q+1} \approx \sigma^q \exp(-\lambda)$. Les simulations montrent que la fréquence d'arrivée des motifs est plus petite dans l'ADN, indiquant que une corrélation de l'ADN, et donnent une mesure de cette différence.
Du travail est en cours pour chacune des approches. Il faudrait par exemple comparer les zones qui ont des corrélations spécifiques avec certains emplacements connus de régions codantes ou comparer les nombres de sous-mots distincts avec ceux générés par des sources markoviennes.

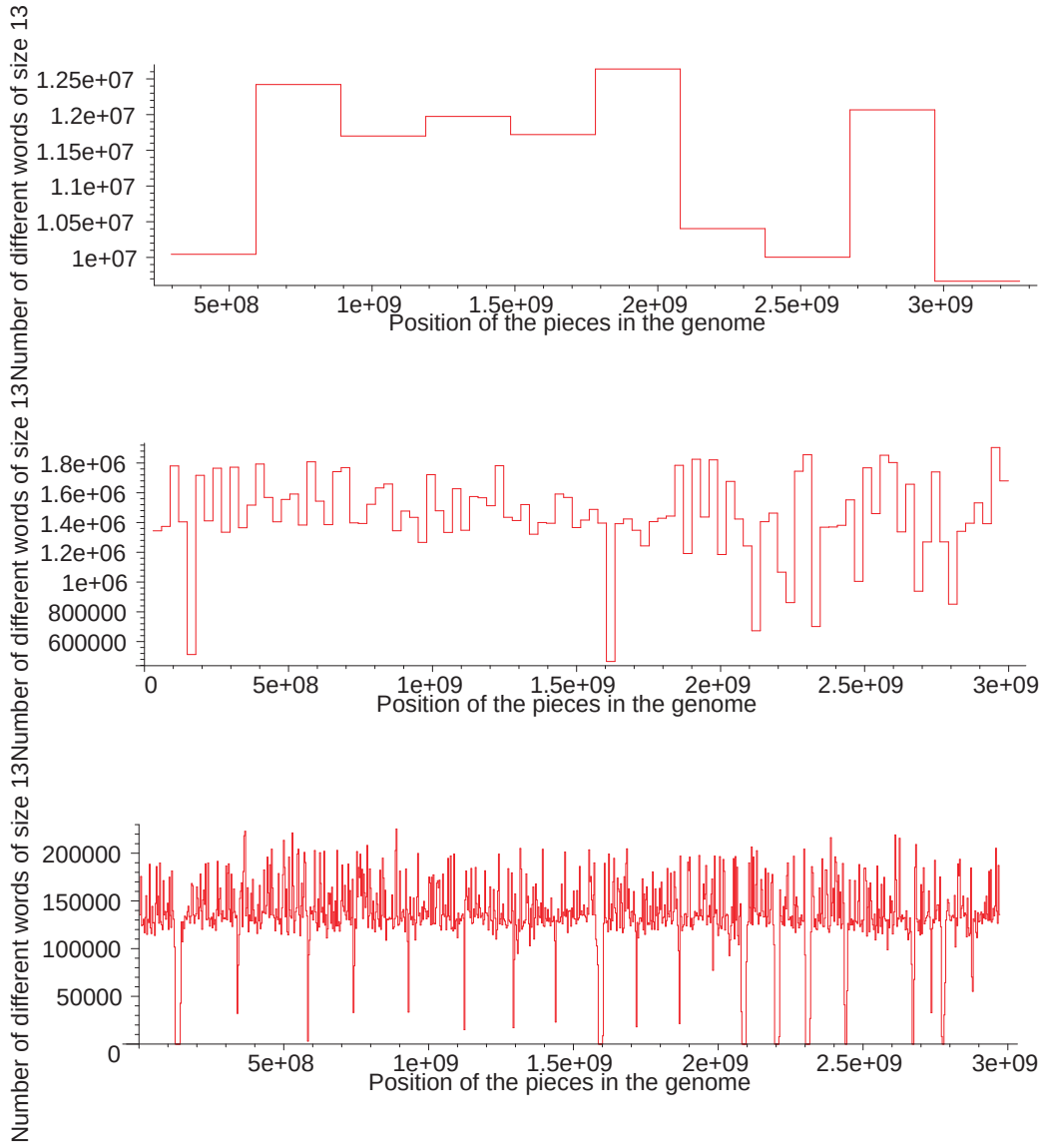


FIG. 3.15 – Mots différents de taille 13 dans des séquences consécutives du génome de $3M$, $30M$ et $300M$ de paires de bases.

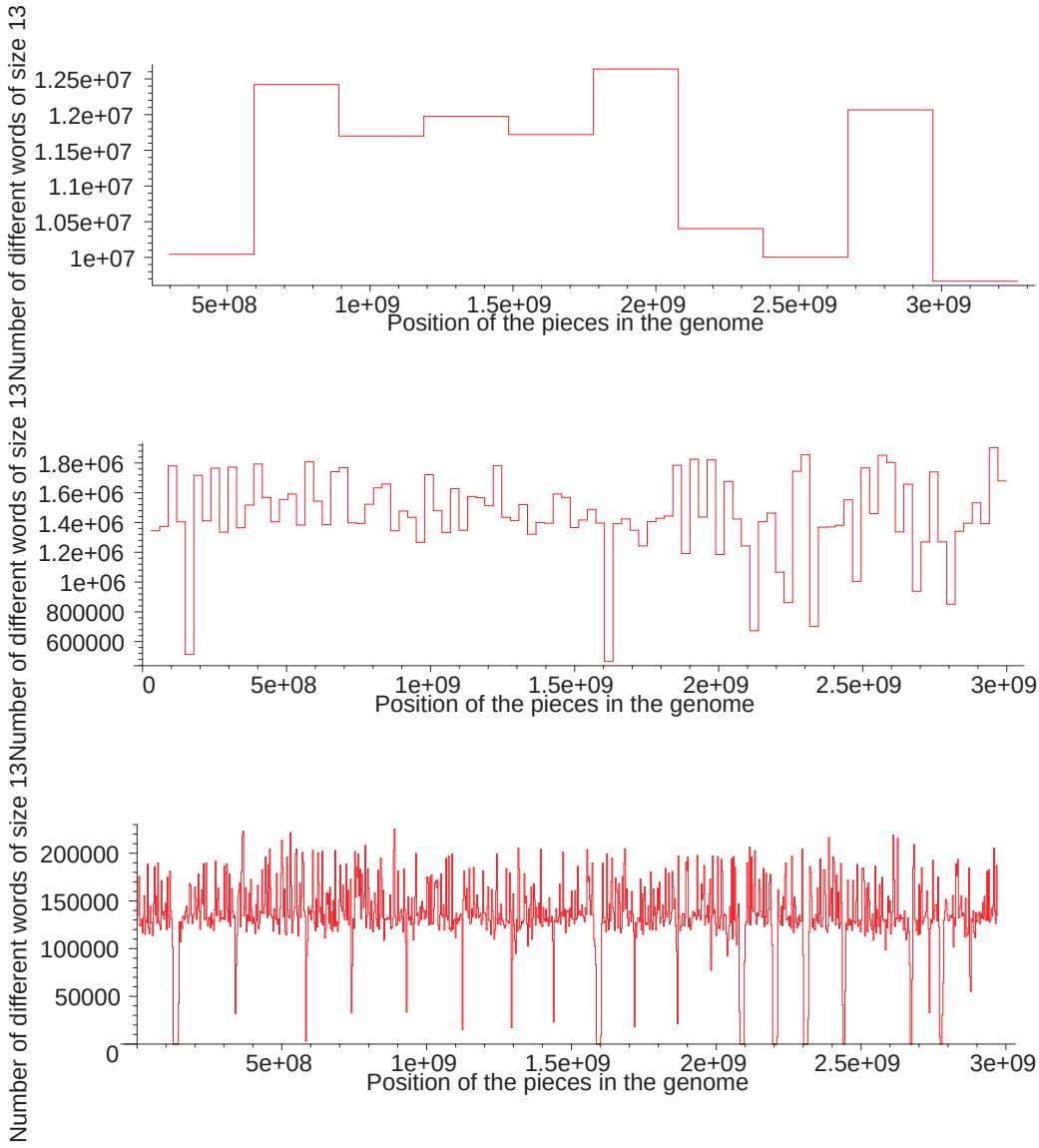


FIG. 3.16 – Mots différents de taille 13 dans des séquences consécutives du génome de $3M$, $30M$ et $300M$ de paires de bases.

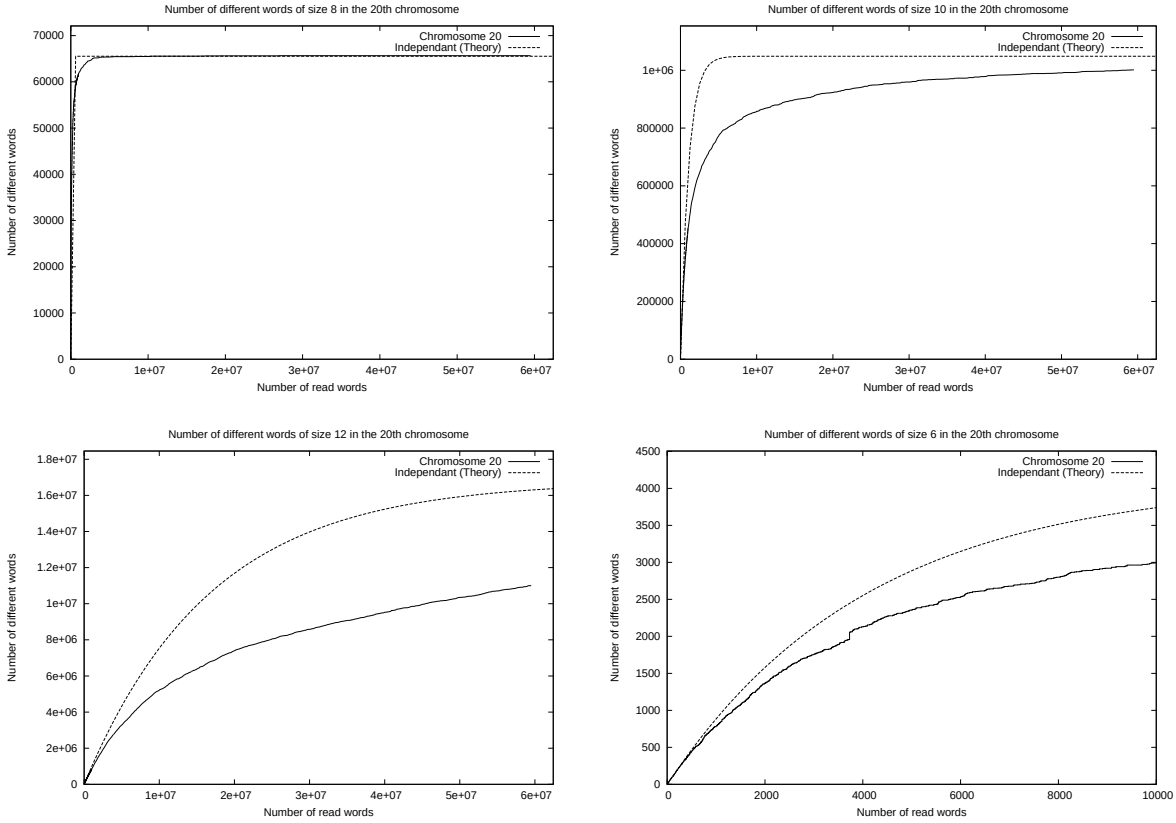


FIG. 3.17 – Apparition des motifs de tailles 6, 8, 10, 12 dans l'ADN du chromosome 20

Chapitre 4

Estimer le nombre de flots actifs dans un flux de données sur une fenêtre coulissante

Ce travail a été réalisé en collaboration avec Éric Fusy du Projet ALGO de l'INRIA Rocquencourt.

mots-clés : *flots actifs, fenêtre coulissante, flux de données, traces du trafic internet.*

Un nouvel algorithme est introduit pour estimer le nombre de flots (ou connections) distincts dans un flux de données. L'algorithme maintient une estimée précise du nombre de flots distincts sur une fenêtre coulissante. Il est facile à implémenter, se parallélise efficacement, et a un très bon compromis entre la mémoire auxiliaire et la précision de l'estimée : une précision relative de l'ordre de $1/\sqrt{m}$ nécessite en gros une mémoire d'ordre $m \ln(n/m)$ mots, où n est une borne supérieure du nombre de flots vus sur la fenêtre coulissante. Par exemple, une mémoire de seulement $64kB$ est suffisante pour maintenir une estimée d'une précision de l'ordre de 4 pourcents pour un flux de plusieurs millions de flots. L'algorithme a été validé par simulations et par expérimentations sur du trafic réel. Il s'avère très efficace pour observer le trafic et détecter des attaques.

Introduction

Dans la dernière décennie, le problème de compter les éléments distincts d'un multi-ensemble est apparu être une opération cruciale dans le contexte des *flux de données (data streams)*. Les éléments d'un flux de données diffèrent de ceux d'une base de donnée en ceci

qu'ils ont un *timestamp* qui indique leur date d'arrivée. Typiquement les éléments sont des *paquets*, chaque paquet appartenant à un *flot (flow)* —aussi appelé connexion— identifié par l'adresse de la source et l'adresse de la destination. Dans ce contexte les paquets sont les éléments du multi-ensemble et les flots en sont les éléments distincts. Estimer le nombre de flots distincts d'un flux de données a de nombreuses applications dans l'étude et la surveillance des réseaux. En effet, c'est tout d'abord une première information statistique sur le trafic simple mais pertinente qui indique si le trafic a beaucoup plus de paquets que de flots —typique quand de grands fichiers sont téléchargés— ou si les nombres de flots et de paquets sont comparables —typique pour un trafic constitué en majorité de courriels. Mais cette information sert aussi à détecter des attaques sur les serveurs, comme les attaques par déni de service (DoS attacks) et par port scans, au cours desquelles un nombre anormal de connexions distinctes sont ouvertes au cours d'une période de temps très courte, voir l'étude détaillée d'Estan et Varghese [EVF03]. À titre d'exemple, le système FlowScan [Plo00] compte le nombre de flots distincts sur certains trafics (en utilisant un algorithme non optimisé) pour détecter de telles attaques. Dans le contexte des flux de données, le multi-ensemble à traiter n'est pas fixe (contrairement au cas des applications pour les bases de données) mais évolue au cours du temps en ayant un nouvel élément à chaque arrivée d'un paquet et en perdant les éléments qui sont périmés. Pour cette raison, le problème doit être résolu pour une *fenêtre coulissante* de taille W, W étant l'intervalle de temps au bout duquel un paquet est périmé. Précisément, on doit pouvoir être capable de répondre, à chaque instant t, à une requête de la forme "Quel est le nombre approché de flots distincts du trafic sur les dernières w unités de temps ($w \leq W$) ?". Remarquez que les algorithmes "statiques" précédents ([FM83, DF03, EVF03, Gir05]) ne peuvent répondre à de telles requêtes : ils devraient être lancés au temps $t-w$ jusqu'au temps t, ce qui voudrait dire qu'il faudrait savoir *à l'avance* qu'il y aura un requête w unités de temps plus tard.

Dans cet article, on développe un algorithme efficace, appelé SLIDING MINCOUNT, pour résoudre le problème d'estimer la cardinalité d'un multi-ensemble sur une fenêtre coulissante. Notre algorithme est construit à partir de l'algorithme "statistique", MINCOUNT, où l'estimée est basé sur la valeur hachée minimale des éléments. L'idée est de maintenir une liste de valeurs qui pourraient devenir le minimum d'une fenêtre dans le futur. En particulier, le minimum courant sur $[t - W, t]$ est toujours dans la liste au temps t, de façon à ce que le minimum puisse être calculé *a posteriori*. La liste, appelée LFPM (pour Liste des Minimums Possibles Futurs), est très simple à maintenir et est étonnamment courte : l'analyse précise de l'exécution complète de l'algorithme (Théorème 9 de la Section 4.3.1) montre que sa taille est logarithmique en le nombre maximal de flots distincts vu sur la fenêtre de temps. Mentionnons qu'il est aussi possible d'adapter les algorithmes probabilistes basés sur les motifs de bits (bitmap patterns) [DF03, FM83] aux fenêtre coulissantes. Le procédé est expliqué rapidement dans l'article de Datar et al [DGIM02]. Ils obtiennent le même ordre de complexité que notre algorithme, bien qu'ils ne donnent pas d'analyse détaillée. Par contre, nous donnons une analyse poussée de la mémoire requise. Cela inclut l'analyse de la mémoire maximale nécessaire sur une très longue période de temps, une

question importante si l'on veut être sûr qu'il n'y ait pas de dépassement des capacités de la mémoire.

En plus, nous fournissons des validations expérimentales variées de notre algorithme. Des simulations avec du trafic idéal montrent un bon accord entre le comportement de l'algorithme et ce qui est prévu par l'analyse. En n'utilisant seulement que $64kB$ de mémoire, Sliding Mincount réussit à estimer le nombre de flots distincts d'un flux de données de 5 millions d'éléments pour une fenêtre coulissante d'un million d'éléments avec de l'ordre de 4%. Des observations sur du trafic réel révèlent que Sliding Mincount peut surveiller sans problème un trafic d'un débit de plusieurs millions de paquets par seconde en temps réel. Sur les traces d'un jour entier (contenant 150 millions de paquets), l'algorithme détecte des pics du nombre de connexions qui sont invisibles en regardant juste le nombre de paquets. C'est d'un intérêt primordial pour détecter les attaques par déni de service qui génèrent beaucoup de connexions pour peu de paquets. Une détection automatique du trafic inhabituel peut facilement être mise en place en déclenchant une alarme quand l'estimée de Sliding Mincount dépasse, par exemple, un certain seuil.

4.1 Préliminaires

Dans cette section, on rappelle rapidement les principes de l'algorithme Mincount, présenté dans les chapitres précédents, qui estime la cardinalité d'un multi-ensemble fixé. Comme notre algorithme Sliding Mincount —présenté en Section 4.2— opère sur les flux de données (data streams), nous utiliserons dans la suite les terminologies de paquets (éléments du multi-ensemble) et de flots (éléments distincts du multi-ensemble), qui sont définis ci-dessous.

4.1.1 Définitions

Un paquet consiste en un *en-tête (header)*, contenant des champs (d'en-tête), et un *corps (body)*, contenant les informations transportées. Un *flot (flow)* est identifié par un ensemble de champs d'en-tête. Un tel identifiant, appelé concisément flowID, est typiquement une paire <source IP, destination IP>, mais peut être aussi de la forme <source IP et port, destination IP et port> si on prend en compte les informations sur les ports (par exemple, pour détecter les attaques par port scans). Deux paquets sont dit appartenir à un même flot s'ils ont le même FlowID.

Pour la suite, on supposera avoir à notre disposition une *fonction de hachage* h qui envoie un FlowID sur une valeur réelle qui "a l'air" distribuée uniformément sur $0, 1$. Remarquez que les l premiers bits du développement binaire d'une telle valeur réelle forment une séquence aléatoire uniforme de l bits indépendants. Ainsi, ce qu'on utilise en pratique est une "bonne" fonction de hachage (c'est-à-dire une fonction approchant l'aléatoire parfait) envoyant un FlowID sur une séquence de l bits qui semble être une séquence aléatoire uniforme de bits et telle que la probabilité d'une collision est négligeable. Cela signifie que

2^l doit être bien plus grand que le nombre de flots distincts à estimer. Pour la plupart des applications, le nombre de flots actifs sur une fenêtre est de l'ordre de quelques millions et il est suffisant de hacher vers des mots de 32 bits. L'hypothèse d'existence de bonnes fonctions de hachage est justifiée en pratique (voir l'étude de la Section 3.2.1).

Soit $S = (p_1, \ldots, p_N)$ un ensemble de paquets et soit n le nombre de flots distincts dans S. Sous les hypothèses sur la fonction de hachage et en ne faisant aucune hypothèse sur la nature du trafic, on peut considérer l'ensemble $(h(p_1), \ldots, h(p_N))$ des valeurs hachées comme construit à partir de n valeurs réelles prise uniformément au hasard dans $[0, 1]$, puis répliquées et permutées arbitrairement. On appelle *multi-ensemble idéal* un tel multi-ensemble. Ainsi,

Estimer le nombre de flots distincts dans un ensemble de paquets sans faire aucune hypothèse sur le trafic revient à estimer la cardinalité d'un multi-ensemble idéal.

4.1.2 L'algorithme MINCOUNT

Nous rappelons ici les principes de l'algorithme probabiliste introduit dans les chapitres précédents, MINCOUNT, qui estime le nombre de valeurs distinctes d'un multi-ensemble idéal avec un bon compromis entre la mémoire et la précision. L'idée est que le minimum des valeur d'un multi-ensemble idéal ne dépend pas de la structure des répétitions des données ni de leur ordre d'apparition et qu'il donne une indication du nombre n des valeurs distinctes du multi-ensemble (intuitivement, le minimum de n réels choisis uniformément et indépendamment dans $[0, 1]$ a de bonnes chances d'être petit si n est grand). Le principe est donc de construire une observable basée sur le minimum et de la combiner avec un *processus de moyennage stochastique*, comme introduit dans [FM83], de façon à obtenir une estimée précise de n. Le moyennage stochastique consiste à simuler l'effet de $m = 2^b$ expériences sur le multi-ensemble et à prendre une moyenne des estimées sur les m expériences. Les b premiers bits d'une valeur x du multi-ensemble sont utilisés pour dirigé x vers une des m *boites* (celle dont l'indice i, écrit en base binaire, correspond aux b bits), et la valeur dirigée dans la boite est la valeur originale x tronquée de ces b premiers bits (ainsi cette valeur est aussi uniforme dans $[0, 1]$). L'algorithme construit alors une estimée précise du nombre de valeurs distinctes à partir des minimums des boites —voir la Figure 4.1 pour un résumé en pseudo-code.

Théorème 7 (Giroire [Gir05]) *Soit $\mathcal{I}$ un multi-ensemble idéal et soit n sa cardinalité. Soit $m = 2^b$ le nombre de boites utilisées par l'estimée. Pour $1 \leq j \leq m$, soit $M^{(j)}$ le minimum des valeurs dirigée vers la boite d'indice j. Alors, si on note Γ la fonction Gamma d'Euler, la quantité*

$$\xi := m\Gamma(2 - 1/m)^{-m} \exp\left(\frac{1}{m}(\ln(M^{(1)}) + \ldots + \ln(M^{(m)})\right) \tag{4.1}$$

peut être calculée en utilisant une mémoire de m mots et est une estimée de la cardinalité de $\mathcal{I}$ dans le sens où :

Algorithm MinCount (with $m = 2^b$ buckets)
Input: $(p_1, \ldots, p_N)$ a set of packets;
Initialize $M^{(1)}, \ldots, M^{(m)}$ to 0;
for i from 1 to N
  $u_i \leftarrow h(p_i)$; [hash p_i to a real value in $[0, 1]$]
  $j \leftarrow$ integer corresponding to the first b bits of u_i;
  $\widetilde{u}_i \leftarrow 2^b u_i - \lfloor 2^b u_i \rfloor$; [$\widetilde{u}_i$ is u_i truncated of its first b bits]
  $M^{(j)} \leftarrow \min(M^{(j)}, \widetilde{u}_i)$;
return $\xi := m\Gamma(2 - 1/m)^{-m} \exp\left(\frac{1}{m} \sum_{j=1}^{m} \ln(M^{(j)})\right)$
  as estimate of the number of distinct flows

FIG. 4.1 – Pseudo-code de l'algorithme MinCount.

- *Il est asymptotiquement non biaisé, c'est-à-dire que* $\mathbb{E}(\xi) \underset{n\to\infty}{\sim} n$.
- *Sa précision relative, définie comme* $\sqrt{\mathbb{V}(\xi)}/\mathbb{E}(\xi)$, *est d'environ* $1.3/\sqrt{m}$.

4.2 Présentation de l'algorithme Sliding Mincount

4.2.1 Introduction

Nous avons vu dans la section précédente que le nombre de flots distincts dans un ensemble fixé de paquets peut être estimé précisément avec une petite mémoire auxiliaire en utilisant une estimée construite à partir des *minimums* des valeurs hachées de m boites. Dans cette section nous nous inspirons de cette idée pour estimer le nombre de flots distincts dans une *fenêtre coulissante*. Ce que nous considérons ici n'est plus un ensemble fixe de données (data set) comme en Section 4.1.2, mais un *flux de données (data stream)*. Un flux de données est une séquence infinie de paquets. De plus, chaque paquet a un *timestamp* indiquant sa date d'arrivée (les timestamps sont en particulier donnés dans les traces de routeurs). Ainsi, en utilisant une fonction de hachage h comme en Section 4.1.2, un flux de données peut être formalisé comme une séquence infinie $< t_k, H_k >_{k\geq 1}$, où t_k est le temps d'arrivée et H_k est la valeur hachée du $k-$ième paquet arrivé (ainsi t_k est croissant). Nous identifions le $k-$ième paquet avec le couple $< t_k, H_k >$. Pour maintenir une estimée sur une fenêtre coulissante, nous utilisons aussi un processus de moyennage stochastique : les b premiers bits de H_k sont utilisés pour diriger le $k-$ième paquet dans une des $m = 2^b$ boites, et le paquet dirigé vers la boite devient $< t_k, H'_k >$ où H'_k est dérivé de H_k en tronquant ses b premiers bits. Ce processus partitionne le flux de données $< t_k, H_k >$ en m flux de données, un pour chaque boite. Si nous sommes capable de maintenir le minimum dans chaque boite, nous serons aussi capable de calculer l'estimée du Théorème 7 sur une fenêtre coulissante. En conséquence, le problème principal à résoudre est de maintenir le minimum d'un flux de nombres réels sur une fenêtre coulissante.

4.2.2 Maintenir le minimum sur une fenêtre coulissante

Le problème de maintenir le minimum des valeurs d'un flux de données $< t_k, H_k >$ sur une fenêtre coulissante peut être énoncé de la façon suivante : étant donné un paramètre W de fenêtre maximale, trouver un processus tel que le minimum des valeurs du flux sur la fenêtre $[t-w, t]$ peut être trouvé à tout temps t et pour n'importe quel $w \leq W$.

La solution que nous proposons est de maintenir une liste de paquets qui *peuvent devenir un minimum* dans une fenêtre future. Après avoir terminé une première version de ce travail, nous avons appris qu'une solution similaire est brièvement indiquée dans un article de Datar *et al* [DGIM02], mais sans aucune analyse. Nous donnons présentation plus détaillée du procédé, avec une terminologie spécifique, et nous analysons la mémoire nécessaire en Section 4.3.

Au temps t, nous considérons l'ensemble de paquets du flux qui sont dans la fenêtre $[t-w, t]$. Nous définissons le *paquet minimum* de la fenêtre comme le dernier paquet arrivé parmi ceux dont la valeur hachée réalise le minimum sur $[t-w, t]$. Un *minimum futur possible* —abrégé en FPM pour Future Possible Minimum— de la fenêtre $[t-w, t]$ est un paquet qui *pourrait* devenir le paquet minimum d'une fenêtre future, si on ne fait aucune hypothèse sur le trafic futur. La liste qui contient tous les minimums futurs possibles de la fenêtre $[t-w, t]$ est appelée LFPM —pour Liste de FPM. En particulier, le paquet minimum de la fenêtre $[t-w, t]$ est dans sa LFPM. Observez que pour deux paquets de $[t-w, t]$, $< t_k, H_k >$ et $< t_{k'}, H_{k'} >$, le paquet $< t_k, H_k >$ ne peut devenir un paquet minimum dans le futur si $t_k < t_{k'}$ et $H_k \geq H_{k'}$. En effet, dans le futur, toute fenêtre contenant $< t_k, H_k >$ contiendra aussi $< t_{k'}, H_{k'} >$ et ainsi $< t_{k'}, H_{k'} >$ empêche $< t_k, H_k >$ de devenir un paquet minimum. Ainsi un FPM $< t_k, H_k >$ doit être tel qu'aucun paquet $< t_{k'}, H_{k'} >$ arrivé après lui ne vérifie $H_{k'} \leq H_k$; c'est-à-dire qu'un FPM *doit être* un record minimum strict (strict minimum record) de la liste de paquets de la fenêtre prise dans le sens inverse du sens chronologique. Réciproquement, si un paquet $< t_k, H_k >$ est un tel record minimum en sens chronologique inverse sur $[t-w, t]$ et si aucun paquet n'arrive plus après le temps t, alors le paquet $< t_k, H_k >$ est le paquet minimum au temps $t_k + W$. En conséquence, on obtient la caractérisation suivante pour la LFPM, voir Figure 4.2 :

Fait 1 *La LFPM contient les records minimums stricts de la liste de paquets sur* $[t-W, t]$ *pris dans le sens opposé du sens chronologique.*

En particulier, les éléments de la LFPM sont strictement croissants de la gauche vers la droite, voir Figure 4.2.

Fait 2 *Si la LFPM peut être maintenue pour une fenêtre coulissante de longueur* W*, alors à toute date* t *et pour n'importe quel* $w \leq W$*, le paquet minimum de la fenêtre* $[t-w, t]$ *peut être extrait de la LFPM : c'est le paquet le plus ancien contenu dans la LPFM dont le timestamp est plus grand que* $t-w$*.*

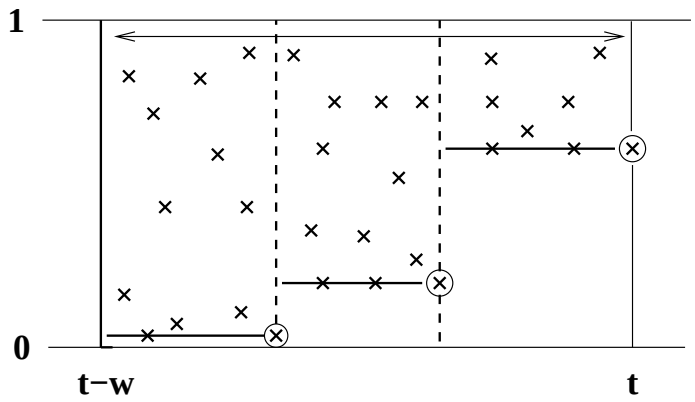


FIG. 4.2 – LFPM et records minimaux.

Le Fait 1 et le Fait 2 rendent possible d'obtenir une boucle interne très simple pour maintenir la LFPM sur une fenêtre coulissante de longueur W. La boucle interne est donnée en Figure 4.3 : quand un paquet arrive, les paquets trop anciens sont effacés à l'Étape 2, et les paquets qui ne sont plus des FPM (en raison de l'arrivée du $n-$ième paquet) sont effacés à l'étape 3. En conséquence de quoi, on voit facilement que l'invariant suivant est conservé à chaque arrivée de paquets :

Fait 3 *La liste maintenue par la boucle interne de la Figure 4.3 est la LFPM.*

Enfin, les Faits 2 et 3 assurent que notre boucle interne donne une solution au problème de maintenir le minimum sur une fenêtre coulissante, formulé au début de la Section 4.2.2. De plus, notre solution est optimale dans le sens où, pour maintenir le minimum sans faire d'hypothèse sur le trafic, on doit avoir tous les FPM de $[t-W, t]$ en mémoire à tout moment t. En effet, si un des FPM $< H_k, t_k >$ manque au temps t et si le trafic est vide après ce temps t, alors $< H_k, t_k >$ sera l'unique paquet réalisant le minimum un petit peu avant le temps $t_k + W$.

Le très joli résultat que nous prouvons en Section 4.3 est que la taille de la liste est étonnamment courte : à n'importe quel temps t, elle est en ordre logarithmique du nombre de valeurs distinctes sur $[t - W, t]$. En plus, la longueur de la LFPM ne fluctue presque pas sur de très longues période de temps.

4.2.3 L'algorithme SLIDING MINCOUNT

L'algorithme SLIDING MINCOUNT, donné en Figure 4.4, combine simplement la méthode pour maintenir le minimum d'un flux de données sur une fenêtre coulissante avec le procédé de moyennage stochastique utilisé par MINCOUNT. $m = 2^b$ boites sont utilisées et on divise le flux de données en m flux, en fonction des b premiers bits de la valeur hachée du paquet.

while(true) do
1) Read packet (H_k, t_k).
2) Delete the packets (H_i, t_i) of L with $t_i < t_k - W$.
3) Delete the packets (H_i, t_i) of L with $H_i \geq H_k$.
4) Add (H_k, t_k) at the end of L.
end do

FIG. 4.3 – La boucle qui met à jour la LFPM.

INTERNAL LOOP:
$L_1 \leftarrow \emptyset \dots L_{2^b} \leftarrow \emptyset$ $[m = 2^b]$
while(true) do
1) Read packet (H_k, t_k).
2) $j \leftarrow$ first b bits of H_k
3) Truncate H_k of its first b bits
4) Delete the packets (H_i, t_i) of L_j with $t_i < t_k - W$.
5) Delete the packets (H_i, t_i) of L_j with $H_i \geq H_k$.
6) Add (H_k, t_k) at the end of L_j.
end do

REQUEST:
Input: "what is approximately the number of distinct flows in the last w units of time ?"
Answer:
for j from 1 to m do
$M^{(j)} \leftarrow$ leftmost hashed value of L_j with timestamp larger than $t - w$
end for
$\xi \leftarrow m\Gamma(2 - \frac{1}{m})^{-m} \exp\left(\frac{1}{m}\sum_{j=1}^{m} \ln(M^{(j)})\right)$
return ξ

FIG. 4.4 – L'algorithme SLIDING MINCOUNT

Pour chaque boite, on garde en mémoire le minimum sur la fenêtre coulissante de taille W en maintenant la LFPM associée à la boite, grâce à la boucle interne de la Figure 4.3. Ensuite, quand il y a une requête au temps t demandant une estimée du nombre de flots distincts dans les dernières w unités de temps (avec $w \leq W$), il suffit simplement d'extraire le minimum des LFPM sur $[t-w, t]$ pour chaque boite. Ceci est fait en suivant la méthode indiquée dans le Fait 2, à savoir en prenant le plus ancien paquet dont le timestamp est plus grand que $t - w$ dans la LFPM de chaque boite. À partir de ces minimums, on calcule la même estimée que celle de MINCOUNT. Ainsi, l'estimée retournée est *exactement la même* que celle qui aurait été obtenue en lançant *a priori* l'algorithme MINCOUNT de $t - w$ à t. Ainsi il a exactement la même précision, comme résumé dans le Théorème 8.

Théorème 8 (Précision de SLIDING MINCOUNT) *Pour n'importe quelle requête demandant le nombre de flots distincts sur les w unités de temps (avec $w \leq W$), l'estimée ξ*

renvoyée par Sliding Mincount *a une précision relative* $SE(\xi)/E(\xi)$ *d'environ*

$$\approx \frac{1.3}{\sqrt{m}}.$$

Par exemple, prendre $m = 2^{10}$ donne des estimées avec une précision d'environ 4 pourcents.

4.2.4 Distributivité

Une caractéristique de Sliding Mincount (comme des algorithmes "statiques" précédents [FM83, DF03, Gir05]) est d'être très bien adapté à la parallélisation. En effet, si le flux de données à mesurer est réparti entre plusieurs routeurs, chaque routeur maintient ses propres LFPM à partir des paquets qu'il reçoit. Ensuite, pour répondre à une requête sur le nombre de flots distincts du trafic total, chaque routeur extrait les valeurs hachées minimales de ses propres LFPM et les envoie à un calculateur central qui calcule l'estimée. Une telle situation est rencontrée très souvent en pratique. Par exemple, dans les réseaux backbone, la quantité de trafic est beaucoup trop importante pour être concentrée sur une seule machine et, ainsi, les paquets arrivant à un point de présence (PoP) sont distribués entre plusieurs routeurs.

4.2.5 Compter le nombre de paquets sur une fenêtre coulissante

Donner le nombre de paquets dans un ensemble donné est direct en utilisant un simple compteur. Cependant, la question devient plus compliquée sur une fenêtre coulissante, car il faut garder en mémoire le temps d'arrivée des paquets. Avec un *léger changement* de Sliding Mincount, il est possible d'estimer le nombre de paquets d'un flux de données sur une fenêtre coulissante : au lieu de hacher les identificateurs des flots —les adresses IP de sources et destinations—, il suffit de hacher les timestamps des paquets. Comme chaque timestamp est unique, l'algorithme ainsi modifié estime maintenant le nombre de paquets de la fenêtre coulissante.

4.3 Analyse de l' algorithme

Comme la précision de Sliding Mincount est la même que celle de Mincount, le problème principal reste d'analyser sa mémoire. La mémoire utilisée par Sliding Mincount est constituée des m LFPM (une par boite). Nous analysons d'abord en Section 4.3.1 la distribution de probabilité de la mémoire à un temps t fixé. Un second question importante, analysée en Section 4.3.2, est l'évolution de la taille des LFPM au cours du temps. En particulier, il apparaît pratique pour l'implémentation d'utiliser des tableaux pour stocker les LFPM en lieu et place de listes. Pour décider quelle taille de tableaux est suffisante pour

éviter un dépassement des capacités de la mémoire, il est crucial d'avoir une idée de la distribution du maximum des tailles des LFPM sur un longue période de temps (par exemple une semaine ou même une année).

4.3.1 Étude de la mémoire requise à un temps t fixé

Nous commençons l'analyse par étudier la taille d'une seule LFPM.

Lemme 4 (Les répétitions n'ont pas d'impact sur la LFPM) *Étant donnée une fenêtre $[t-W,t]$, la LFPM est identique à celle construite en appliquant la boucle interne de la Figure 4.3 sur les seuls paquets arrivés en dernier de chaque flot.*

Preuve Tous les paquets d'un flot sont hachés sur la même valeur. En conséquence, un paquet qui n'est pas le dernier de sa connexion ne peut pas être un record minimum strict pour la liste des paquets considérée en ordre inverse de l'ordre chronologique. □

Lemme 5 (Équivalence avec les permutations aléatoires) *Pour un temps t fixé, soit n le nombre de flots distincts sur la fenêtre courante $[t-W,t]$. Alors, la distribution de probabilité de la taille de la LFPM est la même que la distribution du nombre de records minimum dans une permutation aléatoire de taille n.*

Preuve Le fait que la LFPM est la même que celle construite avec n variables aléatoires i.i.d. uniformes sur $[0,1]$ provient du Corollaire 4 et de l'hypothèse de hasard parfait pour la fonction de hachage. □

Lemme 6 (Taille d'une seule LFPM) *Soit L_n la variable aléatoire donnant la taille d'une seule LFPM au temps t, sachant que le nombre de flots distincts sur $[t-W,t]$ est égal à n. Alors la distribution de L_n satisfait*

$$\mathbb{E}[L_n] = H_n, \text{ avec } H_n \text{ le nombre harmonique, } H_n = \sum_{k=1}^{n} \frac{1}{k} \underset{n\to\infty}{\sim} \ln(n). \tag{4.2}$$

$$\mathbb{V}[L_n] = H_n - \sum_{k=1}^{n} \frac{1}{k^2} \underset{n\to\infty}{\sim} \ln(n), \tag{4.3}$$

Asymptotiquement en n, L_n est une loi gaussienne d'espérance $\ln n$ et de variance $\ln n$, c'est-à-dire

$$L_n \underset{n\to\infty}{=} \ln(n) + \sqrt{\ln(n)}Z, \quad \text{où } Z \in \mathcal{N}(0,1). \tag{4.4}$$

Preuve Le Lemme 5 assures que la distribution de L_n est la même que la distribution du nombre X_n de records minimums dans une permutation aléatoire de taille n. L'analyse de X_n est une problème classique en combinatoire, étudié en premier par Goncharov, voir [FS, II.6,IX.5] : une solution concise est obtenue à partir de la forme close de la fonction caractéristique $f(u) = \sum_k \mathbb{P}(X_n = k)u^k$ de X_n, $f(u) = \frac{1}{n!}u(u+1)\dots(u+n-1)$. L'espérance et la variance sont ensuite extraites en utilisant $\mathbb{E}(X_n) = f'(1)$ et $\mathbb{V}(X_n) = f''(1)+f'(1)-f'(1)^2$. La distribution asymptotique gaussienne est obtenue en normalisant X_n autour de sa moyenne par un facteur contractant de $\sqrt{\mathbb{V}(X_n)}$ et en prouvant que la fonction caractéristique normalisée converge vers un loi gaussienne standard. □

À partir de l'étude de la taille d'une seule LFPM, nous pouvons analyser la mémoire nécessaire à Sliding Mincount qui est la somme des tailles des LFPM des m boites.

Lemme 7 (Taille totale des m LFPM) *Soit L_n^{tot} la variable aléatoire donnant, au temps t, les sommes des tailles des m LFPM (une pour chaque boite), sachant que le nombre de flots distincts sur $[t-W,t]$ est égal à n. Alors la distribution de L_n^{tot} satisfait*

$$\mathbb{E}[L_n^{\text{tot}}] \underset{n\to\infty}{\sim} mH_{\lfloor n/m\rfloor} \underset{n\to\infty}{\sim} m\ln(n/m), \tag{4.5}$$

$$\mathbb{V}[L_n^{\text{tot}}] \underset{n\to\infty}{\sim} m\ln(n/m), \tag{4.6}$$

Asymptotiquement en n, L_n^{tot} est une loi gaussienne d'espérance $m\ln(n/m)$ et de variance $m\ln(n/m)$, c'est-à-dire,

$$L_n^{\text{tot}} \underset{n\to\infty}{=} m\ln(n/m) + \sqrt{m\ln(n/m)}Z, \quad \text{où } Z \in \mathcal{N}(0,1). \tag{4.7}$$

Preuve Asymptotiquement en n, le nombre de flots tombant dans chacune des m boites est égal à n/m, à des fluctuations d'ordre $\sqrt{n/m}$ près. Ainsi, les calculs sont encore vrais (et beaucoup plus simples) si l'on énonce que, au premier ordre, le nombre de flots distincts tombant dans chaque boite est égal à $\lfloor n/m\rfloor$. Dans ce modèle de calcul d'équirépartition, L_n^{tot} est la somme de m variables aléatoires i.i.d. qui sont chacune la taille d'une LFPM construite avec $\lfloor n/m\rfloor$ valeurs distinctes. Selon le Lemme 6, la distribution de la taille d'une telle LFPM est, asymptotiquement en n, égale à $\ln(n/m) + \sqrt{\ln(n/m)}Z$, où Z est une loi normale gaussienne. Ainsi la distribution de L^{tot} est, asymptotiquement en n, égale à $m\ln(n/m) + \sqrt{\ln(n/m)}\sum_{i=1}^{m} Z_i$, où les Z_i sont m lois normales gaussienne i.i.d.. Enfin, il est connu que la somme de m lois normales gaussiennes i.i.d. a la même distribution que $\sqrt{m}Z$, où Z est une loi normale gaussienne. □

Proposition 4 (Mémoire à un temps t fixé) *Soit $n_{\max}$ le nombre maximal de flots distincts qui peuvent être observé sur une fenêtre de longueur W. Alors, au temps t, la mémoire utilisée par* Sliding Mincount *est de l'ordre de $m\ln(n/m)\,(\log_2(n_{\max}/m) + \log_2(W))$, où $m = 2^b$ est le nombre de boites utilisées par* Sliding Mincount *et n est le nombre de flots distincts sur la fenêtre $[t-W,t]$.*

Preuve Premièrement, le Lemme 7 assure que la somme des longueurs des LFPM est de l'ordre de $m\ln(n/m)$. Chaque élément d'une LFPM est un paquet représenté par une valeur hachée tronquée de ces b premiers bits. Comme nous l'avons vu en Section 4.1.2, la fonction de hachage doit envoyer l'identificateur de flot sur une chaîne binaire de taille un peu plus grande que $\log_2(n_{\max})$ de manière à éviter les collisions. De plus, comme les paquets sortent de la fenêtre après W unités de temps, les timestamps peuvent être stockés sur $\log_2(W)$ bits. □

4.3.2 Étude de la mémoire maximale nécessaire pour un temps d'exécution long

Dans cette section, nous étudions le *maximum* de la taille totale L^{tot} des LFPM pour un une exécution longue de SLIDING MINCOUNT. En effet, même si L^{tot} est petit pour un temps fixé —de l'ordre de $m\ln(n/m)$— il pourrait arriver, après un temps d'exécution long, que L^{tot} aie un très fort pic. Comme nous supposons avoir à notre disposition une mémoire auxiliaire limitée, une telle situation en dépasserait les capacités. Nous allons prouver maintenant qu'une telle situation ne peut arriver. Ce qui se passe est que L^{tot} change de valeur très rapidement en quelques paquets, mais ces fluctuations sont petites et *restent petites* même au cours d'une exécution très longue de SLIDING MINCOUNT.

Théorème 9 (Mémoire pour une exécution complète) *Soit $n_{\max}$ une borne supérieure du nombre de flots distincts observés sur une fenêtre de longueur W. Appliquons la boucle interne de* SLIDING MINCOUNT *sur un flux de données long mais fini, de S paquets ($S >> 1$). Alors la valeur maximale prise par la somme L^{tot} des tailles des m LFPMs est de l'ordre d'au plus*

$$m\ln(n_{\max}/m) + \sqrt{m\ln(n_{\max}/m)2\ln(S)}.$$

Avec très forte probabilité, elle ne dépassera jamais cette valeur que de quelques unités.

En conséquence, la mémoire maximale utilisée par SLIDING MINCOUNT *sur son exécution complète est, au plus, de l'ordre de*

$$\left(m\ln(n_{\max}/m) + \sqrt{m\ln(n_{\max}/m)2\ln(S)}\right)\left(\log_2(n_{\max}/m) + \log_2(W)\right).$$

Preuve La Proposition 7 assure que, à chaque arrivée d'un paquet $< H_k, t_k >$, la distribution de la somme des tailles des m LFPM est proche de la distribution de $\ln(n/m) + \sqrt{m\ln(n/m)}Z$, où n est le nombre de flots distincts dans $[t_k - W, t_k]$ et Z est une loi normale. Comme il y a S paquets dans les flux de données, majorer le maximum de L^{tot} sur le flux de donnée revient à majorer le maximum de S lois normales. Cela est réalisé en utilisant le fait que la queue de distribution d'une loi normale décroît très vite : pour $x \geq 1$, $\mathbb{P}(Z \geq x) \leq e^{-x^2/2}$. Ainsi, pour S lois normales $N_1, \ldots, N_S$, nous avons la formule simple

suivante

$$\begin{aligned}\mathbb{P}\left(\max(Z_1,\ldots,Z_S)\geq x\right) &= \mathbb{P}\left((Z_1\geq x)\cup\ldots\cup(Z_S\geq x)\right)\\ &\leq S\cdot\mathbb{P}(Z\geq x)\leq Se^{-x^2/2}.\end{aligned}$$

En conséquence, la probabilité que le maximum est plus grand que x décroît très vite vers 0 quand x dépasse $\sqrt{2\ln(S)}$. □

Comme nous le voyons ici, la fluctuation maximale sur un flux de données complet de S paquets est seulement $\sqrt{2\ln S}$ fois la fluctuation typique pour un temps t fixé, de l'ordre de $\sqrt{m\ln(n/m)}$. Comme la fonction $S\to\sqrt{2\ln S}$ croît *extrêmement lentement*, la fluctuation maximale sur la taille totale des LFPM sera petite même pour *des exécutions très longues*. Par exemple, un temps d'exécution de plusieurs années au lieu d'une heure aura très peu d'effet sur la taille maximale de la mémoire auxiliaire utilisée par Sliding Mincount.

4.4 Validation et Expérimentations sur du trafic réel

4.4.1 Validation par simulations

Pour commencer, nous validons Sliding Mincount à l'aide d'un *trafic idéal* : les paquets ont un taux d'arrivée constant et appartiennent tous à des connexions différentes. Nous simulons un flux de données de 5 millions de paquets et, tous les 100 paquets lus, nous demandons une estimée du nombre de flots distincts vus sur une fenêtre de 1 M de paquets. La Figure 4.5 (gauche) montre ces estimées et la Figure 4.5 (droite) la mémoire utilisée par l'algorithme (correspondant au nombre total d'éléments dans les LFPM). L'algorithme estime le nombre de connexions sans biais et l'erreur standard expérimentale reste dans les bornes attendues de 4% (pour $m=2^{10}$) avec quelques excursions vers 6%. Même remarque pour la taille des LFPM. Observez que l'évolution de la mémoire a plus d'à-coups que celle de l'estimée. En effet, les LFPM sont modifiées à l'arrivée de chaque paquet, alors que l'estimée est modifiée seulement quand la valeur hachée minimale change. Les résultats des tableaux de la Figure 4.6 correspondent à des simulations pour différentes valeurs de m (nombre de boites) avec deux différentes tailles de fenêtre (100 000 paquets pour la figure de gauche et 1 M de paquets pour la figure de droite). Ils montrent une bonne adéquation entre le comportement de l'algorithme et ce que l'analyse prédit. L_{tot}, la moyenne de la taille totale des LFPM pendant l'exécution, est très proche de son espérance $mH_{\lfloor n/m\rfloor}$. L_{max}, la valeur maximale de la taille totale des LFPM pendant l'exécution, est très proche de sa valeur moyenne, validant bien le fait qu'un temps d'exécution très long n'a que très peu d'effet sur la mémoire auxiliaire maximale utilisée par Sliding Mincount. Nous voyons aussi que l'erreur standard expérimentale (Stderr) est proche de son espérance ($\approx 1.3/\sqrt{m}$). Elle décroît d'un facteur 2 quand on utilise 4 fois plus de mémoire, comme prédit par l'analyse. L'algorithme réussit à estimer le nombre de connexions à 4 pourcents près en utilisant seulement $64kB$ (les LFPM ont moins de 8 000 éléments avec un timestamp de 32 bits et une valeur hachée de 32 bits également).

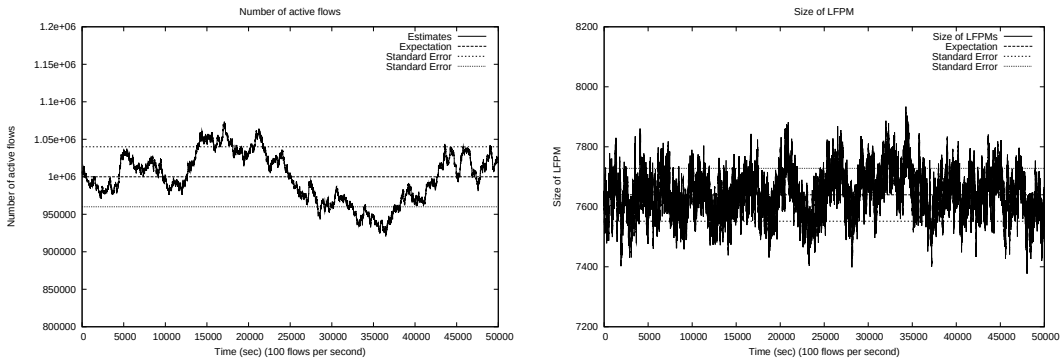


FIG. 4.5 – Comportement de SLIDING MINCOUNT au cours d'une simulation sur un trafic idéal de 5 millions d'éléments, avec une fenêtre d'un million déléments. La figure de gauche montre l'estimée donnée toutes les secondes par SLIDING MINCOUNT. La figure de droite donne la mémoire utilisée par l'algorithme pendant l'exécution ($m = 2^{10}$).

$\log_2 m$	4	6	8	10
$mH_{\lfloor n/m \rfloor}$	149	508	1676	5288
$Ltot$	148.9	507	1676	5293
$Lmax$	199	605	1815	5503
$1.3/\sqrt{m}$	0.32	0.16	0.08	0.04
$StdErr$	0.32	0.14	0.09	0.03

$\log_2 m$	4	6	8	10
$mH_{\lfloor n/m \rfloor}$	186	655	2265	7641
$Ltot$	185.9	653.5	2264	7648
$Lmax$	245	752	2430	7933
$1.3/\sqrt{m}$	0.35	0.16	0.08	0.04
$StdErr$	0.35	0.15	0.07	0.04

FIG. 4.6 – Mémoire moyenne et précision de SLIDING MINCOUNT pour un flux de données de trafic idéal de 5 millions de paquets, avec une fenêtre de 100 milles connexions (droite) et d'un millions de connexions (gauche) pour différents nombres m de boites.

4.4.2 Analyse de Trafic

Nous avons enregistré les traces des paquets arrivant à un routeur d'entrée (gate router) du campus de l'INRIA Rocquencourt. Cela correspond au trafic agrégé de l'activité de 400 ordinateurs qui voit passer typiquement 150 millions de paquets par jour. L'analyse de deux traces est présentée ici : la Trace A correspond au trafic du lundi 3 avril (2006) et la Trace B à celui du dimanche 2 avril. L'algorithme est très rapide : ces traces journalières sont traitées en trois minutes avec un processeur de 1 GHz ($1M$ de paquets par seconde) avec une implémentation même pas optimisée. Ainsi, il ne rencontrerait aucun problème pour l'observation en temps réel de tels trafics. La Figure 4.7 montre les estimées données chaque seconde par SLIDING MINCOUNT pour trois fenêtres différentes : une heure (bas), dix minutes (milieu) et une minute (haut). Avec la fenêtre coulissante d'une heure, nous voyons que le trafic de lundi (gauche) a une forme très différente du trafic de dimanche

(droite). Nous sommes capable de distinguer des pics d'activité correspondant aux sauvegardes automatiques de données pendant la nuit (à 2h et 6h30 pour la Trace A) ainsi qu'aux horaires de travail. Grâce à la fenêtre coulissante d'une minute, nous constatons que ces évolutions correspondent en fait à des pics très forts à l'échelle de la minute.

La Figure 4.8 compare le nombre de connexions distinctes (Figure 4.8 gauche) et le nombre de paquets (Figure 4.8 droite) pour nos trois fenêtres coulissantes de une heure (haut), dix minutes (milieu) et une minute (bas) sur la Trace A —voir la Section 4.2.5 pour la méthodologie pour estimer la nombre de paquets. Le nombre de paquets pendant une heure varie entre 3 millions et 12 millions et en moyenne une connexion a 30 paquets. Remarquez que le nombre de connexions peut augmenter très fortement pendant que le nombre de paquets reste lui presque constant (par exemple, considérez le trafic entre les secondes 20,000 et 28,000). Cela confirme que pouvoir compter le nombre de connexions distinctes donne des informations d'une nature différente que compter le nombre de paquets. Comme énoncé en introduction, cela a des applications importantes pour la surveillance des réseaux —par exemple, pour la détection d'attaque par déni de service (denial of service attacks).

Remerciement Les auteurs voudraient remercier Philippe Flajolet pour des discussions intéressantes et instructives, ainsi que Denis Joiret and André Balsa pour leur aide précieuse pour obtenir les traces du trafic INRIA.

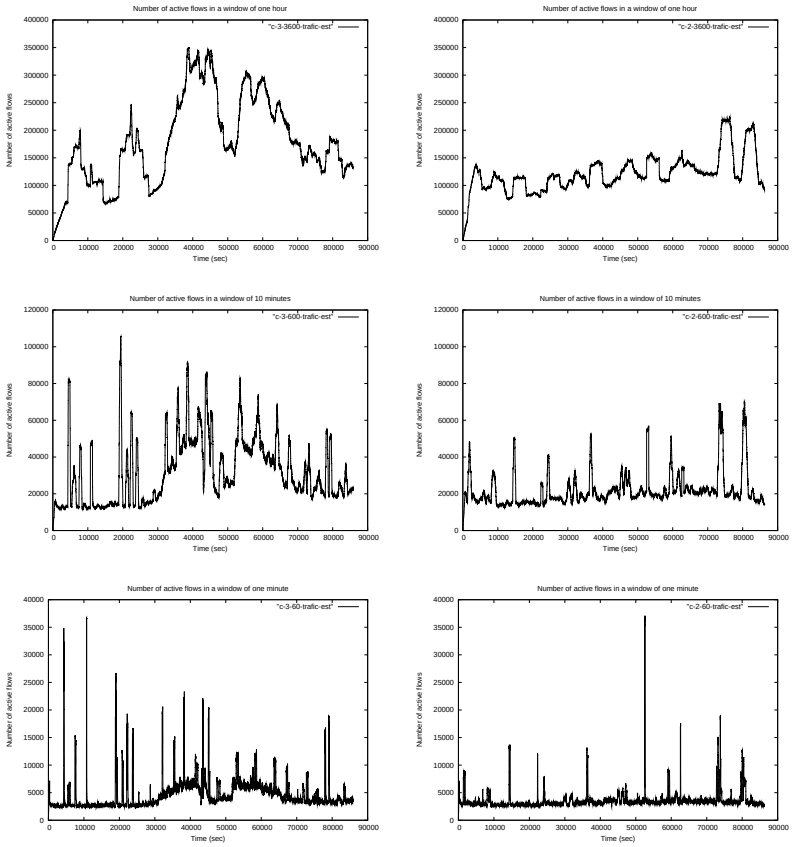


FIG. 4.7 – Estimées du nombre de flots distincts données toutes les secondes par SLIDING MINCOUNT sur la Trace A (lundi 3 avril) (Gauche) et sur la Trace B (dimanche 2 avril) (Droite) pour des fenêtres coulissantes de une heure (Haut), dix minutes (Milieu) et une minute (Bas).

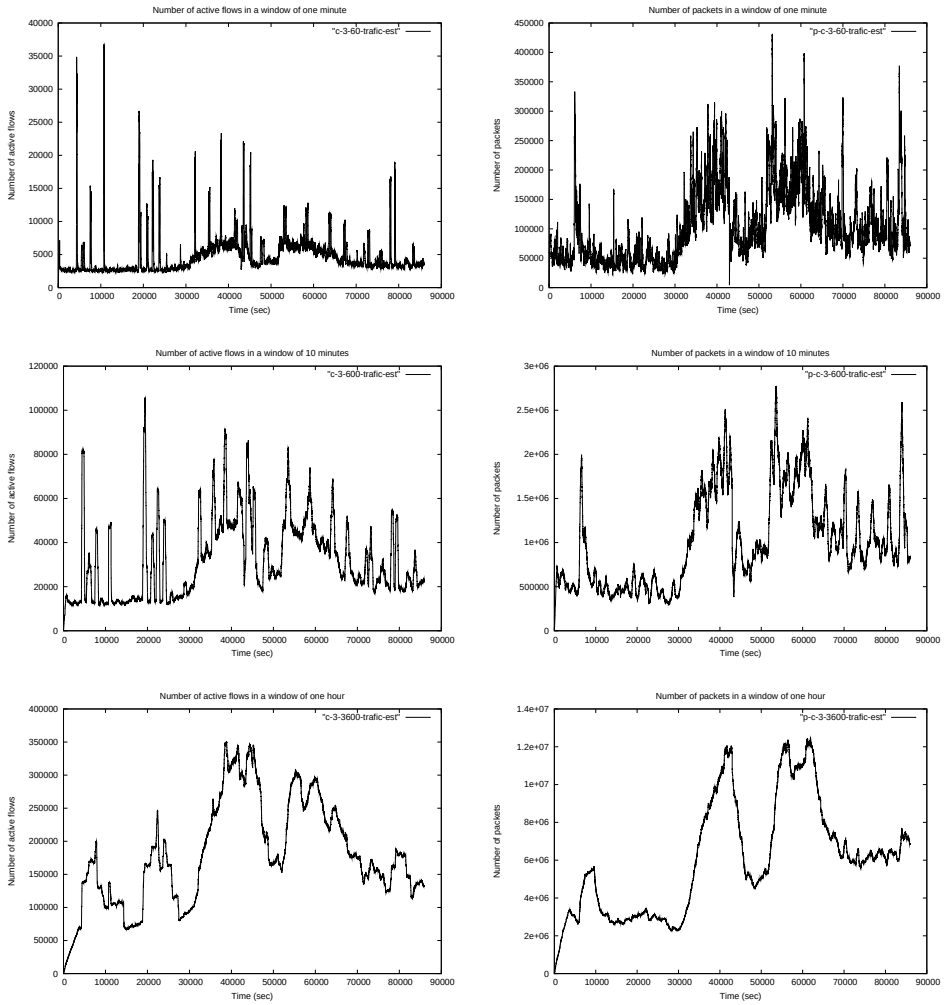


FIG. 4.8 – Comparaison entre le nombre de connexions distinctes (Gauche) et le nombre de paquets (Droite), pour des fenêtres coulissantes de une heure (Haut), dix minutes (Milieu) et une minute (Bas).

Deuxième partie

Réseaux embarqués tolérants aux pannes

Chapitre 5

Conception de Réseaux Tolérants aux Pannes Minimaux : Problème et Constructions

Ce travail a été réalisé en collaboration avec Jean-Claude Bermond et Stéphane Pérennes du Projet MASCOTTE de l'INRIA Sophia Antipolis.

mots-clés : *réseaux tolérants aux pannes, réseaux de commutation (switching networks), routage, redondance TWTA, permutations, connectivité, chemins disjoints.*

Ce chapitre traite de la conception de réseaux embarqués dans des satellites (appelés aussi Traveling wave tube Amplifiers (TWTA)). Ces réseaux doivent connecter les signaux arrivant sur certains des ports du satellites à des amplificateurs, même en cas de panne de certains d'entre eux. Ils sont construits à partir de switchs très coûteux. En raison des coûts de lancement, le but est de donc de trouver des réseaux$-(p, \lambda, k)$ valides avec un nombre minimal de sommets. Un réseau$-(p, \lambda, k)$ est un graphe non-orienté avec $p + \lambda$ entrées, $p + k$ sorties et des sommets internes de degré quatre. Un réseau$-(p, \lambda, k)$ est *valide* si, pour tous choix de p entrées et p sorties, il existe p chemins disjoints en arêtes reliant les entrées aux sorties. Dans ce chapitre, on présente formellement le problème, donne des conditions de validité et présente des méthodes pour construire des réseaux valides proches de minimaux pour de petites valeurs des paramètres —alors que le chapitre suivant présentera des constructions asymptotiques.

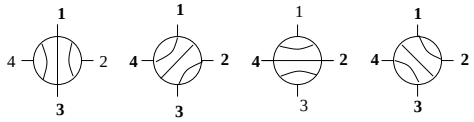


FIG. 5.1 – Un switch peut être dans quatre états différents.

Introduction

Motivation. Le problème, ici, est la conception de réseaux embarqués dans des satellites (appelés aussi Traveling Wave Tube Amplifiers) efficaces. Il a été posé à l'origine par la branche espace d'Alcatel (Alcatel Space Industry) pour des satellites destinés aux transmissions télévisuelles et de vidéos — comme les séries Eutelsat et Astra — ainsi qu'à des applications privées. Les signaux arrivant à un satellite de télécommunication à travers des ports doivent être routés vers des amplificateurs à l'aide d'un réseau embarqué. Une première contrainte — pour des raisons d'ingénérie industrielle — est que ce réseau doit être construit avec des commutateurs ou switchs à quatre liens qui peuvent réaliser les connections montrées en Figure 5.1. Remarquez que deux signaux ne peuvent pas se croiser au travers d'un switch. Néanmoins, comme les amplificateurs cibles sont identiques, il est équivalent de se croiser ou de s'éviter et on peut faire comme si toutes les connexions étaient possibles. D'autres contraintes apparaissent. D'une part les amplificateurs peuvent tomber en panne pendant le temps de vie du satellite et ne peuvent pas être réparés. D'autre part, comme le satellite est en rotation sur lui-même, tous les ports et amplificateurs ne sont pas bien orientés à chaque instant et ne sont donc pas disponibles. Pour ces raisons, on a besoin de plus de ports et d'amplificateurs que de signaux à router.

On peut facilement construire des réseaux répondant à ces contraintes en utilisant deux sélecteurs de n'importe quel degré fixé. (Pour une étude sur les sélecteurs, voir [DH00] ou le travail pionnier de Pippenger [Pip77]). Cependant, comme le lancement de satellites est extrêmement coûteux, il est crucial de minimiser le poids physique des réseaux, c'est-à-dire, pour nous, de minimiser leur nombre de switchs. De plus, comme les switchs sont aussi très coûteux à construire, en économiser même un vaut la peine. Les industries spatiales sont intéressées par la conception de tels réseaux pour des valeurs spécifiques des paramètres. Cependant la théorie générale est intéressante en elle-même.

Problème. Nous considérons ici des *réseaux*, c'est-à-dire des graphes reliant des *entrées (inputs)* à des *sorties (outputs)* et où les sommets représentent les switchs. Nous définissons un *réseau*$-(p, \lambda, k)$ comme un réseau avec $p+\lambda$ entrées et $p+k$ sorties. Un réseau$-(p, \lambda, k)$ est dit *valide*, si, pour tout choix de p entrées et de p sorties, il existe p chemins disjoints en arêtes reliant toutes les entrées choisies à toutes les sorties choisies. Pour des raisons de symétrie, nous supposerons dans la suite que $k \geq \lambda$ et nous noterons $n := p + k$.

Les entrées sont représentées dans les figures par des flèches ($\rightarrow$) et de petites boites

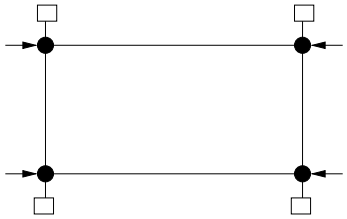


FIG. 5.2 – Un réseau$-(3,1,1)$ valide.

($\square$). Un exemple de réseau$-(3,1,1)$ valide est donné en Figure 5.2. En effet pour toutes les configurations des entrées utilisées et des sorties en panne, les trois entrées utilisées peuvent être redirigées vers les sorties valides. D'autres exemples sont présentés en Figure 5.5.

Remarquez que trouver un réseau minimal est un problème difficile : le nombre de réseaux possibles croît exponentiellement et le simple fait de vouloir tester la validité d'un réseau donné est difficile. En effet, si nous fixons les entrées et les sorties valides, tester la validité se réduit à un problème de flots, mais le nombre de choix possibles pour les entrées et les sorties croît exponentiellement comme nous sommes en présence de coefficients binomiaux. Néanmoins, le problème est dans Co-NP, comme il suffit d'exhiber une mauvaise coupe pour prouver qu'un réseau n'est pas valide. Et, en fait, décider si un réseau$-(p,\lambda,k)$ donné est valide est un problème Co-NP complet, voir [BKP$^+$81] and [Tap95] (réduction au problème de trouver une clique de taille donnée - variation de leur preuve, nous pouvons supposer que le degré maximum des graphes utilisés est quatre).

Nous étudions le cas où les switchs du réseau ont degré quatre (bien que la théorie peut être généralisée pour n'importe quel degré) ce qui est d'un intérêt primordial pour les applications. Le problème est de trouver $N(p,\lambda,k)$, le nombre *minimum* de switchs d'un réseau$-(p,\lambda,k)$ valide et de donner des *constructions* de tels réseaux.

Dans le cas spécifique où $\lambda = k$, on peut concevoir des réseaux avec une propriété particulière : chaque switch relié à un port est aussi relié à un amplificateur. En pratique, quand un port et un amplificateur relié à un même switch sont tous les deux valides, on route l'un sur l'autre. Cela minimise la longueur des signaux et évite les interférences. Ces réseaux sont appelés *réseaux simplifiés*. Observez que dans cette configuration, chaque switch est relié soit à deux switchs soit à quatre.

Contexte. Quand $\lambda = 0$, c'est-à-dire que toutes les entrées sont utilisées, un réseau valide est appelé *sélecteur*. Ce cas a été étudié par exemple dans [BDD02]. Une théorie générale des sélecteurs peut être trouvée dans [BPT] où plusieurs résultats sont obtenus pour de petites valeurs de k. Par exemple, il est prouvé que $N(p,0,4) = \lceil \frac{5p}{4} \rceil$.

Dans [BDH$^+$03] et [DHMP05] le cas des sélecteurs avec des switchs de degré $2k > 4$ est considéré. Dans [BHT06] les auteurs étudient une variante des sélecteurs où quelques signaux sont prioritaires et doivent être envoyés à des amplificateurs offrant une meilleure

qualité de service. Dans [BD02] les auteurs regardent le cas où tous les amplificateurs sont différents et pour lequel une entrée donnée doit être routée sur la sortie qui lui est dédiée, ce problème étant relié aux réseaux de permutations.
Notre travail. On a formalisé le nouveau problème dans lequel les entrées ne sont pas toutes utilisées, trouvé un critère de coupe qui caractérise la validité des réseaux et introduit des méthodes pour prouver des bornes sur les nombres de sommets minimaux des réseaux$-(p, \lambda, k)$. Rappelons que, pour des raisons de symétrie, on suppose que $k \geq \lambda$. Dans la Section 5.2, on présente des méthodes pour construire des réseaux valides proches de minimaux pour de petites valeurs de k ($1 \leq k \leq 8$) pour le problème général et le problème simplifié. En Section 5.3, on présente une méthode générale pour construire des réseaux pour tout k et tout λ.

5.1 Formalisation et Théorème de Construction

Dans cette section, nous définissons le problème de conception (design problem) de réseaux de façon plus formelle et nous introduisons des notations utilisées tout au long de ce travail. Nous énonçons un critère de coupe (Proposition 5) : celui-ci est fondamental parce qu'il caractérise la validité des réseaux$-(p, \lambda, k)$. Il est utilisé de façon intensive pour prouver que des réseaux sont valides. En Sections 5.2 et 6.3 nous l'utilisons aussi pour détecter des motifs interdits ce qui nous amène à des bornes inférieures pour le nombre de switchs des réseaux valides.

5.1.1 Problème de Conception de Réseaux$-(p, \lambda, k)$

Notations. Étant donnée une fonction f, nous définissons $f(A) := \sum_{a \in A} f(a)$ pour tout ensemble fini A. Pour un sous-ensemble W de sommets d'un graphe $G = (V, E)$, nous notons $\Delta(W)$ l'ensemble des arêtes reliant W et $\overline{W} = V \backslash W$, $\delta(W)$ la cardinalité de $\Delta(W)$, et $\Gamma(W)$ l'ensemble des sommets de $\overline{W}$ adjacents à un sommet de W. Plus généralement, nous utilisons la convention que, si un ensemble est désigné par une lettre majuscule, la lettre minuscule correspondante notera sa cardinalité.

Réseaux$-(p, \lambda, k)$ et Réseaux$-(p, \lambda, k)$ valides. Un *réseau*$-(p, \lambda, k)$ est un triplet $\mathcal{N} = \{(V, E), i, o\}$ où (V, E) est un graphe et i, o sont des fonctions entières positives définies sur V appelées fonctions d'entrées (input function) et de sorties (output function) et qui sont telles que pour tout $v \in V$, $i(v) + o(v) + deg(v) = 4$. Le nombre total d'entrées est $i(V) = \Sigma_{v \in V} i(v) = p + \lambda$, et le nombre total de sorties est $o(V) = \Sigma_{v \in V} o(v) = p + k$. Un réseau peut être vu comme un graphe où tous les sommets ont degré 4, et où les feuilles sont les entrées et les sorties. Une *fonction de sorties non en panne (non-faulty output function)* est une fonction o' définie sur V telle que $o'(v) \leq o(v)$ pour tout $v \in V$ et $o'(V) = p$. Une *fonction d'entrées utilisées (used input function)* est une fonction i' définie sur V telle que $i'(v) \leq i(v)$ pour tout $v \in V$ et $i'(V) = p$. Un réseau$-(p, \lambda, k)$ est dit *valide* si pour toute

fonction de sorties non en panne o' et pour toute fonction d'entrées utilisées, il existe p chemins disjoints en arêtes dans G tel que chaque sommet $v \in V$ est le sommet initial de $i'(v)$ chemins et le sommet terminal de $o'(v)$ chemins.

Problème de Conception de Réseaux (Design Problem). Soit $N(p, \lambda, k)$ le nombre minimal de switchs d'un réseau$-(p, \lambda, k)$ valide. Le *Problème de Conception de Réseaux* consiste à déterminer $N(p, \lambda, k)$ et à construire un réseau$-(p, \lambda, k)$ minimum, ou du moins un réseau$-(p, \lambda, k)$ valide avec un nombre de sommets proche de la valeur optimale. Nous introduisons aussi une variante du problème : considérons des réseaux avec $p + \lambda$ switchs reliés chacun à exactement une entrée et une sortie (nous appelons de tels switchs des doublons) et avec $k - \lambda$ switches avec seulement une sortie. Trouver de tels réseaux valides est le *Problème de Conception de Réseaux Simplifié*. Les réseaux de cette sorte sont particulièrement bon pour les applications pratiques. En effet ils ont un processus de routage simplifié, ils minimisent la longueur des chemins et les interférences entre signaux.

5.1.2 Excès, Validité et Critère de Coupe.

Nous montrons que pour vérifier si un réseau est valide, au lieu de résoudre un problème de flot pour chaque configuration possible de pannes de sorties et d'entrées utilisées, il est suffisant de regarder une mesure invariante des sous-ensembles du réseau, l'*excès*, défini comme suit.

Définition 3 (Excès $\varepsilon(W)$, Excès en Entrées $\varepsilon_i(W)$, Excès en Sorties $\varepsilon_o(W)$) *Soit un réseau$-(p, \lambda, k)$ et un de ses sous-ensembles de sommets $W \subset V$. L'*excès en entrées *de W est défini comme*

$$\varepsilon_i(W) := \delta(W) + o(W) - \min(k, o(W)) - \min(i(W), p).$$

*L'*excès en sorties *de W est défini comme*

$$\varepsilon_o(W) := \delta(W) + i(W) - \min(\lambda, i(W)) - \min(o(W), p).$$

Par défaut, on note $\varepsilon(W) := \varepsilon_i(W)$.

Proposition 5 (Critère de Coupe) *Un réseau$-(p, \lambda, k)$ est* valide *si et seulement si, pour tout sous-ensemble $W \subset V$ l'*excès de W *satisfait, $\varepsilon(W) \geq 0$.*

L'intuition est que les signaux arrivant dans W (en nombre au plus $\min(i(W), p)$) doivent être routés soit vers les sorties valides de W (en nombre au moins $o(W) - \min(k, o(W))$) soit vers les liens menant en dehors (en nombre $\delta(W)$). La preuve formelle, omise ici, se réduit à un problème de flot (supply/demand flow problem). Remarquez que, pour le critère de coupe, il est suffisant de considérer seulement les sous-ensembles W connexes qui ont de plus un complémentaire $\overline{W}$ connexe (Cela provient de la sous-modularité de ε).

FIG. 5.3 – Réseaux symétriques $(2,0,2)$ et $(2,2,0)$ valides.

Lemme 8 *Soit un réseau$-(p,\lambda,k)$ et un de ses sous-ensembles $W \in V$. On a $\varepsilon_o(\overline{W}) = \varepsilon_i(W)$.*

Preuve Comme $\delta(\overline{W}) = \delta(W)$, $o(\overline{W}) = p + k - o(W)$, $i(\overline{W}) = p + \lambda - i(W)$,

$$\begin{aligned}
\varepsilon_o(\overline{W}) &= \delta(W) + p + \lambda - i(W) - \min(\lambda, p + \lambda - i(W)) - \min(p + k - o(W), p) \\
&= \delta(W) + p + \lambda - i(W) - (\lambda - i(W) + \min(i(W), p)) - (p - o(W) + \min(k, o(w))) \\
&= \delta(W) + o(W) - \min(k, o(w)) - \min(i(W), p) = \varepsilon_i(W).
\end{aligned}$$

Proposition 6 (Critères de Coupe et Symétrie) *Un réseau$-(p,\lambda,k)$ est valide si et seulement si une des propositions suivantes est vraie.*

1. *Pour tout W, $\varepsilon_i(W) \geq 0$.*
2. *Pour tout W, $\varepsilon_o(W) \geq 0$.*
3. *Pour tout W, avec $o(W) \leq \lceil \frac{p+k}{2} \rceil$ et $i(W) \leq \lceil \frac{p+\lambda}{2} \rceil$, $\varepsilon_i(W) \geq 0$ ET $\varepsilon_o(W) \geq 0$.*
4. *Le réseau est un réseau simplifié et pour tout W, avec $|W| \leq \lceil n/2 \rceil$, $\varepsilon_i(W) \geq 0$ OU $\varepsilon_o(W) \geq 0$.*

Preuve La preuve est directe avec la Proposition 5 et le Lemme 8.

Proposition 7 (Critères de Coupe et Connexité) *Pour tous les critères de la Proposition 6, il est suffisant de ne considérer que les sous-ensembles W connexes qui ont des complémentaires $\overline{W}$ connexes.*

Preuve Si un W a un excès négatif, une de ses composantes connexes a aussi un excès négatif. Si W a un excès négatif, $\overline{W}$ a un excès négatif en sorties, et donc une de ses composantes connexes aussi.

5.1.3 Théorèmes Généraux de Construction

Théorème 10 *Le problème est symétrique en entrée et sorties. C'est-à-dire $\mathcal{N}(p,\lambda,k) = \mathcal{N}(p,k,\lambda)$*

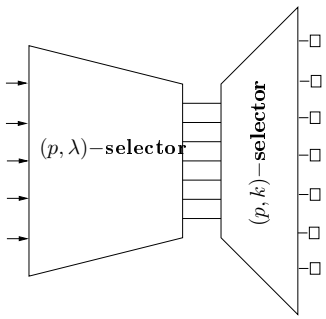


FIG. 5.4 – Un réseau$-(p,\lambda,k)$ construit avec deux sélecteurs en séries.

Preuve Dès qu'on a un réseau$-(p,\lambda,k)$, on a immédiatement un réseau$-(p,k,\lambda)$ qui résulte de l'inversion des entrées et des sorties (voir l'exemple de la Figure 5.3). Sa validité est donnée par la Proposition 6.

Théorème 11 *Un réseau$-(p,\lambda,k)$ valide $\mathcal{R}$ est construit à partir d'un réseau$-(p,k,k)$ valide $\mathcal{R}'$ en lui enlevant $k-\lambda$ entrées.*

Preuve Comme $i_{\mathcal{R}}(W) \leq i_{\mathcal{R}'}(W)$, $\varepsilon_{\mathcal{R}}(W) \geq \varepsilon_{\mathcal{R}'}(W)$, ce qui finit la preuve.

Liens avec les réseaux$-(p,k)$ ou sélecteurs$-(p,k)$. Les réseaux$-(p,\lambda,k)$ pour lesquels toutes les entrées sont utilisées ($\lambda = 0$) ont été étudiés dans [BPT] et [Hav06a] et sont appelés réseaux$-(p,k)$ ou sélecteurs$-(p,k)$. On a

$$N(p,0,k) = N(p,k,0) = N(p,k). \tag{5.1}$$

Théorème 12

$$N(p,\lambda,k) \leq N(p,\lambda) + N(p,k)$$
$$N(p,\lambda,k) \leq O(p+k)$$

Preuve La preuve est constructive. On peut construire un réseau$-(p,\lambda,k)$ valide à partir de deux sélecteurs valides, un sélecteur$-(p,\lambda)$ et un sélecteur$-(p,k)$ comme indiqué en Figure 5.4. L'idée est d'utiliser le premier sélecteur à l'envers en remplaçant ses $p+\lambda$ sorties par nos entrées. Ensuite, on relie les entrées des deux sélecteurs. Les sorties du second sélecteur correspondent à nos sorties. Tout sous-ensemble des entrées de taille p peut être routé vers les p liens centraux par le premier sélecteur. Le second route ces liens vers n'importe quel sous-ensemble de nos sorties. Le réseau est donc valide. De tels réseaux sont nommés réseaux$-(p,\lambda,k)$ 2sélecteurs (Définition 4).

Comme [BPT] et [Hav06a] prouvent que le nombre minimal de switchs d'un sélecteur$-(p,k)$ est linéaire, on obtient le même résultat pour les réseaux$-(p,\lambda,k)$.

Définition 4 (Réseaux$-(p,\lambda,k)$ 2sélecteurs) *Un réseaux$-(p,\lambda,k)$ 2sélecteur est construit avec un sélecteur$-(p,\lambda)$ et un sélecteur$-(p,k)$ en série.*

5.2 Constructions pour les petits cas

On suppose ici que $1 \leq \lambda \leq k$ (voir le Théorème 10 et l'Équation 5.1). Nous présentons tout d'abord des résultats pour $p = 1, 2$ — dans la suite, on supposera $p \geq 3$. On présente des méthodes pour construire des réseaux valides et proches de minimaux pour $1 \leq k \leq 8$. Les preuves sont pour la plupart omises ici, mais la preuve de validité des réseaux$-(p,\lambda,k)$ généraux définis en Section 5.3 est donnée comme exemple typique. La Figure 5.6 résume les constructions présentées.

5.2.1 Définitions préliminaires.

Soit $\mathcal{R}$ un réseau$-(p,\lambda,k)$. On rappelle que $n := p + k$.

Définition 5 (Doublons, R-Switches) *Un* doublon *de $\mathcal{R}$ est un sommet avec $i(v) = o(v) = 1$. Un* R-switch *est un sommet qui n'est pas un doublon.*

Définition 6 (Arêtes de types E_0, E_1 et E_2) *On construit un graphe G associé à $\mathcal{R}$. Ces sommets sont les R-switches de $\mathcal{R}$. Ces arêtes sont de trois sortes, E_0, E_1 et E_2 : les arêtes de $\mathcal{R}$ entre deux R-switches, les arêtes correspondant dans $\mathcal{R}$ à un chemin de longueur deux avec un doublon au milieu et celle correspondant à un chemin de longueur trois avec deux doublons au milieu. (le critère de coupe montre que ce sont les seules arêtes possibles pour un réseau valide).*

5.2.2 Méthode pour construire des réseaux minimaux valides pour $p = 1, 2$ et $\lambda \leq k$

. Quand $p = 1$, nous construisons un réseau avec 2 switchs avec 3 entrées ou 3 sorties et un nombre maximal de switchs avec deux entrées ou sorties tous reliés en une ligne, comme montré en Figure 5.7. Quand $p = 2$, nous construisons un réseau avec un nombre maximal de switchs avec 2 entrées ou sorties connectés en cercle, comme montré en Figure 5.7.

Théorème 13 *$N(1,\lambda,k) = \lceil \frac{\lambda+k}{2} \rceil$ et $N(2,\lambda,k) = \lceil \frac{\lambda+k}{2} \rceil + 1$*

Preuve Ces réseaux, qui ont ces nombres de switchs, sont valides. Quand $p = 1$, soit W un sous-ensemble connexe de V (voir la Proposition 7 pour le choix de W). $\delta(W) \geq 1$, donc $\varepsilon(W) \geq 1 + o(W) - \min(o(W), k) - 1 \geq 0$. Le critère de coupe finit la preuve. Quand $p = 2$, $\delta(W) \geq 2$, donc $\varepsilon(W) \geq 2 + o(W) - \min(o(W), k) - 2 \geq 0$.

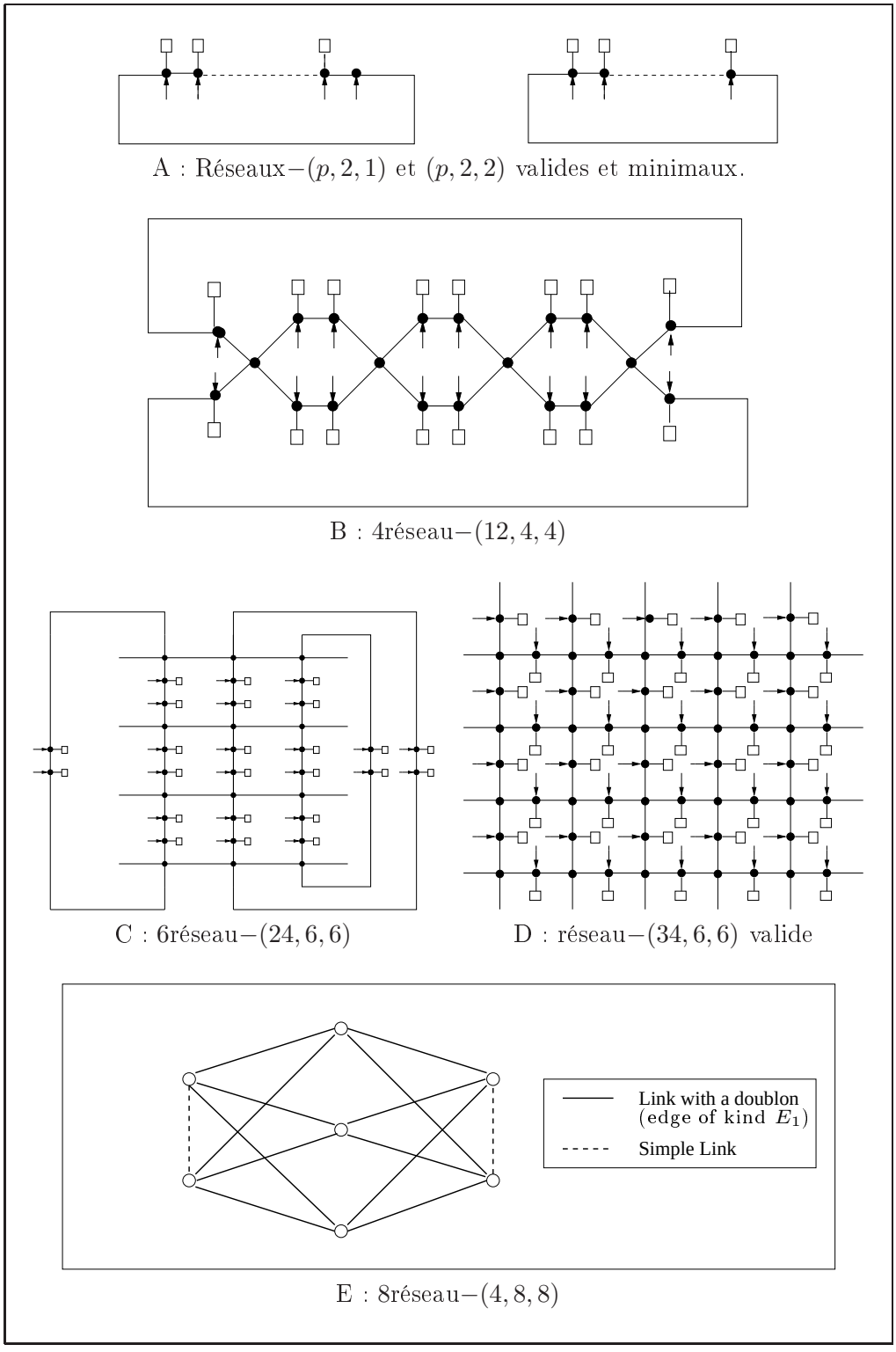


FIG. 5.5 – Constructions pour de petits k

k,λ	Réseaux	Taille	Minimalité ?
$k = 1, 2$	2réseaux	n	minimal
$k = 3, 4$	4réseaux	$n + \frac{n}{4}$	minimal
$k = 5, 6$	6réseaux	$n + \frac{n}{2}$	minimal (réseaux simplifiés)
$k = 7, 8$	8réseaux	$n + \frac{7n}{12}$	minimal (réseaux simplifiés)
pour tout k et λ	réseaux généraux		discussion en Section 5.3

FIG. 5.6 – Résumé des constructions.

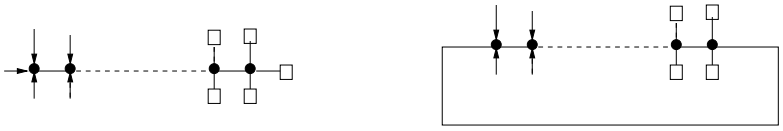


FIG. 5.7 – Réseau$-(1, \lambda, k)$ et $-(2, \lambda, k)$ valides et minimaux.

Ces réseaux sont minimaux. Quand $p = 1$, le réseau est minimal parce que un réseau valide doit être connexe. Quand $p = 2$, si on enlève un sommet, on obtient un sommet v avec deux entrées et une sortie (ou deux sorties et une entrée). $\varepsilon(\{v\}) = 1+1-1-2 = -1$ ($\varepsilon_o(\{v\}) = -1$) et le réseau ne serait pas valide.

5.2.3 Méthode pour construire des réseaux minimaux valides pour $k = 1, 2$, 2réseau.

Un 2réseau est construit avec λ doublons et $k - \lambda$ sommets avec une sortie connectés en cercle, comme montré en Figure 5.5 A.

Théorème 14 $N(p, \lambda, k) = p + 2$

Preuve Les 2réseaux sont valides. Soit $W \in V$. $\delta(W) = 2$. Soit $o(w) \leq k$. On a $\varepsilon(W) = 2 - \min(p, i(W))$. Comme $i(W) \leq o(W) \leq k \leq 2$, $\varepsilon(W) \geq 0$, et le critère de coupe finit la preuve.

Les 2réseaux sont minimaux. Si on enlève un sommet, on obtient un sommet avec $i(v) = 1$ et $o(v) = 2$. Considérons le sous-ensemble W constitué de ce switch $\varepsilon_o(W) = 1 + 1 - 1 - 2 = -1$. Ce réseau ne serait donc pas valide.

5.2.4 Méthode pour construire des réseaux minimaux valides pour $k \leq 3, 4$, 4réseau.

Un 4réseau est construit avec des blocs fait d'un sommet sans entrées ni sorties et de quatre doublons sur deux arêtes de type E_2. Un bloc est connecté à deux autres blocs

identiques en série comme montré dans le 4réseau$-(16,4,4)$ de la Figure 5.5 B. Chaque bloc a 5 sommets, 4 entrées et 4 sorties sauf un bloc ou deux si $(p+k) \mod 4 \neq 0$ or $(p+\lambda) \mod 4 \neq 0$. Le nombre de sommets d'une 4réseau$-(p,\lambda,k)$ est

$$n + \frac{n}{4} + c'_4 - c,$$

avec $c'_4 = \lceil \frac{n \mod 4}{4} \rceil$ et $c = \lfloor \frac{k-\lambda}{2} \rfloor$ (comme le nombre de sommets est $n + \lceil \frac{n}{4} \rceil - c$). $c'_4 \leq 1$ et $c \leq 2$.

Théorème 15 *Les 4réseaux sont valides.*

Théorème 16 *Les 4réseaux sont minimaux. C'est-à-dire*

$$N(p,\lambda,k) \geq n + \frac{n}{4} - c''_4,$$

avec $c''_4 = \frac{k-\lambda}{2} + \frac{k-\lambda}{8}$. *Remarquez que la différence entre le nombre de switchs d'un réseau*$-(p,\lambda,k)$ *et la borne inférieure est au plus* 1 *(*$c'_4 - c + c''_4 \leq 1 - \lfloor \frac{k-\lambda}{2} \rfloor + \frac{k-\lambda}{2} + \frac{k-\lambda}{8} \leq 2$*).*

5.2.5 Méthode pour construire des réseaux minimaux valides pour $k \leq 5,6$, 6réseaux.

Un 6réseau est construit avec des blocs fait de 3 switchs connectés en cercle, chacun d'entre eux étant connecté à deux doublons sur une arête de type E_2. Un bloc est connecté à deux autres blocs identiques en série comme montré dans le 6réseau$-(24,6,6)$ de la Figure 5.5 C. Chaque bloc a 9 sommets, 6 entrées et 6 sorties sauf un ou deux blocs si $(p+k) \mod 4 \neq 0$ ou $(p+\lambda) \mod 4 \neq 0$. Le nombre de switchs d'un 6réseau$-(p,\lambda,k)$ est

$$n + \frac{n}{2} + c'_6 - c,$$

avec $c'_6 = 3\lceil \frac{n \mod 6}{6} \rceil$ et $c = \lfloor \frac{k-\lambda}{2} \rfloor$ (comme le nombre de switchs est $n + 3\lceil \frac{n}{6} \rceil - c$). $c'_6 \leq 3$ et $c \leq 2$.

Théorème 17 *Les 6réseaux sont valides.*

Théorème 18 *Les 6réseaux sont minimaux pour le Problème de Conception Simplifié. C'est-à-dire*

$$N'(p,\lambda,k) \geq n + \frac{n}{2} - c''_6,$$

avec $c''_6 = \frac{k-\lambda}{2} + \frac{k-\lambda}{4}$. *Remarquez que la différence entre le nombre de switchs d'un 6réseau*$-(p,\lambda,k)$ *et la borne inférieure est au plus* 4.

Théorème 19 *Pour le Problème de Conception Générale, nous prouvons*

$$N(p,\lambda,k) \geq n + \frac{3n}{8} - c'',$$

avec $c'' = \frac{k-\lambda}{2} + \frac{3(k-\lambda)}{16}$.

Nous avons aussi trouvé une autre famille de réseaux valides avec le même nombre de switchs. Des sommets sans entrées ni sorties sont reliés en grille sur une sphère avec des arêtes de type E_1, comme montré sur la Figure 5.5 D.

5.2.6 Méthode pour construire des réseaux minimaux valides pour $k = 7, 8$, 8réseaux.

Un 8réseau est construit avec n doublons, $\frac{n}{4}$ sommets dans N_4 et $\frac{n}{3}$ sommets dans N_3. Les sommets dans N_4 et N_3 n'ont ni entrées ni de sorties. Un sommet dans N_4 est relié à quatre sommets dans N_3 par une arête de type E_1 (voir la Définition 6). Un sommet dans N_3 est relié à trois sommets dans N_4 par une arête de type E_1 et à un sommet de N_3 par une arête de E_0. N_3 est divisé en quatre groupes avec la condition que deux sommets reliés par une arête de E_0 sont dans le même groupe. Chaque sommet dans N_4 est relié aux quatre groupes comme montré dans le 8réseau$-(4,8,8)$ avec 19 sommets de la Figure 5.5 E. Le nombre de switchs d'un 8réseau$-(p,\lambda,k)$ est

$$n + \frac{7n}{12} + c'_8 - c.$$

avec $c'_8 = 7\lceil \frac{n \mod 12}{12} \rceil$ et $c = \lfloor \frac{k-\lambda}{2} \rfloor$ (comme le nombre de switchs est $n + 7\lceil \frac{n}{12} \rceil - c$). $c'_8 \leq 7$ et $c \leq 2$.

Théorème 20 *Les 8réseaux sont valides.*

Théorème 21 *Les 8réseaux sont minimaux pour le Problème de Conception Simplifié où les arêtes de type E_2 sont interdites. C'est-à-dire*

$$N'(p,\lambda,k) \geq n + \frac{7n}{12} - c''_8,$$

avec $c''_8 = \frac{k-\lambda}{2} + \frac{7(k-\lambda)}{24}$. *Remarquez que la différence entre le nombre de switchs d'un 8réseau$-(p,\lambda,k)$ et la borne inférieure est au plus* 8.

5.3 Constructions pour tout k et λ. Réseaux$-(p,\lambda,k)$ Généraux

Nous présentons ici les Réseaux$-(p,\lambda,k)$ Généraux (voir la Définition 9). Leurs tailles sont proches d'être minimales pour les petits k (voir la Remarque 1). Ils sont construits à partie des $\nu-$boites introduites dans la Définition 8.

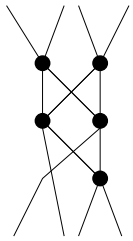


FIG. 5.8 – Un 4-boite minimale.

5.3.1 Préliminaires : $\nu-$boites et réseaux de ν-permutation

La propriété décisive des $\nu-$boites est exprimée dans le Lemme 9.

Définition 7 ($\nu-$graphe) *Un $\nu-$graphe ($\nu \geq 2$) est une paire (G, l) où $G = (V, E)$ est un graphe simple orienté et l une fonction positive définie sur V telle que $l(V) = 2n$ et que pour tout sommet v, $l(v) + deg(v) = 4$.*

Définition 8 ($\nu-$boite) *Une $\nu-$boite est un $\nu-$graphe tel que pour toute fonction entière i définie sur V avec $0 \leq i \leq l$ et $i(V) = n$, il existe n chemins disjoints en arêtes tels que tout sommet v est le début de $i(v)$ chemins et la fin de $l(v) - i(v)$ chemins.*

Lemme 9 *Dans une $\nu-$boite, pour tout sous-ensemble $X \subseteq V$, on a*

$$|\Gamma(X)| \geq \min(l(X), l(\bar{X})).$$

Preuve La preuve se réduit à un problème de flot et peut être trouvée dans [Ser04].

Proposition 8 *Un réseau de $\nu-$permutation est une $\nu-$boite.*

Les preuves des propriétés des $\nu-$boites sont omises ici. Il existe des constructions asymptotiques *linéaires* pour les $\nu-$boites. Cependant, pour de petites valeurs de ν, aucune construction de $\nu-$boites plus petites que le réseau de permutations correspondant n'a été trouvée. Pour $\nu \leq 6$, il a même été prouvé que les réseaux de permutations sont optimaux. Les réseaux de permutations AS-Waksman ont été étudiés et prouvés minimaux dans pour des réseaux de permutations dans [BD99] et [BD02]. Pour ces raisons, nous avons choisi les réseaux de permutations AS-Waksman pour nos $\nu-$boites pour de petits k. À titre d'exemple, une $4-$boite minimale peut être vue en Figure 5.8.

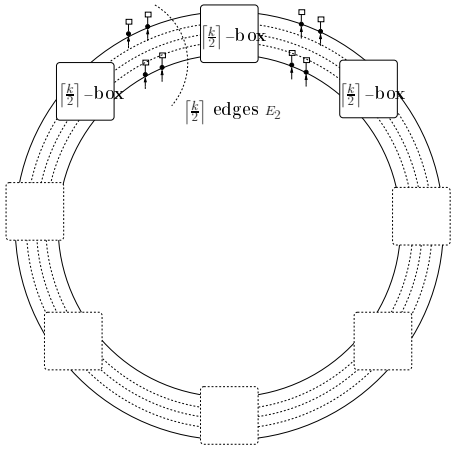


FIG. 5.9 – Schéma des réseaux$-(p, \lambda, k)$ généraux.

5.3.2 Réseaux$-(p, \lambda, k)$ Généraux

Définition 9 (Réseaux$-(p, \lambda, k)$ Généraux) *Un réseau$-(p, \lambda, k)$ général (voir la Figure 5.9) est construit à partir de* $\left\lceil \frac{p+k}{k} \right\rceil$ $\lceil \frac{k}{2} \rceil -$*boites connectées en cercle. Ces boites sont reliées par*

- *un nombre maximum,* $\left\lfloor \frac{p+\lambda}{2} \right\rfloor$*, d'arêtes de type* E_2*,*
- 1 *arête de type* E_1 *si* $p + \lambda$ *est impair et* 0 *sinon,*
- $\left\lfloor \frac{k-\lambda}{2} \right\rfloor$ *arêtes avec un sommet avec deux sorties (arêtes de type* E'_2*),*
- 1 *arête avec un sommet avec une sortie (arête de type* E'_1*), si* $k - \lambda$ *est impair et* 0 *sinon.*
- *le reste de type* E_0 $\left(\lceil \frac{k}{2} \rceil \left\lceil \frac{p+k}{k} \right\rceil - e_2 - e_1 - e'_2 - e'_1 \right)$.

Lemme 10 *Le nombre de switchs d'un réseau$-(p, \lambda, k)$ général est*

$$n + \frac{B_{min}(\lceil \frac{k}{2} \rceil)}{2\lceil \frac{k}{2} \rceil} n + c'_g - c,$$

avec $B_{min}(\nu)$ *le nombre de sommets d'une* $\nu-$*boite minimale, avec* $c'_g = B_{min}\lceil \frac{n \mod 2\lceil \frac{k}{2} \rceil}{2\lceil \frac{k}{2} \rceil} \rceil$ *et* $c = \lfloor \frac{k-\lambda}{2} \rfloor$.

Remarque 1 *Les tailles des réseaux$-(p, \lambda, k)$ généraux pour de petits* k *qui utilisent des réseaux de permutations AS-Waksman pour leur* $\lceil \frac{k}{2} \rceil -$*boites sont indiquées en Figure 5.10. Les réseaux$-(p, \lambda, k)$ généraux sont proches d'être minimaux pour les petits* k *(À comparer avec les réseaux de la Section 5.2).*

k	$3,4$	$5,6$	$7,8$	$9,10$	$13,14$
$Size$	$n+\frac{n}{4}$	$n+\frac{n}{2}$	$n+\frac{5n}{8}$	$n+\frac{8n}{10}$	$2n$

FIG. 5.10 – Tailles des réseaux$-(p, \lambda, k)$ généraux pour de petits k.

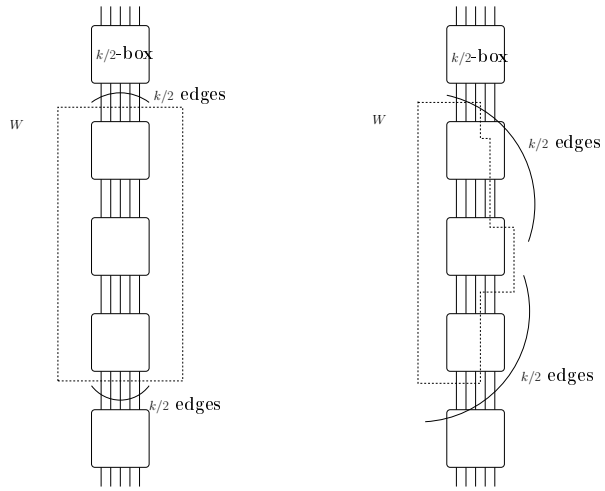


FIG. 5.11 – Intuition de la preuve de validité des réseaux généraux.

Proposition 9 *Un Réseau$-(p, \lambda, k)$ Général est un réseau$-(p, \lambda, k)$ valide.*

L'intuition de la preuve est donnée en Figure 5.11. Quand W contient des boites entières, $\delta(W) = k$. Sinon, on utilise la propriété des boites donnée par le Lemme 9 pour montrer que $\delta(W)$ ne peut pas être plus petit.

Preuve

On prouve d'abord que quand $k = \lambda$ le réseau$-(p, k, k)$ général est valide. On utilise ensuite le Théorème 11 pour construire à partir de lui, un réseau$-(p, \lambda, k)$ général valide quand $\lambda \leq k$, ce qui finit la preuve. Remarquez que le réseau$-(p, k, k)$ général est un réseau simplifié.

Définissons tout d'abord quelques notations. Un niveau d'un boite B_i, on note

- e_i (resp. f_i) le nombre d'arêtes de type E_2 reliant un switch dans $B_i \cap W$ (resp. $B_i \cap \overline{W}$) et un switch dans $(B_{i-1} \cap W) \cup (B_{i+1} \cap W)$ (resp. $(B_{i-1} \cap \overline{W}) \cup (B_{i+1} \cap \overline{W})$).
- e'_i (resp. f'_i) le nombre d'arêtes de type E_2 reliant un switch dans $B_i \cap W$ (resp. $B_i \cap \overline{W}$) et un switch dans $(B_{i-1} \cap \overline{W}) \cup (B_{i+1} \cap \overline{W})$ (resp. $(B_{i-1} \cap W) \cup (B_{i+1} \cap W)$).
- $\delta_i^{\text{box}} := \delta(W \cap B_i, \overline{W} \cap B_i)$
- $\delta_i := \delta(W \cap B_i, \overline{W})$. Remarquez que

$$\delta_i = e'_i + \delta_i^{\text{box}}$$

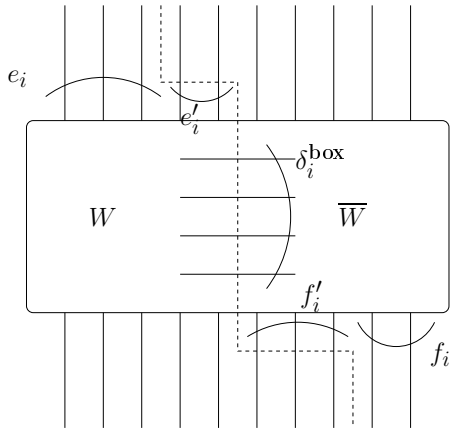


FIG. 5.12 – Boite B_i.

Pour le réseau entier, on a

$$\begin{array}{rcl} 2e_2(W) & = & \sum_i e_i \\ e'_2(W) & = & \sum_i e'_i \\ \delta(W) & = & \sum_i \delta_i \end{array}$$

Dans ce contexte, le Lemme 9 donne

$$\delta_i^{\text{box}} \geq \min(e_i + e'_i, f_i + f'_i).$$

Soit W un sous-ensemble de switchs du réseau$-(p,k,k)$ général. On distingue deux cas
– $k \leq o(W) \leq n-k$. Comme ici $i(W) = o(W)$, on a aussi $i(W) = o(W) \leq n-k = p$. D'où

$$\varepsilon(W) \geq \delta(W) - k.$$

Considérons l'ensemble $\mathcal{B}$ de toutes les boites B_i tel que

$$e_i + e'_i \leq f_i + f'_i.$$

S'il n'existe pas, on utilise $\overline{W}$ à la place de W. On distingue deux cas
– $\mathcal{B}$ contient toutes les boites. Pour chaque boite B_i, le Lemme 9 donne

$$\delta_i \geq e'_i + (e_i + e'_i),$$

$$\delta(W) \geq 2e_2(W) + 2e'_2(W) \geq o(W).$$

Dans ce cas $o(W) \geq k$, d'où $\varepsilon(W) \geq 0$. Si nous travaillions sur $\overline{W}$, on conclut en utilisant $o(W) \leq n-k$, qui donne $o(\overline{W}) \geq k$).

– Sinon, il existe deux boites B_j et B_k (pas nécessairement distinctes) qui satisfont

$$\begin{cases} e_j + e'_j & \geq f_j + f'_j \text{ et } e_{j+1} + e'_{j+1} \leq f_{j+1} + f'_{j+1} \\ e_k + e'_k & \geq f_k + f'_k \text{ et } e_{k-1} + e'_{k-1} \leq f_{k-1} + f'_{k-1} \end{cases}$$

Le Lemme 9 donne

$$\begin{cases} \delta_j + \delta_{j+1} \geq (e'_j + e'_{j+1}) + (f_j + f'_j) + (e_{j+1} + e'_{j+1}) \\ \delta_k + \delta_{k-1} \geq (e'_k + e'_{k-1}) + (f_k + f'_k) + (e_{k-1} + e'_{k-1}) \end{cases}$$

Remarquez que pour toutes les boites B_i, $f_i + e_{i+1} + e'_i + e'_{j+1} \geq \lceil k/2 \rceil$, ce qui donne

$$\delta(W) \geq k.$$

– $o(W) \leq k$ or $o(W) \geq n - k$. Pour la clarté de la preuve, nous ne présentons ici que la preuve pour la cas $p + k = 0 \mod k$. Toutes les boites sont donc identiques avec seulement des arêtes E_2. La preuve complète suit les mêmes idées.
Par symétrie, selon la Proposition 6, nous sommes autorisés à ne considérer que le premier cas

$$\varepsilon(W) \geq \delta(W) - \min(i(W), p) \geq \delta(W) - o(W).$$

Considérons une boite B_i. Si $e_i + e'_i \leq f_i + f'_i$, le Lemme 9 donne $\delta_i \geq e'_i + (e_i + e'_i)$ et

$$\delta(W) \geq 2e_2(W) + 2e'_2(W) \geq o(W)$$

Sinon le Lemme 9 donne $\delta_i \geq e'_i + (f_i + f'_i)$. Il faut maintenant partitionner les arêtes e'_i selon leur nombre de doublons dans W : $e'_i(0)$, $e'_i(1)$, $e'_i(2)$. Comme $2e_i + 2e'_i(2) + e'_i(1) \leq o(W) \leq k$ et $e_i + e'_i(2) + e'_i(1) + e'_i(0) + f_i + f'_i = k$ (nombre d'arêtes d'une boite), on a

$$f_i + f'_i \geq e_i + e'_i(2) - e'_i(0).$$

D'où

$$\delta_i \geq e'_i + e_i + e'_i(2) - e'_i(0) = e_i + 2e'_i(2) + e'_i(1).$$

En considérant toutes les boites, on conclut que

$$\delta(W) \geq 2e_2(W) + 2e'_2(2)(W) + e'_2(1)(W) = o(W).$$

Chapitre 6

Sélecteurs Minimaux et Réseaux Tolérants aux Pannes : Cas Asymptotique

Ce travail a été réalisé en collaboration avec Omid Amini, Florian Huc et Stéphane Pérennes du Projet MASCOTTE de l'INRIA Sophia Antipolis.

mots-clés : *sélecteurs, super-sélecteurs, réseaux tolérants aux pannes, switching networks, routage, expandeurs, connectivité, chemins disjoints.*

Un réseau$-(p, \lambda, k)$ est un graphe non-orienté avec $p + \lambda$ entrées, $p + k$ sorties et des sommets internes de degré quatre. Un réseau$-(p, \lambda, k)$ est *valide* si, pour tous choix de p entrées et p sorties, il existe p chemins disjoints en arêtes reliant les entrées aux sorties. Dans le cas particulier où $\lambda = 0$, un réseau$-(p, \lambda, k)$ est déjà connu sous le nom de *sélecteur (selector)*. Nous souhaitons déterminer $N(p, \lambda, k)$, le nombre minimal de nœuds dans un réseau$-(p, \lambda, k)$ valide. Pour ce faire, nous donnons ici des conditions de validité desquelles se déduisent des bornes inférieures pour $N(p, \lambda, k)$. Nous présentons aussi des constructions, et donc des bornes supérieures, basées sur les expandeurs. Le problème est très sensible aux ordres de λ et k. Par exemple, quand λ et k sont petits devant p, le problème se réduit à éviter certaines configurations locales interdites. Pour des valeurs plus grandes de λ et k, le problème est de trouver des graphes avec une bonne propriété d'expansion pour les petits ensembles. Cela nous amène à introduire un nouveau paramètre appelé *robustesse*. Nous donnons des bornes asymptotiques serrées pour $N(p, \lambda, k)$ pour de nombreux cas.

Introduction

Contexte. Cette étude doit être considérée dans l'esprit de la théorie classique des *superconcentrateurs (superconcentrators)*. Un *concentrateur*-$(p+\lambda, p)$ *(concentrator)* [Val75] est un graphe orienté acyclique $G = (V, E)$ avec $p+\lambda$ nœuds entrées désignés et p nœuds sorties désignés ($\lambda \geq 0$), tel que pour chaque sous-ensemble de p nœuds entrées, il existe p chemins disjoints en arêtes (ou, selon le contexte, disjoints en sommets) qui relient les entrées aux sorties. Un *sélecteur (selector)*, introduit pour la première fois dans [BDD02], est un réseau-$(p, \lambda, 0)$ ($k = 0$). Une théorie générale des sélecteurs peut être trouvée dans [BPT], où plusieurs résultats sont obtenus pour de petites valeurs de λ. Un sélecteur peut être vu comme une version non orientée d'un concentrateur-$(p+\lambda, p)$. Un *n-superconcentrateur* [Pip77] est un graphe orienté acyclique $G = (V, E)$ avec n nœuds entrées désignés et n nœuds sorties désignés, tel que, pour tout ensemble S de $p \leq n$ entrées et tout ensemble T de p sorties, il existe p chemins disjoints en arêtes (ou disjoints en sommets, selon le contexte) reliant S à T. Il existe une vaste théorie des superconcentrateurs (Voir le travail pionner de Pippenger [Pip77] pour une introduction, [AM84, AGM87, AC03] et Schöning [Sch06] pour des constructions, et [BKP$^+$81] pour les problèmes de complexité).

En ce sens, les réseaux-(p, λ, k) généralisent le concept de sélecteurs de la même manière que les superconcentrateurs généralisent celui de concentrateurs. Il est aussi clair que, prendre un superconcentrateur (resp. un concentrateur-$(p+\lambda, p)$) en oubliant les orientations, procure un réseau-(p, λ, k) valide pour n'importe quelle valeur de $k = \lambda$ (resp. n'importe quel λ et $k = 0$). On peut ainsi essayer d'utiliser des superconcentrateurs connus pour construire des réseaux embarqués efficaces, mais, et ce n'est pas étonnant, cela ne donne généralement pas des réseaux minimaux. La différence majeure entre les superconcentrateurs et les réseaux valides généraux est que dans un réseau valide le nombre d'entrées et de sorties qui peuvent tomber en panne est borné (dans notre cas, étant donné une probabilité de panne, pas plus de k pannes n'arriveront au cours de la durée de vie du satellite). Trouver un réseau minimal (un superconcentrateur de faible densité ou ayant un nombre minimal de sommets) est un problème difficile et un domaine actif de recherche (voir les papiers [AM84], [AGM87], [AC03] et la construction récente de [Sch06]). D'un point de vue algorithmique, on peut montrer que le problème de tester si un graphe est un concentrateur-$(p+\lambda, p)$ ou un superconcentrateur est co-NP complet. En fait, ce sont des problèmes complets même quand ils sont réduits au cas spécial important des graphes de taille linéaire par rapport au nombre d'entrées (voir [BKP$^+$81]). On peut montrer, avec le même type d'arguments, que le problème de décider si un réseau-(p, λ, k) est valide ou non est coNP-complet, même réduit aux réseaux 4-réguliers.

Résultats. Nous sommes intéressés dans ce chapitre par les *grands réseaux*, pour lesquels $n = p + k$ tend vers l'infini et k est assez grand.

La construction de réseaux valides optimaux repose très fortement sur les expandeurs. En utilisant ces derniers, nous sommes capables de construire des réseaux simplifiés avec $2n$ switchs dès que n est assez grand et $k \leq c_1 \log n$ pour une constante c_1 (Section 6.1.4). En

Section 6.3.3 nous donnons aussi une borne inférieure d'ordre $2n(1-\epsilon(k))$ où $\epsilon(k)$ tend vers zéro quand k tend vers l'infini (mais $k \leq c_1 \log n$ n'est pas nécessaire). Ainsi, le problème est résolu dans le cas asymptotique pour les réseaux simplifiés pour $k \leq c_1 \log n$.

Pour les réseaux généraux, en utilisant des expandeurs bi-partie, nous obtenons une borne supérieure en $n + \frac{3}{4}n$ quand $k \leq c_2 \log n$ pour une constante c_2. La borne inférieure obtenue est $(n + \frac{2}{3}n)(1 - \epsilon(\lambda, k))$, et nous conjecturons que $n + \frac{3}{4}n$ doit être la vraie valeur.

Nous donnons aussi une construction de sélecteurs (cas $\lambda = 0$) de taille $n + \frac{n}{2}$, ce qui nous donne dans ce cas aussi une borne inférieure serrée.

Pour étendre ces résultats pour des valeurs plus larges de k, nous définissons une propriété d'expansion locale que nous appelons $\alpha-r$obustesse (voir [CRVW02, AHK99] pour des notions proches). Intuitivement, l'α-robustesse d'un graphe G est une version locale du facteur d'expansion. C'est l'entier maximal r_α tel que pour tous les petits sous-ensembles le facteur d'expansion est α, mais pour les sous-ensembles plus-grands leur nombre d'arêtes sortantes est au moins r_α. Pour nos constructions, il est nécessaire que le graphe soit un expandeurs pour les petits sous-ensembles, mais pour les plus grands on a juste besoin d'une connectivité minimale constante avec le reste du graphe. Cette notion est aussi intéressante en elle-même et contient comme cas particulier plusieurs invariants d'expansion comme la longueur de bi-section et la constante de Cheeger (voir [Chu97]). Comme les graphes aléatoires sont de très bons expandeurs, on étudie leur α-robustesse. De cette façon, on généralise le résultat de Bollobàs sur le facteur d'expansion des graphes aléatoires 4-réguliers. À notre connaissance, c'est la première fois qu'est défini et étudié ce nouveau concept d'expansion locale. En l'utilisant, nous présentons ici une construction d'un réseau$-(p, \lambda, k)$ valide avec $3n$ switchs pour $\lambda \leq k \leq \frac{n}{7}$.

6.1 Bornes supérieures : le Problème de Conception de Réseaux

Dans cette section, nous présentons trois constructions avec $2n$, $n + \frac{3}{4}n$ et $n + \frac{n}{2}$ switchs respectivement pour le problème de conception simplifié, le problème de conception général (n'importe quel λ) et pour le problème de conception général quand $\lambda = 0$. Toutes ces constructions sont valides pour $k \leq c \cdot \log n$ (où c est une constante dépendant seulement du facteur d'expansion des graphes 3 et $4-$réguliers). Mais tout d'abord, certaines preuves de bornes supérieures et inférieures —voir la Section 6.1.3 et la Section 6.3.1— sont simplifiées grâce à l'utilisation de la notion de *graphe associé* d'un réseau introduite dans la section suivante.

6.1.1 Graphe Associé.

Les sommets $D \in V$ de degré 2 avec $i(D) = o(D) = 1$ jouent un rôle important. Nous les appelons *doublons*. Un switch qui n'est pas un doublon est appelé un *R-switch*.

Remarquez que pour $k \geq 3$ il n'existe pas de chemins comprenant 3 doublons ou plus consécutifs. En effet si nous considérons l'ensemble W qui consiste en ces doublons, nous avons $\delta(W) = 2$ et $o(W) = i(W) \geq 3$. Le critère de coupe donnerait une contradiction.

Soit $\mathcal{N}$ un réseau$-(p, \lambda, k)$. Nous construisons le *graphe $\mathcal{R}$ associé à $\mathcal{N}$*. Ses sommets sont les $R-$switches de $\mathcal{N}$. En conséquence les arêtes de $\mathcal{R}$ sont de trois sortes, respectivement E_0, E_1 et E_2 : les arêtes de $\mathcal{N}$ entre deux $R-$switches, les arêtes correspondant dans $\mathcal{N}$ à un chemin de longueur 2 avec un doublon au milieu et ceux correspondant à un chemin de longueur 3 avec deux doublons au milieu.

6.1.2 Expandeurs

Un expandeur (voir [DH00] ou [Mur03] pour une étude générale) est un graphe peu dense fortement connecté (highly connected sparse graph). Ils sont utilisés dans des domaines variés de l'informatique et des mathématiques : par exemple pour certaines constructions de codes correcteurs d'erreurs avec des algorithmes de codage et décodage très efficaces, pour la déterminisation (derandomization) d'algorithmes probabilistes, pour la construction de groupes finiment engendrés qui ne peuvent pas être plongés uniformément dans un espace de Hilbert, ... mais ils ont aussi des applications dans des domaines reliés directement au sujet de ce chapitre comme la conception de réseaux super-efficaces explicites ou la construction explicite de graphes avec une grande maille (longueur du plus petit cycle).

Nous présentons ici certains résultats connus sur les expandeurs qui seront utilisés dans les preuves des Sections 6.1.4, 6.1.5 et 6.1.6. Un expandeur se définit formellement comme suit : un (n, r, c)-E-expandeur est un graphe fini $r-$régulier $G = (V, E)$ avec n sommets tel que pour tout ensemble A de sommets de G avec $|A| \leq |V|/2$ nous avons

$$\delta(A) \geq c|A|.$$

Les graphes de Ramanujan sont des exemples très connus d'expandeurs (pour plus sur les graphes de Ramanujan voir [Mor94]). Des constructions explicites de graphes de Ramanujan sont connues pour r de la forme $r = q+1$, avec q une puissance première (en particulier pour $r = 3$ et $r = 4$ d'intérêt direct dans notre cas). Plus précisément il existe des constructions explicites d'une famille infinie $G_i = (V_i, E_i)$ de graphes de Ramanujan telle que $|V_i| \underset{i\to\infty}{\to} \infty$ avec un facteur d'expansion

$$c \geq 1 - \frac{4(r-1)}{r^2}.$$

Cela donne $c \geq \frac{1}{4}$ pour les graphes $4-$réguliers et $c \geq \frac{1}{9}$ pour les graphes $3-$réguliers. La maille des graphes de cette famille satisfait :

$$g(G_i) \geq \frac{2}{3} \log_q |V_i|$$

Il existe aussi une famille $H_i = (W_i, F_i)$ de graphe de Ramanujan *bi-partie* de maille :

$$g(H_i) \geq \frac{4}{3} \log_q |W_i|.$$

Des arguments probabilistes montrent l'existence de graphes bi-partie ou non, avec les mêmes propriétés pour la maille et *même un meilleur facteur d'expansion* pour *toute grande taille (order) de réseaux* (et pas seulement pour les valeurs spécifiques de ces deux familles). Pour les graphes 4−réguliers, des expandeurs avec

$$c \geq \frac{11}{25}$$

existent (voir [Bol88]).

6.1.3 Critère de Coupe pour le Graphe Associé

Pour simplifier les preuves de validité des Sections 6.1.4, 6.1.5 et 6.1.6, on travaille directement sur le graphe associé $\mathcal{R}$ du réseau$-(p, \lambda, k)$ $\mathcal{N} = ((V, E), i, o)$. Cela signifie plus précisément que, quand on applique le critère de coupe pour $\mathcal{N}$, il est en fait suffisant de ne considérer que les sous-ensembles de $\mathcal{R}$.

Nous introduisons une autre notion d'excès, ε', définie pour tout $W \subset V$ comme $\varepsilon'(W) := \delta(W) + o(W) - \min(k, o(W)) - i(W)$. Notez que $\varepsilon'(W) \leq \varepsilon(W)$. On a ainsi que, si pour tout $W \subset V$, $\varepsilon'(W) \geq 0$, alors le réseau est valide. Mais l'équivalence est perdue. La propriété utile de ε' est que, pour tout doublon D, $\varepsilon'(W \cup \{D\}) \leq \varepsilon'(W)$. Il est ainsi suffisant de ne vérifier le critère de coupe que pour les sous-ensembles W de $\mathcal{N}$ comprenant un ensemble de $R-$switches ainsi que tous les doublons des arêtes qui leur sont incidentes de types E_1 et E_2.

6.1.4 Problème de Conception Simplifié - Borne Supérieure en $2n$

Dans cette section, nous utilisons l'existence des expandeurs présentés en Section 6.1.2 pour construire des réseaux$-(p, \lambda, k)$ valides avec $2n = 2(p+k)$ pour n grand et $k \leq c_1 \log n$ (c_1 ne dépendant seulement du facteur d'expansion des expandeurs 4−réguliers, $c_1 = \frac{1}{6}$ quand on utilise des graphes de Ramanujan explicites). De plus nous montrons en Section 6.3 une borne inférieure du même ordre pour le problème de conception simplifié.

Théorème 1 *Soit $n = p + k$, $k \leq \frac{1}{6} \log n$, pour n assez grand, nous avons :*

$$N(p, k, k) \leq 2n$$

Preuve Les résultats sur les expandeurs exposés en Section 6.1.2 énoncent l'existence de $(n, 4, c = \frac{1}{4})-$E-expandeurs, $G = (V, E)$, de maille g, $g \geq \frac{2}{3} \log n$. Soit $k \leq c \cdot g$ et $p = n - k$. Il est bien connu que, dans un graphe 4−régulier, il existe une famille de cycles disjoints en sommets qui couvre tous les sommets de G. Nous appelons cette famille F et ajoutons n nouveaux sommets dans le graphe en découpant chaque arête de F en deux arêtes. Nous

ajoutons une entrée et une sortie sur chaque nouveau sommet, créant ainsi un doublon. Nous avons maintenant un réseau$-(p,k,k)$, $\mathcal{N}$, avec $2n$ switchs.

Prouvons maintenant que ce réseau est valide. Nous utilisons le critère de coupe sur $\mathcal{R}$, qui est exactement G dans ce cas. Remarquez tout d'abord que le réseau est symétrique en entrées et sorties et que, pour tout sous-ensemble $W \subset V$, $i(W) = o(W)$. Ainsi nous avons

$$\varepsilon'(W) = \delta(W) - \min(k, o(W)).$$

En outre, notez que si un réseau est symétrique, il est suffisant de vérifier le critère de coupe sur les seuls sous-ensembles W tels que $|W| \leq |V|/2$ (pour plus de détails sur les variantes du critère de coupe, voir la Section 5.1.2).

- Si $|W| \geq \frac{k}{c}$, alors, par la propriété d'expansion, il y a au moins k arêtes reliant W et $\overline{W}$, et donc $\varepsilon'(W) \geq 0$.
- Dans le cas contraire, si $|W| < \frac{k}{c} \leq g$, nous avons $|W| < g$ et W est alors acyclique. Il y a au plus $|W| - 1$ arêtes à l'intérieur de W et, comme G est $4-$régulier, nous avons $\delta(W) \geq 2|W| + 2$. Soit $e_F(W)$ le nombre d'arêtes de F incidente à un sommet de W. Par construction $o(W) = e_F(W)$. Comme tous les cycles de F sont disjoints, $e_F(W) \leq 2|W|$. Donc $\delta(W) \geq e_F(W) = o(W)$, c'est-à-dire $\varepsilon'(W) \geq 0$.

Le réseau$-(p,k,k)$ est valide.

On peut construire des réseaux$-(p,\lambda,k)$ avec $\lambda \leq k$ en enlevant $k - \lambda$ entrées.

6.1.5 Problème de Conception Général - Borne supérieure en $n + \frac{3}{4}n$

Dans cette section, nous construisons des réseaux$-(p,k,k)$ généraux pour n grand et $k \leq c_2 \log n$ (où c_2 ne dépend que du facteur d'expansion des expandeurs $3-$réguliers). On en déduira aussi ici des réseaux$-(p,\lambda,k)$ pour tout $\lambda \leq k$ en enlevant $k - \lambda$ entrées. Le Théorème 2 donne une borne supérieure en $n + 3/4n$ pour ces réseaux. Les constructions sont basées sur des expandeurs $3-$réguliers bi-partie.

Definition 1 Deux arêtes sont à distance d *si tout chemin qui les contient toutes les deux est de longueur au moins* $d+2$. Un sommet est à distance d d'une arête *si tout chemin qui les contient tous les deux est de longueur au moins* $d+1$.

Lemme 1 *Soit F un graphe bi-partie $3-$régulier $G = (V_1 \cup V_2, E)$ de grande maille ($g = \Theta(\log |V_1|)$) qui est un $(2|V_1|, 3, c)$-E-expandeur et supposons que $2k \leq cg$. Soit $\mathcal{F}$ un ensemble d'arêtes marquées de façon à ce que toutes soient à distance au moins 3. Le réseau $\mathcal{N}$, obtenu à partir de G en ajoutant un doublon sur chaque arête de $\mathcal{F}$, une entrée sur chaque sommet de V_1 et une sortie sur chaque sommet de V_2, est un réseau valide.*

Preuve

Nous utilisons ici le critère de coupe sur le graphe associé comme expliqué en Section 6.1.3. Comme la construction est symétrique en entrées et sorties, il suffit de ne considérer que les sous-ensembles $W \in V$ connexes tels que $|W| \leq \lceil \frac{|V|}{2} \rceil$. Le critère de coupe se

déduit de

$$\varepsilon'(W) = \delta(W) + o(W) - \min(o(W), k) - i(W) \geq 0.$$

Nous distinguons maintenant deux cas pour W.

- cas 1 : $|W| \leq \frac{2k}{c} \leq g$
  Comme $o(W) - min(o(W), k) \geq 0$, nous avons
  $$\varepsilon'(W) \geq \delta(W) - i(W).$$
  Ainsi il est suffisant de montrer que $\delta(W) \geq i(W)$. Comme $|W| \leq g$, il n'y a pas de cycles à l'intérieur de W et donc il y a $|W| - 1$ arêtes dedans ; G est 3−régulier, donc $\delta(W) = |W| + 2$. De plus, $i(W) = v_1(W) + d$, donc nous devons prouver que $v_2(W) + 2 \geq d$.
  Considérons un doublon D incident à W et son arête associée $e(D) = (v_1(D), v_2(D))$ avec $v_1(D) \in V_1$ et $v_2(D) \in V_2$. Si $v_2(D) \in W$, on associe D avec $v_2(D)$. Si $v_2(D) \notin W$, alors $v_1(D) \in W$. On associe à D un voisin de $v_1(D) \in W$. Comme la distance entre deux arêtes de $\mathcal{F}$ est au moins 3, des doublons différents ont des sommets associés eux-aussi différents dans $V_2 \cap W$. Donc $v_2(W) \geq d$.
- cas 2 : $|W| \geq \frac{2k}{c}$, par définition de ε', on a
  $$\varepsilon' \geq \delta(W) + o(W) - k - i(W).$$
  - si $i(W) - o(W) \leq k$, par la propriété d'expansion, nous avons $\delta(W) \geq 2k \geq i(W) - o(W) + k$
  - si $i(W) - o(W) \geq k$, comme le graphe est bi-partie, il y a au moins $3(i(W) - o(W))$ arêtes sortantes. Donc $\varepsilon'(W) \geq 2(i(W) - o(W)) - k \geq k > 0$.

Théorème 2 *(Construction) Soit $n = p + k$, $k \leq \frac{1}{15} \log n$, pour n assez grand, nous avons :*

$$N(p, k, k) \leq n + \frac{3}{4}n$$

Preuve Soit $H = (A, B, E)$, un $(2|A|, 4, c')$-E-expandeur bi-partie de maille $g = \frac{4}{3} \log n$. Soit k, $k \leq c' \cdot g \leq \frac{4}{3} \log n$ (pour l'existence de tels graphes voir la Section 6.1.2). Soit F un couplage complet (complete matching) du complémentaire bi-partie $\overline{H} = (A, B, \overline{E})$ de H, c'est-à-dire, si $(u, v) \in F$, alors $u \in A$, $v \in B$ et u, v ne sont pas adjacent dans H.

Pour chaque arête $e = (a, b)$ de F, on remplace a et b par trois sommets a_1, a_2, a_3 et b_1, b_2, b_3, on ajoute les arêtes (a_1, b_1), (a_1, b_2), (a_1, b_3), (a_2, b_1) et (a_3, b_1). Enfin on relie a_2 (resp b_2) à deux voisins de a (resp b) et a_3 (resp b_3) aux deux autres voisins. Voir la Figure 6.3.1. Nous obtenons ainsi un graphe 3−régulier bi-partie qui est un $(6|A|, 3, \frac{c}{5})$ − E−expandeur. Notez que par construction les arêtes de type (a_1, b_1), avec $(a, b) \in F$ forment un ensemble $\mathcal{F}$ d'arêtes marquées qui sont, par paires, à distance 3. Nous pouvons donc appliquer le Lemme 1 à G avec $\mathcal{F}$ comme ensemble d'arêtes marquées. En résumant nous avons $6|A| + |\mathcal{F}| = 7|A|$ switchs, $|V_1| + |\mathcal{F} = 4|A|$ entrées, $|V_2| + |\mathcal{F} = 4|A|$ sorties. Donc, $N(p, \lambda, k) = 7|A| = \frac{7}{4}n$, avec $n = p + k$.

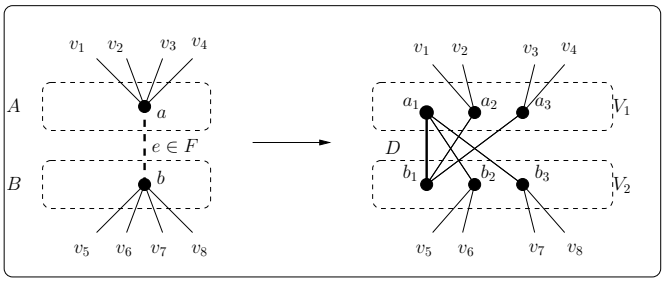


FIG. 6.1 – 'Expansion' d'une paire de sommets sélectionnés du graphe bi-partie (A, B, E).

6.1.6 Problème de Conception Général : $\lambda = 0$ - Borne Supérieure en $n + \frac{n}{2}$

Dans cette section, nous étudions le cas $\lambda = 0$. Dans la littérature, ce type de réseaux est connu sous le nom de sélecteurs. Dans le Théorème 3 nous construisons des réseaux$-(p, 0, k)$ valides avec $n + \frac{n}{2}$ switchs pour n grand et $k \leq c_3 \log n$ (c_3 ne dépendant que du facteur d'expansion des expandeurs 3−réguliers). Ces réseaux sont construits à partir d'expandeurs de sommets (vertex-expanders) bi-partie (voir la Définition 2).

Definition 2 *(V-Expandeur) Un (n, r, d)-V-expandeur est un graphe fini r-régulier $G = (V, E)$ avec $|V| = n$ ($|V_1| = |V_2| = n$ dans le cas d'un graphe bi-partie et $V = V_1 \cup V_2$) tel que pour tout sous-ensemble A de sommets ($A \subset I$ quand G est bi-partie), l'ensemble de voisins de $A, \Gamma(A) = \{v \in V | (v, u) \in E$ pour un $u \in V\}$ satisfait*

$$|\Gamma(A)| \geq |A| + d(1 - |A|/n)|A|$$

Théorème 3 *Soit $n = p + k$, $k \leq \frac{1}{48} \log_2 n$, pour n assez grand, nous avons :*

$$N(p, 0, k) \leq n + \frac{3}{4}n.$$

Preuve

Prenons $G = (V_1, V_2, E)$ un $(n, 3, d = \frac{1}{12})-V-$expandeur bi-partie de maille $g \geq \frac{4}{3} \log n$ (pour l'existence d'un tel graphe voir la Section 6.1.2). Soit $\alpha = \frac{4}{d} = 48$ et k tel que $k \leq d \cdot \frac{g}{2} \leq \frac{g}{12}$ et $k.(2^{\alpha k} + 1) \leq n$.

À chaque sommet de V_1, nous connectons un sommet avec une entrée et deux sorties, qui est dit de type T. À chaque sommet de V_2, nous attachons une entrée. Nous choisissons un sous-ensemble S de k sommets de V_2 tels que la distance entre tout couple d'entre eux

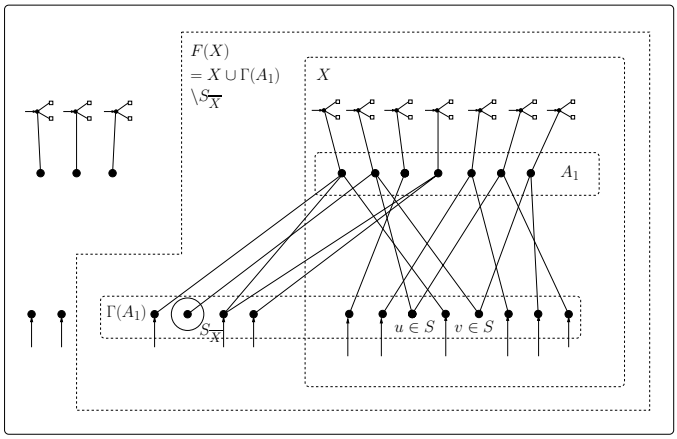


FIG. 6.2 – Schéma de la preuve du Théorème 3.

est au moins $\alpha \cdot k$ (voir Définition 1). En raison du choix de k, nous pouvons choisir les sommets de S l'un après l'autre en enlevant à chaque fois tous les sommets qui sont à distance plus petite que αk des sommets déjà choisis (à chaque étape, on enlève au plus $2^{\alpha k} + 1$ nouveau sommet). Nous enlevons à présent les entrées de tous les sommets de S pour obtenir un réseau$-(p = n - k, 0, k)$ que nous appelons $\mathcal{N} = (V, E, i, o)$.

Il faut maintenant prouver que ce réseau est valide. Soit X un sous-ensemble connexe de V. Soit $\overline{X} = V_1 \cup V_2 \setminus X$, $A_1 = X \cap V_1$, $S_{\overline{X}} = S \cap \Gamma(A_1) \cap \overline{X}$. Nous définissons $Z(X) = X \cup \Gamma(A_1) \setminus S_{\overline{X}}$ (Figure 6.2).

$Z = Z(X)$ est connexe. De plus $\varepsilon(Z) \leq \varepsilon(X)$. Ainsi il suffit de vérifier le critère de coupe pour Z (comme expliqué dans la Section 6.1.3). Soit $A_2 = Z \cap V_2$.

Nous distinguons quatre cas pour $|A_1|$:

- $|A_1| \leq \frac{k}{2}$: comme $o(Z) = 2|A_1| \leq k$ et $\varepsilon(Z) \geq \delta(Z) - i(Z)$, nous avons $\delta(Z) \geq 3|A_2| - 3|A_1|$ et $i(Z) \leq |A_2| + |A_1|$. De plus il n'y a pas de cycle dans Z, donc $|A_2| \geq 2|A_1|$. Donc $\varepsilon(Z) \geq 0$.
- $\frac{k}{2} \leq |A_1| \leq \frac{k}{d} \leq \frac{g}{2}$: nous avons $\varepsilon = \delta(Z) + o - i - k$ et $i \leq |A_1| + |A_2|$. Ainsi $\varepsilon(Z) \geq \delta(Z) - A_2 + A_1 - k$. Comme Z ne contient pas de cycle $\delta(Z) \geq 3|A_2| - 3|A_1|$ donc $\varepsilon \geq 2|A_2| - 2|A_1| - k \geq 2|A_1| - k \geq 0$. Nous utilisons de nouveau le fait que $|A_2| \geq 2|A_1|$.
- $\frac{k}{d} \leq |A_1| \leq n - \frac{k}{d}$: dans ce cas, $\varepsilon \geq \delta(Z) - |A_2| + |A_1| - k$ et $\delta(Z) \geq 3(|A_2| - |A_1|) = |A_2| - |A_1| + 2(|A_2| - |A_1|)$, donc nous avons $\varepsilon \geq 2(|A_2| - |A_1|) - k$. De plus $|A_2| - |A_1| \geq \Gamma_G(A_1) - k$, parce qu'au plus k sommets de $\Gamma_G(A_1)$ sont dans $S \setminus A_2$ (où $\Gamma_G(A_1)$ est dans le voisinage de A_1 dans G). Par la propriété d'expansion, nous avons :

$$|\Gamma_G(A_1)| \geq |A_1| + d(1 - |A_1|/n)|A_1|.$$

Dans ce cas $d(1 - |A_1|/n) \geq 2k$. Ainsi $|\Gamma_G(A_1)| - |A_1| \geq 2k$. Donc $|A_2| - |A_1| \geq k$. Cela donne $\varepsilon \geq 0$.

– $n - \frac{k}{d} \leq |A_1|$: nous pouvons supposer que $\overline{Z}$ est connexe (voir les remarques de la Proposition 5). Comme les sommets de S sont à distance au moins αk et $\alpha k > \frac{k}{d}$, $\overline{Z}$ contient au plus un sommet de S. En conséquence $i \geq |A_2| - k + 1 + |A_1|$ et $\varepsilon \geq \delta(Z) - |A_2| + |A_1| - 1$. En raison de la connexité de Z et de la $3-$régularité, nous avons $|A_2| > |A_1|$. Et donc $\varepsilon \geq 0$.

6.2 Une nouvelle approche générale basée sur la robustesse des graphes pour la conception de réseaux valides

Dans la section précédente, nous avons vu que, dès que n est assez grand et que $k \leq c \log n$ pour une constante c donnée, des réseaux valides proches d'être minimaux peuvent être construits en utilisant des expandeurs. Pour étendre ces résultats pour des valeurs plus larges de k, nous définissons une propriété d'expansion locale que nous appelons $\alpha-r$obustesse (voir [CRVW02, AHK99] pour des notions proches). Cette notion est intéressante en elle-même et contient comme cas particulier plusieurs invariants d'expansion comme la longueur de bi-section et la constante de Cheeger (voir [Chu97]). En l'utilisant, nous présentons ici une construction d'un réseau$-(p, \lambda, k)$ valide avec $3n$ switchs pour $\lambda \leq k \leq \frac{n}{7}$.

6.2.1 Robustesse

La construction de réseaux valides est liée à une propriété plus générale des graphes qui peut être vue comme une *propriété d'expansion bornée* ou propriété d'expansion seulement sur les ensembles de petites tailles. On appelle cette propriété *robustesse*. Étant donné un réel α, l'$\alpha-$robustesse d'un graphe, r_α, est le plus grand entier tel que, pour tout sous-ensemble $X \subset V$ de taille $|X| \leq \frac{|V|}{2}$:

$$\delta(X) \geq \min(\alpha \cdot |X|, r_\alpha).$$

Pour $\alpha = 1$, on parle simplement de robustesse. En d'autres mots, si un graphe est $\alpha-$robuste, ses petits sous-ensembles ont un facteur d'expansion de α, alors que ces grands sous-ensembles ont juste au moins r_α arêtes sortantes.

L'étude de la robustesse d'un graphe est une approche perpendiculaire aux études classique. Au lieu de rechercher des graphes de facteurs d'expansion maximaux, on fixe l'expansion et on se demande quelle est la taille maximale des sous-ensembles qui la satisfont. Néanmoins ce problème est lié à des problèmes classiques : il recouvre celui de la longueur de bi-section (bisection-width), étudiée par plusieurs auteurs (voir par exemple [Alo93] et [MP01]), qui a de nombreuses applications importantes ; il est bien évidemment lié aux expandeurs et, par exemple, comme il existe des graphes de facteur d'expansion $\frac{1}{2}(d-2\sqrt{d-1})$

(graphes de Ramanujan), pour $\alpha \leq \frac{1}{2}(d - 2\sqrt{d-1})$, il existe des graphes de robustesse $r_{\frac{1}{2}(d-\sqrt{d-1})} \geq \frac{n}{2}$.

On se pose deux problèmes principaux. D'une part, on aimerait se faire une idée de la robustesse d'un graphe général, c'est-à-dire exhiber des bornes supérieures en montrant l'existence, dans certains cas, de sous-ensembles qui violent l'$\alpha-$robustesse. D'un autre côté, pour de nombreuses applications, on voudrait être capable de trouver des graphes de grande robustesse. De premiers exemples de tels graphes sont bien sûr les expandeurs de facteur d'expansion c qui donnent des graphes de $c-$robustesse maximale, égale à $\frac{n}{2}$ où n est le nombre de sommets du graphe.

En utilisant la méthode du premier moment, approche proche de celle de Bollobás dans [Bol88], on peut trouver une borne inférieure pour la robustesse des graphes aléatoires $2k-$réguliers. Ainsi, on montre en Section 6.2.2 le théorème suivant,

Théorème 22 *Les graphes aléatoires $4-$réguliers d'ordre n ont une robustesse supérieure à $n/14$ avec une probabilité tendant vers 1 quand n tend vers l'infini.*

Le même type de méthodes donne des résultats pour l'$\alpha-$robustesse, pour $\alpha \neq 1$. Ainsi, pour quelques valeurs, on obtient les bornes suivantes.

$\mathcal{R}_1$	$\mathcal{R}_{4/5}$	$\mathcal{R}_{2/3}$	$\mathcal{R}_{3/5}$	$\mathcal{R}_{11/20}$	$\mathcal{R}_{11/25}$
$n/14$	$n/5.77$	$n/3.66$	$n/3.00$	$n/2.62$	$n/2$

La première correspond au premier théorème et la dernière dit simplement que les graphes aléatoires $4-$réguliers ont asymptotiquement un facteur d'expansion d'au moins $11/25$.

On utilise ensuite les résultats sur la robustesse pour étendre les résultats de bornes supérieures pour des nombres de pannes,k, plus grands. L'idée est de prendre un graphe d'$\alpha-$robustesse k et de choisir les arêtes qui vont recevoir les entrées et les sorties (souvent sous forme de doublons) pour obtenir des réseaux valides pour $k = r_\alpha$. Ainsi, Théorème 23 implique

Théorème 4 *Pour $k \leq \frac{n}{7}$, il existe des réseaux$-(p, \lambda, k)$ valide de taille $3n$.*

Preuve Soit un graphe $4-$régulier hamiltonien G d'ordre $2n$ et de robustesse $\frac{n}{7}$. On peut extraire un couplage parfait à partir d'un cycle hamiltonien et ajouter un doublon sur chaque arête de ce couplage. Le réseau résultant est un réseau$-(p = n - k, k, k)$ valide (direct par le critère de coupe) avec $3n$ sommets. Plus généralement, on prouve ainsi que l'on peut, par un choix d'arêtes à bonne distance, construire des réseaux valides pour tout $k \leq n/2$.

6.2.2 Robustesse des graphes aléatoires $4-$réguliers

Le résultat principal prouvé dans cette section est le suivant que la *robustesse maximale* d'un *graphe $4-$régulier*, $\mathcal{R}_{\max}$, est telle que

$$\mathcal{R}_{\max} \geq \frac{n}{14}.$$

Il provient directement du théorème suivant

Théorème 23 *Les graphes aléatoires $4-$réguliers d'ordre n ont une robustesse supérieure à $n/14$ avec une probabilité tendant vers 1 quand n tend vers l'infini.*

Ces graphes aléatoires sont obtenus comme unions de k cycles hamiltoniens [GKW04]. Le résultat provient de leurs propriétés et sa preuve est donnée dans la suite.

Propriété de coupe des graphes de $\mathcal{G}(k,n)$

On appelle $\mathcal{G}(k,n)$ l'ensemble de tous les graphes d'ordre n qui sont une union de k cycles hamiltoniens. Il y a $n!^k$ éléments dans $\mathcal{G}(k,n)$. On note $G_{k,n}$ un de ces éléments. On considère la probabilité uniforme sur cet ensemble. Le graphe $G_{k,n}$ a n sommets, est régulier de degré $2k$ —il peut avoir des arêtes multiples. Avant d'énoncer le résultat principal de cette section (Lemme 12), on s'intéresse à certaines propriétés des permutations aléatoires.

Lemme 11 *Soit π une permutation cyclique avec n éléments. On note $I(s,m)$ le nombre d'ensembles de taille s qui définissent m segments sur le cycle associé à π. $I(s,m)$ ne dépend que de la taille de π et*

$$I(s,m) = \binom{s-1}{m-1}\binom{n-s}{m} + \binom{n-s-1}{m-1}\binom{s}{m} = \binom{s}{m}\binom{n-s}{m}\left(\frac{m}{s} + \frac{m}{n-s}\right).$$

Preuve Le nombre de façon d'écrire un nombre t comme somme ordonnée de p entiers non nul est $cut(t,p) = \binom{t+p-1}{p-1}$. En partant du sommet 0 du cycle, l'ensemble S peut être codé par deux suites de nombres : la liste ordonnée des longueurs des segments dans S et celle des longueurs des segments dans son complémentaire $\overline{S}$.

- Si le sommet 0 appartient à $\overline{S}$, alors S est associée à exactement m segments sur le chemin qui ont tous une longueur d'au moins 1 et $\overline{S}$ est associé à $m+1$ segments qui ont tous une longueur d'au moins 1 à part le dernier qui peut être de longueur 0. $\overline{S}$ peut donc être codé en divisant $|\overline{S}| - m = n - s - m$ en $m+1$ entiers non nuls. Il y a exactement $cut(n-s-m, m+1)$ façons de le faire. Pour S, on l'écrit comme somme ordonnée de m entiers non nuls. Il existe $cut(s-m,m) = \binom{s-1}{m-1}$ possibilités. Le nombre d'ensembles est donc $cut(n-s-m,m+1)\cdot cut(s-m,m) = \binom{s-1}{m-1}\binom{n-s}{m}$.
- Si $0 \in S$, la situation est symétrique. Il suffit de remplacer s par $n-s$. On a donc $cut(s-m,m+1)cut(n-s-m,m) = \binom{s}{m}\binom{n-s-1}{m-1}$.

Lemme 12 *Soit $G_{n,k}$ un graphe tiré aléatoirement dans $\mathcal{G}_{k,n}$. Soit S un ensemble de s sommets de $G_{n,k}$. On a*

$$\mathbb{P}(|\Gamma(S)| = 2x) = \prod_{k_1+\ldots+k_m=x} I(s,k_i) / \binom{n}{s}^k$$

Preuve $G_{n,k}$ est constitué de k cycles $C_1, C_2, \ldots C_k$ indépendants. Soit $m_i(S)$ le nombre d'intervalles que S définit sur C_i. On a $\mathbb{P}[m_i(S) = k_i] = I(s, k_i)$ et

$$|\Gamma(S)| = 2 \sum_{i=1,2,\ldots m} m_i(S).$$

D'où le résultat. Remarquez que la frontière d'un sous-ensemble de $G_{n,k}$ est toujours paire car le graphe est défini comme une union de cycles.

Robustesse des graphes de $\mathcal{G}_{2,n}$

On s'intéresse ici aux graphes de $\mathcal{G}_{2,n}$ formés par deux cycles aléatoires. On montre qu'ils ont *asymptotiquement* une *robustesse linéaire*, de facteur plus grand que $n/14$ —preuve du Théorème 23. La preuve utilise la méthode du premier moment et s'articule en deux temps : on étudie d'abord les petits sous-ensembles (Lemme 13) puis les grands (Lemme 14).

Petits sous-ensembles

Lemme 13 *Pour n grand, tout sous-ensemble de taille $s \leq n/14$ a une frontière de taille au moins s avec probabilité $u > 1/2$.*

Preuve Un sous-ensemble S de taille s est dit *mauvais* si sa frontière est plus petite que s. On appelle $W(n, s)$ la probabilité qu'il existe un sous-ensemble mauvais de taille $s \leq n/14$. On a

$$W(n,s) \leq \binom{n}{s} \sum_{r \leq s/2} \sum_{m_1+m_2=r} I(s, m_1) I(s, m_2) / \binom{n}{s}^2.$$

Comme $I(s, m_1)I(s, m_2) \leq I(s, (m_1 + m_2)/2)$, il vient

$$W(n,s) \leq \sum_{r \leq s/2} r (I(s, r/2))^2 / \binom{n}{s}.$$

Comme $r \leq s/2$, on a $I(s, r) \leq I(s, s/4)$.

$$\begin{array}{rcl} W(n,s) & \leq & W(n,s) \leq \frac{s^2}{8} \left(I(s, s/4)\right)^2 / \binom{n}{s} \\ W(n,s) & \leq & \frac{s^2}{2} \binom{s}{s/4} \binom{n-s/4}{s/4}^2 / \binom{n}{s} \end{array}$$

On utilise maintenant la variante suivante de la formule de Stirling prouvée par Robbins (1955) :

$$n! = \left(\frac{n}{e}\right)^n \sqrt{2\pi n} \cdot e^{\alpha_n}, \quad \text{avec} \quad \frac{1}{12n+1} < \alpha_n < \frac{1}{12n}.$$

D'où

$$W(n,s) \leq X(n,s) := \frac{s^{3s}(n-s)^{3n-3s}}{(s/4)^s (3s/4)^{3s/2} (n - 5s/4)^{2n-5s/2} n^n} \cdot \beta,$$

où $\beta := \frac{16(n-s)\sqrt{s(n-s)}}{3\pi\sqrt{2\pi n}(4n-5s)} \cdot e^{\gamma}$ avec $\gamma := 3\alpha_s + 3\alpha_{n-s} - 4\alpha_{s/4} - 2\alpha_{n-5s/4} - \alpha_n$ et α_i est un réel de l'intervalle $[1/(12i+1), 1/(12i)]$. Posons $n = as$. On obtient

$$X(n,s) = g(a)^s \cdot \beta,$$

avec

$$g(a) := \frac{(a-1)^{3(a-1)}}{1/4 \cdot (3/4)^{3/2} \cdot (a-5/4)^{2a-5/2} \cdot a^a}.$$

La fonction g est une fonction décroissante de a et $g(13.80) =$... Donc $g(14) < 1$.

- Pour $s \in [-\frac{4\log n}{\log(g(14))}, \frac{n}{14}]$, on a

$$X(n,s) \leq C \cdot \frac{1}{n^4}\beta = O\left(\frac{1}{n^2}\right).$$

  Et donc $W(n,s) \leq C \cdot \frac{1}{n^2}$ pour une constante C.
- Pour $s \leq -\frac{4\log n}{\log(g(14))}$, on a $a > C \cdot n/\log n$ pour une constante C. On a donc

$$X(n,s) \leq C(n/s)^{-s/2} \cdot \beta.$$

  Pour $s \geq 4$, on a donc $X(n,s) \leq C \cdot 1/n$. Pour $s = 1,2,3$, $W(n,s) = 0$ car le graphe a une connectivité de 4 avec probabilité 1.

On a donc, pour n grand,

$$\sum_{s \leq n/14} W(n,s) \leq \sum_{-\frac{4\log n}{\log(g(14))} \leq s \leq \frac{n}{14}} X(n,s) + \sum_{s \leq -\frac{4\log n}{\log(g(14))}} X(n,s) < C \cdot n \cdot \frac{1}{n^2} + C \cdot \log(n) \cdot \frac{1}{n} < \frac{1}{2}.$$

Grands sous-ensembles

Lemme 14 *Pour n grand, tout sous-ensemble de taille $n/14 \leq s \leq 13n/14$ a une frontière de taille au moins $n/14$ avec probabilité $u > 1/2$.*

Preuve Un sous-ensemble plus grand que $n/14$ est *mauvais* si sa frontière est plus petite que $n/14$. Soit $W'(n,s)$ la probabilité qu'il existe un sous-ensemble mauvais.

$$\begin{aligned} W'(n,s) &\leq \textstyle\sum_{i \leq n/28} \sum_{k_1+k_2} I(s,k_1)I(s,k_2)/\binom{n}{s} \\ W'(n,s) &\leq \tfrac{(n/14)^2}{8} I(s,n/56)^2/\binom{n}{s}. \end{aligned}$$

En définissant, pour $1/14 \leq a \leq 1 - 1/14$,

$$\delta(a) := \frac{\left(\binom{a \cdot n}{n/56}\binom{(1-a)n}{n/56}\right)^2}{\binom{n}{a \cdot n}}$$

on peut écrire

$$W'(n, an) \leq \frac{1}{8}\left(\frac{n}{14}\right)^2 \delta(a).$$

En utilisant les mêmes méthodes que pour les petits sous-ensembles, on obtient

$$W'(n, an) \leq h(a)^n \beta',$$

où β' est une fonction rationnelle de n et a, et h est défini comme suit

$$h(a) := \frac{a^{3a}(1-a)^{3-3a}}{(1/56)^{1/14}(a-1/56)^{2a-1/28}(1-a-1/56)^{2-2a-1/28}}\beta'.$$

Comme $h(1/14) < 1$ et que h est une fonction décroissante, il vient

$$W'(n, an) \leq h(1/14)^n \beta'.$$

Comme β' est rationnelle, pour n grand, $W'(n, an) \leq 1/2n$. Et donc

$$\sum_{n/14 \leq s \leq 13n/14} W'(n, s) < \frac{1}{2},$$

ce qui prouve le lemme.

6.3 Bornes Inférieures

Dans cette section on partitionne les switchs selon leurs nombres d'entrées et de sorties (représentés respectivement par des flèches $\rightarrow$ et de petites boites $\square$) comme montré dans la Figure 6.3 : S, S_i, S_o, V_i, V_o, D et T. Par exemple, un switch v est dans D si $i(v) = o(v) = 1$. Rappelez vous que nous avons appelé doublon un tel switch et qu'un switch qui n'est pas un doublon est appelé *R-switch*. Des applications directes du critère de coupe montrent qu'il n'existe aucun autre type de switchs dans un réseau valide et que, dès que $\lambda \geq 1$, les switchs de type T ne sont plus autorisés. On rappelle la convention de notation qu'une lettre minuscule indique la cardinalité de l'ensemble désigné par la lettre majuscule correspondante.

Trois équations fondamentales lient ces types de switchs : l'Équation 6.1 (*équation de partition des switchs (switch partition equation)*) qui compte le nombre N de switchs du réseau, l'Équation 6.2 (*équation des entrées (input equation)*) qui compte le nombre d'entrées et l'Équation 6.3 (*équation des sorties (output equation)*) qui compte le nombre de sorties.

$$N = s + s_i + s_o + d + v_i + v_o + t \tag{6.1}$$

$$p + \lambda = s_i + 2v_i + d + t \tag{6.2}$$

$$p + k = s_o + 2v_o + d + 2t \tag{6.3}$$

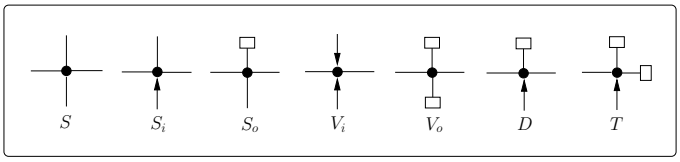


FIG. 6.3 – Types de switchs

Dans la Section 6.3.1, nous prouvons un théorème préliminaire fondamental, le Théorème 5. Son point principal est que $N(p, \lambda, k) \geq \frac{3}{2}n + \frac{d}{2} - \varepsilon(n)$ (où $\varepsilon(n)$ tend vers zéro quand n tend vers l'infini). Des applications directes de ce théorème donnent des bornes inférieures pour les réseaux généraux et simplifiés (Théorèmes 6, 7). En Sections 6.3.4 et 6.3.5, nous avons réussi à obtenir une meilleure borne de $n + \frac{2}{3}n - \varepsilon(n)$ dans deux cas : quand λ tends vers l'infini (Théorème 8) et quand certains types de switchs ne sont pas permis dans les réseaux (Théorème 9). En Section 10, nous montrons que la borne peut même être montée à $n + \frac{3}{4}n - \varepsilon(n)$ pour d'autres réseaux en utilisant des arguments de majorité sur des graphes bi-partie : étant donné un graphe bi-partie (A, B, E) avec B partitionné en deux sous-ensembles B_0 et B_1, si, pour tout ensemble X de petite taille ($\leq k$) nous avons $b_0(X) \geq b_1(X)$, alors nous avons pour le réseau tout entier $b_o \geq 2b_1 + \epsilon(k)$.

6.3.1 Théorème Préliminaire Fondamental

L'idée est ici de découper le réseau en grands et petits composants. On considère chaque ensemble constitué d'un grand composant et des petits qui lui sont adjacents. On utilise alors le critère de coupe pour obtenir des équations qui lient les différents types de switchs à l'intérieur, nous donnant une information locale sur les configurations possibles des switchs. Ensuite, on utilise l'existence d'une quasi-partition du réseau 2 pour obtenir une borne pour le réseau entier.

Definition 3 (q-quasi-partition, voir [DHMP06]) *Soit $G = (V, E)$ un graphe et q un entier positif. Une* q-quasi-partition de G *est une famille $Q = \{A_1, A_2, \ldots, A_m\}$ de sous-ensembles de V, telle que :*

(i) pour tout $1 \leq i \leq m$, le sous-graphe $G[A_i]$ induit par A_i est connexe ;

(ii) pour tout $1 \leq i \leq m$, $\frac{q}{2} \leq |A_i| \leq q$;

(iii) $V = \bigcup_{i=1}^{m} A_i$ et $\sum_{i=1}^{m} |A_i| \leq |V| + |\{A_i, |A_i| > \frac{2q}{3}\}|$.

Lemme 2 *[[DHMP06]] Soit q un entier positif et G un graphe connexe d'ordre au moins $\frac{q}{2}$. Alors G admet une q-quasi-partition.*

Remarque 1 *Si G a plusieurs composants connexes de taille au moins $\frac{q}{2}$, en appliquant le lemme à chaque composant et en utilisant l'additivité des deux côtés de l'Équation (ii), on obtient une $q-$quasi-partition de G.*

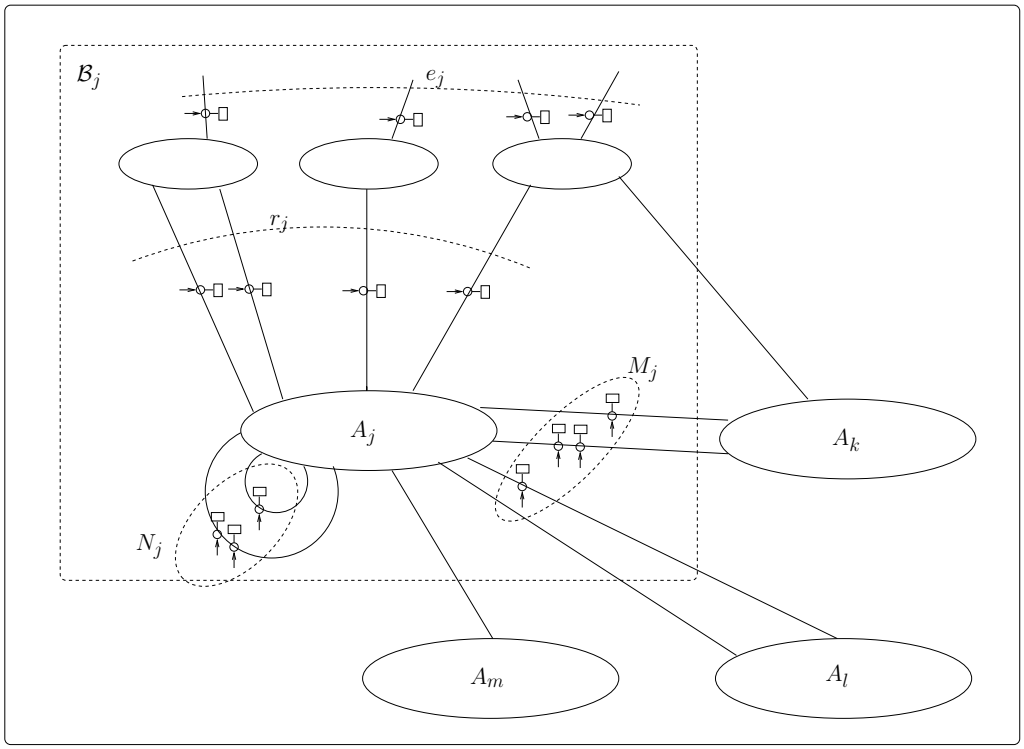


FIG. 6.4 – Schéma de la preuve du Théorème 5

Remarque 2 *Soit Q une quasi-partition de G (voir le Lemme 2). Soit $t = |\{A_i, |A_i| > \frac{2q}{3}\}|$ et $v = |V|$. Alors*

1. $m \leq \frac{2(v+t)}{q}$ *et*
2. $t \leq \frac{v}{\frac{2q}{3}-1}$.

Definition 4 [H-composants, petits et grands H-composants, H-composants adjacents] *Soit un réseau$-(p, \lambda, k)$ et son graphe associé $\mathcal{R}$. Considérons H le sous-graphe de $\mathcal{R}$ qui ne contient que les arêtes de E_0. Un H-*composant *de $\mathcal{R}$ est un composant connexe de H. Un H-composant est dit* grand *(respectivement* petit*) s'il a plus que (resp. strictement moins que) q switchs, avec q le plus grand entier qui satisfait $2(q + (2q + 2)q) + 2 \leq k - 1$. Remarquez que $q \sim \frac{\sqrt{k}}{2}$. Deux H-composants C_1 et C_2 sont dits* adjacents *s'il existe une arête de $\mathcal{R}$ reliant un $R-$switch dans C_1 and un autre dans C_2.*

Proposition 10 *1. Un petit H-composant n'a pas d'arête sortante de type E_2.*

2. *Un petit H-composant n'a pas d'entrée à l'intérieur.*
3. *Deux petits H-composants ne peuvent être adjacents.*

Preuve Soit C un petit H-composant. Nous utilisons le critère de coupe de la Section 6.1.3 à l'ensemble $\tilde{C}$ de $\mathcal{N}$ obtenu à partir de C en ajoutant dans $\mathcal{N}$ les doublons des arêtes de type E_1 et E_2 incidentes aux $R-$switches de C.

Comme $k \leq p$ et $|C| \leq q \approx \frac{\sqrt{k}}{2}$, $o(C) \leq \frac{k}{2} \leq p$, le critère de coupe se réduit à

$$\delta(\tilde{C}) \geq i(\tilde{C}).$$

Soit e_1 (res. e_2) le nombre d'arêtes sortantes de type E_1 (resp. E_2) incidentes aux $R-$switches de C.

- Par définition d'un petit composant $\delta(\tilde{C}) = e_1+e_2$. Nous avons $i(\tilde{C}) \geq e_1+2e_2+i(C)$. Ainsi par le critère de coupe, on obtient $e_2 = 0$, ce qui prouve 1), et $i(C) = 0$, ce qui prouve 2).
- Soit C' un autre petit $H-$composant. Si C et C' sont reliés par $f \geq 1$ arêtes, alors soit $W = \tilde{C} \cup \tilde{C'}$, nous avons $i(W) \geq e_1 + e'_1 - f$ et $\delta(W) \leq e_1 + e'_1 - 2f$ et donc $\delta(W) < i(W)$ ce qui donne une contradiction.

Théorème 5 *Dans un réseau $\mathcal{N}$ valide avec $k \leq \frac{n}{2}$, nous avons*

$$N(p,\lambda,k) \geq \left(\frac{3}{2}n - (k-\lambda)\right)\left(1 - \frac{7}{2\sqrt{k}} + O\left(\frac{1}{k}\right)\right) + \frac{d}{2}\left(1 + \frac{7}{2\sqrt{k}} + O\left(\frac{1}{k}\right)\right)$$

où $n = p + k$.

Preuve Selon le Lemme 2 et la Remarque 1, l'union des grands $H-$composants de $\mathcal{R}$ admet une q-quasi-partition $Q = \{A_1, \ldots, A_m\}$. Donc chaque A_j est connexe et de taille $\frac{q}{2} \leq |A_j| \leq q$.

Entre toutes les arêtes avec doublons, on distingue :
- les arêtes de type R reliant un A_j et un petit $H-$composant,
- celles de type M entre deux A_j et A_k distincts et
- celles de type N à l'intérieur d'un A_j.

Nous introduisons l'ensemble $\mathcal{B}_j$ comprenant A_j et tous les petits $H-$composants qui lui sont adjacents. Soit
- r_j le nombre d'arêtes reliant A_j et ses petits H-composants,
- e_j le nombre d'arêtes sortantes des petits composants de $\mathcal{B}_j$,
- M_j le nombre de doublons sur les arêtes de type M et

- N_j le nombre de doublons sur les arêtes de type N.

Comme deux petits composants ne peuvent pas être adjacents, remarquez que toutes les arêtes r_j et e_j sont de type R.

Appliquons maintenant le critère de coupe à $\mathcal{B}_j$

$$\delta(\mathcal{B}_j) + o(\mathcal{B}_j) - \min(k, o(\mathcal{B}_j)) - \min(i(\mathcal{B}_j), p) \geq 0.$$

Comme $k \leq \frac{n}{2}$, nous avons $i(\mathcal{B}_j) \leq p$. Nous montrons d'abord que $o(\mathcal{B}_j) < k$. Pour $o(\mathcal{B}_j) \geq k$, le critère de coupe se réduit à $\delta(\mathcal{B}_j) \geq k$. De plus A_j est connexe et de taille plus petite que q. Il a donc au plus $2q+2$ arêtes sortantes et le nombre de petits H-composants de $\mathcal{B}_j$ est au plus $2q+2$. Comme la taille d'un petit $H-$composant est plus petite que q, le nombre de sommets de $\mathcal{B}_j$ est au plus $q + (2q+2)q$. Donc le nombre d'arêtes sortantes $\delta(\mathcal{B}_j)$ est au plus $2(q + (2q+2)q) + 2$. Le choix de q donne

$$k \leq \delta(\mathcal{B}_j) \leq 2(q + (2q+2)q) + 2 \leq k - 1,$$

une contradiction.

Donc $o(\mathcal{B}_j) < k$. Le critère de coupe est maintenant équivalent à $\delta(\mathcal{B}_j) \geq i(\mathcal{B}_j)$. Pour des raisons de clarté, la référence à A_j est omise dans la prochaine équation et ainsi, par exemple, $e_j = e_j(A_j)$. Nous avons $\delta(\mathcal{B}_j) = \delta'(\mathcal{B}_j) + e_j$ où δ' est le nombre d'arêtes sortantes de $\mathcal{B}_j$ incidentes à A_j. En utilisant les définitions des types de switchs, nous avons

$$\delta' = 4|A_j| - 2e(A_j) - r_j - 2N_j - s_i - s_o - 2v_i - 2v_o - 3t$$

Comme A_j est connexe, $e(A_j) \geq |A_j| - 1$, donc :

$$\delta' \leq 2|A_j| + 2 - r_j - 2N_j - s_i - s_o - 2v_i - 2v_o - 3t.$$

Pour le nombre d'entrées dans $\mathcal{B}_j$, nous avons :

$$i(\mathcal{B}_j) = e_j + r_j + M_j + N_j + s_i + 2v_i + t.$$

Le critère implique alors que :

$$2|A_j| + 2 \geq 2s_i + s_o + 4v_i + 2v_o + 4t + 3N_j + 2r_j + M_j \tag{6.4}$$

Le nombre total de doublons est d. Tous les doublons sont de types R, M ou N. Les doublons de M sont comptés de deux façons différentes A_j et A_k. D'où $\sum_{j=1}^{m} M_j + 3N_j + 2r_j \geq 2d$. Ainsi, en prenant la somme sur j de toutes les Équations 6.4, on obtient

$$2\sum_{j=1}^{m} |A_j| + 2m \geq 2s_i + s_o + 4v_i + 2v_o + 4t + 2d.$$

Les équations sur les entrées et sur les sorties (Équations 6.2 et 6.3) donnent $2s_i + 4v_i + 2t + 2d = 2n - 2(k - \lambda)$ et $s_o + 2v_o + 2t = n - d$. Donc

$$2\sum_{j=1}^{m} |A_j| + 2m \geq 3n - d - 2(k - \lambda) \tag{6.5}$$

La famille $\{A_j\}$ forme une quasi-partition de $\mathcal{R}$. Soit $t := |\{A_j, |A_j| > \frac{2q}{3}\}|$ et v le nombre de sommets de $\mathcal{R}$. Alors par la Remarque 2, nous avons

$$m \leq \frac{2(v+t)}{q}$$

$$t \leq \frac{v}{\frac{2q}{3} - 1}$$

Par définition d'une quasi-partition :

$$\sum_{i=j}^{m} |A_j| \leq v + |\{A_j, |A_j| > \frac{2q}{3}\}| = v + t.$$

En utilisant toutes ces équations avec l'Équation 6.5, cela donne

$$2v\left(1 + \frac{1}{\frac{2q}{3} - 1}\right) \geq \frac{q}{q+2}(3n - d - 2(k - \lambda))$$

$$2v \geq n\frac{q}{q+2}\frac{2q-3}{2q} = (3n - d - 2(k - \lambda))\left(1 - \frac{7}{2q+4}\right).$$

En utilisant $N \geq v + d$, on obtient

$$N \geq \left(\frac{3}{2}n - (k - \lambda)\right)\left(1 - \frac{7}{2q+4}\right) + \frac{d}{2}\left(1 + \frac{7}{2q+4}\right)$$

Enfin

$$N \geq \left(\frac{3}{2}n - (k - \lambda)\right)\left(1 - \frac{7}{2\sqrt{k}} + O\left(\frac{1}{k}\right)\right) + \frac{d}{2}\left(1 + \frac{7}{2\sqrt{k}} + O\left(\frac{1}{k}\right)\right).$$

6.3.2 Problème de Conception : $\lambda = 0$ - Borne Inférieure en $n + n/2$

Théorème 6 *Dans un réseau valide, $\mathcal{R}$, quand $k \to \infty$ avec $k \leq \frac{n}{2}$, nous avons*

$$N(p, \lambda, k) \geq n + \frac{n}{2} + O(\frac{n}{\sqrt{k}}).$$

Preuve La preuve est déduite directement du Théorème 5.

On obtient une borne serrée dans ce cas des réseaux avec $\lambda = 0$ (voir la borne supérieure en Section 6.1.6).

6.3.3 Problème de Conception Simplifié : $\lambda \geq 1$ - Borne Inférieure en $2n$

Théorème 7 *Dans le cas simplifié (où les switchs de types S_i, S_o, V_i, V_o sont interdits), quand $k \to \infty$ avec $k \leq \frac{n}{2}$, nous avons*

$$N(p, \lambda, k) \geq 2n + O(\frac{n}{\sqrt{k}}).$$

où $n = p + k$. Plus précisément

$$\begin{aligned} N(p, \lambda, k) \geq 2n &\left(1 - \frac{7}{4\sqrt{k}} + O\left(\frac{1}{k}\right)\right) \\ &+ \frac{3}{2}(k - \lambda)\left(1 + \frac{7}{6\sqrt{k}} + O\left(\frac{1}{k}\right)\right). \end{aligned}$$

Preuve La preuve se déduit du Théorème 5 et de $d = n - (k - \lambda)$ (pour le cas simplifié).

6.3.4 Problème de Conception : $\lambda \geq 1$ - $\lambda \to \infty$ - Borne Inférieure en $n + \frac{2}{3}n$

Nous montrons ici que, quand $\lambda \to \infty$, $N(p, \lambda, k) \geq n + \frac{2}{3}n + O(\frac{n}{\sqrt{\lambda}})$ (Théorème 8). Tout d'abord, nous donnons une borne sur le nombre de switchs de types V_i et V_o en utilisant la remarque suivante :

Remarque 3 *Quand $\lambda = 0$, les switchs de type V_i ne sont pas présents dans un réseau$-(p, \lambda, k)$ valide minimum. Comme montré en Figure 6.5, ils peuvent être enlevés pour construire un nouveau réseau$-(p, \lambda, k)$ valide avec v_i switchs en moins.*

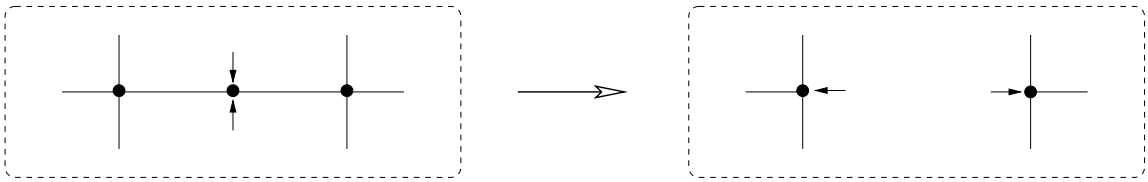


FIG. 6.5 – Quand $\lambda = 0$, les switchs de type V_i peuvent être enlevés.

Lemme 3 *Quand $\lambda \to \infty$ et $k \to \infty$,*

$$\begin{cases} v_i & \leq & N - \frac{3}{2}n - \frac{d}{2} - \frac{k-\lambda}{2} + \lambda O(\frac{n}{\sqrt{k}}) \\ v_o & \leq & N - \frac{3}{2}(n-k+\lambda) - \frac{d}{2} + k - \frac{\lambda-k}{2} + O(\frac{n}{\sqrt{\lambda}}) \end{cases}$$

Preuve Imaginons que nous ayons une réseau$-(p, \lambda, k)$ valide avec N switchs et v_i sommets de V_i. On obtient un réseau$-(p, 0, k)$ valide en enlevant λ entrées choisies de n'importe quelle manière. Ce nouveau réseau a au moins $v_i - \lambda$ switchs de type V_i. Par la Remarque 3, on peut enlever ces derniers et obtenir un réseau$-(p, 0, k)$ valide avec $N - v_i + \lambda$ switchs. Le Théorème 5 donne

$$N - v_i + \lambda \geq \frac{3}{2}n + \frac{d}{2} + \frac{k-\lambda}{2} + O(\frac{n}{\sqrt{k}}).$$

D'où le résultat. Un argument de symétrie (dans le sens où l'échange des entrées et des sorties donne un réseau$-(p, k, \lambda)$ valide) donne la seconde équation.

Théorème 8 *Quand $\lambda \to \infty$ et $k \to \infty$*

$$N(p, \lambda, k) \geq n + \frac{2}{3}n + O(\frac{n}{\sqrt{\lambda}}).$$

Preuve L'équation de partition des switchs et les deux Équations 6.2, 6.3 donnent ici

$$N = 2n - (k - \lambda) - v_i - v_o - d + s.$$

Le Lemme 3 donne

$$\begin{aligned} N &\geq 2n - (k-\lambda) - d + s - 2N + 3n + d \\ &\quad + O(\frac{n}{\sqrt{k}}) + O(\frac{n}{\sqrt{\lambda}}) - \frac{3}{2}(k-\lambda) - \lambda - k \\ N &\geq \frac{5}{3}n + O(\frac{n}{\sqrt{k}}) + O(\frac{n}{\sqrt{\lambda}}) - \frac{5}{2}k + \frac{\lambda}{2}. \end{aligned}$$

6.3.5 Problème de Conception : $\lambda \geq 1$ - $v_i = v_o = 0$ - Borne Inférieure en $n + \frac{2}{3}n$

Remarque 4 *Dès que $\lambda \geq 1$, les switchs de type W ($i(v) = 1$ et $o(v) = 2$) sont interdits.*

Preuve Preuve directe par le critère de coupe ($\varepsilon_o(v) = 1 + 1 - 1 - 2 = -1$).

Quand $v_i = v_o = 0$, l'équation sur les entrées (Équation 6.2) devient $n = d + s_i$, l'équation sur les sorties (Équation 6.3) devient $n = d + s_o$. D'où $s_i = s_o$ et l'équation de répartition des switchs (Equation 6.1) donne

$$N = 2n - d. \tag{6.6}$$

Théorème 9 *Quand $\lambda \geq 1$ et qu'aucuns switchs de types V_i et V_o ne sont autorisés, et quand $k \to \infty$ avec $k \leq \frac{n}{2}$, nous avons*

$$N(p, \lambda, k) \geq n + \frac{2}{3}n + O(\frac{n}{\sqrt{k}}).$$

Preuve

Le Théorème 5 donne

$$2n - d \geq \frac{3}{2}n + \frac{d}{2} + O(\frac{n}{\sqrt{k}})$$

$$\frac{n}{3} \geq d + O(\frac{n}{\sqrt{k}}).$$

L'Équation 6.6 donne

$$N \geq \frac{5}{3}n + O(\frac{n}{\sqrt{k}}).$$

6.3.6 Problème de Conception : $\lambda \geq 1$ - $s = v_i = v_o = 0$, $G = (I, O, E)$-bipartite - Borne Inférieure en $n + \frac{3}{4}n$

Théorème 10 *Soit $\mathcal{N}$ un réseau de N switchs de graphe associé $\mathcal{R} = (V = A \cup B, E)$ tel que $\mathcal{R}$ est bi-partie, avec tous les sommets de A de type S_o et tous les sommets de B de type S_i. Quand $\lambda \geq 1$ et que les switchs de types S, V_i et V_o sont permis, on obtient*

$$N \geq n + \frac{3}{4}n + O(\frac{n}{\sqrt{k}}).$$

Preuve Étant donné un graphe $R = (V = A \cup B, E)$ ternaire bi-partie tel que tous les sommets de A ont une sortie et tous les sommets de B ont une entrée. On partitionne les sommets de B en deux ensembles B_1, et B_0. Un sommet de B_1 est adjacent à une arête de type E_1 ou E_2, au contraire des sommets de B_0.

Pour un sous-ensemble $X \subset A$, nous utilisons les notations suivantes : $B_i(X) = \Gamma(X) \cap B_i, i = 0, 1$ et $b_i = |B_i|$.

Soit X un sous-ensemble de A tel que $F(X) = X \cup \Gamma(X)$ est connexe et de plus $6|X| \leq k$. Nous avons donc moins de k sorties. Pour remplir le critère de coupe pour les petits sous-ensembles (moins que k sorties), il faut

$$b_0(X) \geq b_1(X) - 3 + 3c_{F(X)} \tag{6.7}$$

où $c_{F(X)}$ est l'ensemble d'arêtes de retour du sous-graphe induit par $F(X)$.

Notre but est de prouver que cela ne peut arriver que si $b_0 + O(\frac{1}{\sqrt{k}}) \geq 2b_1$.

Soit Y un ensemble de y sommets de B_0 tel que $Y \cup \Gamma(Y)$ est connexe. On a $|\Gamma(Y)| = 2y + 1 - c_{F(Y)}$. Il y a $3y + 3 - 3c$ arêtes sortant de $F(Y)$, divisées en αy vers des sommets de type B_0 et $3y + 3 - 3c - \alpha y$ vers des sommets de type B_1. On considère dans G le sous-graphe connexe induit par $Z = F(Y) \cup (\Gamma(F(Y) \cap B_1)$. On a $(6 - \alpha)y + 3 - 3c_{F(Y)}$ arêtes à l'intérieur de Z. Le nombre de sommets dans $\Gamma(F(Y) \cap B_1$ est donc $(6-\alpha)y+3-3c_{F(Y)} - y - (2y+1-c_{F(Y)}) - c_Z = (3-\alpha)y+2-2c_{F(Y)} - c_Z$. Si on pose $X = \Gamma(Y)$ dans l'Équation 6.7, on obtient :

$$\begin{array}{rcl} (\alpha+1)y & \geq & b_0(X) \geq b_1(X) - 3 + 3c_{F(X)} \\ & \geq & (3-\alpha)y + 2 - 2c_{F(Y)} - c_Z - 3 + 3c_{F(X)} \\ & \geq & (3-\alpha)y - 1 - 2c_{F(Y)} + 2c_{F(X)} \end{array}$$

où on utilise dans la dernière inégalité le fait que $c_Z \leq c_{F(X)}$; parce que $c_{F(Y)} \leq c_{F(X)}$, on a

$$\alpha \geq 1 - \frac{1}{2y}.$$

On considère tous les composants connexes du graphe induit par $B_0 \cup A$ et on prend une q-quasi-partition des grands composants pour un q, $q = O(k)$ tel que tous les composants sont de la forme $F(Y)$ pour un Y sous-ensemble de B_0. On compte maintenant les arêtes reliant B_0 et B_1. Pour un composant D de la quasi-partition avec y sommets de B_0, on trouve au moins $(3 + \alpha)y \geq 4y - \frac{1}{2}$ arêtes vers B_0 et au plus $2y + \frac{7}{2}$ vers B_1 avec une extrémité dans D. Globalement, si m est le nombre de composants, on obtient (à quelques ajustements près dus à des arguments de quasi-partition) $4|A| - \frac{m}{2} = 3b_0$ arêtes vers B_0 et $2A + \frac{7m}{2} = 3b_1$ arêtes vers B_1. D'où $b_0 \geq 2b_1 + O(\frac{1}{\sqrt{k}})$.

6.4 Conclusion

Dans ce chapitre nous avons proposé des constructions de réseaux$-(p, \lambda, k)$ et donné des bornes inférieures pour leur taille. Le problème de conception apparaît dirigé par deux

contraintes : une locale qui interdit de petits motifs et un autre liée à une propriété globale d'expansion du réseau. Cela nous a amené à définir un paramètre d'expansion d'un graphe : la $\alpha-$robustesse. Ce paramètre est une généralisation de l'expansion en arête commune. En utilisant des graphes de $2-$robustesse égal à $\Theta(\log n)$ nous avons construit des réseaux simplifiés quasiment optimaux. De façon similaire, quand $k \leq \frac{n}{7}$, en utilisant des graphes de grande $1-$robustesse, on a proposé de bons réseaux. Malgré notre bonne compréhension du problème dans le cas des petites valeurs de k ($\leq \frac{n}{7}$), de nombreuses questions restent ouvertes quand k est grand. Par exemple :

- Pour un α fixé, trouver des constructions explicites de graphes ayant une grande α-robustesse.
- Existe-t-il une construction explicite donnant des graphes robustes à la manière du nouveau produit Zig-Zag de [RVW02] ? Est-ce que le produit Zig-Zag conserve la robustesse ?
- Quelle est la bi-section maximale d'un graphe $4-$régulier ? Dans [MP01], il est prouvé que la longueur de bi-section maximale d'un graphe 4-régulier d'ordre n est au plus $\frac{2n}{5}$. Cela montre que l'α-robustesse est $< \frac{n}{2}$ pour $\alpha > \frac{4}{5}$. Est-il possible d'obtenir des résultats similaires pour de plus grandes valeurs de α, assurant, par exemple, une robustesse maximale de $\frac{n}{3}$?
- Quel est le nombre minimal de switchs d'un *n-super-sélecteur* (voir [Hav06b, Sch06] pour des constructions) ?

Remerciements. Les auteurs voudraient remercier Jean-Claude Bermond, Stéphan Thomassé et Frédéric Havet pour des discussions utiles et stimulantes.

.1 La Méthode de Laplace pour les Sommes

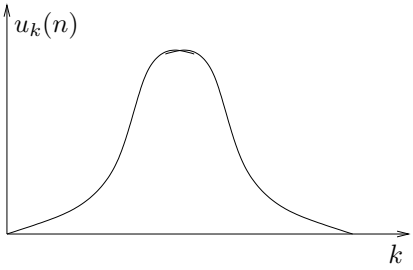
Dans quels cas l'utiliser ?

Pour calculer une somme de la forme

$$\sum_k u_k(n)$$

avec n grand.

Il faut aussi que les $u_k(n)$ aient une forme de cloche (souvent normale).



n fixé mais grand

Étapes de la méthode.

1. Déterminer un équivalent de $u_{k_{max}}(n)$, la valeur maximale des $u_k(n)$.
2. Déterminer un équivalent de $\frac{u_{k_{max}+l}(n)}{u_{k_{max}}(n)}$.
3. Déterminer un *domaine central* et prouver que la somme sur la *périphérie* est nulle.
4. Calculer la somme en l'approximant par une intégralle.

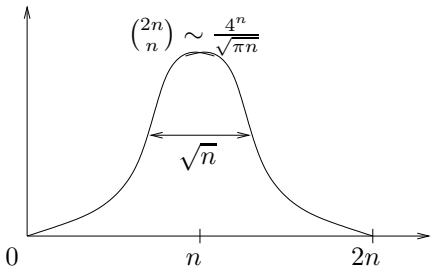
Un exemple.

On veut calculer S_n, la somme des binomiales. Plus précisément, on veut retrouver le résultat connu.

$$S_n = \sum_{k=0}^{2n} \binom{2n}{k}.$$

1. On sait que $\binom{2n}{k}$ est maximum pour $k = n$ et

$$\binom{2n}{n} = \frac{(2n)!}{(n!)^2} \sim \frac{(2n)^{2n}}{(n^n)^2} \frac{\sqrt{2\pi 2n}}{(\sqrt{2\pi n})^2} \sim \frac{4^n}{\sqrt{\pi n}}$$



2. l fixé :

$$\begin{aligned}\frac{\binom{2n}{n+l}}{\binom{2n}{n}} &= \frac{n!n!}{(n+l)!(n-l)!} = \frac{n(n-1)\cdots(n-l+1)}{(n+l)(n+l-1)\cdots(n+1)} \\ &= \frac{1(1-1/n)\cdots(1-(l-1)/n)}{(1+1/n)\cdots(1+l/n)} = \frac{\exp(\ln(1-1/n)+)\cdots\ln(1-(l-1)/n)}{\exp(\ln(1+1/n)+\cdots+\ln(1+l/n))} \\ &= \frac{\exp(-1/n-\cdots-(l-1)/n)}{\exp(1/n+\cdots+l/n)} = \frac{\exp(-1/nl(l-1)/2)}{\exp(1/nl(l+1)/2)} \\ &= \exp(-\frac{l^2}{n})\end{aligned}$$

vrai si $l << n$ donc on va définir un domaine central.

3.
$$\frac{S_n}{\binom{2n}{n}} = \sum_{l=-n}^{n} \exp(-\frac{l^2}{n})$$

On définit un domaine central DC $-\sqrt{n}\ln n \leq l \leq \sqrt{n}\ln n$. On a

$$\frac{S_n}{\binom{2n}{n}} = \sum_{DC} + \sum_{P}$$

et on va montrer que $\sum_P \underset{n\to\infty}{\to} 0$. Si $l \notin DC$ [maximum à la frontière en $\sqrt{n}\ln n$]

$$\exp(-\frac{l^2}{n}) \leq \exp(-\frac{(\sqrt{n}\ln n)^2}{n}) = \exp(-(\ln n)^2)$$

d'où

$$\sum_P \exp(-\frac{l^2}{n}) \leq 2n \cdot \exp(-(\ln n)^2) \underset{n\to\infty}{\to} 0.$$

On a

$$\frac{S_n}{\binom{2n}{n}} = \sum_{DC} \exp(-\frac{l^2}{n}).$$

4. On va utiliser la propriété qu'une somme de Riemann tend vers une intégrale quand son pas tend vers 0.

$$\sum_{k \text{ par pas de } h \text{ petits}} f(k) \underset{h\to 0}{\sim} \frac{1}{h}\int f(x)dx$$

Ainsi si l'on pose $l = x\sqrt{n}$, $\sum_{l \in DC \text{ pas de } 1} \exp(-\frac{l^2}{n})$ devient $\sum_{x \text{ par pas de } \frac{1}{\sqrt{n}}} \exp(-x^2)$. d'où

$$\frac{S_{2n}}{\binom{2n}{n}} \underset{n\to\infty}{\to} \frac{1}{\frac{1}{\sqrt{n}}} \int_{\sqrt{n}-\ln n}^{\sqrt{n}+\ln n} e^{-x^2} dx$$

d'où

$$\frac{S_{2n}}{\binom{2n}{n}} \underset{n\to\infty}{\to} \sqrt{n}\sqrt{\pi}$$

$$S_{2n} \underset{n\to\infty}{\to} \sqrt{n}\sqrt{\pi}\frac{4^n}{\sqrt{\pi n}} = 4^n.$$

Bibliographie

[ABG+06] O. Amini, J.-C. Bermond, F. Giroire, F. Huc, and S. Pérennes. Design of minimal fault tolerant networks : Asymptotic bounds. In *Huitièmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (AlgoTel'06)*, pages 37–40, Trégastel, France, May 2006.

[AC03] N. Alon and M. Capalbo. Smaller explicit superconcentrators. *Proc. of the Fourteenth Annual ACM-SIAM SODA*, pages 340–346, 2003.

[AGHP06a] O. Amini, F. Giroire, F. Huc, and S. Pérennes. Minimal selectors and fault tolerant networks. In *In preparation*, 2006.

[AGHP06b] O. Amini, F. Giroire, F. Huc, and S. Pérennes. Minimal selectors and fault tolerant networks. Research report, INRIA Research Report HAL-00082015, July 2006.

[AGM87] N. Alon, Z. Galil, and V. D. Milman. Better expanders and superconcentrators. *J. Algorithms*, 8 :337–347, 1987.

[AHK99] N. Alon, P. Hamburger, and A. V. Kostochka. Regular honest graphs, isoperimetric numbers, and bisection of weighted graphs. *European Journal of Combinatorics*, 1999.

[Alo93] N. Alon. On the edge-expansion of graphs. *Combinatroics, Probability and Computing*, 11 :1–10, 1993.

[AM84] N. Alon and V. D. Milman. Eigenvalues, expanders and superconcentrators. *Proc. 25th Annual Symp. on Foundations of Computer Science (FOCS), Singer Island, Florida,*, IEEE(1984) :320–322, 1984.

[BD99] B. Beauquier and E. Darrot. Arbitrary size Waksman networks. In *AlgoTel*, pages 95–100, 1999.

[BD02] B. Beauquier and E. Darrot. Arbitrary size Waksman networks and their vulnerability. *Parallel Processing Letters*, 3-4 :287–296, 2002.

[BDD02] J-C. Bermond, E. Darrot, and O. Delmas. Design of fault tolerant onboardnetworks in satellites. *Networks*, 40 :202–207, 2002.

[BDH+03] J-C. Bermond, O. Delmas, F. Havet, M. Montassier, and S. Perennes. Réseaux de télécommunications minimaux embarqués tolérants aux pannes. In *Algotel*, pages 27–32, 2003.

[BGP06] J-C. Bermond, F. Giroire, and S. Pérennes. Design of minimal fault tolerant networks : Constructions. In *In preparation*, 2006.

[BHT06] J-C. Bermond, F. Havet, and D. Tóth. Fault tolerant on board networks with priorities. *Networks*, 47(1) :9–25, 2006.

[BKP+81] M. Blum, R. Karp, C. Papadimitriou, O. Vornberger, and M. Yannakakis. The complexity of testing whether a graph is a superconcentrator. *Inf. Proc. Letters*, 13 :164–167, 1981.

[Bol88] B. Bollobás. The isoperimetric number of random regular graphs. *European Journal of Combinatroics*, 9, no. 3 :241–244, 1988.

[BPT] J.-C. Bermond, S. Perennes, and Cs. D. Tóth. Fault tolerant on-board networks with priorities. *Manuscrit*.

[BYJK+02] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *RANDOM '02 : Proceedings of the 6th International Workshop on Randomization and Approximation Techniques*, pages 1–10. Springer-Verlag, 2002.

[Chu97] F. R. K. Chung. *Spectral Graph Theory*, volume No. 92. CBMS, 1997.

[CLKB04] Jeffrey Considine, Feifei Li, George Kollios, and John Byers. Approximate aggregation techniques for sensor databases. In *ICDE '04 : Proceedings of the 20th International Conference on Data Engineering*, page 449, Washington, DC, USA, 2004. IEEE Computer Society.

[CRVW02] M. Capalbo, O. Reingold, S. Vadhan, and A. Wigderson. Randomness conductors and constant-degree lossless expanders. In *STOCS*, pages 659 – 668, 2002.

[DF03] M. Durand and P. Flajolet. Loglog counting of large cardinalities. In *Proceedings of the European Symposium on Algorithms*, 2003.

[DGIM02] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. *SIAM J. Comput.*, 31(6) :1794–1813, 2002.

[DH00] D. Z. Du and F. K. Hwang, editors. *Notes on the Complexity of Switching Networks*, volume 42. Kluwer Academic Publishers, 2000.

[DHMP05] O. Delmas, F. Havet, M. Montassier, and S. Pérennes. Design of fault tolerant on-board networks. *Research report RR 1345-05, Laboratoire bordelais de recherche en informatique (LaBRI), Unité Mixte de Rec herche CNRS (UMR 5800)*, 2005.

[DHMP06] O. Delmas, F. Havet, M. Montassier, and S. Pérennes. Design of fault tolerant on-board networks. *Research report INRIA RR 5866, submitted*, 2006.

[Dur04] Marianne Durand. *Combinatoire analytique et algorithmique des ensembles de données*. PhD thesis, École polytechnique, 2004.

[EVF03] C. Estan, G. Varghese, and M. Fisk. Bitmap algorithms for counting active flows on high speed links. In *Technical Report CS2003-0738*, UCSE, Mars 2003.

[FDL+01] C. Fraleigh, C. Diot, B. Lyles, S. Moon, P. Owezarski, K. Papagiannaki, and F. Tobagi. Design and Deployment of a Passive Monitoring Infrastructure. In *Passive and Active Measurement Workshop*, Amsterdam, April 2001.

[FG07] E. Fusy and F. Giroire. Estimating the number of active flows in a data stream over a sliding window. In *2007 SIAM Workshop on Analytic Algorithms and Combinatorics (ANALCO07)*, 2007.

[Fla85] Philippe Flajolet. *Approximate Counting : A detailed analysis.*, pages 113–134. BIT 25, 1985.

[Fla97] Philippe Flajolet. Adaptive sampling. In M. Hazewinkel, editor, *Encyclopaedia of Mathematics*, volume Supplement I, page 28. Kluwer Academic Publishers, Dordrecht, 1997.

[FM83] P. Flajolet and P. N. Martin. Probabilistic counting. In *Proceedings of the 24th Annual Symposium on Foundations of Computer Science*, pages 76–82. IEEE Computer Society Press, 1983.

[FS] P. Flajolet and R. Sedgewick. Analytic combinatorics. Available at `http ://algo.inria.fr/flajolet/Publications/book051001.pdf`. Preliminary version of the forthcoming book.

[Gib01] Phillip B. Gibbons. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In *The VLDB Journal*, pages 541–550, 2001.

[Gir03] Frédéric Giroire. Etude de problèmes combinatoires liés à l'analyse du génome : Séquençage et polymorphisme, 2003. Mémoire de DEA, Université Paris VI.

[Gir05] Frédéric Giroire. Order statistics and estimating cardinalities of massive data sets. In Conrado Martínez, editor, *2005 International Conference on Analysis of Algorithms*, volume AD of *DMTCS Proceedings*, pages 157–166. Discrete Mathematics and Theoretical Computer Science, 2005.

[Gir06a] F. Giroire. Directions to use probabilistic algorithms for cardinality for dna analysis. In *Journées Ouvertes Biologie Informatique Mathématiques (JOBIM 2006)*, pages 3–5, July 2006.

[Gir06b] F. Giroire. Extended hit counting to estimate cardinality. In *Proceedings of WWW/Internet 2006, Murcia, Spain, october 2006.*, 2006.

[GKW04] C. Greenhill, J. H. Kim, and N. C. Wormald. Hamiltonian decompositions of random bipartite regular graphs. *Journal of Combinatorial Theory Series B*, no. 2, 90 :195–222, 2004.

[GNTD03] F. Giroire, A. Nucci, N. Taft, and C. Diot. Increasing the robustness of ip backbones in the absence of optical level protection. In *Proceedings of INFOCOM*, 2003.

[Gol82] S.W. Golond. *Shift register sequences*. Aegean Park Press, revised version, 1982.

[GTK01] Lise Getoor, Benjamin Taskar, and Daphne Koller. Selectivity estimation using probabilistic models. In *SIGMOD Conference*, 2001.

[Hav06a] F. Havet. Repartitors, selectors and superselectors. *Submitted to Journal of Interconnection Networks*, 2006.

[Hav06b] F. Havet. Repartitors, selectors and superselectors. *Submitted to Journal of Interconnection Networks*, 2006.

[IDGM01] G. Iannaccone, C. Diot, I. Graham, and N. McKeown. Monitoring very high speed links. In *ACM SIGCOMM Internet Measurement Workshop*, San Francisco, CA, November 2001.

[Knu98] D. E. Knuth. *The Art of Computer Programming, Volume III : Sorting and Searching, 2nd edition*, volume 3. Addison-Wesley, 1998.

[LN83] R. Lidl and H. Niederreiter. *Finite fields*. Cambridge University Press, 1983.

[Mor77] R. Morris. *Counting large numbers of events in small registers*, pages 840–842. Communications of the ACM 21 10, 1977.

[Mor94] M. Morgenstern. Existence and explicit constructions of $q+1$ regular ramanujan graphs for every prime power q. *Journal of Combinatorial Theory Series B*, 62 :44–62, 1994.

[MP01] B. Monien and R. Preis. Upper bounds on bissection width of 3- and 4- regular graphs. *Mathematical foundations of computer science*, Lecture Notes in Comput. Sci., 2136, Springer, Berlin :524–536, 2001.

[Mur03] M. Murty. Ramanujan graphs. *Journal of the Ramanujan Mathematical Society*, 18 :33–52, 2003.

[Pip77] N. Pippenger. Superconcentrators. *SIAM Journal on Computing*, 6 :298–304, 1977.

[Plo00] D. Plonka. Flowscan : A network traffic flow reporting and visualization tool. In *USENIX LISA*, pages 305–317, 2000.

[RR00] S. Rahman and E. Rivals. Exact and efficient computation of the expected number of missing and common worlds in random texts. In *Proceedings of the 11th Symposium on Combinatorial Pattern Matching*, 2000.

[RVW02] O. Reingold, S. Vadhan, and Z. Wigderson. Entropy waves, the zig-zag product and new constant-degree expanders. *Ann. of Math.(2)*, 155(1) :157–187, 2002.

[Sch06] Uwe Schöning. Smaller superconcentrators of density 28. *Inform. Process. Lett. 98*, 4 :127–129, 2006.

[Ser04] J. Serror. Réseaux d'interconnexion : tolérance aux pannes. Rapport de stage de Maîtrise, 2 mois, encadrant J.-C. Bermond, ENS Paris, 2004.

[Tap95] C. Tap. *A survey on graf theory*, pages 115 – 136. The Pof Corporation, 1995.

[Val75] Leslie G. Valiant. On non-linear lower bounds in computational complexity. *STOC*, pages 45–53, 1975.

[WZT90] K.-Y. Whang, B. T. V. Zanden, and H. M. Taylor. A linear-time probabilistic counting algorithm for database applications. In *TODS 15, 2*, pages 208–229, 1990.