# *AlgoTel 2005*
# *Septièmes rencontres francopohones sur les Aspects Algorithmiques des Télécommunications*

11, 12 et 13 mai 2005 – Presqu'île de Giens

---

AlgoTel 2005 est la septième édition d'une série de rencontres fructueuses: à Roscoff en 1999, à la Rochelle en 2000, à Saint-Jean-de-Luz en 2001, à Mèze en 2002, à Banyuls en 2003 et à Batz-sur-mer en 2004.

AlgoTel réunit ainsi depuis sept ans la communauté universitaire, institutionnelle et industrielle souhaitant partager ses compétences et ses résultats récents traitant de la résolution de problèmes fondamentaux issus du monde des réseaux et des télécommunications au moyen de techniques algorithmiques. Ces techniques sont issues de domaines variés comme les mathématiques discrètes, la théorie des graphes, l'optimisation combinatoire, les systèmes distribués et répartis, l'analyse probabiliste, la géométrie stochastique ou la simulation à évènements discrets.

Pour sa septième édition, AlgoTel a mis particulièrement l'accent sur :

- Algorithmique pour les réseaux d'interactions
- Allocation de ressources (fréquences, codes, etc.)
- Configuration et routage dans les réseaux à grande échelle.
- Gestion de la mobilité et réseaux sans fil
- Réseaux de capteurs
- Métrologie de graphes et réseaux
- Pair-à-Pair

Cette septième édition a également été l'occasion d'une plus grande ouverture vers la communauté *réseaux* et vers des thématiques pluridisciplinaires (réseaux sociaux notamment). Cette ouverture, voulue depuis longtemps, se reflète dans la composition du comité scientifique (voir ci-dessous), dans les articles publiés dans les actes, et dans les thématiques prioritaires ci-dessus. Nous sommes convaincus qu'elle sera l'occasion de rapprochements fructueux et d'ouverture à des problématiques nouvelles de première importance.

## Comité scientifique

Nous souhaitons vivement remercier les membres du comité scientifique qui ont accepté de participer à la sélection des papiers. La qualité des communications présentées cette année a été assurée grâce à leur lecture rigoureuse et leurs remarques constructives.

- Houssem Assadi, FT R&D, Issy-les-Moulineaux
- Alain Barrat, LPT, CNRS et université Paris 11, Orsay

- Pascal Berthou, LAAS, CNRS, Toulouse
- David Coudert, MASCOTTE, I3S/INRIA, Sophia Antipolis
- Jean-Michel Fourneau, PRISM, CNRS et université Versailles-Saint Quentin, Versailles
- Timur Friedman, LIP6, CNRS et université Paris 6, Paris
- Isabelle Guerin-Lassous, CITI, CNRS, INSA Lyon et INRIA, Lyon
- Jens Gustedt, Algorille/LORIA, CNRS, universités de Nancy et INRIA, Nancy
- Nicolas Hanusse, LaBRI, CNRS et université Bordeaux 1, Bordeaux
- Matthieu Latapy, LIAFA, CNRS et université Paris 7, Paris (co-président)
- Fabrice Le-Fessant, COMETE, INRIA, Rocquencourt
- Clémence Magnien, CREA, CNRS et Ecole Polytechnique, Palaiseau
- Damien Magoni, LSIIT, CNRS et université Strasbourg 2, Strasbourg
- Olivier Marce, Alcatel R&I, Paris
- Philippe Owezarski, LAAS, CNRS, Toulouse (co-président)
- Christophe Paul, LIRMM, CNRS et université Montpellier 2, Montpellier
- Christophe Prieur, LIAFA, CNRS et université Paris 7, Paris
- David Simplot-Ryl, IRCICA/LIFL, CNRS, université Lille 1 et INRIA, Lille
- Sandrine Vial, LaMI, CNRS et Université d'Évry-val-d'Essonne, Évry

## Comité d'organisation

Nous tenons à remercier les membres du comité d'organisation pour l'énergie qu'ils ont dépensée à l'organisation de ces rencontres afin de nous offrir le meilleur accueil possible.

- Guillaume Chelius, INRIA, Lyon (co-président)
- Jean-Loup Guillaume, LIAFA et FT R&D, Paris
- Corinne Julien, I3S, Sophia Antipolis
- Hervé Rivano, CNRS, Sophia-Antipolis (co-président)
- Dany Sergeant, INRIA, Sophia-Antipolis

## Partenaires d'AlgoTel 2005

AlgoTel 2005 a été organisée avec le soutien de partenaires et organismes privés et publics. Cet intérêt et ce témoignage de reconnaissance sont un encouragement pour la poursuite et l'organisation de ce colloque, dans le futur. Nous les en remercions.

- le GDR ARP du CNRS,
- le CITI – INSA de Lyon
- France Télécom
- l'INRIA
- l'I3S
- Alcatel
- KoBaFiS
- la région PACA
- le CNRT Télius

# Table des matières

# *Last recent research advances in energy-efficient broadcasting in wireless ad hoc networks*

David Simplot-Ryl

*IRCICA/LIFL INRIA Futurs, Lille*

---

De nombreux protocoles ont été proposés pour le routage dans les réseaux ad hoc. Il existe plusieurs classifications des protocoles de routage dans les réseaux ad-hoc, la plus classiques d'entre elles consiste à considérer les trois grandes catégories : réactifs, proactifs hybrides. La diffusion (broadcast) est un mécanisme fondamental utilisé aussi bien dans les algorithmes de routage proactifs que réactifs. Dans les algorithmes proactifs, elle est utilisée pour diffuser les informations nécessaires. Les algorithmes réactifs font appel à l'inondation afin de découvrir les routes. Les protocoles de diffusion sont également utilisés en dehors des protocoles de routage comme, par exemple, pour la dissémination d'informations pour la découverte de service. Il est donc primordial de chercher à optimiser les protocoles de diffusion, c'est-à-dire d'utiliser le moins de ressources réseaux pour diminuer la surcharge tout en assurant une diffusion maximale. Dans cet exposé, nous verrons les dernières avancées sur l'optimisation de la diffusion en distinguant trois grandes familles : la diminution du nombre d'émissions, l'ajustement de portées et les antennes directionnelles.

**Références :**


- J. Cartigny, D. Simplot-Ryl, and I. Stojmenovic. *An adaptive localized scheme for energy-efficient broadcasting in ad hoc networks with directional antennas.* In Proc. 9th IFIP International Conference on Personal Wireless Communications (PWC 2004), (Delft, The Netherlands, 2004), I. Niemegeers and S. Heemestra de Groot, Eds., Lecture Notes in Computer Science, vol. 3260, pp. 399-413. Best paper award.
- F. Ingelrest, and D. Simplot-Ryl. *Localized Broadcast Incremental Power Protocol for Wireless Ad Hoc Networks.* In Proc. 10th IEEE Symposium on Computers and Communications (ISCC 2005), (Cartagena, Spain, 2005), to appear.
- F. Ingelrest, D. Simplot-Ryl, and I. Stojmenovic. *A dominating sets and target radius based localized activity scheduling and minimum energy broadcast protocol for ad hoc and sensor networks.* In Proc. 3rd IFIP Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET 2004), (Bodrum, Turkey, 2004).
- F. Ingelrest, D. Simplot-Ryl, and I. Stojmenovic. *Energy efficient broadcasting in wireless mobile networks.* In Resource Management in Wireless Networking, M. Cardei, I. Cardei, and D.-Z. Du, Eds, Kluwer. 2005.
- D. Simplot-Ryl, I. Stojmenovic, and J. Wu. *Energy Efficient Backbone Construction, Broadcasting, and Area Coverage in Sensor Networks.* In Handbook of Sensor Networks. I. Stojmenovic, Ed., John Wiley & Sons, New York, 2005. (to appear).

# *Peer sharing behaviour in the eDonkey network and Design of Server-less File Sharing Systems*

Anne-Marie Kermarrec

*IRISA, Rennes*

---

Peer-to-peer file sharing systems have grown to the extent that they now generate most of the Internet traffic, way ahead of Web traffic. Understanding workload properties of peer to peer systems becomes necessary to optimize their performance. In this talk, we present the analysis of an eDonkey peer to peer file sharing system workload of over 350 terabytes of data, gathered mainly in Europe between December 2003 and February 2004. The trace is based on an active crawl of over 50,000 client caches per day, where a cache consists of files a peer agrees to share. We first present summary statistics on file replication and popularity, and peer geographic distribution. We also analyze temporal properties of the workload, namely the evolution of file popularity, and of peer cache contents. We are able to analyze fine properties such as the evolution of the overlap between distinct peer caches thanks to the massive information obtained from the crawler. We focus on the emergent clustering properties of such workloads, in terms of similarity of interest between peers. We assess such clustering properties via both direct metrics and an indirect operational index. The latter measures how frequently peers could retrieve files they sought from caches of appropriately selected “semantic neighbours”. We observe the presence of a significant amount of interest-based clustering. This suggests several potential optimizations of current peer to peer networks by exploiting this property.

# *Réseaux sociaux, pratiques culturelles et outils de communication*

Dominique Cardon

*Laboratoire de sociologie des usages/France Télécom R&D,* domi.cardon@francetelecom.com

Une des voies d'entrée privilégiée pour étudier les usages des outils de communication est l'analyse des pratiques de sociabilités des individus. Dans cet exposé, on reconstituera l'évolution des différentes enquêtes mettant en relation le trafic téléphonique des personnes et leurs caractéristiques sociodémographiques et géographiques. La reconstitution de cette histoire des différents travaux sur la sociabilité permettra de mettre en avant les apports des analyses en terme de réseaux sociaux à la compréhension des usages « entrelacés » des différents outils de communication qui se sont complexifiés avec le développement de la téléphonie mobile et de l'Internet.

## 1 Téléphone et face-à-face

Le lien entre rencontre et appel téléphonique semble aujourd'hui aller de soi, tant les téléphones et les autres moyens de communication sont devenus partie intégrante de nos contacts quotidiens. Néanmoins les premières enquêtes sociologiques sur les réseaux sociaux n'ont pas inclus ces sociabilités médiatisées dans leur protocole de saisie des comportements relationnels. Ainsi, la première grande enquête française sur les contacts sociaux de 1983 a limité son spectre aux seules rencontres en face à face pour étudier le capital social des Français. Ce choix n'était pas seulement l'écho d'un faible intérêt pour les contacts médiatisés à l'époque de la recherche, mais il reflétait également un parti pris plus général, accordant une prépondérance dans la construction du lien social aux interactions en face à face, comme seul idéaltype de contact interpersonnel. Un appel téléphonique apparaissait, dans cette perspective, comme une forme réduite et contrainte de l'interaction au regard des contacts « directs ». L'introduction de la téléphonie comme média de la sociabilité au sein des enquêtes sur les contacts interpersonnels a d'abord été jugé suspecte. Tout s'est en effet passé comme si les enquêtes sur les pratiques téléphoniques initiées par la DGT et celles sur le trafic téléphonique conduites au CNET (le centre de recherche de France Télécom) avaient d'abord du faire la preuve de la cohérence de leurs résultats avec les enquêtes sur les contacts en face-à-face. Loin d'une opposition ou, à tout le moins, de la constitution d'un univers autonome et séparé, il est vite apparu que les échanges téléphoniques étaient très étroitement articulés aux relations sociales ordinaires des personnes et qu'ils permettaient d'en dessiner l'architecture de manière économique et efficace. Pour se faire accepter comme nouveau support méthodologique dans l'étude des sociabilités, les enquêtes sur le trafic téléphonique ont ainsi du démontrer que l'univers social du téléphone n'était pas différent de l'espace relationnel des personnes, tout en prouvant par ailleurs qu'il permettait d'enrichir la connaissance des variables structurantes de la distribution des sociabilités au sein des espaces conjugaux, familiaux et amicaux.

## 2 L'entrelacement des usages : vers une approche élargie de l'entretien des sociabilités

L'approche des sociabilités s'est cependant considérablement complexifiée avec l'émergence de nouveaux outils de communication ajoutant au répertoire communicationnel du téléphone du foyer les téléphones cellulaires, les systèmes de messagerie textuelle ou vocale et tous les outils utilisables à partir d'un téléphone, d'un PDA ou d'un ordinateur connecté à l'Internet qui peuvent mettrent en relation avec des

proches ou des inconnus. La diversification de l'offre de technologies de contact ne permet plus de se reposer sur une opposition entre la rencontre physique et l'appel téléphonique, mais elle oblige à faire place à des trajectoires beaucoup plus complexes dans lesquelles l'entrelacement de multiples médias de communication est mis au service de l'entretien des relations sociales. Aussi, avant de disposer d'enquêtes de grande ampleur à vocation représentative des effets de ces nouvelles technologies relationnelle sur l'organisation des sociabilités, est-il utile de relever les phénomènes que font apparaître une série de recherches portant sur les populations les plus engagées dans l'usage de ces nouvelles technologies, à savoir les internautes et les jeunes. A cet égard, trois dimensions de ces formes relationnelles « entrelacées » peuvent être isolées : la connexion continue (ou présence connectée), l'enchevêtrement des espaces de communication électronique et les articulations entre pratiques de communication et pratiques culturelles et de loisir.

## 3 Réseaux sociaux, pratiques de communication et activités culturelles et de loisirs

Des démarches d'enquête visant à reconstituer la sphère relationnelle des personnes ont permis de cartographier la taille et la forme des réseaux relationnels des individus. A partir de différents exemples, on montrera l'étroite relation qui existe entre l'organisation du système relationnel des personnes et leurs pratiques culturelles et de loisirs. Les interactions entre le type de pratiques culturelles et la forme du réseau de sociabilité constituent en effet un point d'ancrage décisif des « trajectoires d'usages » des instruments de communication. De manière formelle, on distinguera : (1) les configurations « polarisées », dans lesquelles plusieurs types de pratiques culturelles différentes sont conduites avec un même réseau de relation ; (2) les configurations « spécialisées », dans lesquelles un type spécifique de pratiques est réservé de façon quasi exclusive à un type de réseau de relation ; enfin (3) les situations « distribuées », dans lesquelles un type de pratiques culturelles est partagé avec plusieurs cercles du réseau relationnel. Chacune de ces configurations renvoie à des modes de mise en contact sensiblement différents et donc à des arbitrages particuliers dans le choix des outils de communication.

# *Session WDM & Graphes*

Mardi 11 mai 2005, 10:15 - 11:45

---

# *Exploration par un automate fini de réseaux anonymes étiquetés*†

R. Cohen[1], P. Fraigniaud[2], D. Ilcinkas[2], A. Korman[1], et D. Peleg[1]

[1] *Dept. of Computer Science, Weizmann Institute, Israel*
[2] *CNRS, LRI, Université Paris-Sud, France.*

**Résumé.** Cet article traite de l'exploration d'un réseau par un agent mobile, ou *robot*, modélisé par un automate fini. Le robot ne dispose pas de connaissances préalables sur la topologie du réseau, ni même sur sa taille. Sa tâche consiste à explorer chacun des nœuds du réseau. Il a été montré que, pour tout robot $\mathcal{R}$ à $K$ états, et pour tout $d \geq 3$, il existe un réseau de degré maximum $d$ et d'au plus $K+1$ nœuds que $\mathcal{R}$ ne réussit pas à explorer. Ce papier s'intéresse à la possibilité d'aider le robot en ajoutant des étiquettes de petite taille aux nœuds grâce à un calcul préalable. Nous décrivons un algorithme d'exploration qui, étant donné des étiquettes appropriées sur 2 bits (en fait des étiquettes de trois valeurs différentes), permet à un robot d'explorer tous les réseaux. De plus, nous décrivons un algorithme linéaire calculant un tel étiquetage. Nous montrons également comment modifier notre méthode d'étiquetage afin qu'un robot puisse explorer tous les réseaux de degré constant, avec des étiquettes de seulement 1 bit (donc des étiquettes de deux valeurs différentes seulement). Autrement dit, alors qu'il n'existe pas de robot capable d'explorer tous les réseaux (non étiquetés) de degré maximum 3, il existe un robot $\mathcal{R}^*$ et un moyen de colorier les nœuds en noir et blanc tel que $\mathcal{R}^*$ réussit à explorer tous les réseaux coloriés de degré maximum 3.

Une version étendue de cet article sera publiée dans ICALP 2005 [3].

**Keywords:** exploration, labyrinthe, automate fini, agent mobile, robot, étiquetage.

## 1 Background and model

Let $\mathcal{R}$ be a finite automaton, simply referred to in this context as a *robot*, moving in an unknown graph $G = (V, E)$. The robot has no a priori information about the topology of $G$ and its size. To allow the robot $\mathcal{R}$, visiting a node $u$, to distinguish between its edges, the $d = \deg(u)$ edges incident to $u$ are associated to $d$ distinct *port numbers* in $\{0, \ldots, d-1\}$, in a one-to-one manner. The port numbering is given as part of the input graph, and the robot has no a priori information about it. For convenience of terminology, we henceforth refer to "the edge incident to port number $l$ at node $u$" simply as "edge $l$ of $u$". (Clearly, if this edge connects $u$ to $v$, then it may also be referred to as "edge $l'$ of $v$" for the appropriate $l'$.) The robot has a transition function $f$, and a finite number of states. If $\mathcal{R}$ enters a node of degree $d$ through port $i$ in state $s$, then it switches to state $s'$ and exits the node through port $i'$, where

$$(s', i') = f(s, i, d).$$

The objective of the robot is to *explore* the graph, i.e., to visit all its nodes. The task of visiting all nodes of a network is fundamental in searching for data stored at unknown nodes or when looking for defective components.

The first known algorithm designed for graph exploration was introduced by Shannon [9]. Since then, several papers have been dedicated to the feasibility of graph exploration by a finite automaton. Rabin [7] conjectured that no finite automaton with a finite number of pebbles can explore all graphs (a *pebble* is a marker that can be dropped at and removed from nodes). The first step towards a formal proof of Rabin's

†Les deuxièmes et troisièmes auteurs ont reçu le support des projets Fragile de de l'ACI Sécurité Informatique, PairAPair de l'ACI Masse de Données, et Grand Large de l'INRIA.

conjecture is generally attributed to Budach [2], for a robot without pebbles. Blum and Kozen [1] improved Budach's result by proving that a robot with three pebbles cannot perform exploration of all graphs. Kozen [6] proved that a robot with four pebbles cannot explore all graphs. Finally, Rollik [8] gave a complete proof of Rabin's conjecture, showing that no robot with a finite number of pebbles can explore all graphs. The result holds even when restricted to planar 3-regular graphs. Without pebbles, it was proved [5] that a robot needs $\Theta(D \log \Delta)$ bits of memory for exploring all graphs of diameter $D$ and maximum degree $\Delta$. On the other hand, if the class of input graphs is restricted to trees, then exploration is possible even by a robot with no memory (i.e., zero states), simply by DFS using the transition function $f(i,d) = i+1 \bmod d$ (see, e.g., [4]).

The ability of dropping and removing pebbles at nodes can be viewed alternatively as the ability of the robot to dynamically *label* the nodes. If the robot is given $k$ pebbles, then, at any time of the exploration, $\sum_{u \in V} |l_u| \leq k$ where $l_u$ is the label of node $u$ and $|l_u|$ denotes the size of the label in unary. This paper considers the effects of allowing the system designer to assign labels to the nodes in a preprocessing stage, and using these labels to guide the exploration by the robot. The transition function $f$ is augmented to utilize labels as follows. If $\mathcal{R}$ in state $s$ enters a node of degree $d$, labeled by $l$, through port $i$, then it switches to state $s'$ and exits the node through port $i'$, where

$$(s', i') = f(s, i, d, l).$$

This model can be considered stronger than Rabin's pebble model since labels are given in a preprocessing stage whereas in Rabin's model the automaton starts with all its pebbles. But it can also be considered weaker since, once assigned to nodes, the labels cannot be modified.

In this paper, we consider settings where it is expected that the graph will be visited by many exploring robots, and consequently, the system designer would like to preprocess the graph by leaving (preferably small) road-signs, or *labels*, that will aid the robots in their exploration task. As possible scenarios one may consider a network system where finite automata are used for traversing the system and distributing information in a sequential manner.

More formally, we address the design of *exploration labeling schemes*. Such schemes consist of a pair $(\mathcal{L}, \mathcal{R})$ such that, given any graph $G$ with any port numbering, the algorithm $\mathcal{L}$ labels the nodes of $G$, and the robot $\mathcal{R}$ explores $G$ with the help of the labeling produced by $\mathcal{L}$. In particular, we are interested in exploration labeling schemes for which : (1) the preprocessing time required to label the nodes is polynomial, (2) the labels are short, and (3) the exploration is completed after a small number of edge-traversals. Note that we are only interested in the preprocessing time. The algorithm $\mathcal{L}$ may require a global knowledge of the graph or a large amount of memory.

## 2 Our results

As a consequence of Budach's result, any exploration labeling scheme must use at least *two* different labels. Our main result states that just *three* labels (e.g., three colors) are sufficient for enabling a robot to explore all graphs. Moreover, we show that our labeling scheme gives to the robot the power to stop once exploration is completed, although, in the general setting of graph exploration, the robot is not required to stop once the exploration has been completed, i.e., once all nodes have been visited. In fact, we show that exploration is completed in time $O(m)$, i.e., after $O(m)$ edge traversals, in any $m$-edge graph.

For the class of bounded degree graphs, we design an exploration scheme using even smaller labels. More precisely, we show that just *two* labels (i.e., 1-bit labels) are sufficient for enabling a robot to explore all bounded degree graphs. The robot is however required to have a memory of size $O(\log \Delta)$ to explore all graphs of maximum degree $\Delta$. The completion time $O(\Delta^{O(1)} m)$ of the exploration is larger than the one of our previous 2-bit labeling scheme, nevertheless it remains polynomial.

All these results are summarized in Table 1. The two mentioned labeling schemes require polynomial preprocessing time.

| Label size (#bits) | Robot's memory (#bits) | Time (#edge-traversals) |
|---|---|---|
| 2 | $O(1)$ | $O(m)$ |
| 1 | $O(\log\Delta)$ | $O(\Delta^{O(1)}m)$ |

**TAB. 1:** Summary of main results.

# 3 A 2-bit exploration-labeling scheme

Our first result is to describe a 3-valued exploration labeled scheme for a robot with constant memory.

**Theorem 1** *There exists a robot with the property that for any graph $G$, it is possible to color the nodes of $G$ with three colors (or alternatively, assign each node a 2-bit label) so that using the labeling, the robot can explore the entire graph $G$, starting from any given node and terminating after identifying that the entire graph has been traversed. Moreover, the total number of edge-traversals by the robot is $\leq 20m$.*

We first describe the labeling scheme $\mathcal{L}$ and then the exploration algorithm. The node labeling is in fact very simple ; it uses three labels, called colors, and denoted WHITE, BLACK, and RED. Let $D$ be the diameter of the graph.

Pick an arbitrary node $r$. Node $r$ is called the *root* of the labeling $\mathcal{L}$. Nodes at distance $d$ from $r$, $0 \leq d \leq D$, are labeled WHITE if $d \mod 3 = 0$, BLACK if $d \mod 3 = 1$, and RED if $d \mod 3 = 2$.

The neighbor set $\mathcal{N}(u)$ of each node $u$ can be partitioned into three disjoint sets : (1) the set $\mathsf{pred}(u)$ of neighbors closer to $r$ than $u$ ; (2) the set $\mathsf{succ}(u)$ of neighbors farther from $r$ than $u$ ; (3) the set $\mathsf{sibling}(u)$ of neighbors at the same distance from $r$ as $u$. We also identify the following two special subsets of neighbors :

- $\mathsf{parent}(u)$ is the node $v \in \mathsf{pred}(u)$ such that the edge $\{u,v\}$ has the smallest port number at $u$ among all edges leading to a node in $\mathsf{pred}(u)$.
- $\mathsf{child}(u)$ is the set of nodes $v \in \mathsf{succ}(u)$ such that $\mathsf{parent}(v) = u$.

For the root, set $\mathsf{parent}(r) = \emptyset$.

Note that for every node $u$ with label $\mathcal{L}(u)$, and for every neighbor $v \in \mathcal{N}(u)$, the label $\mathcal{L}(v)$ uniquely determines whether $v$ belongs to $\mathsf{pred}(u)$, $\mathsf{succ}(u)$ or $\mathsf{sibling}(u)$.

Furthermore one can design a procedure called Check_Edge that allows a robot with constant memory to identify precisely whether an edge incident to a node $u$ leads to $\mathsf{parent}(u)$ or to a node in $\mathsf{child}(u)$.

The functions parent and child induce a spanning tree of the graph. One can then design a quite simple robot that can perform a depth-first search of the tree, using a constant number of memory bits.

# 4 A 1-bit exploration-labeling scheme for bounded degree graphs

We now describe an exploration labeling scheme using only 1-bit labels. This scheme requires a robot with $O(\log\Delta)$ bits of memory for the exploration of graphs of maximum degree $\Delta$. More precisely, we prove the following.

**Theorem 2** *There exists a robot with the property that for any graph $G$ of degree bounded by a constant $\Delta$, it is possible to color the nodes of $G$ with two colors (or alternatively, assign each node a 1-bit label) so that using the labeling, the robot can explore the entire graph $G$, starting from any given node and terminating after identifying that the entire graph has been traversed. The robot has $O(\log\Delta)$ bits of memory, and the total number of edge-traversals by the robot is $O(\Delta^{(O(1)}m)$.*

As for $\mathcal{L}$, pick an arbitrary node $r \in V$, called the *root*. Nodes at distance $d$ from $r$ are labeled as a function of $d \mod 8$. Partition the nodes into eight *classes* by letting

$$C_i = \{u \in V \mid \mathrm{dist}_G(r,u) \mod 8 = i\}$$

for $0 \leq i \leq 7$. Node $u$ is colored white if $u \in C_0 \cup C_2 \cup C_3 \cup C_4$, and black otherwise. Let

$$\tilde{C}_1 = \{u \mid \mathrm{dist}_G(r,u) = 1\}$$

$$\widehat{C} = \{r\} \cup \{u \in C_2 \mid \mathrm{dist}_G(r,u) = 2 \text{ and } \mathcal{N}(u) = \tilde{C}_1\}.$$

**Lemma 1** *There is a local search procedure enabling a robot of $O(\log\Delta)$ bits of memory to decide whether a node $u$ belongs to $\widehat{C}$ and to $\tilde{C}_1$, and to identify the class $C_i$ of every node $u \notin \widehat{C}$.*

**Proof.** Let **B** (resp., **W**) be the set of black (resp., white) nodes which have all their neighbors black (resp., white). The class $C_1$ and the classes $C_3,\ldots,C_7$ can be described using the colors of the node and of the other nodes at distance at most 4. For example, we provide the caracterization of the class $C_1$ : $u \in C_1 \Leftrightarrow u$ is black, $u$ has no neighbor in **B**, and $u$ has a white neighbor $v$ that has no neighbor in **W**.

Based on those characterizations, the classes $C_1$ and $C_3,\ldots,C_7$ can be easily identified by a robot of $O(\log\Delta)$ bits, via performing a local search. Moreover, the sets $\tilde{C}_1$ and $\widehat{C}$ can also be characterized as follows :

- $u \in \tilde{C}_1 \Leftrightarrow u \in C_1$ and $u$ has no node in $C_7$ at distance $\leq 2$ ;
- $u \in \widehat{C} \Leftrightarrow N(u) \subseteq \tilde{C}_1$ and every node $v$ at distance $\leq 2$ from $u$ satisfies $|N(v) \cap \tilde{C}_1| \leq |N(u)|$.

Using this we can deduce :

- $u \in C_0 \setminus \widehat{C} \Leftrightarrow u \notin (\cup_{i=3}^{7} C_i) \cup C_1$ and $u$ has a neighbor in $C_7$ ;
- $u \in C_2 \setminus \widehat{C} \Leftrightarrow u \notin \widehat{C}$, has a neighbor in $C_1$, but has no neighbor in $C_7$.

It follows that a robot of $O(\log\Delta)$ bits can identify the class of every node except for nodes in $\widehat{C}$. □

**Proof of Theorem 2.** Due to Lemma 1, all instructions of the exploration algorithm using labeling $\mathcal{L}$ can be executed using labeling $\mathcal{L}'$, but for the cases not captured in Lemma 1, i.e., $\widehat{C}$.

To solve the problem of identifying the root, we notice that each of the nodes in $\widehat{C}$ can be used as a root, and all the others can be considered as leaves in $C_2$. Thus, when leaving the root, the robot should memorize the port $P$ by which it should return to the root. When the robot arrives at a node $u \in \tilde{C}_1$ through a tree edge and has to explore the root, it leaves immediately through port $P$ and deletes the contents of $P$, then it goes down through the next unexplored port if one is left. When the robot is in a node $u \in \tilde{C}_1$ and has to explore some child, it will skip the port $P$. □

# 5 Concluding Remarks

Our results let open a very nice problem : are single bit labels sufficient for ensuring the traversal of all graphs by a finite ($O(1)$ memory) robot ?

# Références


[1] M. Blum and D. Kozen. On the power of the compass (or, why mazes are easier to search than graphs). In 19th Symposium on Foundations of Computer Science (FOCS), pages 132-142, 1978.

[2] L. Budach. Automata and labyrinths. Math. Nachrichten, pages 195-282, 1978.

[3] R. Cohen, P. Fraigniaud, D. Ilcinkas, A. Korman and D. Peleg. Label-guided Graph Exploration by a Finite Automaton. In 32nd International Colloquium on Automata, Languages and Programming (ICALP), 2005, to appear.

[4] K. Diks, P. Fraigniaud, E. Kranakis, and A. Pelc. Tree Exploration with Little Memory. J. Algorithms 51(1) :38-63, 2004.

[5] P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, and D. Peleg. Graph Exploration by a Finite Automaton. In 29th International Symposium on Mathematical Foundations of Computer Science (MFCS), LNCS 3153, 451-462, 2004.

[6] D. Kozen. Automata and planar graphs. In Fund. Computat. Theory (FCT), 243-254, 1979.

[7] M.O. Rabin, Maze threading automata. Seminar talk presented at the University of California at Berkeley, October 1967.

[8] H.A. Rollik. Automaten in planaren Graphen. Acta Informatica 13 :287-298, 1980 (also in LNCS 67, 266-275, 1979).

[9] C. Shannon. Presentation of a maze-solving machine. In 8th Conf. of the Josiah Macy Jr. Found. (Cybernetics), pages 173-180, 1951.

# *Stratégies d'encerclement connexes dans un réseau*

Pierre Fraigniaud[†] and Nicolas Nisse[‡]

*CNRS*
*Laboratoire de Recherche en Informatique*
*Université Paris Sud*
*91405 ORSAY, France*
{*pierre,nisse*}*@lri.fr*

Le problème de l'encerclement dans les réseaux a été introduit par Parson (1976) : étant donné un réseau "contaminé" (par exemple dans lequel un intrus s'est introduit), l'*encerclement* du réseau est le nombre minimum d'agents nécessaires pour "nettoyer" le réseau (c'est-à-dire capturer l'intrus). Une stratégie d'encerclement est dite connexe si à chaque étape de la stratégie, l'ensemble des liens nettoyés induit un sous-réseau connexe. Les stratégies d'encerclement connexes sont essentielles si l'on souhaite assurer des communications sûres entre les agents. Dans le cas des réseaux en arbres, Barrière *et al.* (2002, 2003) ont prouvé que le rapport entre l'encerclement connexe et l'encerclement est majoré par 2, et que cette borne est optimale. Dans cet article, nous donnons une borne pour ce rapport dans le cas des réseaux arbitraires. Pour cela nous utilisons une notion cruciale de théorie des graphes : la largeur arborescente. L'égalité entre la largeur arborescente connexe d'un graphe et sa largeur arborescente découle du théorème de Parra et Scheffler (1995). Nous donnons ici une preuve constructive de cette égalité. Plus précisemment, nous proposons un algorithme qui étant donnés un graphe $G$ de $n$ sommets et une décomposition arborescente de largeur $k$ de $G$, calcule en temps $O(n\,k^3)$ une décomposition arborescente connexe de largeur $\leq k$ de $G$. Une conséquence importante de notre résultat est qu'il permet de borner par $\lceil \log n \rceil + 1$ le rapport entre encerclement connexe et encerclement d'un réseau de $n$ nœuds.

**Keywords:** Décomposition arborescente, Décomposition en branches, Décomposition linéaire, Stratégie d'encerclement

## 1 Introduction

In *graph searching* (see, e.g.,[3, 4, 8, 12]), a fugitive is hidden in a graph $G$. A team of *searchers* is aiming at capturing this fugitive. The fugitive is assumed to be arbitrary fast, and permanently aware of the positions of the searchers. To capture the fugitive, three operations are allowed for the searchers. These basic operations, called *search steps*, are the following:

- **1:** place a searcher on a vertex,
- **2:** remove a searcher from a vertex,
- **3:** move a searcher along an edge.

There are several variants of the problem of graph searching. To clear an edge $e = \{u,v\}$ in the *edge-search* variant, a searcher must traverse this edge from one end $u$ to $v$. The edge $e$ is preserved from recontamination if either another searcher remains in $u$, or all edges incident to $e$ in $u$ are *cleared*. That is, a *cleared* edge $e$ is recontaminated if there exists a path between $e$ and a contaminated edge, with no searcher

[†]Additional supports from the INRIA Project "Grand Large", and from the Project PAIRAPAIR of the ACI "Masse de Données".
[‡]Additional supports from the Project FRAGILE of the ACI "Sécurité & Informatique".

on any vertex of this path. An edge-search strategy is a sequence of search steps that capture the fugitive (i.e. a strategy that completly clears the graph). The graph searching problem asks for the design of search strategies using a *minimum number of searchers*. The mimimal number of searchers for which there exists a strategy in a graph $G$ is the *search number* $\mathbf{s}(G)$ of $G$. The search-number varies depending on the relative power of the fugitive and the searchers.

— If the searchers are permanently aware of the position of the fugitive, then the optimal size of the team is essentially the treewidth of the graph [3]. The *treewidth* of a graph is a central concept in the theory of Graph Minors developed by Robertson and Seymour (see, e.g., [10]). Roughly speaking, the treewidth, $\mathbf{tw}(G)$, of a graph $G$ measures "how far" the graph $G$ is from a tree. More formally, a *tree-decomposition* of a graph $G$ is a pair $(T,X)$ where $T$ is a tree, and $X = \{X_v, v \in V(T)\}$ is a collection of subsets of $V(G)$ satisfying the following three conditions:

- **C1:** $V(G) = \cup_{v \in V(T)} X_v$;
- **C2:** For any edge $e$ of $G$, there is a set $X_v$ such that both end-points of $e$ are in $X_v$;
- **C3:** For any triple $u, v, w$ of nodes in $V(T)$, if $v$ is on the path from $u$ to $w$ in $T$, then $X_u \cap X_w \subseteq X_v$.

Condition **C3** can be rephrased as: for any node $x$ of $G$, $\{v \in V(T) \mid x \in X_v\}$ is a subtree of $T$. The sets $X_v$'s are often called *bags*. The *width*, $\omega(T,X)$, of a tree-decomposition $(T,X)$ is defined as $\max_{v \in V(T)} |X_v| - 1$, i.e., the width of $(T,X)$ is roughly the maximum size of its bags. The treewidth $\mathbf{tw}(G)$ is defined as $\min \omega(T,X)$ where the minimum is taken over all tree-decompositions $(T,X)$ of $G$. Beside its own interest, the treewidth has several important applications. In particular, it is known that several NP-hard problems can be solved in polynomial time if instances are restricted to graphs of bounded treewidth. Actually, on graphs of treewidth at most $k$, where $k$ is fixed, every decision or optimization problem expressible in monadic second-order logic has a linear algorithm [5].

— If the searchers are unaware of the position of the fugitive, then the optimal size of the team, $\mathbf{s}(G)$, is essentially the *pathwidth* of $G$ [3]. A path-decomposition of $G$ is a tree-decomposition $(T,X)$ of $G$, where $T$ is a path. The pathwidth $\mathbf{pw}(G)$ is defined as $\min \omega(T,X)$ where the minimum is taken over all path-decompositions $(T,X)$ of $G$. For any graph $G$, we have [3]: $\mathbf{pw}(G) \leq \mathbf{s}(G) \leq \mathbf{pw}(G) + 2$.

## 2 Our Objective

It has been argued (cf., e.g., [1, 2] and the references therein) that several practical applications (e.g., network security, speleological rescue, etc.) requires the search strategy be *connected*, i.e., at any time of the search strategy, the portion of the searched graph is a connected subgraph. In particular, this property insures safe and secure communications among the searchers across the cleared area.

Formally, a search strategy is *connected* if the set of *cleared* edges induces a connected subgraph at every step of the search. Another way to define such strategies is not to allow operation **2**, and to force all searchers to start from the same vertex. Assuming that the searchers are unaware of the position of the fugitive, the *connected* search number $\mathbf{cs}(G)$ of the graph $G$ is the minimum number of searchers for which a connected search strategy exists for $G$. In [2], Barrière *et al.* showed that the non-connectness helps. In other words, there exists graphs for which optimal connected search strategy requires more searchers than optimal search strategy. Nevertheless, [2] proved that, for any tree $T$,

$$\mathbf{cs}(T) \leq 2\,\mathbf{s}(T) - 2,$$

and that this bound is tight.

Using the concept of connected branchwidth, Fomin *et al.* [6] have shown that, for any connected $m$-edge graph $G$, the *connected* search number, $\mathbf{cs}(G)$, satisfies

$$\mathbf{cs}(G)/\mathbf{s}(G) \leq \lceil \log m \rceil + 1. \tag{1}$$

However, this bound has not been proved to be tight, and it is conjectured that

$$\mathbf{cs}(G)/\mathbf{s}(G) \leq 2$$

for any connected graph $G$. Our objective is therefore to improve the bound of Eq. 1. For that purpose, we have revisited the several notions of connectedness for graph decompositions.

# 3 Connectedness of graph decomposition

Connectedness has been defined directly on many graph decompositions. In particular, a *branch-decomposition* of a graph $G$ is a tree $T$ whose all internal nodes have degree 3, with a one-to-one correspondence between the leaves of $T$ and the edges of $G$. Given an edge $e$ of $T$, removing $e$ from $T$ results in two trees $T_1^{(e)}$ and $T_2^{(e)}$, and an *e-cut* is defined as the pair $\{E_1^{(e)}, E_2^{(e)}\}$, where $E_i^{(e)} \subset E(G)$ is the set of leaves of $T_i^{(e)}$ for $i = 1,2$. The width of $T$ is defined as $\omega(T) = \max_e |\delta(E_1^{(e)})|$ where the maximum is taken over all $e$-cuts in $T$, and where, for any edge-set $E$, $\delta(E)$ denotes the set of nodes with one extremity in $E$ and the other in $E(G) \setminus E$. The branchwidth $\mathbf{bw}(G)$ of $G$ is then $\min \omega(T)$ where the minimum is taken over all branch-decompositions $T$ of $G$. A branch-decomposition is *connected* if, for any of its $e$-cut, the two subgraphs of $G$ induced by $E_1^{(e)}$ and $E_2^{(e)}$ are connected. A major result about branchwidth is that if a 2-edge-connected $G$ has a branch-decomposition of width $k$, then it has a connected branch-decomposition of width $\leq k$ [11]. Therefore, for any 2-edge-connected graph $G$, there is equality between its connected branchwidth, $\mathbf{cbw}(G)$, and its branchwidth, $\mathbf{bw}(G)$. This equality is the argument used to derive Eq. 1.

We address the connectivity constraint directly on tree-decompositions (cf. Section 1 for the definition). An $e$-cut of a tree-decomposition $(T,X)$ of a graph $G$ is defined as the pair $\{X_1^{(e)}, X_2^{(e)}\}$, where $X_i^{(e)} \subseteq V(G)$ is the set of nodes in $\{X_v, v \in V(T_i^{(e)})\}$ for $i = 1,2$. A tree-decomposition is connected if, for any of its $e$-cuts, the two subgraphs $G[T_1^{(e)}]$ and $G[T_2^{(e)}]$ of $G$, induced by $X_1^{(e)}$ and $X_2^{(e)}$, respectively, are connected. The connected treewidth, $\mathbf{ctw}(G)$, of a connected graph $G$, is defined as the minimum width of any connected tree-decomposition of $G$. It is well known [7] that a clique tree $(T,X)$ of a minimal triangulation $H$ of a connected graphe $G$ is a tree decomposition with width $\mathbf{tw}(G)$ of $G$. Moreover, Parra and Scheffler proved in [9], that the set $\Delta_H$ of minimal separators of $H$ is exactly the set of pairwise parallel minimal separators in $G$ and that for any $S \in \Delta_H$, $S$ induces the same connected components in $H$ and $G$. This implies that $(T,X)$ is *connected*, and thus we get that

$$\mathbf{tw}(G) = \mathbf{ctw}(G) \tag{2}$$

for any connected graph $G$. This result extends to treewidth the result on carvings (and therefore branchwidth) in [11]. Importantly, the equality $\mathbf{ctw} = \mathbf{tw}$ holds for any connected graph, whereas the equality $\mathbf{cbw} = \mathbf{bw}$ holds for 2-edge-connected graphs only.

# 4 Our Results

We first give a constructive proof for the equality $\mathbf{ctw}(G) = \mathbf{tw}(G)$, for any connected graph $G$. This equality between treewidth and connected treewidth is obtained via the design of a polynomial-time algorithm transforming a tree-decomposition into a connected tree-decomposition of same width. More precisely, we prove the following:

**Theorem 1** *There exists a $O(Nk^3)$-time algorithm that, given any $N$-node tree-decomposition of a connected graph $G$, of width $k$, returns a connected tree-decomposition of $G$, of width $\leq k$.*

The algorithm of theorem 1 is linear in the case of graphs with bounded treewidth. A consequence of the equality between treewidth and connected treewidth is that we propose a polynomial-time algorithm which, given a connected graph $G$ and a tree decomposition with width $\mathbf{tw}(G)$ of $G$, computes a connected search strategy using at most $\mathbf{s}(G)$ $(\lceil \log n \rceil + 1)$ searchers, thus improving [6]. More precisely, we proved the following:

**Theorem 2** *For any connected $n$-node graph $G$, $\frac{\mathbf{cs}(G)}{\mathbf{s}(G)} \leq \lceil \log n \rceil + 1$.*

**Sketch of the Proof.** Let $G$ be an $n$-node connected graph, and let $(T,X)$ be an optimal connected tree-decomposition of $G$, i.e., of width $\mathbf{tw}(G)$. We use a result in [10] to show that, for any tree-decomposition $(T,X)$ of an $n$-node graph $G$, there exists a vertex $v \in V(T)$ (or an edge $e = (u,v) \in E(T)$) such that removing $v$ (or $u$ and $v$) from $T$ results in a forest $T_1, \cdots, T_r$, such that for every $i = 1 \cdots r$, the subgraph $G_i$ induced by $\bigcup_{v \in T_i} X_v$ has at most $n/2$ vertices. Using this result, we prove, by induction on $n$, that $\mathbf{cs}(G) \leq 1 + \mathbf{tw}(G)\lceil \log n \rceil$. Then, since $\mathbf{s}(G) \geq \mathbf{pw}(G) \geq \mathbf{tw}(G) = \mathbf{ctw}(G)$, we get $\mathbf{cs}(G) \leq 1 + \mathbf{s}(G)\lceil \log n \rceil$, and thus $\mathbf{cs}(G)/\mathbf{s}(G) \leq 1 + \lceil \log n \rceil$. □

# 5 Conclusion and further works

Barrière *et al.* proved in [2], that for any tree $T$, $\mathbf{cs}(T) \leq 2\ \mathbf{s}(T) - 2$. In this paper, we proved that $\mathbf{cs}(G)/\mathbf{s}(G) = O(\log n)$ for any $n$-nodes connected graph $G$. However, we conjecture that there exists a constant $k$ such that, for any connected graph $G$, $\mathbf{cs}(G)/\mathbf{s}(G) \leq k$. In fact, we even conjecture that $k = 2$:

**Conjecture:** For any connected graph $G$, $\mathbf{cs}(G)/\mathbf{s}(G) \leq 2$.

Obviously, one can generalize the definition of connected tree-decomposition to $k$-connected tree-decomposition, for any $k \geq 1$. In this case, it is required that every $e$-cut of the decomposition induces a $k$-connected graph. The $k$-connected treewidth $\mathbf{ctw}_k(G)$ of a $k$-connected graph $G$ is the smallest $w$ for which there is a $k$-connected tree-decomposition of $G$, of width $w$. One can show that, for any $k \geq 2$, there exists a $k$-connected $n$-node graph $G$ such that $\mathbf{ctw}_k(G)/\mathbf{tw}(G) \geq \Omega(n)$. Hence Eq. 2 cannot be trivially generalized to connectivities greater than 1. We however ask the following question:

**Open problem:** Is there a function $f$ such that $\mathbf{ctw}_k(G) = \mathbf{tw}(G)$ for any $f(k)$-connected graph $G$?

# References


[1] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Capture of an intruder by mobile agents. In 14th ACM Symp. on Parallel Algorithms and Architectures (SPAA), pages 200-209, 2002.

[2] L. Barrière, P. Fraigniaud, N. Santoro, and D. Thilikos. Connected and Internal Graph Searching. In 29th Workshop on Graph Theoretic Concepts in Computer Science (WG), Springer-Verlag, LNCS 2880, pages 34–45, 2003.

[3] D. Bienstock. Graph searching, path-width, tree-width and related problems (a survey). DIMACS Series in Discrete Mathematics and Theoretical Computer Science 5, pages 33-49, 1991.

[4] D. Bienstock and P. Seymour. Monotonicity in graph searching. Journal of Algorithms 12, pages 239-245, 1991.

[5] B. Courcelle, J. Makowsky, and U. Rotics. Linear-time solvable optimization problems on graphs of bounded cliquewidth. In 24th Workshop on Graph-Theoretic Concepts in Computer Science (WG), LNCS 1517, pages 1-16, 1998.

[6] F. Fomin, P. Fraigniaud, D. Thilikos. The Price of Connectedness in Expansions. Technical Report LSI-04-28-R, UPC Barcelona, 2004.

[7] M. C. Golumbic. Algorithmic graph theory and perfect graphs. Computer Science and Applied Mathematics, 1980.

[8] T. Parson. Pursuit-evasion in a graph. Theory and Applications of Graphs, Lecture Notes in Mathematics, Springer-Verlag, pages 426-441, 1976.

[9] A. Parra and P. Scheffler. Characterizations and algorithmic applications of chordal graphe embeddings. Discrete Applied Mathematics 79, pages 171-188, 1997.

[10] N. Robertson and P. D. Seymour. Graph minors II, Algorithmic Aspects of Tree-Width. Journal of Algorithms 7, pages 309-322, 1986.

[11] P. Seymour and R. Thomas. Call routing and the rat-catcher. Combinatorica 14(2), pages 217–241, 1994.

[12] K. Skodinis. Computing optimal linear layouts of trees in linear time. In 8th European Symp. on Algorithms (ESA), Springer-Verlag, LNCS 1879, pages 403-414, 2000.

# *Rerouting requests in WDM networks*[†]

D. Coudert[1] and S. Pérennes[1] and Q.-C. Pham[2] and J.-S. Sereni[1]

*1-MASCOTTE, I3S-CNRS/INRIA/UNSA, 2004 Route des Lucioles, BP93, F-06902, Sophia-Antipolis Cedex*
*2-Ecole Normale Supérieure, 45 rue d'Ulm, F-75230 Paris cedex 05*

We model a problem related to routing reconfiguration in WDM networks. We establish some similarities and differences with two other known problems: the pathwidth and the pursuit problem. We then present a distributed linear-time algorithm to solve the problem on trees. Last we give the solutions for some classes of graphs, in particular complete $d$-ary trees and grids.

**Keywords:** process number, pursuit problem, pebbling, rerouting, WDM

## 1 Introduction

Usually, when connexion requests are added or removed from a network, for instance a WDM network, the routing of older connexions is not modified. Hence it is likely that after some additions and removings, the overall use of resources is far from optimal. In particular, a new request may be rejected, even if it could be added up to a whole rerouting of older requests. So operators have to reorganise regularly the routing of all requests so as to make better use of the resources. Here we are interested in the problem of going from one routing to another without loss of services.

Given a network, a set of requests $I$ and two different routings for it in the network, $R_1$ and $R_2$, we want to switch from routing $R_1$ to routing $R_2$. Let $u$ and $v$ be two requests. We denote by $R_i(u)$ (resp. $R_i(v)$) the routing of request $u$ (resp. $v$) in $R_i, 1 \leq i \leq 2$. If $R_2(u) \cap R_1(v) \neq \emptyset$, i.e. the routing of request $u$ in $R_2$ uses resources already used by the routing of request $v$ in $R_1$, then obviously the request $v$ has to be rerouted before we can reroute request $u$. However, a request might be switched to an intermediate route, that uses available resources. For instance, the operator may reserve a dedicated wavelength in the network for temporary routes. We assume that each request cannot be switched to more than one temporary route, that is the next routing of a request routed on a temporary route has to be its final routing. When a request that was previously switched to a temporary route reaches its final routing, then the freed resources can be used again, for another request. While independent switching of requests can be made simultaneously, we will consider, for matter of exposition, that only one request is switched per unit of time.

The problem is modelled as follows: we construct a directed graph $D = (V,A)$, where each vertex $u$ corresponds to one request, and there is an arc from vertex $u$ to vertex $v$ if and only if $R_2(u) \cap R_1(v) \neq \emptyset$. A vertex is said to be *processed* as soon as its corresponding request has been rerouted. We introduce the notion of Temporary Memory Unit (TMU): routing the request $u$ on an intermediate route corresponds to putting the vertex $u$ in a TMU. So a vertex can be processed if and only if all its outneighbours are either processed or in TMU's. Note that a vertex without any outneighbour can be processed at any time. There are two basic operations: process a vertex according to the preceding rule ; put a vertex in a temporary memory unit. Figure 1 shows the processing steps of a graph using one TMU.

We underline the fact that once placed in a TMU, a vertex cannot recover its original state: it has to be processed. Nevertheless, it can occupy its TMU as long as desired. Processing a vertex which occupies a TMU frees the TMU, so that it can immediately be used by another vertex. The digraph is said to be *processed* when all its vertices have been processed. The problem is hence to find a suitable order to

[†]This work has been partially funded by European project IST FET CRESCCO, the ACI-SI PRESTO and the CRC CORSO with France Telecom

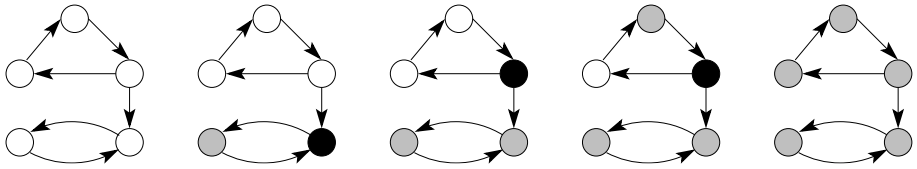


**Fig. 1:** Processing of a graph: processed vertices are in grey and vertices in TMU are in black.

process all the vertices. If we don't want to use any TMU, then it is clear that such a vertex ordering exists if and only if the digraph is acyclic; and in this case a processing order can be found in linear time. On the contrary, if we can use an arbitrary large number of TMU's, then we can first put all vertices in TMU's and then process them in any order. We aim at minimising the number of TMU's simultaneously in use. Notice that the problem is upper bounded by the *minimum forward vertex set number*, that is the smallest number of vertices which intersect all directed cycles.

The *process number* of $D$ is the minimum number of temporary memory units for which there exits a process strategy for $D$. We denote it by $p(D)$. A process strategy which uses $p$ (resp. at most $p$, resp. at least $p$) TMU's is called a $p$-process strategy (resp. $(\leq p)$-process strategy, resp. $(\geq p)$-process strategy). Remark that the digraphs we are dealing with can have loops. This may increase the process number by at most one, and it is straightforward to construct a loopless digraph $D'$ such that $p(D) = p(D')$. When $D$ is symmetric, we will work for convenience on the underlying undirected graph $G = (V,E)$.

This problem recalls the *pursuit problem*, introduced by Breisch [Bre67] and Parsons [Par78a, Par78b]. It has been well studied since, see for instance [MHG$^+$88, BS91, LaP93, KP86, BFFS02]. Given a graph $G = (V,E)$, the goal is to clear the graph using the minimal number of searchers. The basic operations are the following: place a searcher on a vertex ; remove a searcher from a vertex ; and move a searcher along an edge. An edge is said to be *contaminated* if it is capable of harbouring a fugitive. An edge is *cleared* by placing a searcher at one end (as a *guard*) and moving a second searcher along the edge. If all the other edges incident to the guarded endpoint are already clear, then the guard can clear the contaminated edge alone by moving along it. Once cleared, an edge remains clear as long as every path from it to a contaminated edge is blocked by at least one guard. If it is not the case, the edge is *recontaminated*. The graph is cleared as soon as all the edges are simultaneously clear. A *search strategy* is a sequence of pebbling operations that will clear the graph. The *search number* $s(G)$ of a graph is the minimum number of searchers for which a search strategy exists. The following result of Turner (which is proved in [EST94] and was first mentioned as a personal communication of Turner in [KP86]), establishes a link between the pursuit problem and the *vertex separator* (also known as the *pathwidth* problem, see for instance [DPS02]):

**Proposition 1 ([EST94])** *For any graph $G$, $vs(G) \leq s(G) \leq vs(G)+2$, where $vs(G)$ denotes the vertex separator of $G$.*

To provide some basic examples, let us mention that for any path $P$, $vs(P) = s(P) = 1$ while $p(P) = 2$ except if $P$ is of length at most 3 in which case $p(P) = 1$. If $\mathcal{S}$ is any star, then $vs(\mathcal{S}) = 1 = p(\mathcal{S})$ while $s(\mathcal{S}) = 2$.

## 2 Relation to known problems

**Proposition 2** *For any digraph $D$, $vs(D) \leq p(D) \leq vs(D)+1$.*

**Proof**. Consider a $p$-process strategy for $D$, and let $L$ be the order in which vertices are processed. Notice that if we stop the strategy just after the $i^{th}$ vertex has been processed, then any non-processed vertex having a processed neighbour must be in a TMU. As this is true for all $1 \leq i < |V|$, this exactly means that the vertex separator of $(D,L)$ is $p$, so $vs(D) \leq p(D)$.
Let $L$ be an ordering of the vertices of $D$, and say the vertex separator of $(D,L)$ is $vs$. We consider a process strategy for $D$ which consists of processing the vertices in the increasing order induced by $L$. The first vertex can be processed by putting its at most $vs$ neighbours in temporary memory units by definition. Suppose $i \leq 1$ vertices have been processed. We denote by $P$ the set of processed vertices, $M$ the set of vertices

in TMU's and $v$ the next vertex to be processed. If $v \notin M$, then as the vertex separator of $(D,L)$ is $vs$, $|M \cup (N^+(v) \setminus P)| \leq vs$, so we can put all the outneighbours of $v$ which are not in $M \cup S$ in TMU's and process $v$. This will use at most $vs$ TMU's simultaneously. If $v \in M$, then we have $|M \setminus \{v\} \cup (N^+(v) \setminus P)| \leq vs$, so putting all the neighbours of $v$ not in $M \cup S$ in TMU's will use at most (and possibly) $vs+1$ TMU's. □

As the vertex separator problem is APX [DKL87], the preceding result shows that the process number problem also is. The following proposition induces a construction which enforces that each parameter grows by 1.

**Proposition 3** *Let $G_1, G_2$ and $G_3$ be three connected graphs such that $vs(G_i) = vs, s(G_i) = s$ and $p(G_i) = p, 1 \leq i \leq 3$. We construct the graph $G$ by putting one copy of each of the $G_i$, and we add one vertex $v$ that has exactly one neighbour in each of the $G_i, 1 \leq i \leq 3$. Then $vs(G) = vs+1, s(G) = s+1$ and $p(G) = p+1$.*

**Proof**. We only show the proof for the process number. It is evident how to process $G$ with $p+1$ TMU's. Consider a $k$-process strategy for $G$. Without loss of generality, we can assume that $G_i$ is the $i^{th}$ graph of $G_1, G_2$ and $G_3$ to have $p$ of its vertices simultaneously in temporary memory units. Remark that once a vertex of $G_i$ has been put in a TMU, then there is always at least one vertex of $G_i$ in a TMU until $G_i$ is processed. So the first time $p$ vertices of $G_2$ occupy TMU's, $G_1$ must have been totally processed and no vertex of $G_3$ has been either processed or put in a TMU. Thus the vertex $v$ must be in a TMU. This gives that $k \geq p+1$. □

Using Proposition 3, one can show:

**Theorem 1** *Let $vs$ be a positive integer, $s \in \{vs, vs+1, vs+2\}$ and $p \in \{vs, vs+1\}$. There exists a graph $G_{vs}$ such that $vs(G_{vs}) = vs, s(G_{vs}) = s$ and $p(G_{vs}) = p$.*

So all the cases covered by Propositions 1 and 2 occur. These differences are created by reflecting an initial difference on a small graph. However, if we focus on structural properties of graphs that can be searched or processed with a fixed number of searchers or TMU's, we actually get important differences. In [MHG+88], all graphs with search number at most 3 are characterised. In particular, it is shown that a biconnected graph has search number at most 3 if and only if it is outerplanar and bipolar. On the opposite, the complete bipartite graph $K_{3,3}$ can be 3-processed. Until now, we obtained a minor-excluded characterisation for all graphs that can be 2-processed. We now pay more attention to the case of graphs that can be 3-processed.

# 3 Optimal algorithm on trees

Both the vertex separator and the search number problem can be solved in linear time on trees [Sko03, BFFS02]. We present here a distributed linear-time algorithm which computes the process number of trees and gives an optimal strategy. Let us first introduce some parameters and make a basic observation on them: let $T$ be a tree, and $v$ a vertex of $T$. If $u_1, \ldots, u_k$ are neighbours of $v$, we denote by $T_v(u_1, \ldots, u_k)$ the maximum subtree of $T$ rooted at $v$ such that the neighbours of $v$ are precisely $u_1, \ldots, u_k$. We denote by $p_v^F(T)$ (resp. $p_v^L(T)$) the minimum number of TMU's needed to process the tree $T$ under the constraint that the first (resp. last) step of the strategy must be to put $v$ in a TMU (resp. to process $v$). It is clear from the definition that for any vertex $v$ of any tree $T$, we have $p(T) \leq p_v^F(T), p_v^L(T) \leq p(T)+1$. The following lemma plays a key role in our algorithm.

**Lemma 1** *Consider a tree $T$ which is not a star. Let $v$ be a vertex of $T$, and $u_1, \ldots, u_k, k \geq 2$ be neighbours of $v$ in $T$. For any $i \in \{1, \ldots, k\}$, we denote by $p_i$ (resp. $p_i^F$, resp. $p_i^L$) the number $p(T_{u_i}(N(u_i) \setminus \{v\}))$ (resp. $p(T_{u_i}^F(N(u_i) \setminus \{v\}))$, resp. $p(T_{u_i}^L(N(u_i) \setminus \{v\}))$). We assume that the vector $(p_1, p_1^F, p_1^L, \ldots, p_k, p_k^F, p_k^L)$ is maximum for the lexicographic order among all the numberings of the neighbours. If $p_1^F \geq p_1^L$, then $p(T_v(u_1, \ldots, u_k)) = \max(2, p_1^L, p_2^F, p_3+1)$ else $p(T_v(u_1, \ldots, u_k)) = \max(2, p_1^F, p_2^L, p_3+1)$. Furthermore, we have $p_v^F(T_v(u_1, \ldots, u_k)) = \max(2, p_1^L, p_2+1)$ and $p_v^L(T_v(u_1, \ldots, u_k)) = \max(2, p_1^F, p_2+1)$.*

Here is how the algorithm goes. Each vertex $v$ has a list $L(v)$ of 4 values, and is uniquely identified. At the beginning, all the leaves are in the state `active`. All other vertices are in the state `ready`. Every leaf has its values initialised to `[0,0,0,0]`, values of the other vertices being `nil`. Each vertex having all

its neighbours but one `active` computes its own values. Let $u_1,\dots,u_k$ be the neighbours of $v$ which are `active`. Then $L(v)[1] = p(T_v(u_1,\dots,u_k)), L(v)[2] = p^F_v(T_v(u_1,\dots,u_k))$ and $L(v)[3] = p^L_v(T_v(u_1,\dots,u_k))$. So we compute them using Lemma 1, and some simple init rules (that may use $L(v)[4]$). So as to record an optimal strategy, the vertex also records the order of its neighbours when computing its values. A vertex which is `ready` stays `ready` until it has received information from all its neighbours but one. Then it computes its values, sends them and becomes `active`. If an `active` vertex receives information from its last neighbour, then if the identifier of the neighbour is lower than its own identifier it just ignores it. Otherwise it computes again its values, using all the information it has collected now, becomes `done` and sends a special message so that any `active` vertex becomes `done`. Eventually, either there will be a step at which exactly one vertex has not computed its values yet, or exactly two vertices compute their values at the last step, and the updating process ensures only one will compute its values using the information from all its neighbours. In any case, if $v$ is the last vertex to compute its values, we will have $L(v)[1] = p(T)$.

**Theorem 2** *The preceding algorithm computes all the values and records the strategy in sequential time $O(n)$, and distributively with $O(n)$ messages.*

We note that this result induces an upper bound (along with a corresponding strategy) for processing outerplanar graphs, since the dual graph of an outerplanar graph is a tree.

# 4 Exact process number for some classes of graphs.

**Theorem 3** $(i)$ *The process number of the complete bipartite graph $K_{m,n}$ is* $\min(m,n)$.
$(ii)$ *The process number of the complete binary tree of height $h$ is $h-1$, except if $h \in \{1,2\}$ in which case it is $h$.*
$(iii)$ *For any $d \geq 3$, the process number of the complete $d$-ary tree of height $h$ is $h$.*
$(iv)$ *The process number of the grid of size $m \times n$ is* $\min(m,n)+1$, *except if $n = m = 2$ in which case it is* 2.
$(v)$ *The process number of the pyramid of size $n$ is $\lceil \frac{n}{2} \rceil + 1$, except if $n \in \{2,3\}$ in which case it is $n-1$.*
$(vi)$ *Any triangulated outerplanar graph whose dual graph is a caterpillar of maximum degree* 3 *can be* 3*-processed.*

# References


[BFFS02] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Capture of an intruder by mobile agents. In *14th ACM Symp. on Parallel Algorithms and Architectures*, 2002.

[Bre67] R. L. Breisch. An intuitive approach to speleotopology. *Southwestern Cavers*, 6:72–78, 1967.

[BS91] D. Bienstock and P. Seymour. Monotonicity in graph searching. *J. Algorithms*, 12(2):239–245, 1991.

[DKL87] N. Deo, S. Krishnamoorthy, and M. A. Langston. Exact and approximate solutions for the gate matrix layout problem. *IEEE Transactions on Computer-Aided Design*, 6:79–84, 1987.

[DPS02] J. Díaz, J. Petit, and M. Serna. A survey of graph layout problems. *ACM Computing Surveys*, 34(3):313–356, 2002.

[EST94] J. A. Ellis, I. H. Sudborough, and J. S. Turner. The vertex separation and search number of a graph. *Inform. and Comput.*, 113(1):50–79, 1994.

[KP86] L. M. Kirousis and C. H. Papadimitriou. Searching and pebbling. *Theoret. Comput. Sci.*, 47(2):205–218, 1986.

[LaP93] A. S. LaPaugh. Recontamination does not help to search a graph. *J. Assoc. Comput. Mach.*, 40(2):224–245, 1993.

[MHG+88] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou. The complexity of searching a graph. *J. Assoc. Comput. Mach.*, 35(1):18–44, 1988.

[Par78a] T. D. Parsons. Pursuit-evasion in a graph. In *Theory and applications of graphs (Proc. Internat. Conf., Western Mich. Univ., Kalamazoo, Mich., 1976)*, pages 426–441. Lecture Notes in Math., Vol. 642. Springer, Berlin, 1978.

[Par78b] T. D. Parsons. The search number of a connected graph. In *Proceedings of the Ninth Southeastern Conference on Combinatorics, Graph Theory, and Computing (Florida Atlantic Univ., Boca Raton, Fla., 1978)*, Congress. Numer., XXI, pages 549–554, Winnipeg, Man., 1978. Utilitas Math.

[Sko03] Konstantin Skodinis. Construction of linear tree-layouts which are optimal with respect to vertex separation in linear time. *J. Algorithms*, 47(1):40–59, 2003.

# *Groupage sur un chemin orienté*

Séverine Petat [†] and Marie-Emilie Voge [‡]

*Projet Mascotte, I3S-CNRS/INRIA/Université de Nice-Sophia Antipolis, 2004 Route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex, France*

Nous présentons plusieurs approches, heuristiques, programmation linéaire mixte et en nombres entiers et décomposition de graphes, pour étudier un problème de groupage sur un chemin (orienté). On dispose d'un chemin et d'un ensemble quelconque de requêtes unitaires. L'objectif est l'installation d'un ensemble minimum de tubes (arcs joignant deux sommets du chemin) de capacité $C$, permettant d'acheminer toutes les requêtes. Ce problème est un problème de dimensionnement de réseaux qui reste NP-difficile malgré ses hypothèses restrictives.

**Keywords:** groupage, routage, network design, GMPLS, WDM

## 1 Introduction

Le groupage de trafic (ou grooming) est un principe qui consiste à agréger des flux dans des flux de débit supérieur. On retrouve cette idée en particulier dans les réseaux GMPLS [IET] et WDM [DR02, BCM03].

En termes de graphes, les entrées du problème de groupage sont modélisées par un graphe support $G$ muni de capacités, un ensemble de requêtes $R$ de débit donné entre des paires de sommets de $G$, un routage de ces demandes sur $G$ et un graphe "virtuel", le graphe des tubes disponibles, muni de coûts fixes et de capacités. Un tube représente une connexion entre deux sommets de $G$, c'est à dire un chemin fixé entre ces deux sommets. Une requête peut entrer (respectivement sortir) dans un tube uniquement au sommet origine (respectivement destination) du tube.

Grouper les demandes dans des tubes équivaut à trouver un ensemble de tubes tel que le chemin emprunté par chaque requête se décompose en une succession de tubes. Le groupage doit respecter des contraintes de capacités à deux niveaux différents. D'une part, la somme des capacités des tubes passant sur une arête du graphe support ne doit pas dépasser la capacité de cette arête. D'autre part, la somme des débits des requêtes empruntant un tube doit être inférieure à la capacité de ce tube.

Plusieurs critères d'optimisation ont été étudiés, comme la minimisation du nombre d'ADM (Add Drop Multiplexer) dans [BCM03], la minimisation des capacités totales ou maximales utilisées sur le réseau support, etc. Ici nous voulons minimiser le nombre de tubes nécessaires avec un chemin pour graphe support. Ce critère est celui considéré dans [BDPS03] où le graphe support est un anneau.

Nos hypothèses permettent de classer le groupage dans les problèmes de dimensionnement de réseau (Network Design). La littérature sur le Network Design est abondante et les résolutions proposées sont souvent basées sur une approche polyédrale comme dans [BG96, KM01], cependant des algorithmes d'approximation et des heuristiques ont aussi été étudiés en particulier dans [GMM95].

Nous définissons le groupage sur le chemin précisement en section 2.1. Nous avons abordé ce problème sous différents angles, comme la recherche de briques élémentaires (analogue à [BDPS03]) qui ne sera pas développée ici, ou la programmation linéaire en nombres entiers, ou enfin en proposant des heuristiques (section 3.2) dont certains résultats expérimentaux sont présentés en section 4.

## 2 Problème du groupage sur un chemin

### *2.1 Définition du problème*

Le problème de groupage, en général, requiert la donnée de trois éléments :

[†] recherche soutenue par le CRC CORSO avec France Telecom et le projet européen CRESCO
[‡] ACI Sécurité PRESTO avec CNRS

**Un graphe support** Ici, le graphe support considéré $G = (V,A)$ est un chemin orienté composé de $n$ sommets numérotés de 0 à $(n-1)$, $V = \{0,1,2,...,(n-1)\}$ et des arcs $A = \{(i,i+1) \forall\ 0 \leq i < n-1\}$. Pour simplifier, on supposera que la capacité des arcs est infinie, c'est-à-dire qu'autant de tubes que nécessaire peuvent emprunter un arc donné.

**Un ensemble de requêtes R** $R$ est un ensemble de paires ordonnées de sommets de $G$. Ici, les requêtes sont de la forme $(i,j)$ avec $i < j$ et $i,j \in V$. Pour simplifier, nous considérons des requêtes simples et unitaires, c'est à dire qu'il existe au plus une demande de valeur 1 entre deux sommets donnés.

**Un ensemble de tubes disponibles T** Tous les tubes de la forme $(i,j)$ avec $i < j$ et $i,j \in V$ sont autorisés en autant d'exemplaires que nécessaire. En revanche, leur capacité $C$, ou facteur de groupage, est uniforme et fait partie des données. Au plus $C$ requêtes peuvent donc emprunter le même tube. Le coût d'un tube est unitaire et fixe, il ne dépend pas du nombre de requêtes qui l'utilisent. En d'autres termes le coût d'un ensemble de tubes est son cardinal.

On souhaite installer un ensemble de tubes de cardinal minimum, sur le graphe support, constituant un groupage réalisable des demandes. Pour illustrer nos propos, la figure 1 montre deux groupages différents pour un graphe des requêtes donné, dans le cas où le graphe support est un chemin.

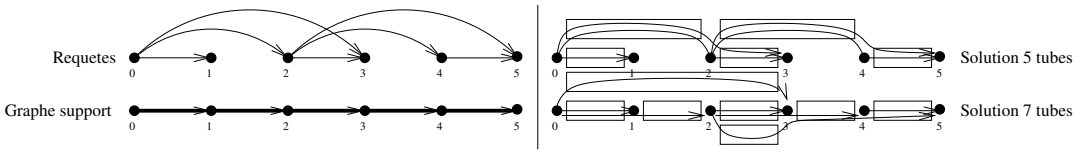



**FIG. 1:** Illustration du problème de groupage sur un chemin pour $C = 2$

Le groupage sur le chemin et sur l'anneau diffèrent uniquement par leurs graphes support. Les résultats de [BDPS03], comme la NP-complétude et les bornes sur le nombre de tubes optimal, n'utilisent pas les propriétés du graphe support et sont donc valables pour le chemin. En l'occurence, le nombre de tubes optimal est majoré par le nombre de requêtes $R$ et minoré par $\frac{2R}{C+1}$.

## 2.2 Programmation linéaire en nombres entiers

Le groupage sur le chemin s'interprète comme un problème de dimensionnement de réseau consistant, d'une façon générale, à trouver un ensemble d'arcs de coût minimum, nos tubes, entre des sommets connus pour router des requêtes données.

Ainsi, pour le groupage sur le chemin, deux formulations classiques en programmation linéaire en nombres entiers existent. Bien que la formulation arcs-chemins comporte un nombre exponentiel de variables, elle est plus efficace que la formulation compacte sommets-arcs (1), grâce à la génération de colonnes.

Nous avons utilisé le modèle sommets-arcs suivant pour calculer le nombre de tubes optimal et le modèle arcs-chemins avec un ensemble restreint de chemins pour en obtenir rapidement une borne supérieure.

Soient $T = (V,A_T)$ le graphe des tubes disponibles, $R$ l'ensemble des requêtes, $x_a$ le nombre d'exemplaires du tube $a$ utilisés et $f_a^{st}$ le flot associé à la demande $(st)$ circulant sur un exemplaire du tube $a$. $\delta^+(s)$ (resp. $\delta^-(s)$) est l'ensemble des tubes sortant (resp. entrant) du sommet $s$.

$$
\left\{
\begin{array}{lll}
Min & \displaystyle\sum_{a \in A_T} x_a & \\
t.q. & \displaystyle\sum_{a \in \delta^+(u)} f_a^{st} - \sum_{a \in \delta^-(u)} f_a^{st} = \left\{ \begin{array}{llll} 0 & si & u \neq s,\ u \neq t & \\ 1 & si & u = s & \\ -1 & si & u = t & \end{array} \right. & \forall u \in V,\ \forall (s,t) \in R \subseteq V^2 \\
 & \displaystyle\sum_{(st) \in R} f_a^{st} \leq C * x_a & \forall a \in A_T \\
 & f_a^{st} \in \{0,1\} & \forall (s,t) \in R,\ \forall a \in A_T \\
 & x_a \in IN & \forall a \in A_T .
\end{array}
\right. \quad (1)
$$

Nous avons formulé la conjecture suivante, qui permettrait de remplacer ce modèle en nombres entiers par un programme mixte (MIP) en relâchant l'intégrité du flot, et ainsi de le résoudre plus rapidement.

**Conjecture 2.1** *Le nombre de tubes optimal est identique, que le flot soit réel ou entier.*

Aucun des tests effectués n'a invalidé la conjecture, cependant il existe une instance pour laquelle la solution du programme relaxé n'est pas réalisable pour le programme entier. Pour la rendre réalisable, permuter le flot entre les requêtes (les rerouter) ne suffit pas, il faut en effet modifier l'ensemble de tubes.

### *2.3 Heuristiques*

Nous proposons deux heuristiques gloutonnes pour résoudre le groupage sur le chemin.

**Heuristique 1** La $i^{eme}$ itération de cette heuristique consiste à ajouter aux tubes choisis précédemment un ensemble de tubes permettant de router toutes les requêtes de longueur $i$ : si les tubes existants ne permettent pas de router une requête, alors le tube le plus court possible[§] permettant de la router est ajouté.

Prenons l'exemple 1 avec $C = 2$. La première itération ajoute un tube de longueur 1 pour chaque requête de longueur 1, i.e. les tubes $(0,1)$[¶], $(2,3)$, $(4,5)$. Dans chacun de ces tubes il reste la place de router une autre requête. A la deuxième itération, les tubes placés ne suffisent pas à router les requêtes $(0,2)$ et $(2,4)$, il faut donc ajouter de nouveaux tubes. Pour router la requête $(0,2)$ on peut ajouter soit le tube $(0,2)$, soit le tube $(1,2)$, l'heuristique choisit $(1,2)$ car c'est le plus court des deux. De même on ajoute $(3,4)$ qui permet avec le tube $(2,3)$ déjà placé de router la requête $(2,4)$. Ainsi le tube $(0,1)$ est complet car il contient les requêtes $(0,1)$ et $(0,2)$. De même pour le tube $(2,3)$. A la troisième itération, on obient la solution en 7 tubes donnée à la figure 1.

En pratique cette heuristique donne de bons résultats, mais une requête peut emprunter beaucoup de petits tubes, comme la requête $(2,5)$ dans l'exemple qui emprunte les trois tubes $(2,3)$, $(3,4)$ et $(4,5)$. Or dans un réseau de télécommunication réel, le respect de la qualité de service impose souvent un nombre de sauts maximum pour une route. Le principe de la deuxième heuristique permet de remédier à cet inconvénient.

**Heuristique 2** L'idée de la deuxième heuristique est de regrouper une longue requête $(i, j)$ avec des requêtes formant un plus court chemin[‖] entre $i$ et $j$ dans le graphe des requêtes privé de $(i, j)$. Un tube est ajouté lorsqu'entre deux sommets $C$ requêtes ont pu être regroupées ou qu'aucun regroupement n'est possible. Dans l'exemple 1 avec $C = 2$, il n'existe qu'un chemin, $(0,2)$ et $(2,3)$, entre les sommets 0 et 3 pour la requête $(0,3)$. Entre 0 et 2 les requêtes $(0,2)$ et $(0,3)$ sont donc regroupées, le tube $(0,2)$ est ajouté, de même pour $(2,3)$.

Si pour une requête on ne trouve pas de chemin, l'heuristique 2 la décompose en deux requêtes telles qu'il existe un chemin, le plus long possible en nombre d'arcs, pour l'une des deux seulement.

Supposons que la requête $(1,5)$ existe dans l'exemple 1. Il n'y a pas de chemin entre 1 et 5 mais il en existe entre 2 et 5 ($(2,5)$ ou $(2,4)$ et $(4,5)$) et aussi entre 4 et 5. Deux décompositions sont possibles, soit $(1,2)$ et $(2,5)$ car il existe au moins un chemin de 2 à 5 et aucun de 1 à 2, soit $(1,4)$ et $(4,5)$. L'heuristique choisit la première car la requête $(2,5)$ issue de cette décomposition est plus longue que $(4,5)$.

L'heuristique 2 appliquée à l'exemple 1 donne la solution en 5 tubes.

## 3 Résultats

Nous avons comparé les méthodes de résolution évoquées précédemment, les heuristiques (H1, H2), les programmes en nombres entiers sommets-arcs (NAI, programme 1) et arcs-chemins (API) et le programme mixte sommets-arcs (NAF). Pour la formulation arcs-chemins nous n'avons généré qu'un ensemble restreint de chemins (variables du programme) par requête, le nombre de tubes fourni par le solveur Ilog Cplex est donc une borne supérieure.

Nous avons effectué des tests pour différents facteurs de groupage sur des graphes de requêtes complets, i.e. pour tous sommets $i, j$ du graphe support tels que $i < j$ la demande $(ij)$ existe.

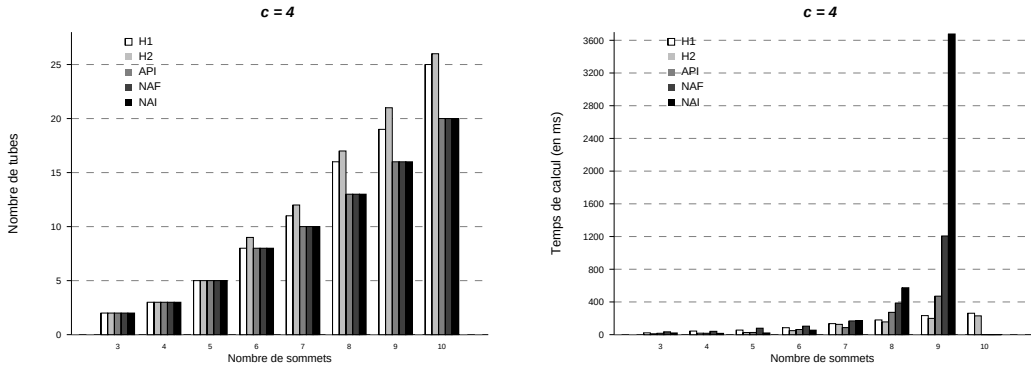
---

[§] Le plus court tube en nombre d'arcs.

[¶] La première coordonnée indique le sommet origine du tube et la seconde indique le sommet destination du tube.

[‖] Le plus court chemin en nombre de requêtes traversées.

Les graphiques suivants présentent le nombre de tubes obtenu par les différentes méthodes ainsi que les temps de calcul associés en millisecondes pour $C = 4$. Pour dix sommets les temps de calculs nécessaires aux résolutions des trois programmes mixtes dépassent l'heure et n'apparaissent pas sur le graphique.




Notons que la conjecture est vérifiée pour ces exemples, les valeurs obtenues par les programmes NAI et NAF sont identiques. Par contre, pour certains cas le calcul est plus rapide avec les variables de flots entières qu'avec les variables de flots réelles. Cependant la différence n'est pas significative et ne remet pas en cause l'intérêt de la relaxation NAF. Elle s'avère très intéressante en particulier pour un graphe à dix sommets et un facteur de groupage $C = 2$ puisque le temps de calcul passe de plus de 8h30 pour NAI à un peu plus de 55 minutes pour la relaxation NAF. Ces résultats montrent aussi que les heuristiques sont satisfaisantes car elles fournissent très rapidement des solutions proches de l'optimal.

# 4 Conclusion

Le problème de groupage que nous avons étudié, bien que fortement simplifié, reste difficile et long à résoudre pour des graphes dépassant dix sommets. D'où l'intérêt de nos heuristiques, qui fournissent très rapidement des solutions proches de l'optimal.

Beaucoup reste à faire sur ce problème, en particulier la démonstration de la conjecture et des tests supplémentaires des heuristiques sur des instances de grande taille. Le travail déjà effectué sur les briques élémentaires nous donne une infinité d'instances qui nous permettrons d'effectuer ces tests.

Enfin, le problème pour des graphes supports différents, comme les arbres reste à explorer.

# Références

[BCM03] J.-C. Bermond, D. Coudert, and X. Munoz. Traffic grooming in unidirectional wdm ring networks : the all-to-all unitary case. In *ONDM*, pages 1135–1153, 2003.

[BDPS03] J.-C. Bermond, O. DeRivoyre, S. Pérennes, and M. Syska. Groupage par tubes. In *Conference ALGOTEL2003, Banyuls, May 2003*, pages 169–174, 2003.

[BG96] D. Bienstock and O. Günlück. Capacited network design-polyhedral structure and computation. *Informs journal on Computing*, 8, 1996.

[DR02] R. Dutta and G. N. Rouskas. Traffic grooming in wdm networks : past and future. *IEEE Networks*, 16(6) :46–56, november/december 2002.

[GMM95] M. Grötschel, C.L. Monma, and M.Stoer. *Handbooks in Operations Research and Management Science*, volume 7 :Network Models. Elsevier publisher, 1995.

[IET] IETF. *http ://www.ietf.org/internet-drafts/draft-ietf-ccamp-gmpls-architecture-07.txt*.

[KM01] H. Kérivin and A.R. Mahjoub. On survivable network polyhedra. *submitted to Discrete Mathematics*, 2001.

# *Internet : mesures actives de l'Internet et analyse*

Mardi 11 mai 2005, 16:00 - 17:30

---

# *Describing and Simulating Internet Routes*

Jérémie Leguay, Timur Friedman, Kavé Salamatian

*LIP6 – CNRS and Université Pierre et Marie Curie*
*8, rue du Capitaine Scott, 75015 Paris, France*
*Tel. +33 1 44 27 71 34, Fax +33 1 44 27 53 53*
{*jeremie.leguay,timur.friedman,kave.salamatian*}*@lip6.fr*

This paper introduces relevant statistics for the description of routes in the internet, seen as a graph at the interface level. Based on the observed properties, we propose and evaluate methods for generating artificial routes suitable for simulation purposes. The work in this paper is based upon a study of over seven million route traces produced by CAIDA's *skitter* infrastructure.

**Keywords:** Network measurements, graphs, statistical analysis, modeling, simulation.

## 1 Introduction

Realistic modeling of routes in the internet is a challenge for network simulation. Until now, one has had to choose one of the three following approaches to simulate routes: (1) use the shortest path model, (2) explicitly model the internet hierarchy, and separately simulate inter- and intra-domain routing, or (3) replay routes that have been recorded with a tool like *traceroute*. All of these methods have serious drawbacks. The first method does not reflect reality since routes have not the same properties as shortest paths [Pax97], mainly because of routing policies [TGS01]. The second method is limited by our ability to explicitly simulate the internet hierarchy [TGJ+02, LAWD04]. Finally, the third method is not suitable if routes from a large number of sources are to be simulated. Today's route tracing systems employ at most a few hundred sources. CAIDA's *skitter* [HPMc02, CAI] infrastructure, for instance, produces an extensive graph suitable for simulations, but it based on routes from just thirty sources.

This paper's principal contribution is a new approach to modeling routes in the internet, one that does not share the drawbacks just described. We suggest using an actual measured graph of the internet topology, such as the graph generated by skitter. Between random chosen sources and destinations, we suggest generating artificial routes with a model chosen to reflect statistical properties of actual routes.

The remainder of this paper is organized as follows. Sec. 2 describes the data set that we have used. Sec. 3 proposes the set of statistical properties to describe routes in the internet. Sec. 4 proposes the models we use to simulate routes based on these properties. Sec. 5 evaluates those models, and Sec. 6 concludes the paper.

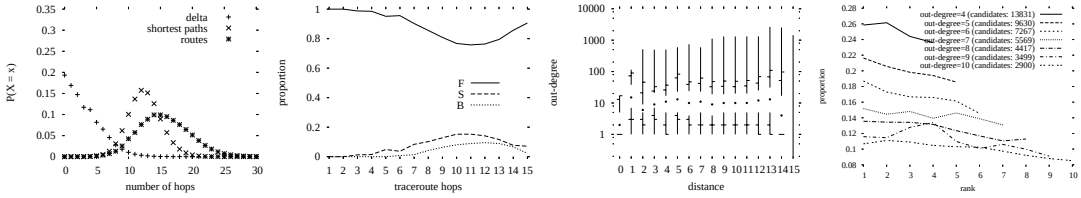
## 2 The data set

This study uses skitter data from July 2nd 2003. The data was collected from 23 servers targeting 594,262 destinations. We obtained the corresponding IP graph by merging the results of the 7,075,189 traceroutes conducted on that day. This graph captures the small-world, clusterized, and scale-free nature of the internet already pointed out for instance in numerous publications [JB02, FFF99]. In particular, the average distance is approximately 12.54 hops, and the degree distribution is well fitted by a power law of exponent 1.97.

## 3 Statistical properties of routes

This section presents a set of properties for statistical description of internet routes. These properties motivate the models of Sec. 4. Several properties have already been studied in previous work, and the work here

serves to evaluate and update them.




(a) Distributions of route lengths, shortest path lengths, and their differences

(b) Hop direction in 15-hop routes (F: Forward, S: Stable, B: Backward)

(c) Quantile plots for the out-degree of nodes along routes of length 15.

(d) Choice of next hop as a function of its degree ranking.

Figure 1: Statistical properties of internet routes.

## 3.1 Route lengths

It is well known that routes are not shortest paths: they are not optimal in general. Fig. 1(a) shows the length distributions of the routes in our data set, and of the corresponding shortest paths. It also shows the distribution of the difference (*delta*) between the length of a route and the corresponding shortest path. The mean length of 15.57 hops for routes in this data set fits closely Paxson's observations [Pax97, Pax96] on a data set from nine years prior. The shortest paths have a mean length of 12.55 hops (11.4 hops if the graph is considered to be undirected).

## 3.2 Hop direction

When a packet travels from one router to another, it may move closer to its destination, but also it may move farther, or it may move to an interface that is at the same distance from the destination as one it just left. Likewise, the distance from the source may increase, decrease, or stay constant. We will call these behaviors the *hop direction*. Note that in shortest paths hops always increase the distance from the source and decrease the distance to the destination. Fig. 1(b) shows the portion of forward, backward, and stable hops at each hop distance for routes of 15 hops (the most numerous ones). Note that, as one would expect, the first and last few hops are generally forward because there are few alternatives. On the contrary, in the core of the network a significant proportion of the hops do not go closer to the destination. This type of behavior has already been described as the product of policy-based routing in the core of the internet [TGS01].

## 3.3 Degree evolution along a route

Recent work has shown that in many real-world complex networks, most of the short paths between pairs of nodes tend to pass through the highest degree nodes [KYHJ02, GLM04]. These observations lead us to ask how the node degree evolves along a route. Fig. 1(c) shows[†] how node degree evolves for routes of length 15. It reveals that a typical route does not pass through the highest degree nodes, though a certain number of routes do pass through some very high degree nodes. There is a peak in median out-degree observable at distance 1. The median falls at distance 2, rises again, and then stays fairly flat out to distance 13, with a median degree of about 10. This leads us to the following interpretation: the hosts have low degree, they are connected at their first hop router to relatively high degree nodes which play the role of access points, and then packets are routed in a core network where the degree does not depend much on the distance from the source or from the destination.

Furthermore, one may wonder if there is a simple local rule that can be observed for the degree evolution. In particular, if there is a choice of next hop interface along a route, is there a correlation between the degree rank of an interface and its probability of being chosen? Fig. 1(d) plots the probability that a packet travels

[†] In Fig. 1(c), dots indicate the median. Vertical lines run from the min to Q1 and from Q3 to the max. Tick marks indicate the $5^{th}$, $10^{th}$, $90^{th}$ and $95^{th}$ percentiles.

to an interface's *i*-th ranked neighbor, where the neighbors are ranked from highest out-degree to lowest. This figure highlight such a correlation.

# 4 Route models

The previous section provides a set of simple statistical tools to capture some properties of routes in the internet. We now propose two models designed to capture these features.

## *4.1 Random deviation model*

The random deviation model is based upon the idea that a route usually follows a shortest path, but might occasionally deviate from it. We modeled this using one single parameter, $p$, the probability at any point of deviating from the current shortest path to the destination, if such a deviation is possible. We found $p = 0.2$ to work well. A random deviation route from source $s$ to destination $d$ is therefore based upon a shortest path $u$ from $s$ to $d$. At each hop, with probability $1 - p$, the route continues along $u$. But with probability $p$ it will, if possible, deviate off $u$ to another path.

## *4.2 Node degree model*

The basic idea is that a path which goes preferentially towards high degree nodes tends to see most nodes very rapidly [KYHJ02]. The node degree model is based upon a similar approach, as follows. Two paths are computed, one starting from the source and the other from the destination. The next node on the path is always the highest degree neighbor of the current node. The computation terminates when we reach a situation where a node is the highest degree neighbor of its own highest degree neighbor. One can show that this is the only kind of loop can occur. Then, one of two cases applies: either the two paths have met at a node, or they have not. In the first case, the route produced by the model is the discovered path (both paths are truncated at the meet up node, and are merged). In the second case, we compute a shortest path between the two loops, and then obtain the route by merging the two paths and this shortest path, removing any loops. This method has already been proposed [BLP04] as an efficient way to compute short paths in complex networks.

# 5 Evaluation

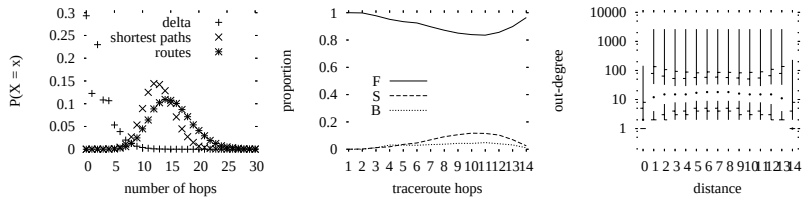
This section compares the performance of the random deviation and node degree models. For each model, we chose at least 60,000 (source, destination) pairs at random from amongst the nodes of the graph and generated an artificial route for each of them. We compute the same statistics on these routes as we had computed for actual routes in Sec. 3. Fig. 2 shows the statistics for each model.

Comparing the route length distributions, we find that both models generate distributions that are symmetric, average somewhat higher than the shortest path distribution, and have tails similar to the actual route length distribution shown in Fig. 1(a). Mean route length is 15.15 for the random deviation model and it is 14.96 for the node degree model. Lengths of paths generated with the node degree model tail off somewhat quicker than in reality, but the degree of fidelity is nonetheless remarkable given that the length distributions are not explicitly part of the model.

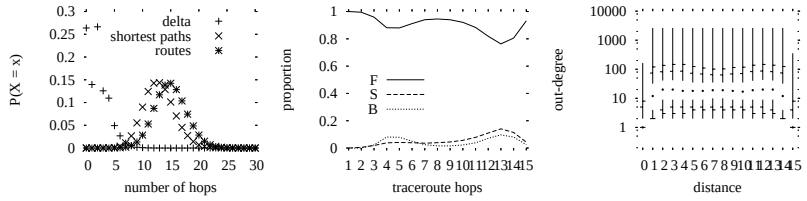
Looking at the hop directions for the most frequent route length, we found that the curves for the random deviation model better match the shapes of the curves for real routes shown in Fig. 1(b). Hops are mostly forward near the source, but dip to around 80% roughly ten hops out (whereas in reality the portion of forward hops dips to around 80% at eleven or twelve hops out). This is in marked contrast to hop directions produced by the node degree model because forward hops dip much sooner and a bit less steadily. But overall portions of forward, stable, and backward hops closely match reality for both models.

The node degree model shines compared to the random deviation model in capturing the evolution of the out-degree close to a route's source. Routes generated with this model show the peak in the out-degree before settling down to a median value. The peak is reached at distance 2 rather than at the first hop router.

Based upon this comparison to real routes, we can state that the random deviation and node degree models do a reasonable job of emulation, though each model captures some aspects better than others, and their strengths are different. Both models clearly out-perform the shortest path model.




(a) Lengths (r.d.) (b) Lengths (r.d) (c) Hop direction (r.d.)




(d) Out-degree (n.d.) (e) Hop direction (n.d.) (f) Out-degree (n.d.)

Figure 2: Experiments using the *random deviation model* (top) and the *node degree model* (bottom) on the undirected skitter graph using sources and destinations chosen at random from amongst all the nodes in the graph.

# 6 Conclusion and future work

The main contribution of this paper has been to propose new alternatives for the simulation of routes in the internet: the use of simple models that capture non-trivial statistical properties of routes. Future work along these lines might include the development of models that explicitly incorporate some additional characteristics, such as the clustering coefficienti, or that captures something of the dynamics of internet routes.

# References

[BLP04] E. Bampis, , M. Latapy, and F. Pascual. Computing short paths in scale-free networks. 2004. preprint.

[CAI] CAIDA. skitter. a tool for actively probing the Internet, `http://www.caida.org/tools/measurement/skitter/`.

[FFF99] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *Proc. ACM SIGCOMM*, 1999.

[GLM04] J.-L. Guillaume, M. Latapy, and C. Magnien. Comparison of failures and attacks on random and scale-free networks. In *Proceedings of the 8th International Conference on Principles of Distributed Systems (OPODIS)*, 2004. http://www.liafa.jussieu.fr/~latapy/Publis/.

[HPMc02] B. Huffaker, D. Plummer, D. Moore, and k claffy. Topology discovery by active probing. In *Proc. Symposium on Applications and the Internet (SAINT)*, January 2002.

[JB02] S. Jin and A. Bestavros. Small-world internet topologies: Possible causes and implications on scalability of end-system multicast. Tech. Report BUCS-2002-004, Boston Univ. Computer Sci., 2002.

[KYHJ02] B. J. Kim, C. N. Yoon, S. K. Han, and H. Jeong. Path finding strategies in scale-free networks. *Phys. Rev. E 65, 027103*, 2002.

[LAWD04] L. Li, D. Alderson, W. Willinger, and J. Doyle. A first-principles approach to understanding the internet's router-level topology. In *Proc. ACM SIGCOMM*, 2004.

[Pax96] V. Paxson. End-to-end routing behavior in the Internet. In *Proc. ACM SIGCOMM*, 1996.

[Pax97] V. Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM Trans. on Networking*, 5(5):601–615, October 1997.

[TGJ+02] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Network topology generators: Degree-based vs. structural. In *Proc. ACM SIGCOMM*, 2002.

[TGS01] H. Tangmunarunkit, R. Govindan, and S. Shenker. Internet path inflation due to policy routing. In *Proc. SPIE ITCom*, 2001.

# *How accurate are* `traceroute`*-like Internet mappings ?*

Luca Dall'Asta[1], Ignacio Alvarez-Hamelin[1,4], Alain Barrat[1], Alexei Vázquez[2] and Alessandro Vespignani[3]

[1]*Laboratoire de Physique Théorique, Bâtiment 210, Université de Paris-Sud, 91405 ORSAY Cedex France*
[2]*Nieuwland Science Hall, University of Notre Dame, Notre Dame, IN 46556, USA.*
[3]*School of Informatics and Department of Physics, University of Indiana, Bloomington, IN 47408, USA*
[4] *Facultad de Ingeniería, Universidad de Buenos Aires, Paseo Colón 850, C 1063 ACV Buenos Aires, Argentina*

Mapping the Internet generally consists in sampling the network from a limited set of sources by using `traceroute`-like probes. This methodology, akin to the merging of different spanning trees to a set of destinations, has been argued to introduce uncontrolled sampling biases that might produce statistical properties of the sampled graph which sharply differ from the original ones[1, 2, 3]. In this paper we study numerically how the fraction of vertices and edges discovered in the sampled graph depends on the particular deployments of probing sources. The results might hint the steps toward more efficient mapping strategies.

**Keywords:** Traceroute, Internet exploration, Topology inference

## 1 Introduction

In the absence of accurate Internet maps, researchers rely on a general strategy that consists in acquiring local views of the network from several vantage points and merging these views in order to get a presumably accurate global map. By using this strategy, a number of research groups have generated maps of the Internet [4, 5, 6, 7, 8], that have been used for the statistical characterization of the network properties. Defining $\mathcal{G} = (V,E)$ as the sampled graph of the Internet with $N = |V|$ vertices and $|E|$ edges, it is quite intuitive that the Internet is a *sparse* graph in which the number of edges is much lower than in a complete graph; i.e. $|E| \ll N(N-1)/2$. Equally important is the fact that the average distance, measured as the shortest path, between vertices is very small. This is the so called *small-world* property, that is essential for the efficient functioning of the network. Most surprising is the evidence of a skewed and heavy-tailed behavior for the probability that any vertex in the graph has degree $k$ defined as the number of edges linking each vertex to its neighbors. In particular, in several instances, the degree distribution appears to be approximated by $P(k) \sim k^{-\gamma}$ with $2 \leq \gamma \leq 2.5$ [9]. Evidence for the heavy-tailed behavior of the degree distribution has been collected in several other studies at the router and AS level [10, 11, 12, 13, 14] and have generated a large activity in the field of network modeling and characterization [15, 16, 17, 18, 19].

While `traceroute`-driven strategies are very flexible and can be feasible for extensive use, the obtained maps are undoubtedly incomplete. Along with technical problems such as the instability of paths between routers and interface resolutions [20], typical mapping projects are run from relatively small sets of sources whose combined views are missing a considerable number of edges and vertices [14, 21]. In particular, the various spanning trees are specially missing the lateral connectivity of targets and sample more frequently vertices and links which are closer to each source, introducing spurious effects that might seriously compromise the statistical accuracy of the sampled graph. These *sampling biases* have been explored in numerical experiments of synthetic graphs generated by different algorithms[1, 2, 3, 24].

It was shown in [22] that the map accuracy depends on the underlying network *betweenness centrality* [†] distribution. We substantiate the analytical finding of [22] with a throughout exploration of maps obtained varying the number of source-target pairs on networks models with different topological properties.

## 2 Optimization of mapping strategies

Let us consider sparse undirected graphs denoted by $G = (V, E)$. In particular, we will consider two main classes of graphs: *i) Homogeneous graphs* in which the degree distribution $P(k)$ has small fluctuations and a well defined average degree; *ii) Heterogeneous graphs* for which $P(k)$ is a broad distribution with heavy-tail and large fluctuations.

The most widely known model for homogeneous graphs is given by the classical Erdös-Rényi (ER) model [23]: in such random graphs $G_{N,p}$ of $N$ vertices, each edge is present in $E$ independently with probability $p$. We generated ER graphs with $p = 1/N$, where $N = 10^4$.

In opposition to the previous case, heterogeneous graphs are characterized by connectivity distributions spanning various orders of magnitude, with a heavy-tail at large $k$. While we do not want to enter the detailed definition of heavy-tailed distribution we have considered two classes of such distributions: (i) *scale-free* or Pareto distributions of the form $P(k) \sim k^{-\gamma}$ (RSF), and (ii) Weibull distributions (WEI) $P(k) = (a/c)(k/c)^{a-1}\exp(-(k/c)^a)$. In both cases, we have generated the corresponding random graphs by using the algorithm proposed by Molloy and Reed [25]. The parameters used are $a = 0.25$ and $c = 0.6$ for the Weibull distribution, and $\gamma = 2.3$ for the RSF case, and all graphs have $N = 10^4$ nodes.

It was shown in [22] that it is possible to have a general qualitative understanding of the efficiency of network exploration and the induced biases on the statistical properties. The quantitative analysis of the sampling strategies, however, is a much harder task that calls for a detailed study of the discovered proportion of the underlying graph and the precise deployment of sources and targets. In this perspective, very important quantities are the fraction $N^*/N$ and $E^*/E$ of vertices [‡] and edges discovered in the sampled graph, respectively. In our study the parameters of interest are the density $\rho_T = N_T/N$ and $\rho_S = N_S/N$ of targets and sources. An appropriate quantity representing the level of sampling of the networks is $\varepsilon = \frac{N_S N_T}{N}$, that measures the density of probes imposed to the system.

This finding hints toward a behavior that is determined by the number of sources and targets, $N_S$ and $N_T$. Any quantity is thus a function of $N_S$ and $N_T$, or equivalently of $N_S$ and $\rho_T$. This point is clearly illustrated in Fig. 1, where we report the behavior of $E^*/E$ and $N^*/N$ at fixed $\varepsilon$ and varying $N_S$ and $\rho_T$. The curves exhibit a non-trivial behavior and since we will work at fixed $\varepsilon = \rho_T N_S$, any measured quantity can then be written as $f(\rho_T, \varepsilon/\rho_T) = g_\varepsilon(\rho_T)$. Very interestingly, the curves show a structure allowing for local minima and maxima in the discovered portion of the underlying graph.

This feature can be explained by a simple symmetry argument. The model for `traceroute` is symmetric by the exchange of sources and targets, which are the endpoints of shortest paths: an exploration with $(N_T, N_S) = (N_1, N_2)$ is equivalent to one with $(N_T, N_S) = (N_2, N_1)$. In other words, at fixed $\varepsilon = N_1 N_2/N$, a density of targets $\rho_T = N_1/N$ is equivalent to a density $\rho'_T = N_2/N$. Since $N_2 = \varepsilon/\rho_T$ we obtain that at constant $\varepsilon$, experiments with $\rho_T$ and $\rho'_T = \varepsilon/(N\rho_T)$ are equivalent obtaining by symmetry that any measured quantity obeys the equality $g_\varepsilon(\rho_T) = g_\varepsilon\left(\frac{\varepsilon}{N\rho_T}\right)$. This relation implies a symmetry point signaling the presence of a maximum or a minimum at $\rho_T = \varepsilon/(N\rho_T)$. We therefore expect the occurrence of a symmetry in the graphs of Fig.1 at $\rho_T \simeq \sqrt{\varepsilon/N}$. Indeed, the symmetry point is clearly visible and in quantitative good agreement with the previous estimate in the case of heterogeneous graphs. On the contrary, homogeneous underlying topology have a smooth behavior that makes difficult the clear identification of the symmetry point. Moreover, unique shortest path probes create a certain level of correlations in the exploration that tends to hide the complete symmetry of the curves.

The previous results imply that at fixed levels of probing $\varepsilon$ different proportions of sources and targets may achieve different levels of sampling. This hints to the search for optimal strategies in the relative

[†] The betweenness represents the all-to-all traffic situation.

[‡] The measured quantities have the symbol $*$, to distinguish from the original ones.

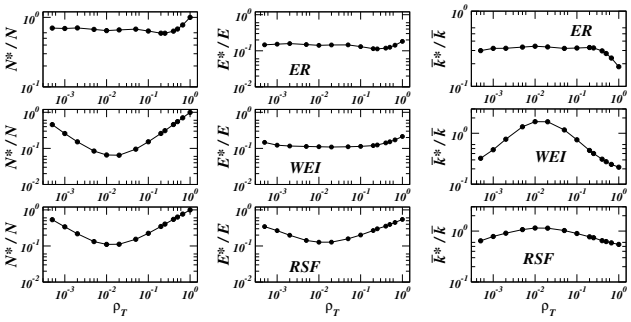



**Fig. 1:** Behavior as a function of $\rho_T$ of the fraction of discovered edges and vertices in explorations with fixed $\varepsilon$ (here $\varepsilon = 2$). Since $\varepsilon = \rho_T N_S$, the increase of $\rho_T$ corresponds to a lowering of the number of sources $N_S$. The plots on the right show the fraction of the normalized average degree $\overline{k}^*/\overline{k}$.

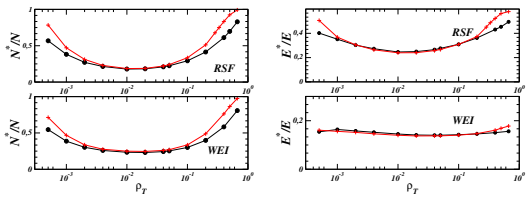



**Fig. 2:** Behavior as a function of $\rho_T$ of the fraction of discovered edges and vertices in explorations with fixed $\varepsilon$ (here $\varepsilon = 2$). The circles correspond to a random deployment of sources and targets while the crosses are obtained when sources and targets are vertices with lowest betweenness vertices.

deployment of sources and targets. The picture, however, is more complicate if we look at other quantities in the sampled graph. In Fig.1 we show the behavior at fixed $\varepsilon$ of the average degree $\overline{k}^*$ measured in sampled graphs normalized by the actual average degree $\overline{k}$ of the underlying graph as a function of $\rho_T$. The plot shows also in this case a symmetric structure. By comparing the data of Fig.1 we notice that the symmetry point is of a different nature for different quantities: the minimum in the fraction of discovered edges corresponds to the best estimate of the average degree. In other words, the best level of sampling is achieved at particular values of $\varepsilon$ and $N_S$ that are conflicting with the best sampling of other quantities.

The evidence purported in this section hints to a possible optimization of the sampling strategy. The optimal solution, however, appears as a trade-off strategy between the different level of efficiency achieved in competing ranges of the experimental setup. In this respect, a detailed and quantitative investigation of the various quantities of interest in different experimental setups is needed in order to pinpoint the most efficient deployment of source-target pairs depending on the underlying graph topology. While such a detailed analysis lies beyond the scope of the present study, an interesting hint comes from the analytical results of [22]: since vertices with large betweenness have typically a very large probability of being discovered, placing the sources and targets preferentially on low-betweenness vertices (the most difficult to discover) may have an impact on the whole process. This is what we investigate in Fig. 2 in which we report the fraction of vertices and edges discovered by either a random deployment of sources and targets or a deployment on the lowest-betweenness vertices. It is apparent that such a deployment allows to discover larger parts of the network. Of course the procedure used is unrealistic since identifying low-betweenness vertices is not an easy task. The usual correlation between connectivity and betweenness however indicates that the exploration of a real network could be improved by a massive deployment of sources using low-connectivity vertices.

# 3 Conclusions and outlook

The rationalization of the exploration biases at the statistical level provides a general interpretative framework for the results obtained from the numerical experiments on graph models. In general, exploration strategies provide sampled distributions with enough signatures to distinguish at the statistical level between graphs with different topologies. It is of major importance to define strategies that optimize the estimate of the various parameters and quantities of the underlying graph. In this paper we have shown that the proportion of sources and targets may have an impact on the accuracy of the measurements even if the number of total probes imposed to the system is the same. For instance, the deployment of a highly distributed infrastructure of sources probing a limited number of targets may result as efficient as few very powerful sources probing a large fraction of the addressable space [26]. The optimization of large network sampling is therefore an open problem that calls for further work aimed at a more quantitative assessment of the mapping strategies both on the analytic and numerical side.

# References

[1] A. Lakhina, J. W. Byers, M. Crovella and P. Xie, "Sampling Biases in IP Topology Measurements," Technical Report BUCS-TR-2002-021, Department of Computer Sciences, Boston University (2002).

[2] A. Clauset and C. Moore, "Accuracy and Scaling Phenomena in Internet Mapping," Phys. Rev. Lett. **94**, 018701 (2005).

[3] T. Petermann and P. De Los Rios, "Exploration of Scale-Free Networks - Do we measure the real exponents?," Eur. Phys. J. B **38** 201-204 (2004).

[4] The National Laboratory for Applied Network Research (NLANR), sponsored by the National Science Foundation. (see http://moat.nlanr.net/).

[5] The Cooperative Association for Internet Data Analysis (CAIDA), located at the San Diego Supercomputer Center. (see http://www.caida.org/home/).

[6] Topology project, Electric Engineering and Computer Science Department, University of Michigan (http://topology.eecs.umich.edu/).

[7] SCAN project at the Information Sciences Institute (http://www.isi.edu/div7/scan/).

[8] Internet mapping project at Lucent Bell Labs (http://www.cs.bell-labs.com/who/ches/map/).

[9] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On Power-law Relationships of the Internet Topology," ACM SIGCOMM '99, Comput. Commun. Rev. **29**, 251–262 (1999).

[10] R. Govindan and H. Tangmunarunkit, "Heuristics for Internet Map Discovery," Proc. of IEEE Infocom 2000, Volume 3, IEEE Computer Society Press, 1371–1380, (2000).

[11] A. Broido and K. C. Claffy, "Internet topology: connectivity of IP graphs," San Diego Proceedings of SPIE International symposium on Convergence of IT and Communication. Denver, CO. 2001

[12] G. Caldarelli, R. Marchetti, and L. Pietronero, "The Fractal Properties of Internet," Europhys. Lett. **52**, 386 (2000).

[13] R. Pastor-Satorras, A. Vázquez, and A. Vespignani, "Dynamical and Correlation Properties of the Internet," Phys. Rev. Lett. **87**, 258701 (2001); A. Vázquez, R. Pastor-Satorras, and A. Vespignani, "Large-scale topological and dynamical properties of the Internet," Phys. Rev. E .**65**, 066130 (2002).

[14] Q. Chen, H. Chang, R. Govindan, S. Jamin, S. J. Shenker, and W. Willinger, "The Origin of Power Laws in Internet Topologies Revisited," Proceedings of IEEE Infocom 2002, New York, USA.

[15] A. Medina and I. Matta, "BRITE: a flexible generator of Internet topologies," Tech. Rep. BU-CS-TR-2000-005, Boston University, 2000.

[16] C. Jin, Q. Chen, and S. Jamin, "INET: Internet topology generators," Tech. Rep. CSE-TR-433-00, EECS Dept., University of Michigan, 2000.

[17] S. N. Dorogovtsev and J. F. F. Mendes, *Evolution of networks: From biological nets to the Internet and WWW* (Oxford University Press, Oxford, 2003).

[18] P.Baldi, P.Frasconi and P.Smyth, *Modeling the Internet and the Web: Probabilistic methods and algorithms*(Wiley, Chichester, 2003).

[19] R. Pastor-Satorras and A. Vespignani, *Evolution and structure of the Internet: A statistical physics approach* (Cambridge University Press, Cambridge, 2004).

[20] H. Burch and B. Cheswick, "Mapping the internet," IEEE computer, **32**(**4**), 97–98 (1999).

[21] W. Willinger, R. Govindan, S. Jamin, V. Paxson, and S. Shenker, "Scaling phenomena in the Internet: Critically examining criticality," Proc. Natl. Acad. Sci USA **99** 2573–2580, (2002).

[22] L. Dall'Asta, I. Alvarez-Hamelin, A. Barrat, A. Vzquez, A. Vespignani "Traceroute-like exploration of unknown networks: a statistical analysis" in Proc of Combinatorial and Algorithmic Aspects of Networking and the Internet August 5 - 7, 2004, Banff, Canada, to apear in LCNS.

[23] P. Erdös and P. Rényi, "On random graphs I," Publ. Math. Inst. Hung. Acad. Sci. **5**, 17 (1960).

[24] J.-L. Guillaume and M. Latapy, "Relevance of Massively Distributed Explorations of the Internet Topology: Simulation Results," Proc. Infocom 2005 (to appear).

[25] M. Molloy and B. Reed, "A critical point for random graphs with a given degree sequence," Random Struct. Algorithms **6**, 161 (1995). M. Molloy and B. Reed, "The size of the giant component of a random graph with a given degree distribution," Combinatorics, Probab. Comput. **7**, 295 (1998).

[26] http://www.tracerouteathome.net/

# *Techniques de localisation géographique d'hôtes dans l'Internet*

Bamba Gueye *, Artur Ziviani **, Mark Crovella ***, Serge Fdida*

** Université Pierre et Marie Curie*
*Laboratoire d'Informatique de Paris 6*
*Paris France*
*** Laboratório Nacional de Computação Cientifica (LNCC)*
*Petrópolis - RJ - Brasil*
**** Department of Computer Science*
*Boston University*

L'inférence de la localisation géographique d'hôtes dans l'Internet permet l'émergence de nouvelles applications très variées. Jusqu'à présent, la localisation d'un hôte cible est fournie par la position des hôtes références, hôtes dont on connaît les positions géographiques. Ainsi le nombre d'endroits possibles où on peut localiser un hôte cible est égal au nombre d'hôtes références, conduisant ainsi à un espace discret de réponses. Nous proposons une technique de Localisation Géographique basée sur la Multilatération (LGM) pour inférer la position géographique d'un hôte cible. La multilatération permet d'obtenir un espace continu d'endroits possibles où on peut localiser un hôte contrairement aux approches précédentes. LGM transforme les mesures de délai en distance géographique `surestimée`, malgré les délais supplémentaires dûs aux congestions, et la non linéarité des chemins entre les hôtes. LGM utilise la multilatération avec ces distances géographiques surestimées pour inférer la position de l'hôte. Les résultats obtenus montrent que LGM est plus performante que les précédentes techniques de localisation et, de surcroît, est capable d'attribuer une zone de confiance à chaque hôte localisé.

**Keywords:** Localisation, multilatération, mesures de délai

## 1 Introduction

Avec le développement des nouvelles technologies de l'information, des nouveaux services ont fait leur apparition, notamment des services dits de "proximité" basés sur la localisation des clients. Nous pouvons citer comme exemple la publicité ciblée, la sélection automatique de la langue à la connexion, la diffusion de contenu suivant une politique géographique, et l'acceptation d'une transaction bancaire seulement à partir d'un endroit pré-établi. Les techniques de localisation basées sur des mesures tentent de déterminer la position géographique d'un hôte en se basant sur la connaissance de son adresse IP. C'est ainsi que [PS01] proposent d'utiliser la position de l'hôte référence, hôte dont on connaî t la position géographique, le plus proche en terme de délai comme possible localisation de l'hôte cible. Avec cette approche l'ensemble des endroits où on peut localiser un hôte cible est limité par le nombre d'hôtes références. Nous obtenons ainsi un ensemble discret de réponses.

Nous proposons une nouvelle technique de Localisation Géographique basée sur la Multilatération (LGM) pour résoudre ce problème. En effet, la multilatération permet d'estimer une position en utilisant un nombre suffisant de distances à partir de quelques points immobiles. Dès lors, elle fournit un ensemble continu d'endroits où on peut localiser la cible au lieu d'un espace discret de réponses. Connaissant la distance géographique *surestimée* entre la cible et chaque hôte référence, LGM fournit à l'instar du système de positionnement par satellites (GPS) [EM99] une estimation de localisation.

Pour l'évaluation de LGM nous avons utilisé les mesures de délai d'hôtes localisés à travers les États Unis et l'Europe de l'Ouest. Les résultats montrent que LGM est plus performante en précision que les

précédentes techniques de localisation géographique, basées sur des mesures. Ainsi l'erreur médiane de distance obtenue est inférieure à 25 km pour l'ensemble des hôtes localisés en Europe de l'Ouest, et 100 km pour ceux localisés aux États Unis. Nous avons remarqué que dans la plupart des cas, la zone de confiance que LGM fournit est raisonnable, car assimilable à la superficie d'un petit pays comme la Belgique en Europe ou un petit état comme le Maryland aux États Unis.

Ce papier est organisé comme suit. La section 2 dresse l'état de l'art du domaine et montre les contributions que LGM a apportées. Dans la section 3 nous présentons la technique LGM et ses différentes caractéristiques. La section 4 illustre les différents résultats obtenus en appliquant LGM. Enfin la section 5 conclut notre travail et présente quelques perspectives pour le long terme.

# 2 Techniques de localisation géographique

## *2.1 État de l'art*

La RFC 1876 [DVGD96] propose d'ajouter des informations de localisation dans les noms DNS (Domaine Name Server). Cependant cette proposition ne fut pas largement adoptée, car les administrateurs n'étaient pas trop motivés pour ajouter des enregistrements de localisation dans les bases de données DNS. Padmanabhan et Subramanian [PS01] quant à eux ont développé trois techniques pour inférer la localisation géographique d'un hôte. La première technique GeoTrack infère la localisation de l'hôte cible à partir de son nom DNS ou bien celui de l'hôte le plus proche. Les noms DNS dans Internet contiennent parfois certaines indications sur la localisation. Par exemple bcr1-so-2-0-0.Paris.cw.net indique un routeur localisé à Paris. Toutefois l'estimation de localisation peut être imprécise car le dernier routeur reconnaissable qui donne sa position comme estimation n'est pas forcement proche de la cible.

La deuxième technique, GeoCluster, se base sur l'hypothèse que tous les hôtes qui se trouvent à l'intérieur d'un même cluster sont co-localisés. Connaissant la localisation de quelques hôtes qui s'y trouvent, grâce à une base de données contenant des associations d'adresses IP et leurs localisations, elle déduit la localisation du cluster en entier. Son efficacité dépend de la véracité des informations se trouvant dans la base de données utilisée. Ces informations étant fournies par les utilisateurs sont peu fiables.

La troisième technique, GeoPing, est la plus proche de LGM, et exploite une possible corrélation entre délai et distance géographique. L'hypothèse de base de GeoPing est que des hôtes ayant un délai similaire par rapport à d'autres hôtes fixes (des serveurs sondes par exemple) tendent à être situés dans une même zone géographique. Ainsi, la localisation de l'hôte cible est assimilable à la position de l'hôte référence, qui a la mesure de délai la plus similaire à l'hôte dont on veut déterminer sa localisation. Le nombre d'endroits possibles, où on peut localiser l'hôte cible, est alors limité au nombre d'hôtes références, d'où un espace discret de réponses. Par conséquent, le nombre et le placement des hôtes références jouent un rôle important dans la précision de l'estimation de localisation [ZFdRD04]. L'amélioration de la technique GeoPing passe par une augmentation du nombre d'hôtes références. Dans la section 4 nous comparons LGM à l'approche DNS et à la technique GeoPing.

## *2.2 Contributions*

LGM est la première technique dans le domaine de la localisation à utiliser la multilatération pour inférer la position d'un hôte. Ses principales contributions sont :

- LGM établit une relation dynamique entre les adresses IP et leur localisation géographique grâce à des mesures de délai faites périodiquement entre les hôtes références.
- L'apport principal de la technique LGM est sa capacité à transformer les mesures de délai en distances géographiques surestimées, utilisées par la multilatération. Ainsi, en utilisant la multilatération, nous obtenons un espace continu d'endroits où on peut localiser un hôte contrairement aux autres techniques de localisation basées sur des mesures de délai.
- LGM fournit également une zone de confiance pour chaque hôte localisé offrant aux applications qui l'utilisent la possibilité d'évaluer la précision de l'estimation par rapport à leurs exigences.

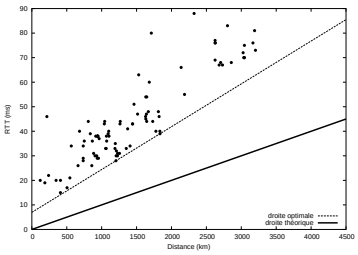



**FIG. 1:** Exemple montrant la relation entre distance géographique et délai.

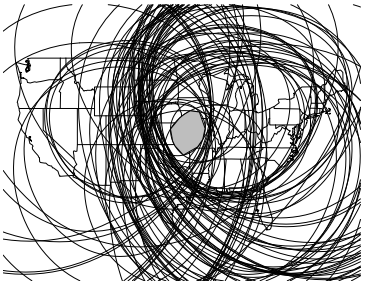


**FIG. 2:** Exemple de zone d'estimation de la localisation d'un hôte.

# 3 La Multilatération : idée générale

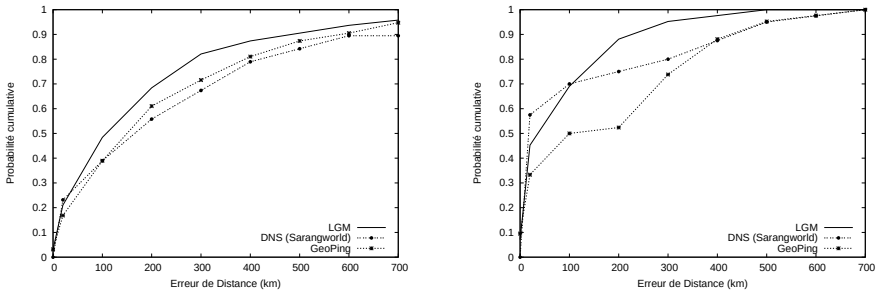
Soit un ensemble $\mathcal{H} = \{H_1, H_2, \ldots, H_K\}$ de $K$ hôtes références. Connaissant les délais entre un hôte cible et ces hôtes références, notre principal objectif est d'estimer les distances géographiques correspondantes. En effet, le délai de bout en bout est la somme des délai de propagation et de transmission, et des délais d'attente dans les files des routeurs.Au délai effectif mesuré s'ajoute donc un délai supplémentaire induit par ces distorsions. Ainsi, l'estimation de distance fournie par LGM est appelée par définition distance géographique surestimée, car étant la somme de la distance géographique réelle et de la distance induite par les distorsions.

La Figure 1 illustre un exemple choisi parmi les résultats décrits dans la section 4. L'axe des abscisses représente la distance géographique réelle et l'axe des ordonnées le délai mesuré entre un hôte référence $H_i$ et les autres hôtes références restants. Supposons qu'il existe un chemin linéaire entre l'hôte référence $H_i$ vers les autres hôtes références restants et que les données sont contraintes à aucun autre facteur à part le délai de propagation. On devrait avoir une droite de la forme $y = mx + b$ où $b = 0$ puisqu'il n'y a pas de délai additionnel et $m$ n'est rien d'autre que la vitesse de transmission des données dans le support physique. La "droite théorique" illustrée dans la Figure 1 montre ce cas. Cependant dans la réalité ce chemin linéaire existe rarement à cause des politiques de routage et des goulots d'étranglement. Ainsi pour modéliser la relation entre délai et distance géographique, nous définissons une "droite optimale" pour chaque hôte référence $H_i$ comme la droite $y = m_i x + b_i$ qui est la plus proche, mais en dessous de tous les points $(x,y)$ et dont l'ordonnée à l'origine *i*.e. $b_i$ n'est pas négative. Ainsi [GZCF04] montre comment les droites optimale et théorique sont construites. Chaque hôte référence utilise sa propre droite optimale pour convertir le délai obtenu, entre l'hôte cible et lui, en distance géographique surestimée.

# 4 Resultats

Pour nos expériences, nous avons utilisé des hôtes de RIPE [RIP] localisés en Europe Occidentale et de NLANR [NLA] localisés aux États Unis. Chaque ensemble contient respectivement 42 et 95 hôtes. Nous construisons la matrice de délai de chaque ensemble qui contient le RTT minimum entre les hôtes. Les hôtes références de chaque ensemble jouent à tour de rôle l'hôte cible à localiser et les hôtes références restants tentent de le localiser.

La Figure 2 montre un exemple extrait à partir de nos résultats obtenus et illustre la méthodologie de LGM. Elle illustre l'ensemble des 94 cercles utilisés pour estimer la localisation d'un hôte AMP situé à Lawrence, en Kansas aux États Unis. La région grise illustrée dans dans la figure 2 représente la zone d'intersection $\mathcal{R}$ de ces cercles. Le polygone approximant cette région $\mathcal{R}$ (*voir* [GZCF04]) est la zone de confiance que LGM associe à chaque estimation de localisation et son centre le point d'estimation de l'hôte cible. Après avoir trouvé la position d'estimation de chaque hôte cible, nous avons calculé l'erreur de distance qui représente la différence entre la position estimée et la position réelle de l'hôte cible $\tau$. Nous avons comparé nos résultats avec ceux obtenus par une méthode basée sur les noms DNS (*voir* le projet SarangWorld Traceroute [Sar]) et par GeoPing qui utilise un espace discret de réponses [PS01]. La Figure 3 montre la fonction de probabilité cumulative de l'erreur de distance obtenue en utilisant LGM, la




(a) Hôtes situés aux États Unis.

(b) Hôte situés en Europe de l'Ouest.

**FIG. 3:** Erreur de distance de LGM, de la méthode DNS et de GeoPing.

méthode basée sur le DNS et GeoPing. LGM dépasse en précision et la méthode basée sur le DNS et la technique GeoPing. Ainsi, l'erreur médiane de distance obtenue pour les hôtes références localisés aux E.U est inférieure à 100 km tandis que pour l'Europe Occidentale elle est inférieure à 25 km. Alors que pour la technique GeoPing cette erreur est de 150 km pour les E.U et 100 km pour l'Europe Occidentale.

# 5 Conclusion

Cet article montre une comparaison des différentes techniques de localisation. Les résultats obtenus illustrent que LGM est plus performante que les précédentes techniques de localisation géographique. LGM montre que la transformation des mesures de délai en distance géographique surestimée est possible. Transformer les mesures de délai en distance géographique surestimée avec précision est un challenge en raison de beaucoup de particularités inhérentes à l'utilisation d'Internet. La localisation à partir ou vers des hôtes situés un peu partout dans l'Internet, par exemple PlanetLab [pla], est envisagé pour nos travaux à long terme. De même que l'utilisation d'une base de données, où on enregistre les couples IP-Localisation des hôtes déjà localisés, afin d'éviter des mesures répétitives.

# Références

[DVGD96] Christopher Davis, Paul Vixie, Tim Goowin, and Ian Dickinson. A means for expressing location information in the domain name system. *Internet RFC 1876*, January 1996.

[EM99] Per Enge and Pratap Misra. Special issue on global positioning system. *Proceedings of the IEEE*, 87(1) :3–15, January 1999.

[GZCF04] Bamba Gueye, Artur Ziviani, Mark Crovella, and Serge Fdida. Constraint-based geolocation of internet hosts. In *Proc. of the ACM Sigcomm Internet Measurement Conference - IMC'2004*, Taormina, Sicily, Italy, October 2004.

[NLA] NLANR Active Measurement Project. `http://watt.nlanr.net/`.

[pla] *PlanetLab*. `http://www.planet-lab.org`.

[PS01] Venkata N. Padmanabhan and Lakshminarayanan Subramanian. An investigation of geographic mapping techniques for Internet hosts. In *Proc. of the ACM SIGCOMM'2001*, San Diego, CA, USA, August 2001.

[RIP] RIPE Test Traffic Measurements. `http://www.ripe.net/ttm/`.

[Sar] Sarangworld Traceroute Project. `http://www.sarangworld.com/TRACEROUTE/`.

[ZFdRD04] Artur Ziviani, Serge Fdida, José Ferreira de Rezende, and Otto Carlos Muniz Bandeira Duarte. Improving the accuracy of measurement-based geographic location of Internet hosts. *Computer Networks, Elsevier Science*, 2004. Accepted for publication.

# A CIDR Prefix Stopping Rule for Topology Discovery

Benoit Donnet and Timur Friedman

*Univeristé Pierre & Marie Curie, Laboratoire LiP6-CNRS*
*8, rue du Capitaine Scott*
*75015 Paris - France*

Recently, a first step towards a highly distributed IP-level topology discovery tool has been made with the introduction of the Doubletree algorithm. Doubletree is an efficient cooperative algorithm that allows the discovery of a large portion of nodes and links in the network while strongly reducing probing redundancy on nodes and destinations as well as the amount of probes sent. In this paper, we propose to reduce more strongly the load on destinations and, more essentially, the communication cost required for the cooperation by introducing a probing stop rule based on CIDR address prefixes.

**Keywords:** metrology, topology, traceroute, scalability, prefix

## 1 Introduction

This is a time when highly distributed applications are in full expansion. Among others, we can cite *SETI@home* [ACK+02] (probably the first one and the most famous) and *FOLDING@home* [LSSP02].

The network measurement community is not an exception to this fashion. Some measurement tools have already been released as daemons or screen savers. For instance, recently, we saw the introduction of *NETI@home* [SR04], an application collecting network performance statistics from end-systems.

Tools allowing for topology discovery at the IP level, based on *traceroute* [Jac89], are becoming more distributed. There is a number of well known systems, such as *skitter* [HPMC02], *Ripe NCC TTM* [GGK+01] or *NLANR AMP* [MBB00]. However, the need to increase the number of traceroute sources (the monitors) in order to obtain more complete topology measurement is felt [CM, LBCX03].

The idea of placing a tracerouting tool inside a screen saver, an idea first suggested by Jörg Nonnenmacher as reported by Cheswick et al. in [CBB00] should allow one to quickly obtain a structure of a considerable size. A publicly downloadable measurement tool, DIMES [Y. ct], has been released in September 2004.

Such a large structure has, however, inherent scaling problems. For example, if all the monitors trace towards the same destination, it could easily appear as a distributed denial of service (DDoS) attack. Furthermore, such a system must avoid consuming undue network resources. However, before the development of the Doubletree algorithm, little consideration had been given to how to perform large-scale topology discovery efficiently and in a network-friendly manner.

Doubletree [DRFC05] is based on the tree-like structure of routes in the internet. Routes leading out from a monitor towards multiple destinations form a tree-like structure rooted at the monitor. Similarly, routes converging towards a destination from a set of monitors form also a tree-like structure, but rooted at the destination.

Doubletree acts to avoid retracing the same routes through these structures. A monitor that applies Doubletree probes hop by hop so long as it encounters previously unknown interfaces. However, once it encounters a known interface, it stops, assuming it has touched an already known part of the tree, and the rest of the path to the root is also known. Backwards and forwards probing are thus used. These two probing schemes make use of stop sets. The first one, used during backwards probing and called the *local stop set*, consists of all interfaces already seen by that monitor. Forwards probing uses the *global stop set*

of (interface, destination) pairs accumulated from all monitors. This global stop set is shared amongst all the monitors, in order to keep track of what was already discovered. A monitor that implements Doubletree starts probing for a destination at some number of hops $h$ from itself. It first probes forwards from $h$ and then, backwards from $h-1$. The initial value $h$ is computed based on a probability $p$ of hitting a destination with the first probe sent $h$ hops towards that destination. For a range of $p$ values, Doubletree is able to reduce measurement load by approximately 70% while maintaining interface and link coverage above 90%.

One issue impeding a large-scale deployment of Doubletree is the communication overhead required by sharing the global stop set amongst monitors. For instance, tracing from only 24 monitors towards just 50,000 destinations with $p = 0.05$ will require roughly 20 megabytes [DFC05] for an uncompressed global stop set. This could lead to unacceptable scalability problems when increasing both number of monitors and number of destinations. To reduce the global stop set size, we investigate a lossy compression method: the *Bloom filters* [Blo70]. We found that using Bloom filters allows to reduce the size by a factor of 17.3 with very little loss in node and link coverage.

In this paper, we propose to replace a stopping rule based on destination addresses with a stopping rule based on the *CIDR address prefixes* [FLYV93] of destinations. The idea is to aggregate the destinations set into subnetworks, i.e. we filter each destination address and associate them to a subnetwork with the use of the CIDR address prefixes. Each monitor will probe all the destinations in each subnetwork. In addition to that, the global stop set will contain (interface, destination_prefix) pairs.

The rest of the paper is organized as follow: in Sec. 2, we present our methodology and our results and we conclude in Sec. 3.

# 2 Doubletree with CIDR

## 2.1 Methodology

Skitter data from the beginning of August 2004 serves as the basis of our work. This data set is composed of traceroutes gathered from 24 monitors scattered around the world. All the monitors share a common destination list of 971,080 IPv4 addresses. Each destination is probed in turn by each monitor. To cycle through the destination list, it takes usually three days. For our studies, in order to reduce computing time and hard disk space to a manageable level, we decided to work on a limited destination subset of 50,000 items randomly chosen amongst the whole set.

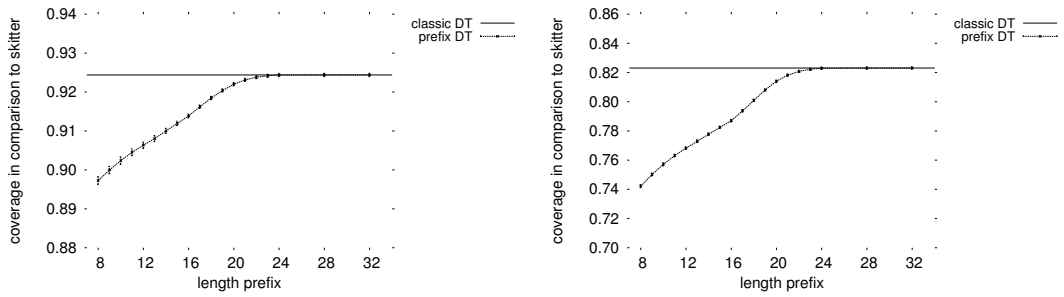
We conduct simulations based on the skitter data, applying Doubletree, as described in [DRFC05]. A single experiment uses traceroutes from all 24 monitors to a common set of 50,000 destinations chosen at random. Each data point represents the average value over fifteen runs of the experiment, each run using a different set of 50,000 destinations. No destination is used more than once over the fifteen runs. We determine 95% confidence intervals for the mean based, since the sample size is relatively small, on the Student $t$ distribution. These intervals are typically, though not in all cases, too tight to appear on the plots.

We use $p = 0.05$, which is a value that belongs to the range of $p$ values that our previous work identified as providing a good compromise between coverage accuracy and redundancy reduction. We test all prefixes length from /8 to /24, as well as lengths /28 and /32 (i.e. full IPv4 addresses). Each monitor probes each destination and records in the global stop set (interface, destination_prefix) pairs instead of (interface, destination) pairs. Compared to classic Doubletree, we only change the global stop set stop rule. Each result considered is compared, in Sec. 2.2, with classic Doubletree.

## 2.2 Results

Fig. 1 shows the main performance metric for a probing system: its coverage of the nodes and links in the network. It illustrates how the nodes and links coverage vary in function of the prefix length. A value of 1.0 (not shown here) would mean that application of Doubletree with the given prefix length had discovered exactly the same set of nodes and links as skitter. As already pointed out in our previous work, the use of Doubletree implies a small accuracy loss in the link and node coverage compared to skitter.

The lowest level of performance is reached for the /8 prefix. In our data set, on average, there are thirteen /8 subnetworks. As these subnetworks are quite large, monitors are stopped early in their probing. The loss




(a) Nodes

(b) Links

**Fig. 1:** Coverage when using prefixes

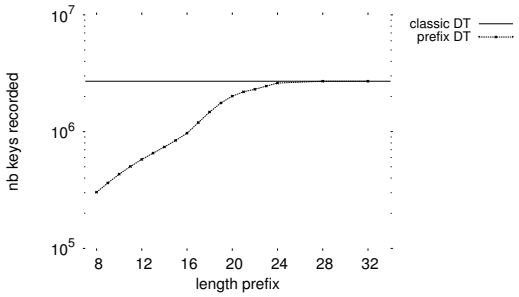



**Fig. 2:** Global stop set size when using prefixes

| | /8 | /16 | /24 | /32 |
|---|---|---|---|---|
| Prefix DT | 2.31 | 7.40 | 19.87 | 20.61 |
| Prefix DT w. BF | 0.361 | 0.689 | 1.192 | 1.192 |
| Classic DT | 20.61 | | | |
| Classic DT w. BF | 1.192 | | | |

**Tab. 1:** Global stop set size comparison (in MB)

of accuracy, however, is not so dramatic. The link coverage is 0,742 instead of 0,823 and node coverage is 0,897 instead of 0,924. We believe that the coverage level is still high due to the way exploration is performed by the first monitor to probe the network. Indeed, this first monitor uses an empty stop set (by definition of Doubletree) and is thus never stopped in its exploration. We further note that performance improves with prefix length until reaching nearly the same accuracy as classic Doubletree with /24 prefixes.

We believe that the loss of accuracy, compared to classic Doubletree, is essentially located within the subnetworks containing destinations but also inside the core of the network, where duplicated links (and the associated nodes) are missed due to the prefix based stopping rule. Typically, probes reach a very few number of destinations in each subnetwork but, in general, they are stopped at the ingress routers. We miss thus essentially the vast majority of destinations located in a given subnetwork. However more nodes and links may be missed if the network structure of the subnetwork is more complex, i.e. the subnetwork is not only composed of an ingress router that connects destinations with the rest of the network.

Fig. 2 shows the number of pairs recorded in the global stop set (in log-scale) as of a function of CIDR block prefixes. We can see that there is a strong reduction for low prefixes. For instance, if we consider a /8 prefix, the global stop set will only contain, in average, 302,854 keys. As each key is recorded as a 64 bit value, it corresponds to a stop set of around 2.31MB. Compared to the classic Doubletree, there is a compression factor of 8.9.

In addition to the mechanism presented in this paper, we could also implement the global stop set as a Bloom filter without losing much coverage accuracy [DFC05, Sec. 3]. Table 1 compared the global stop set implemented as a set of pairs and as a Bloom filter. It also compares classic Doubletree with the mechanism presented in this paper. We see that coupling the prefix based stop rule with a Bloom filter implementation of the global stop set introduces a very strong reduction in the global stop set size. For instance, using a /8 prefix stop rule gives, compared to classic Doubletree, a compression factor of 57.1.

# 3 Conclusion

In this paper, we present an improvement to the Doubletree probing algorithm. By using stop rules based on address prefixes, we show that we are able to strongly reduce communication between monitors while maintaining an acceptable level of coverage accuracy. Further, if we use this simple mechanism with a global stop set implemented as a Bloom filter, we still reduce the global stop set size to very low proportions.

The next prudent step for future work would be to test the algorithms that we describe here on an infrastructure of intermediate size, on the order of hundreds of monitors. We have developed a tool called *traceroute@home* that we plan to deploy in this manner.

# References

[ACK+02] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. SETI@home: An experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, Nov. 2002.

[Blo70] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[CBB00] B. Cheswick, H. Burch, and S. Branigan. Mapping and visualizing the internet. In *Proc. of the 2000 USENIX Annual Technical Conference*, San Diego, California, USA, Jun. 2000.

[CM] A. Clauset and C. Moore. Traceroute sampling makes random graphs appear to have power law degree distributions. arXiv:cond-mat/0312674 v3 8 Feb. 2004.

[DFC05] B. Donnet, T. Friedman, and M. Crovella. Improved algorithms for network topology discovery. In *Proc. of Passive and Active Measurement Workshop (PAM)*, Boston, USA, Mar. 2005.

[DRFC05] B. Donnet, P. Raoult, T. Friedman, and M. Crovella. Efficient algorithms for large-scale topology discovery. In *Proc. of ACM SIGMETRICS 2005*, Banff, Canada, Jun. 2005.

[FLYV93] V. Fuller, T. Li, J. Yu, and K. Varadhan. Classless inter-domain routing (CIDR): an address assignment and aggregation strategy. RFC 1519, Internet Engineering Task Force, Sept. 1993.

[GGK+01] F. Georgatos, F. Gruber, D. Karrenberg, M. Santcroos, A. Susanj, H. Uijterwaal, and R. Wilhelm. Providing active measurements as a regular service for ISPs. In *Proc. of PAM*, 2001.

[HPMC02] B. Huffaker, D. Plummer, D. Moore, and k Claffy. Topology discovery by active probing. In *Symposium on Applications and the Internet*, Nara City, Japan, Jan. 2002.

[Jac89] V. Jacobsen. traceroute, 1989.

[LBCX03] A. Lakhina, J. Byers, M. Crovella, and P. Xie. Sampling biases in IP topology measurements. In *Proc. of IEEE Infocom '03*, 2003.

[LSSP02] S. M. Larson, C. D. Snow, M. Shirts, and V. S. Pande. FOLDING@home and GENOME@home: Using distributed computing to tackle previously intractable problems in computational biology. In *Computational Genomics*, 2002.

[MBB00] A. McGregor, H.-W. Braun, and J. Brown. The NLANR network analysis infrastructure. *IEEE Communications Magazine*, 38(5):122–128, may 2000.

[SR04] C. R. Simpson, Jr. and G. F. Riley. NETI@home: A distributed approach to collecting end-to-end network performance measurements. In *Proc. of PAM*, 2004.

[Y. ct] Y. Shavitt et al. DIMES, ongoing project.

# *Ad Hoc & topologie dynamique*

Mardi 11 mai 2005, 18:00 - 19:30

---

# *Auto-stabilisation dans les réseaux ad hoc* †

N. Mitton[1], E. Fleury[1], I. Guérin-Lassous[1] et S. Tixeuil[2]

[1] *INRIA Ares - CITI, INSA de Lyon, 69621 Villeurbanne, France - prenom.nom@insa-lyon.fr*
[2]*LRI - CNRS UMR 8623 & INRIA Grand Large, 91405 Orsay, France - Sebastien.Tixeuil@lri.fr*

Dans cet article, nous choisissons un algorithme d'organisation de réseaux sans fil multi-saut, dit de *clusterisation*, et nous montrons que cet algorithme est auto-stabilisant. Nous proposons également certaines améliorations pour réduire le temps de stabilisation.

**Keywords:** réseaux sans fil multi-saut, clusterisation, auto-organisation, auto-stabilisation, extensibilité, DAG

## 1 Introduction

Les réseaux sans-fil multi-sauts sont des réseaux radio mobiles sans aucune infrastructure fixe. Leur fonctionnement est basé sur les principes d'auto-organisation et d'auto-stabilisation. Les protocoles actuels proposés pour un contexte multi-sauts fonctionnent correctement sur des réseaux de taille moyenne mais deviennent beaucoup trop coûteux sur de grands réseaux. Un défi majeur actuellement dans ce domaine concerne l'élaboration de solutions qui "passent à l'échelle", *i.e.* qui fonctionnent correctement sur de grands réseaux. Pour les protocoles de routage multi-sauts, une des solutions proposées pour tenter de résoudre ce problème de passage à l'échelle est d'organiser le réseau en regroupant géographiquement des nœuds proches en *clusters* (on parle alors de *clusterisation*) et en utilisant des approches différentes au sein des clusters et entre les clusters [KVCP97]. Dans la majorité des techniques de clusterisation proposées, chaque nœud choisit localement un chef. Tous les nœuds ayant choisi le même chef appartiennent au même cluster et le chef est appelé *cluster-head*. L'élection de ce chef peut se faire en se basant sur un critère d'identité (*p.ex.* le plus petit identifiant), sur un critère de connexité (*p.ex.* le degré ou le diamètre) ou sur un critère mixte identité/connexité. La maintenance de ces clusters repose elle aussi sur des critères fixes comme un diamètre ou rayon fixe ou un nombre constant de nœuds par cluster. Cependant, ces solutions ne sont pas adaptées aux grands réseaux puisque d'une part elles peuvent générer un nombre inutilement élevé de clusters et que d'autre part une toute petite modification de la topologie (comme le mouvement d'un nœud par exemple) peut conduire à de nouveaux calculs et reconstructions. Dans [MBF04], un nouveau critère de densité est proposé. Les auteurs montrent que cette métrique permet de limiter les reconstructions de clusters inutiles et se révèle plus stable que d'autres métriques face à la mobilité des nœuds.

Dans tous ces travaux de clusterisation, les auteurs testent la stabilité et la robustesse de leur algorithme face à la mobilité des nœuds par simulations, mais jamais selon une approche théorique. Dans cet article, nous cherchons à étudier la robustesse des algorithmes de clusterisation avec une approche théorique. Pour cela, nous partons de l'algorithme proposé dans [MBF04] puisqu'il présente de bonnes propriétés de robustesse et nous montrons que, sous certaines hypothèses, l'algorithme est auto-stabilisant. Nous proposons également l'ajout de certaines règles permettant d'améliorer la robustesse tout en conservant le caractère auto-stabilisant. Ces propriétés sont ensuite validées par simulations. Il faut noter que l'algorithme de [MBF04] est un point de départ et que cette étude pourrait aussi être appliquée à d'autres protocoles de clusterisation.

†Ces travaux sont soutenus par le FSN du ministère de la recherche via le projet FRAGILE de l'ACI Sécurité et Informatique.

# 2 Quelques rappels

Étant donnée la limitation sur le nombre de pages, il n'est pas possible de décrire notre étude dans son intégralité. Le lecteur intéressé par ce sujet peut se reporter à [MFGLT05]. Toutes les preuves des théorèmes et des lemmes suivants y sont décrites, ainsi que toutes les simulations réalisées. Dans cette section, nous donnons les notations utilisées par la suite, nous décrivons brièvement la métrique de densité choisie et la formation des clusters basée sur cette métrique et nous rappelons quelques pré-requis pour l'étude d'auto-stabilisation.

**Modèle et hypothèses.** Le système est modélisé par un ensemble de nœuds $V$. Chaque nœud $p$ a un identifiant unique et peut communiquer avec un sous-ensemble $N_p \subseteq V$ de nœuds défini par la portée du signal radio de $p$ ; $N_p$ est appelé le *voisinage* du nœud $p$, $p \notin N_p$. Nous supposons les liens de communication bidirectionnels : $q \in N_p$ ssi $p \in N_q$. Définissons $N_p^1 = N_p$ et, pour $i > 1$, le $i$-voisinage de $p$ $N_p^i = N_p^{i-1} \cup \{ r \mid (\exists q \in N_p^{i-1}, r \in N_q) \}$. De plus, nous supposons que la répartition des nœuds est telle qu'il existe une constante connue $\delta$ telle que pour tout nœud $p$, $|N_p| \leq \delta$. Ce contrôle de densité peut être réalisé en ajustant la portée de transmission des nœuds dans des zones trop peuplées.

**La métrique de densité et formation des clusters.** La notion de densité, introduite dans [MBF04] tente de caractériser l'importance « relative » d'un nœud dans son propre voisinage. Cette notion permet d'absorber de petits changements locaux de topologie, en considérant le rapport entre le nombre de liens et le nombre de nœuds dans un voisinage. La densité d'un nœud $p \in V$ est la suivante : $d_p = \frac{|e=(v,w) \in E \text{ tq. } w \in \{p\} \cup N_p \text{ et } v \in N_p|}{|N_p|}$. Chaque nœud calcule localement sa valeur de densité et la diffuse périodiquement à ses voisins. S'il possède une densité plus forte que tous ses voisins, il s'élit chef, sinon, il élit comme père son voisin de plus forte densité. En cas d'égalité, le plus petit identifiant départage les ex æquo. Le chef sera le nœud s'étant choisi comme son propre père.

**Auto-stabilisation** Une description plus complète de l'étude de l'auto-stabilisation d'un algorithme est donnée dans [HT04]. Nous gardons les mêmes hypothèses que dans [HT04], à savoir que nous disposons d'une implantation de CSMA/CA, pour l'accès au médium radio, qui satisfait les points suivants : il existe une constante $\tau > 0$ telle que la probabilité d'une transmission de trame sans collision est supérieure ou égale à $\tau$. Nous décrivons les algorithmes sous la forme de règles gardées : $G \rightarrow S$ représente une telle règle, où $G$ est un prédicat sur les variables locales d'un nœud, $S$ une affectation de ces mêmes variables locales. Si le prédicat $G$ (la garde) est vrai, l'affectation $S$ est exécutée, sinon elle est ignorée. L'exécution du système consiste pour chaque nœud à évaluer répétitivement ses règles gardées. Nous supposons que chaque règle activable est exécutée en un temps constant. Certaines variables des nœuds sont dites *partagées*. Suivant le schéma présenté en [HT04], les nœuds diffusent périodiquement les valeurs de leurs variables partagées. Nous supposons également que le schéma de [HT04] est utilisé pour obtenir sur chaque nœud $N_p$ et $N_p^2$.

# 3 Étude d'auto-stabilisation

## *3.1 Construction d'un DAG à hauteur constante*

Dans l'algorithme choisi, comme dans tout algorithme utilisant l'identifiant des nœuds comme critère de décision finale, le pire cas se rencontre quand tous les nœuds ont la même valeur de décision (comme le degré ou la densité) et que les identifiants des nœuds sont uniques et mal distribués. L'algorithme peut alors ne construire qu'un seul cluster dont le diamètre est aussi grand que celui du réseau, le temps de stabilisation dépendant de ce diamètre. Pour pallier cet inconvénient, il peut s'avérer utile de donner aux nœuds d'autres identifiants, choisis dans un espace de noms constant et plus petit, de façon à ce que les identifiants soient localement uniques. Un DAG (Directed Acyclic Graph) peut alors être construit à partir de ces nouveaux noms en orientant les arêtes entre les voisins du plus grand identifiant vers le plus petit. Notre construction de DAG est basée sur la technique aléatoire décrite dans [HT04], mais utilise un espace

de noms beaucoup plus petit $\gamma$ ($|\gamma| = \delta^6$ dans [HT04], tandis que $\delta^2$ ou même $\delta$ est suffisant dans notre cas). Soit $Id_p$ une variable partagée appartenant au domaine $\gamma$, correspondant au *nom* du nœud $p$ dans le DAG. Soit $Cids_p = \{\bowtie Id_q \mid q \in N_p\}$, une variable pour déterminer le nom des nœuds voisins, où $\bowtie Id_q$ réfère à la copie en cache de la variable partagée $Id_q$ sur le nœud $p$. Supposons que $\mathsf{random}(S)$ choisit avec une probabilité uniforme un élément de $S$. Le nœud $p$ utilise la fonction suivante pour calculer $Id_p$ :

$$\mathsf{newId}(Id_p) = \begin{cases} \bowtie Id_p & \text{si } \bowtie Id_p \notin Cids_p \\ \mathsf{random}(\gamma \setminus Cids_p) & \text{sinon} \end{cases}$$

L'algorithme de construction d'un DAG à hauteur constante est le suivant : $\mathsf{N1} : \mathit{true} \rightarrow Id_p := \mathsf{newId}(Id_p)$.

**Theorème 1.** *L'algorithme N1 stabilise avec probabilité* 1 *en un temps constant vers un DAG de hauteur inférieure ou égale à* $|\gamma| + 1$.

Un compromis est à faire pour déterminer le paramètre $\gamma$ : plus la valeur de $|\gamma|$ est grande, plus le temps de convergence de N1 est faible mais plus la hauteur du DAG est importante. Une hauteur de DAG importante augmente le temps de stabilisation des règles R1 et R2 (*cf.* Section 3.2).

## 3.2 Formation de clusters suivant la densité

Chaque nœud $p$ maintient deux variables partagées : $d_p$ et $\mathcal{H}(p)$ où $d_p$ est la densité du nœud $p$ et $\mathcal{H}(p)$ son cluster-head. Nous définissons $\prec$ tel que $p \prec q$ si et seulement si $d_p < d_q$ ou $(d_p = d_q) \wedge (Id_q < Id_p)$. $\max_\prec$ représente la valeur du maximum de $\prec$. Quand un nœud $p$ calcule $\prec$ ou $\max_\prec$, il utilise les valeurs de cache de son voisinage ($\bowtie Id_p = Id_p$ et $\bowtie d_p = d_p$).

Nous définissons maintenant la fonction d'élection du cluster-head :

$$\mathsf{clusterHead} = \begin{cases} Id_p & \text{if } \forall q \in N_p, q \prec p \\ \mathcal{H}(\max_\prec \{q \in N_p\}) & \text{sinon} \end{cases}$$

L'algorithme s'exécute comme suit :

$$\begin{array}{ll} \mathsf{R1}: & \mathit{true} \rightarrow d_p := \mathsf{density} \\ \mathsf{R2}: & \mathit{true} \rightarrow \mathcal{H}(p) := \mathsf{clusterHead} \end{array}$$

**Lemme 1.** *Partant de n'importe quelle configuration initiale, chaque nœud $p$ a une valeur de densité* correcte *$d_p$ en un temps borné constant.*

**Lemme 2.** *Partant de n'importe quelle configuration initiale, chaque nœud $p$ a une valeur* correcte *pour $\mathcal{H}(p)$ en un temps borné constant.*

**Remarque :** Nous sommes également en train d'établir une borne supérieure sur le temps de convergence de l'algorithme.

## 3.3 Améliorer la stabilité

Afin d'améliorer la stabilité de notre algorithme, nous introduisons des règles de décision supplémentaires lors de l'élection du cluster-head. Premièrement, lorsque deux nœuds $u$ et $v$ sont en compétition pour devenir le père d'un troisième nœud $w$, l'élu sera prioritairement celui choisi précédemment par $w$ (s'il existe), et sinon celui ayant le plus faible identifiant de DAG (comme défini dans la Section 3). Cette nouvelle règle préserve la structure de notre preuve de stabilisation, puisqu'il suffit de définir $\prec$ comme $p \prec q$ si et seulement si $(d_p < d_q)$ ou $(d_p = d_q) \wedge (\mathcal{H}(q) = Id_q) \wedge (\mathcal{H}(p) \neq Id_p)$ ou $(d_p = d_q) \wedge (\mathcal{H}(p) \neq Id_p) \wedge (\mathcal{H}(q) \neq Id_q) \wedge (Id_q < Id_p)$. De plus, la hauteur du nouveau DAG$_\prec$ est similaire à celle de l'ancien. Deuxièmement, si un nœud $p$ est 1-voisin de deux cluster-heads différents $u$ et $v$ (qui ne sont donc pas voisins), il initie une fusion entre les clusters de $u$ et $v$ : si $p$ a choisi $v$ comme cluster-head, cela signifie que $u \prec v$ et donc $v$ reste cluster-head contrairement à $u$. Cela assure que *(i)* un cluster-head n'est pas trop excentré dans son propre cluster, *(ii)* un cluster a un diamètre supérieur ou égal à 2, et *(iii)* que deux cluster-heads sont distants d'au moins trois sauts. Encore une fois, ces règles préservent notre preuve de stabilisation puisqu'il suffit d'utiliser la fonction alternative :

$$\mathsf{clusterHead} = \begin{cases} Id_p & \text{si } (\forall q \in N_p, q \prec p) \wedge (\forall q \in N_p^2 | \mathcal{H}(q) = Id_q \implies q \prec p) \\ \mathcal{H}(\max_\prec \{q \in N_p\}) & \text{sinon} \end{cases}$$

# 4 Simulations

D'après l'hypothèse réalisée sur l'accès au médium radio, nous pouvons dire que chaque nœud est en mesure de diffuser localement une trame et d'en recevoir une de chacun de ses voisins en un temps borné $\Delta(\tau)$, appelé une *étape*. Après une étape, chaque nœud connaît ses 1-voisins. Après deux étapes, il connaît ses 2-voisins et peut calculer sa valeur de densité et après trois étapes, il connaît son père. Le nombre d'étapes nécessaires à un nœud pour connaître l'identité de son cluster-head dépend directement de la distance qui l'en sépare et est borné par la profondeur de l'arbre. Pour la construction du DAG et l'attribution des *noms* des nœuds, chaque nœud se choisit aléatoirement un *nom* entre 0 et $\delta^2$. Il compare alors ce nom à celui de ses voisins. Si deux voisins ont le même nom, le nœud dont l'identifiant "normal" est le plus petit se choisit un autre nom et ainsi de suite jusqu'à ce qu'il n'existe aucune paire de nœuds voisins portant le même nom.

Afin d'évaluer les performances de l'algorithme et d'estimer l'apport de l'introduction d'un DAG, nous avons réalisé des simulations où les nœuds sont soit déployés aléatoirement suivant un processus de Poisson dans un carré $1 \times 1$ avec plusieurs valeurs de portée de transmission $R$ et d'intensité du processus $\lambda$ soit disposés sur une grille. Chaque statistique est la moyenne sur 1000 simulations. Nous pouvons d'abord noter que construire un DAG n'est pas coûteux, puisque cela prend en moyenne deux étapes. Nous avons ensuite évalué le nombre de cluster-heads par unité de surface, la hauteur de l'arbre de clusterisation et l'excentricité du cluster-head dans son cluster en nombre de sauts. Pour le déploiement aléatoire, l'excentricité moyenne d'un cluster-head et la hauteur de l'arbre varient peu en fonction de la portée radio, donc un nœud connaît l'identité de son cluster-head en un temps faible et constant. Dans ce cas, l'utilisation d'un DAG n'est d'aucune aide car les valeurs de densité étant uniformément réparties et rarement égales, l'identifiant du nœud est rarement utilisé pour sélectionner le père. Dans le cas de la distribution sur une grille, les identifiants sont choisis croissant de la gauche vers la droite et du bas vers le haut. Tous les nœuds intérieurs ont alors la même densité et le choix du père se fait en fonction de l'identifiant. Dans un pareil cas, la construction du DAG est utile (cf Tab 1) car elle permet de réduire drastiquement le nombre d'étapes nécessaires avant la stabilisation (puisqu'elle réduit la hauteur des arbres de clusterisation).

| | $R = 0.05$ | | $R = 0.08$ | | $R = 0.1$ | |
|---|---|---|---|---|---|---|
| | Avec DAG | Sans DAG | Avec DAG | Sans DAG | Avec DAG | Sans DAG |
| # clusters | 52.8 | 1.0 | 29.3 | 1.0 | 18.5 | 1.0 |
| excentricité | 3.4 | 29.1 | 4.1 | 19.1 | 3.6 | 6.5 |
| hauteur d'arbre | 3.7 | 83.4 | 4.7 | 100.5 | 4.5 | 32.1 |

**TAB. 1:** Caractéristiques des clusters dans une grille.

Pour tester les règles introduites en Section 3.3, nous avons réalisé des simulations où les nœuds bougent aléatoirement à des vitesses aléatoires. Nous avons calculé le pourcentage de cluster-heads restant cluster-heads après chaque série de mouvement. Pour une mobilité de nœud entre 0 et $1.6m/s$ (piétons), ce pourcentage est d'environ 82% avec nos règles supplémentaires contre 78% sans. Pour une mobilité de nœud entre 0 et $10m/s$ (voitures), ce pourcentage est de 31% avec les nouvelles règles contre 25% sans.

# Références

[HT04] T. Herman and S. Tixeuil. A distributed tdma slot assignment for wireless sensor networks. In *Proceedings of the First Workshop on Algorithmic Aspects of Wireless Sensor Networks (AlgoSensors'2004)*, number 3121 in Lecture Notes in Computer Science, pages 45–58, Turku, Finland, July 2004. Springer-Verlag.

[KVCP97] P. Krishna, N. H. Vaidya, M. Chatterjee, and D. K. Pradhan. A cluster based approach for routing in dynamic networks. In *ACM SIGCOMM*, pages 49–65, April 1997.

[MBF04] N. Mitton, A. Busson, and E. Fleury. Self-organization in large scale ad hoc networks. In *3nd Med-Hoc-Net*, june 2004.

[MFGLT05] Nathalie Mitton, Eric Fleury, Isabelle Guerin-Lassous, and Sebastien Tixeuil. Self-stabilization in self-organized multihops wireless networks. In *WWAN'05*, june 2005.

# *Une analyse de la sélection des MPR dans OLSR*

A. Busson[1], N. Mitton[2] and E. Fleury[2]

[1] *IEF - CNRS - Orsay - anthony.busson@ief.u-psud.fr*
[2] *INRIA/ARES - CITI/INSA Lyon - nom.prenom@insa-lyon.fr*

OLSR est un protocole de routage pour réseaux ad-hoc. Il introduit le concept de Multi-Point Relais (MPR) pour minimiser le trafic de contrôle et limiter les effets de bords de la diffusion dans un tel réseau. Nous nous intéressons ici aux performances de la sélection de ces MPR. Nous en analysons le nombre et la répartition spatiale.

**Keywords:** ad hoc, OLSR, MPR, analyse, distribution de Palm

## 1 Introduction

De nos jours, de nombreuses technologies pour les réseaux sans fils apparaissent, telles 802.11[†], hiperlan ou bluetooth[‡]. Avec elles, de nouvelles possibilités ont émergé comme de permettre aux entités du réseau de communiquer sans infrastructure physique. Deux nœuds n'étant pas à portée radio l'un de l'autre utilisent alors des noeuds intermédiaires pour relayer leurs données. Ceci suppose l'utilisation d'un protocole de routage permettant à un noeud de trouver une route valide vers n'importe quelle destination. Du fait des caractères intrinsèques de tels réseaux où les nœuds peuvent à tout moment disparaître ou se déplacer, les protocoles de routage classiques du monde filaire ne sont pas adaptés. Les algorithmes doivent être adaptatifs tout en limitant les échanges coûteux en énergie. OLSR [1], protocole récemment standardisé du groupe MANET[§] de l'IETF est un protocole de routage pro-actif pour réseaux ad-hoc *i.e.* il maintient en permanence une vue de la topologie du réseau sur chaque nœud lui permettant de calculer une route disponible à tout moment vers n'importe quelle entité. Il introduit le concept de Multi-Point Relais (MPR) afin de minimiser le trafic de contrôle et limiter les messages redondants lors d'une diffusion à tous les nœuds du réseau. Chaque nœud sélectionne ses MPR parmi les nœuds de son voisinage de telle sorte qu'ils couvrent l'intégralité de son 2-voisinage. Les informations sur la topologie du réseau sont émises uniquement par les MPR réduisant ainsi le trafic de contrôle sur le réseau. De plus, seuls les MPR sont autorisés à retransmettre les paquets de diffusion, limitant ainsi le nombre d'émetteurs.

Dans cet article, nous nous intéressons à l'algorithme de sélection de ces MPR (Section 2), en analysant le nombre moyen de MPR sélectionnés ainsi que leur distribution spatiale (Section 3). La majeure partie des travaux menés sur les performances d'OLSR s'attachent à l'efficacité du protocole de routage en lui-même ou des différentes techniques de diffusion utilisant les MPR ([2, 5, 6, 4]). Peu d 'articles ont étudié les performances de la sélection des MPR. Une analyse sur la ligne est menée en [8], une large borne supérieure du nombre de MPR sélectionnés dans un graphe unité est donnée dans [3]. Dans [7], les auteurs proposent et analysent plusieurs variantes de l'algorithme de sélection dans le but d'optimiser certaines quantités (bande passante, recouvrement, etc.). Comme nous le montrerons, la majeure partie des MPR est sélectionnée durant la première étape de l'algorithme. Or, cette étape ne peut être contournée puisqu'elle permet de couvrir les nœuds isolés et doit être exécutée en premier lieu afin de minimiser le nombre de MPR à sélectionner. De ce fait, les résultats de tous ces algorithmes s'en trouvent assez similaires, particulièrement en ce qui concerne le nombre de MPR par nœud.

[†] standards.ieee.org/getieee802/802.11.html
[‡] www.bluetooth.org/specifications.htm
[§] http ://www.ietf.org/html.charters/manet-charter.html

## 2 La sélection des MPR

Soit $N(u)$ le voisinage d'un nœud $u$, *i.e.* l'ensemble des nœuds à portée radio de $u$. On suppose les liens radio bidirectionnels : si $v \in N(u)$ alors $u \in N(v)$. On note $N_2(u)$ le 2-voisinage du nœud $u$, *i.e.* l'ensemble des nœuds voisins d'au moins un nœud de $N(u)$ mais n'appartenant pas à $N(u)$ : $N_2(u) = \{v \text{ t.q. } \exists w \in N(u) \,|\, v \in N(w) \backslash \{u\} \cup N(u)\}$. Pour $v \in N(u)$, soit $d_u^+(v)$ le nombre de nœuds de $N_2(u)$ appartenant à $N(v)$ : $d_u^+(v) = |N_2(u) \cap N(v)|$. Pour $v \in N_2(u)$, soit $d_u^-(v)$ le nombre de nœuds de $N(u)$ appartenant à $N(v)$ : $d_u^-(v) = |N(u) \cap N(v)|$. Chaque nœud $u$ choisit ses MPR parmi $N(u)$ de façon à couvrir tous les nœuds de $N_2(u)$. La sélection optimale des MPR est un problème NP-complet ([6]). Nous décrivons ici brièvement l'algorithme le plus fréquemment utilisé : le *Simple Greedy MPR Heuristic* (plus détaillé dans [9]). Il est composé de deux étapes. La première sélectionne les nœuds $v \in N(u)$ qui couvrent des "nœuds isolés" de $N_2(u)$. Un nœud $w \in N_2(u)$ est dit isolé si $d_u^-(w) = 1$ c'est à dire si il est couvert par un unique point $v$ de $N(u)$. La deuxième étape sélectionne parmi les nœuds restants de $N(u)$ ceux permettant de couvrir le plus grand nombre de nœuds de $N_2(u)$ non encore couverts. Ainsi, si $MPR(u)$ est l'ensemble des MPR sélectionnés par $u$, on a : $\bigcup_{v \in MPR(u)} N(v) = \{u\} \cup N(u) \cup N_2(u)$. On notera que la première étape de l'algorithme est obligatoire si l'on souhaite couvrir tout le 2-voisinage, donc seule la seconde étape peut être améliorée en vue d'une optimisation de la sélection.

## 3 Analyse

Nous nous intéressons aux propriétés des MPR d'un nœud particulier. Ce point sera le point situé à l'origine du plan. Soit $B(0,R)$ une boule de rayon $R \in \mathbb{R}^{*+}$ centrée en l'origine du plan. Nous distribuons des points dans cette boule suivant un processus de Poisson d'intensité $\lambda > 0$ et y ajoutons le point 0 à l'origine. Soit $d(u,v)$, la distance euclidienne entre $u$ et $v$. Il existe un lien bidirectionnel entre $u$ et $v$ si et seulement si $d(u,v) \leq R$. Nous utilisons les notations introduites dans la Section 2 pour définir le 1-voisinage (resp. le 2-voisinage) de 0 : $N$ (resp. $N_2$). $N$ est donc constitué des points du processus de Poisson situés dans $B(0,R)$. Nous notons $A(r)$ l'aire intersection de deux boules $B(x,R)$ et $B(y,R)$ telles que $d(x,y) = r$ et $A_1(u,R,r)$ l'aire couverte par la superposition de deux boules $B(x,R)$ et $B(y,u)$ telles que $d(x,y) = r$. Leurs formules mathématiques ainsi que l'ensemble des démonstrations des propositions qui suivent pourront être trouvées dans [9].

Dans la proposition suivante, nous considérons un point uniformément distribué dans $B(0,R)$ (resp. $B(0,2R) \backslash B(0,R)$) et donnons certaines quantités associées. Il correspondent à un point du processus choisi arbitrairement dans $B(0,R)$ (resp. $B(0,2R) \backslash B(0,R)$).

**Proposition 1.** *Soient u un point uniformément réparti dans $B(0,R)$ et v un point uniformément réparti dans $B(0,2R) \backslash B(0,R)$ :*

$$\mathbb{E}\left[d_0^+(u)\right] = \lambda R^2 \tfrac{3\sqrt{3}}{4} \qquad \mathbb{E}\left[d_0^-(v)\right] = \lambda R^2 \tfrac{\sqrt{3}}{4} \qquad \mathbb{E}\left[d_0^-(v) | v \in N_2\right] = \tfrac{\mathbb{E}\left[d_0^-(v)\right]}{\mathbb{P}(d_0^-(v)>0)}$$

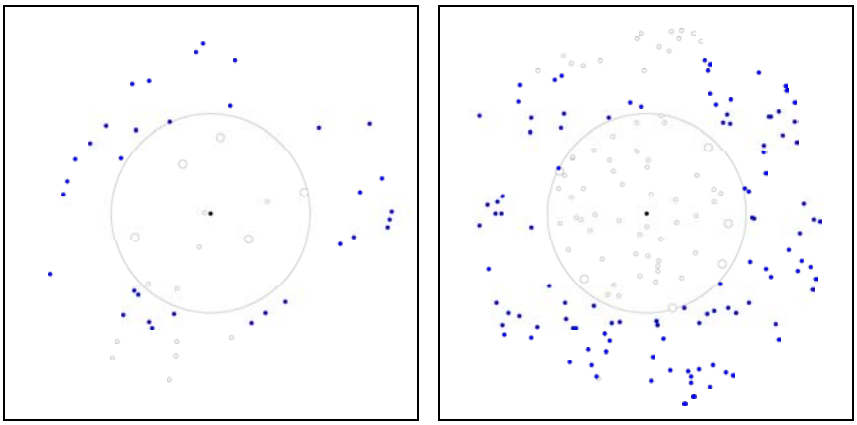
$$\mathbb{E}[|N|] = \lambda \pi R^2 \qquad \mathbb{E}[|N_2|] = 3\lambda\pi R^2 \left(1 - \tfrac{2}{3R^2} \int_R^{2R} exp\{-\lambda A(r)\} r dr\right)$$

Soit $MPR_1(u)$ l'ensemble des MPR de $u$ sélectionnés lors de la première étape de l'algorithme. Nous donnons ici le nombre moyen de points isolés $v$ : $\{v \in N_2 \mid d_0^-(v) = 1\}$ (Proposition 2). Mais puisque cette quantité ne nous donne qu'une borne supérieure sur le cardinal de $MPR_1$ nous donnons une borne inférieure dans la Proposition 3. Nous nous intéressons également à la distribution spatiale des points $MPR_1$. Pour un point $u \in N$ à distance donnée de l'origine, nous encadrons la probabilité que $u \in MPR_1$ (Proposition 4).

**Proposition 2.** *Soient v un point uniformément réparti dans $B(0,2R) \backslash B(0,R)$ et D l'ensemble des points v tels que $d_0^-(v) = 1$,*

$$\mathbb{P}\left(d_0^-(v) = 1\right) = \tfrac{2}{3R^2} \int_R^{2R} \lambda A(r) exp\{-\lambda A(r)\} r dr \qquad \mathbb{P}\left(d_0^-(v) = 1 | v \in N_2\right) = \tfrac{\mathbb{P}\left(d_0^-(v)=1\right)}{\mathbb{P}\left(d_0^-(v)>0\right)}$$

$$\text{et} \qquad \mathbb{E}[|D|] = 2\pi\lambda^2 \int_R^{2R} A(r) exp\{-\lambda A(r)\} r dr$$



(a) $\lambda\pi = 15$ (b) $\lambda\pi = 45$

**FIG. 1:** Sélection des MPR avec $\lambda\pi = 15$ et $\lambda\pi = 45$.

**Proposition 3.** *Soit u un point uniformément réparti dans* $B(0,R)$.

$$\mathbb{E}[|MPR_1|] \geq 2\lambda\pi\mathbb{P}\left(d_0^+(u) > 0\right)\int_0^R \int_R^{R+r} f(x,r,R) \times \exp\{-\lambda\left(2\pi R^2 - A_1(R,x,R)\right)\} r dx dr$$

$$\text{avec } f(x,r,R) = -\frac{\lambda}{1-\exp\{-\lambda\left(A_1(R,r,R)-\pi R^2\right)\}}\left[\frac{\partial}{\partial x}A_1(x,r,R) - 2\pi x\right]\exp\{-\lambda\left(A_1(x,r,R)-\pi x^2\right)\}$$

*De plus, puisqu'il existe au moins un point isolé par point de* $MPR_1$*, le nombre moyen de points isolés donne une borne supérieure :* $\mathbb{E}[|MPR_1|] \leq \mathbb{E}[|D|]$.

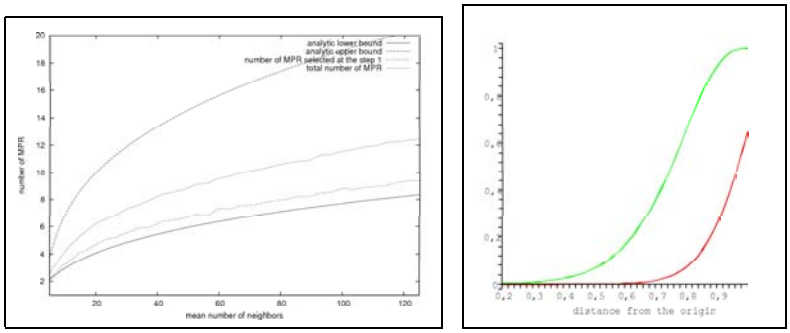
**Proposition 4.** *Soit u un point tel que* $d(0,u) = r, r \leq R$*. Nous distribuons en plus de ces deux points un processus de Poisson de paramètre* $\lambda$ *dans* $B(0,2R)$.

$$\mathbb{P}(u \in MPR_1) \geq \left(1 - \exp\{-\lambda(\pi R^2 - A(r))\}\right) \times \int_R^{R+r} f(v,r,R)\exp\{-\lambda(2\pi R^2 - A_1(R,v,R))\} dv$$

$$\mathbb{P}(u \in MPR_1) \leq 1 - \left(1 - \exp\{-\lambda\frac{A(R+r)}{2}\}\right)^2$$

# 4 Résultats numériques et simulations

Comme dans les sections précédentes, les nœuds du réseau sont distribués dans $B(0,2R)$ suivant un processus de Poisson d'intensité $\lambda > 0$. Nous fixons $R = 1$ et ajoutons un point en 0. Nous étudions alors pour ce point le nombre de MPR moyen sélectionnés à chaque étape. La Figure 1 représente des échantillons du modèle pour différentes valeurs de $\lambda\pi$, $\lambda\pi$ étant le degré moyen des nœuds du réseau. Sur ces figures, les points du cercle central représentent l'ensemble $N$, les points plus gros étant les $MPR_1$. Les points en dehors du cercle sont les points de $N_2$ : en noir ceux couverts par les $MPR_1$, en blanc ceux couverts lors de la deuxième étape. Nous remarquerons que dans tous les cas, quasiment l'intégralité de $N_2$ est couverte par les points de $MPR_1$. En effet, un nombre appréciable de points isolés donnent naissance à un certain nombre de points $MPR_1$. Ces derniers semblent être distribués régulièrement et près de la frontière de $B(0,R)$, ce qui confirme les résultats de la Proposition 4. C'est pourquoi les points de $MPR_1$ couvrent une large partie de $N_2$. La Figure 2(a) donne le nombre moyen de MPR et $MPR_1$ obtenus par simulation. Nous observons que près de 75% des MPR sont dans $MPR_1$, ce qui confirme que les $MPR_1$ couvrent la quasi totalité du 2-voisinage. Elle montre aussi que la borne inférieure est très près des résultats obtenus par simulation. L'encadrement de la probabilité pour un point $u$ d'appartenir à $MPR_1$ (Proposition 4), nous permet de




(a) Comparaison de la taille moyenne de $MPR_1$ et du nombre total de MPR obtenue par simulation et comparaison avec les bornes analytiques (le tout fonction de $\lambda\pi$).

(b) Bornes supérieure et inférieure de la probabilité que $u \in MPR_1$ en fonction de $d(0,u)$.

**FIG. 2:** Nombre moyen de MPR et leur distribution spatiale.

montrer que les points de $MPR_1$ se trouvent très près de la frontière de $B(0,R)$. Ce phénomène est illustré Figure 2(b) pour $d(0,u)$ variant de 0.2 à 0.999 et pour $\lambda = 15$. Cette tendance s'affirme lorsque l'intensité du processus croît. Lors d'une diffusion de messages à tout le réseau, une partie de la redondance reçue par les nœuds est liée à la surface des intersections des zones de portée des MPR. Puisque la distance entre un point et ses MPR est élevée, ces aires d'intersection sont minimales de même que la redondance induite.

# 5 Conclusion

Nous avons proposé une analyse de l'algorithme de sélection des MPR dans OLSR, montrant que près de 75% d'entre eux étaient sélectionnés lors de la première étape de l'algorithme. Cette étape étant indispensable et incompressible, toutes les variantes visant à améliorer l'algorithme mènent à des performances similaires. Mais cette propriété soulève un problème de robustesse : si on considère un nœud $u$ dont un MPR disparaît, il y a 75% de chances que ce MPR permettait d'atteindre un nœud isolé qui ne sera donc plus atteint par $u$. Nous avons également mis en évidence le fait que ces MPR étaient situés à la limite des portées radio, limitant ainsi le recouvrement des surfaces couvertes par chacun de ces nœuds. Ces résultats ont été présentés pour un modèle particulier, en utilisant un processus de Poisson. Nous espérons pouvoir analyser d'autres modèles et les comparer.

# Références

[1] Optimized Link State Routing Protocol (OLSR). T. Clausen and P. Jacquet, RFC 3626, October 2003.

[2] Simulation Results of the OLSR Routing Protocol for Wireless Network. A. Laouiti, P. Muhlethaler, A. Najid, E. Plakoo, Med-Hoc-Net, Sardegna, Italy 2002.

[3] Performances of multipoint relaying in ad hoc mobile routing protocols. P. Jacquet, A. Laouiti, P. Minet, L. Viennot, Networking 2002, Pise (Italy) 2002.

[4] L. Lovasz, On the ratio of optimal integral and fractional covers, Discrete Mathematics, 13, 1975, 383-390.

[5] The Optimized Link State Routing Protocol, Evaluation through Experiments and Simulation. T.H. Clausen, G. Hansen, L. Christensen and G. Behrmann, in WPCM'01.

[6] Performance Analysis of OLSR MPR Flooding in Two Ad Hoc Wireless Network Models. P.Jacquet, A. Laouiti, P. Minet and L. Viennot, RR-4260, INRIA, September 2001, RSRCP journal special issue on Mobility and Internet.

[7] Performance Evaluation of Approximation Algorithms for MPR Selection. B. Mans and N. Shrestha, Med-Hoc-Net'04, Bodrum, Turkey, June 27-30, 2004.

[8] Flooding techniques in mobile Ad-Hoc networks. E. Baccelli and P. Jacquet. RR-5002, INRIA, 2003.

[9] An analysis of the Multi-Point Relays selection in OLSR. A. Busson, N. Mitton, E. Fleury. RR-5468, INRIA, Janvier 2005.

# *Influence de l'auto-organisation sur la capacité des réseaux ad hoc*

Hervé Rivano, Fabrice Theoleyre, Fabrice Valois

*CNRS - Projet I3S/INRIA MASCOTTE, INRIA Sophia Antipolis, Email : Herve.Rivano@sophia.inria.fr*
*CITI - Projet INRIA ARES - INSA Lyon, Email : {fabrice.theoleyre,fabrice.valois}@insa-lyon.fr*

Les réseaux ad hoc tirent parti de la collaboration des nœuds pour acheminer des informations. Si de nombreuses approches ont vu le jour, la problématique du routage demeure un point crucial. Deux approches se détachent : une première résidant dans une vision *à plat* du réseau et une seconde, plus récente, où le routage repose sur une auto-organisation du réseau. Il s'agit de fournir une solution d'organisation afin de tirer parti des propriétés structurelles et d'améliorer des services tels que le routage. Les performances obtenues sont intéressantes bien que les auto-organisations réduisent le nombre de liens radio effectivement utilisés. Nous proposons donc ici de quantifier les changements, en terme de bande passante disponible, entre un réseau à plat et un réseau structuré.

**Keywords:** réseaux ad hoc, auto-organisation, routage, capacité, optimisation, simulation

## 1 Introduction et motivations

Les réseaux ad hoc sont des réseaux radio multi-sauts où chaque mobile collabore à la vie du réseau afin de permettre l'acheminement d'information [1]. Le problème clef est alors la conception d'un protocole de routage distribué. Classiquement, les protocoles de routage ont une vision à plat : tous les nœuds sont indifférenciés et participent activement à l'acheminement des paquets, sans tenir compte de la naturelle hétérogénéité d'un tel réseau. Dans OLSR [2], chaque nœud envoie des paquets de topologie dans tout le réseau afin de construire proactivement des routes optimales. De nombreux travaux se sont alors focalisés sur la notion d'auto-organisation des réseaux ad hoc [3, 4, 5]. Parmi les travaux sur l'auto-organisation, nous avons introduit dans [4] la notion de topologie virtuelle fondée sur la construction d'un ensemble dominant connexe modélisant une dorsale, à partir de laquelle nous proposons la création de zones de services (*cluster*). L'évaluation du comportement d'une telle topologie virtuelle montrant de bonnes propriétés de stabilité et de persistance dans le temps, nous avons proposé VSR, une solution de routage basée sur cette auto-organisation [5]. En dehors de la partie évaluation de performances (longueur moyenne des routes, taux de livraison de paquets, etc.), il est important d'étudier l'impact des solutions d'organisation sur le trafic que peut écouler le réseau ad hoc en terme de capacité. En effet, les solutions d'auto-organisation permettent de simplifier la topologie du réseau en réduisant le nombre de liens radio utilisés.

La prochaine section introduira quelques hypothèses de travail et les notations utilisées par la suite. La section 3 détaillera la modélisation de l'émission d'un paquet. L'évaluation de la capacité sera donnée dans la section 4. Nous conclurons alors ce travail en donnant quelques perspectives dans la section 5.

## 2 Hypothèses et modélisation du trafic

Pour la modélisation du partage de la ressource radio entre les nœuds, nous ferons les hypothèses suivantes sur le comportement du médium radio et le routage dans le réseau :

1. Couche MAC idéale et canal radio parfait, i.e. canal sans erreur.
2. Communication *unicast* bidirectionnelle : un flux du nœud $u$ au nœud $v$ génère un trafic d'acquittement identique à un facteur $\alpha$ près de $v$ à $u$, et empêche les voisins de $u$ et $v$ d'accéder au médium.
3. Une requête $(f_{u,v})$ sur $u \rightarrow v$ génère une réponse sur la route inverse $v \rightarrow u$, d'intensité $\beta f_{u,v}$.
4. Trafic de contrôle émis en *broadcast* empêchant tous les 2-voisins de la source d'émettre ou recevoir.

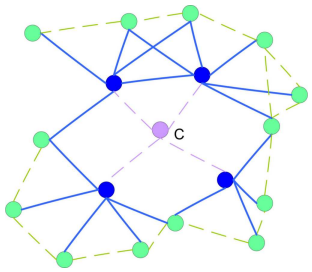



**FIG. 1:** 2-Voisinage centré sur le nœud C

**Notations** On note *BP* la bande passante radio disponible, i.e. le volume maximum de trafic pouvant être écoulé sur le canal. Pour chaque connexion $p$ dans le réseau, on note $f(p)$ son débit.

Soit $u$ un nœud du réseau. On note $T(u)$ le trafic total émis par $u$ : $T(u) = \sum_{v\in\Gamma(u)} T(u,v) + T_c(u)$ où :

- $\Gamma^k(u)$ est le k-voisinage de $u$, i.e. l'ensemble des nœuds à moins de $k$ sauts de $u$. On note $\Gamma^1(u) = \Gamma(u)$.
- $T(u,v)$ est le trafic *unicast* de $u$ vers $v$. Celui ci est composé des
  - paquets de données transmis le long des chemins passant par $(u,v)$, $\sum_{p\ni(u,v)} f(p)$,
  - paquets de réponse pour les données envoyées sur les chemins passant par $(v,u)$, $\sum_{p\ni(v,u)} \beta f(p)$,
  - acquittements des paquets envoyés sur l'arc $(v,u)$, $\alpha\left(\sum_{p\ni(v,u)} f(p) + \sum_{p\ni(u,v)} \beta f(p)\right)$
- $T_c(u)$ est le trafic de contrôle et maintenance de topologie envoyé par $u$ à ses voisins en mode *broadcast*
- $\Delta^k(u)$ est le nombre de k-voisins, i.e. $|\Gamma_k(u)|$. On note, $\Delta(u) = \Delta^1(u)$.

# 3 Modélisation de l'accès concurrent au médium radio

Nous souhaitons évaluer la capacité du réseau en terme de quantité de trafic généré. Pour cela, nous exprimons des ensembles de contraintes s'exereçant sur la capacité disponible aux communications passant par le voisinage d'un nœud $c$ quelconque (cf. figure 1). Le premier ensemble sous-estime cette capacité, tandis que le deuxième est un cas "idéal".

À partir de ces contraintes locales, on obtient un programme linéaire donnant, selon l'ensemble utilisé, une borne supérieure ou inférieure à la capacité globale du réseau.

**Borne inférieure** Le partage de la bande passante radio peut être sous-estimé de la façon suivante :

$$\forall c,\ \forall u \in \Gamma^2(c), \qquad T(u) \leq \frac{BP}{\Delta^2(c)} \tag{1}$$

$$\forall u,\ \forall v \in \Gamma(u)/\{c\}, \quad T(u,v) \leq \frac{\sum_{v\in\Gamma(u)} T(u,v)}{\Delta(u)} \quad \Rightarrow \frac{T_c(u)}{\Delta(u)-1} + T(u,v) \leq \frac{T(u)}{\Delta(u)-1} \tag{2}$$

Localement, la bande passante est équitablement répartie entre tous les nœuds en situation potentielle de brouillage : le centre et ses 2-voisins quand ils émettent du trafic. Toutes les communications sont supposées envoyées de façon séquentielle. La deuxième contrainte exprime tout d'abord qu'un nœud ne peut pas envoyer vers un voisin $v$ plus que le trafic moyen vers tous les 1-voisins, i.e. la bande passante est équitablement répartie. De plus, le trafic global qu'un nœud envoie à $v$ est composé du trafic de données vers $v$ et d'une partie du trafic de contrôle, équitablement réparti entre tous les 1-voisins. Un trafic sur une arête bloquant toutes les arêtes du 2-voisinage, le trafic de *broadcast* suit la même modélisation pour les interférences que le trafic *unicast*.

**Borne Supérieure** Nous considérons un modèle optimiste de répartition de la bande passante fondé sur l'hypothèse (1) de couche MAC idéale : il suppose, entre autre, que deux nœuds en contention ne tentent jamais d'accéder simultanément au canal, et que la bande passante est utilisée de façon optimale. Le canal radio se comporte comme une entité centralisée qui donnerait une portion équitable de la bande passante à chaque nœud. Il est important de noter que plusieurs nœuds peuvent communiquer simultanément, ce qui est lié à la notion de stable dans un graphe [6].

La sélection des communications ayant accès au médium se fait en les traitant dans un ordre aléatoire équiprobable. La communication (u,v) est sélectionnée si elle n'est pas bloquée par une communication précédemment sélectionnée, i.e. aucune autre communication dont une des extrémités est voisine de u ou v n'a été sélectionnée auparavant.

Cet algorithme génère des ensembles de communications, maximaux pour l'inclusion, pouvant s'établir sans brouillage réciproque. Si on note $G_c$ le graphe des communications dans la zone de contention d'un nœud $c$ ($\Gamma^2(c)$), ces ensembles sont donc des stables maximaux du graphe $L_{1,2}(\angle(G_c))$ † ‡. Enfin, si on note $T(I)$ la bande passante allouée au stable $I$, la bande passante disponible pour $c$ est :

$$T(c) \leq BP - \sum_I T(I) - \sum_{u \in \Gamma(c)} T_c(u) \tag{3}$$

Il n'est toutefois pas souhaitable d'avoir une modélisation qui utilise explicitement une variable par stable : il en existe un nombre exponentiel. Notons que $T(I)$ est proportionnel à la probabilité $P(I)$, probabilité que le stable $I$ soit sélectionné par le canal. P(I) n'étant calculable qu'à partir de la connaissance de tous les stables, le calcul de P(I) est alors exponentiel. Nous proposons de construire une estimation de P(I), notée $freq(I)$, telle que $freq(I)$ est obtenue statistiquement à partir d'un nombre significatif de tirages aléatoires d'ordre de communications. Ainsi : $T(I) = freq(I) \cdot \left(BP - T(c) - \sum_{u \in \Gamma(c)} T_c(u)\right)$.

La bande passante du lien $(u,v)$ est la somme des bandes passantes allouées aux stables contenant $(u,v)$ :

$$T(u,v) \leq \sum_{I \ni (u,v)} T(I) \leq \left(BP - T(c) - \sum_{x \in \Gamma(c)} T_c(x)\right) \cdot \sum_{I \ni (u,v)} freq(I) \tag{4}$$

$\sum_{I \ni (u,v)} freq(I)$ étant exactement la probabilité que la communication $(u,v)$ soit sélectionnée par le canal :

$$T(u,v) \quad \leq \quad \left(BP - T(c) - \sum_{x \in \Gamma(c)} T_c(x)\right) \cdot freq(u,v) \tag{5}$$

Où $freq(u,v)$ est l'estimation statistique de $P(u,v)$, probabilité que le lien $(u,v)$ soit sélectionné.

Il est nécessaire de prendre en compte, en sus, le trafic de contrôle nécessaire aux différents protocoles de routage. Lorsqu'un voisin de $c$ émet un paquet de contrôle, $c$ ne peut pas émettre simultanément (déjà traduit dans l'équation 5). Enfin, les 2-voisins de $c$, dans leur proportion de temps de parole, répartissent leur trafic de contrôle et de données. Ainsi nous avons la contrainte supplémentaire :

$$\forall w \in \Gamma^2(c),\ \left(\sum_{u \in \Gamma(c)} T(w,u)\right) + T_c(w) \leq \left(BP - T(c) - \sum_{x \in \Gamma(c)} T_c(x)\right) \cdot \sum_{v \in \Gamma(c)} freq(w,v) \tag{6}$$

# 4 Résultats

Des simulations sous OPNET des 2 protocoles de routage VSR et OLSR ont permis d'obtenir toutes les données d'entrées : topologie radio, surcoût protocolaire, routes construites. Nous considérons 2 types de réseau : un réseau ad hoc où un nœud peut communiquer potentiellement avec tous les autres, et un réseau hybride où un nœud ne communique que vers un nœud particulier, le point d'accès. Ces données d'entrée sont ensuite injectées dans Cplex [7] après avoir traduit les équations de contraintes précédentes.

Nous proposons les fonctions d'évaluation suivantes :

- Max-sum ($Max \sum_{(u,v)} T(u,v)$) cherchant à maximiser la quantité globale de flux généré. Ainsi, un flux n'étant comptabilisé que par la source, les flux de longueur unitaire seront privilégiés.
- Max-sum-dist ($Max \sum_{(u,v)} T(u,v).l(u,v)$) cherchant à maximiser la quantité globale de flux émis ou relayé. En tenant compte de la longueur des chemins ($l(u,v)$), nous traitons de façon équivalente les chemins courts et longs.

---

† $\angle$ le *linegraph* de G(V,E) est le graphe G'(V',E') tel que les sommets V' de G' sont les arêtes E, et il existe une arête entre 2 sommets de V' ssi les arêtes correspondantes possèdent un sommet commun dans G

‡ $L_{1,2}(X)$représente le graphe avec les sommets de X, tel qu'il existe une arête entre 2 sommets s'ils sont à distance 1 ou 2 dans X.

| | | Borne Inférieure | | Borne Supérieure | |
|---|---|---|---|---|---|
| | | à plat (OLSR) | organisé (VSR) | à plat (OLSR) | organisé (VSR) |
| Réseau ad hoc | Max-sum | 8706 | 8540 | 44284 | 38201 |
| | Max-sum-dist | 7458 | 7351 | 39445 | 34453 |
| Réseau hybride | Max-sum | 886 | 1087 | 5167 | 4987 |
| | Max-sum-dist | 518 | 784 | 3429 | 3524 |

**TAB. 1:** Flux maximum écoulés en kbps

Les bornes inférieures et supérieures, quoique différentes, sont toujours du même ordre de grandeur. Nous pouvons remarquer que la capacité d'un réseau ad hoc semble plus importante que celle d'un réseau hybride, les routes étant plus distribuées, et de longueur potentiellement faible. Dans un réseau ad hoc, une organisation à plat semble pouvoir écouler plus de trafic. Néanmoins, la capacité d'un réseau organisé, bien que supprimant certains liens radio, semble pouvoir offrir une capacité proche. Dans un réseau hybride, l'ordre s'inverse : une approche organisée permettrait d'acheminer plus de trafic en répartissant plus les routes. Une auto-organisation ne diminuerait donc pas de façon significative la capacité d'un réseau, tout en proposant, d'après [4], une vue plus simplifiée et plus stable de la topologie.

# 5 Conclusions et perspectives

Dans cette étude, nous nous intéressons à l'évaluation de la capacité d'un réseau ad hoc. Nous appelons capacité le flux maximum de données que peut écouler une topologie en tenant compte des contraintes radio, du routage et des paquets de contrôle. Deux types de réseaux ad hoc sont étudiés : un réseau *à plat* échangeant pro-activement des paquets de topologie dans tout le réseau (OLSR) puis un exemple de réseau auto-organisé construisant des routes mixant proactif et réactif (VSR). Nous avons proposé un modèle de partage du médium radio centré sur un nœud et son 2-voisinage. L'estimation est basée sur le calcul d'une borne inférieure et une borne supérieure. La borne inférieure traduit une contention où la bande passante est partagée équitablement et de façon exclusive entre tous les nœuds du 2-voisinage tandis que la borne supérieure tient compte du fait que plusieurs nœuds du 2-voisinage peuvent communiquer simultanément sans pour autant se brouiller mutuellement. Nous observons qu'une organisation n'est pas préjudiciable à la capacité du réseau. Néanmoins, ces résultats nécessitent de profondes investigations afin de comprendre le lien entre propriétés structurelles de la topologie (connexité, densité moyenne, variance de la densité, etc.) et la variation de la capacité du réseau. De plus, il apparaît nécessaire, à l'issue de cette première étude, d'affiner l'estimation des bornes.

# Références

[1] I. Chlamtac, M. Conti, J. J.-N. Liu, *Mobile ad hoc networking : imperatives and challenges* Ad hoc Networks Journal, 13-64, 2003.

[2] T. Clausen and P. Jacquet. Optimized link state routing protocol (olsr). RFC 3626, IETF, 2003.

[3] J. Wu and F. Dai. A distributed formation of a virtual backbone in manets using adjustable transmission ranges. In *International Conference on Distributed Computing Systems (ICDCS)*, Tokyo, Japan, March 2004. IEEE.

[4] F. Theoleyre, F. Valois, *A Virtual Structure for Mobility Management in Hybrid Networks* Wireless Communications and Networking Conference (WCNC), Atlanta, USA, March 2004. IEEE

[5] F. Theoleyre, F. Valois, *Virtual Structure Routing in Ad Hoc Networks* International Conference in Communications (ICC), Seoul, Korea, May 2005. IEEE

[6] C. Berge, *The Theory of Graphs,* Dover Publications Inc., 2001.

[7] ILOG CPLEX. http ://www.ilog.com/products/cplex/index.cfm. (v7.5).

# *Robust Routing in Changing Topologies*

Pascal Berthomé, Lynda Gastal and Abdel Lisser

*Université de Paris-Sud, Laboratoire de Recherche en Informatique, CNRS UMR 8623*
*Bâtiment 490,*
*F-91405 Orsay Cedex, France*

Nous proposons dans cet article une nouvelle version du problème de plus court chemin robuste. L'incertitude, qui ne concerne que les poids des arcs d'un graphe dans la version initiale, s'applique ici également à la structure même du graphe. Le problème consiste alors à trouver une séquence de chemins minimisant la somme des poids des chemins plus la somme des coût de transformation de deux chemins consécutifs. Après avoir formalisé le problème et établi sa complexité, nous donnons sa formulation mathématique ainsi qu'une relaxation. Nous proposons un algorithme polynômial pour les instances constituées de deux graphes et dans lesquelles le coût de désinstallation d'un arc est inférieur à son poids.

**Keywords:** robustness, robust shortest path, networks

## 1 Introduction

Recently the introduction of the uncertainty has appeared in numerous applications. This notion captures the natural evolution of general systems. The complexity lies in the number of parameters that are allowed to vary in order to cope with the characteristics of the problem studied.

In this context, the Absolute Robust Shortest Path problem consists in finding a path corresponding to the minimum maximum weight over a set of scenarii. Each scenario corresponds to a pre-determined set of edge weights. The motivation for studying this problem comes from telecommunications where a communication network is used to send packets from a given source to a given destination. The aim is to determine a shortest path between some given source and destination under some criteria (total delay, congestion, ...). The network manager has to choose a robust path where the total delay is acceptable regardless of the realized congestion [KY97].

The common approach of the robustness consists in finding a robust solution which minimizes the maximum regret, *i.e.*, the worst case scenario. Several approaches have been proposed. Averbakh [Ave04] focuses on minmax regret for optimization over uniform matroid, Ben-Tal, El Ghaoui and Nemirovski [BTN00, BT, BTNEGR00] work on ellipsoidal uncertainty, Bertismas and Sim [BS04] investigate robust optimization with control of the conservation of a solution, Kouvelis and Yu [KY97] focus on minmax regret robust optimization and Yaman and Karasan [PKY03] also work on absolute and relative robustness for spanning tree problem.

In Robust Shortest Path problem, the uncertainty is only related to link weights and does not take into account the possible evolutions of the network topology. Based on the market quick evolutions, telecommunication networks may increase and sometimes decrease when operators sell a part of their networks. Thus, an existing route between two protagonists may appear less profitable after adding new connections or nodes, and it may have to be reconsidered with the network evolution. Further cost may appear during this evolution, especially for resource desallocation and re-allocation. Considering this other definition of robustness, we introduce an alternative problem in which the uncertainty is also related to the structure of the network itself. Here, routing consists in finding an optimal sequence of paths taking into account the network topology variations. Moreover, changing existing routing may create local disturbance which can

be formulated in terms of cost. Thus, a sequence of paths will be optimal if it minimizes its total weight cost and the number of disturbances. We call this problem Robust Routing in Changing Topologies, *RRCT* for short.

We show in Section 2 that this problem is NP-complete even with simplified assumptions and we give its mathematical formulation and a combinatorial relaxation. In Section 3 we give a polynomial algorithm for an instance of a two-graph sequence, where the first graph is a simple path and disturbance cost is simplified. Finally, we draw a conclusion and address open questions in Section 4.

# 2 Definition

Throughout this paper, networks will be considered as weighted graphs.

## 2.1 RRCT problem

We consider a sequence of $n$ directed and weighted graphs $(G_i)_{i\in\{1,..,n\}}$. Each graph is defined by its set of edges $E_i$ and its set of vertices $V_i$. This sequence of graphs is such that $E_i \subset E_{i+1}$ and $V_i \subseteq V_{i+1}$ . We note $G_i \subset G_{i+1}$. Let $n_i$ be the number of nodes of graph $G_i$ and $m_i$ its number of edges. In addition, an edge has an installation cost and an uninstallation cost. We define $w : E \to \mathbb{N}$ as the weight function, $\iota : E \to \mathbb{N}$ as the installation cost function and $\delta : E \to \mathbb{N}$ as the uninstallation cost function. Let $s$ and $t$ be a source node and a destination node respectively. Note also that $s$ and $t$ belong to $G_i$ for any $i$.

The problem consists in finding a collection of paths $(X_i)_{i\in\{1,...n\}}$, one per graph, each linking source node $s$ to destination node $t$. In the rest of the paper, $S_n$ denotes the collection $(X_i)_{i\in\{1,...,n\}}$. The aim of *RRCT* problem is to minimize total weight plus transition cost between consecutive paths. The transition cost is composed of the uninstallation cost of the subset of edges of $X_i$ which are not in $X_{i+1}$ plus the installation cost of the subset of edges of $X_{i+1}$ that are not in $X_i$. Formally, we have to minimize the following function:

$$\varphi(S_n) = \sum_{i=1}^{n} \sum_{x\in X_i} w(x) + \sum_{x\in X_1} \iota(x) + \sum_{i=1}^{n-1} \Big( \sum_{x\in X_i, x\notin X_{i+1}} \delta(x) + \sum_{x\in X_{i+1}, x\notin X_i} \iota(x) \Big) \tag{1}$$

The Robust Routing in Changing Topologies problem can be formulated as a decision problem:
Robust Routing in Changing Topologies problem (*RRCT*)

**Instance:**

$n$ graphs $G_i = (V_i, E_i)$ such that $G_i \subset G_{i+1}\ \forall i \in \{1 \dots n-1\}$

$w : E \to \mathbb{N}$, $\iota : E \to \mathbb{N}$, $\delta : E \to \mathbb{N}$

two vertices $s$ and $t$; and an integer $b$

**Question:**

Does there exist a sequence $S_n$ of $n$ paths between $s$ and $t$ such that $\varphi(S_n) \leq b$?

**Theorem 1 ( [BGL05a])** *RRCT is NP-complete even for a two-graph sequence and when the first one is a simple path.*

The proof of this theorem is given in [BGL05a]. The reduction is based on Hamiltonian Circuit Problem [GJ79].

## 2.2 Mathematical formulation

We now formulate *RRCT* as a quadratic program with linear constraints. Let $x_{ij}^k$ be the binary variable equal to 1 if the edge between the nodes $i$ and $j$ of graph $G_k$ is part of path $X_k$, and 0 otherwise. The objective function can be formulated as:

$$\varphi(S_n) = \sum_{k=1}^{n} \left( \sum_{(i,j)\in E_k} w(ij) x_{ij}^k + \sum_{(i,j)\in E_k} \delta(ij) x_{ij}^k (1 - x_{ij}^{k+1}) + \sum_{(i,j)\in E_{k+1}} \iota(ij) x_{ij}^{k+1} (1 - x_{ij}^k) \right) \tag{2}$$

*RRCT* can be formulated as follows:

$$(RRCT)\begin{cases} \min \varphi_n \\ s.t. \\ \quad \sum_{j\in V} x_{ij}^k - \sum_{j\in V} x_{ji}^k = \begin{cases} 1 \; if \; i = u \\ -1 \; if \; i = v \\ 0 \; otherwise \end{cases} & \forall k \in \{1\dots n\} \quad \forall i \in V & (3) \\ \quad \sum_{(i,j)\in E(H)} x_{ij}^k \leq |H| - 1 & \forall H \subset G_k \quad \forall k \in \{1\dots n\} & (4) \\ \quad x_{ij}^k \in \{0,1\} & \forall i,j \in V_k \quad \forall k \in \{1\dots n\} \end{cases}$$

Constraints 3 represent the classical flow constraints and ensure that the solution contains a path from $s$ to $t$ within each graph $G_i$. Constraints 4 eliminate potential circuits. Indeed, some circuits can appear since the objective function contains negative terms. However, Constraints 4 introduce an exponential number of constraints.

In the following, we assume that Constraints 4 can be relaxed in order to solve *RRCT* by direct methods. It can however be handled within a decomposition scheme where cuts are added if a circuit is detected. The model then becomes:

$$(RRCTr)\begin{cases} \min \varphi_n \\ s.t. \\ \quad \sum_{j\in V} x_{ij}^k - \sum_{j\in V} x_{ji}^k = \begin{cases} 1 \text{ if } i = u \\ -1 \text{ if } i = v \\ 0 \text{ otherwise} \end{cases} & \forall k \in \{1\dots n\} \quad \forall i \in V & (5) \\ \quad x_{ij}^k \in \{0,1\} & \forall i,j \in V_k \quad \forall k \in \{1\dots n\} \end{cases}$$

*RRCTr* is a quadratic 0-1 program problem. It can be solved by several approaches. Among all, tight relaxation can be used to obtain good lower bounds. Recent studies, such as Goemans [Goe97] and Rendl [BTNEGR00], show the efficiency of semidefinite relaxation for solving quadratic 0-1 programming, namely, Quadratic Knapsack problem, Max Cut problem.... However, we show in [BGL05b] that semidefinite relaxation can only be used for small instances.

# 3 Polynomial cases

In this section, we identify two cases in which *RRCT* turns to be polynomially solvable. The first one is based on the basic formulation given in the previous section, and the second one presents a case in which a more efficient algorithm can be used.

## 3.1 Link between both formulations

**Theorem 2** *If the optimal solution of RRCTr problem is composed of n paths between s and t, then this solution is optimal for RRCT problem.*

**Proof:** Let $\tilde{S}$ be the optimal solution of *RRCTr* and $S$ the optimal solution of *RRCT*. It is easy to see that $\tilde{S}$ is an upper bound of *RRCT* since it is a feasible solution of *RRCT* problem. It is also a lower bound of *RRCT* since it is a feasible solution of *RRCTr*. □

This theorem provides a simple way to determine whether our instance belongs to the polynomially solvable cases. Since the constraints are simply flow constraints and disjointed linearization constraints, the linearization of the problem induces that the resulting linear program is unimodular, and consequently the binary solution can be obtained in polynomial time by solving the linear relaxation

We define the following algorithm:

1. Solve the relaxed program *RRCTr* (5). Let $P_i$ denote part of the solution which is a path between $s$ and $t$ in graph $G_i$ and $C_i$ denote the part of the solution which is an independent cycle in graph $G_i$.
2. Return $(P_i)_{i\in\{1,\dots,n\}}$

### 3.2 A path and a graph

We showed that *RRCT* problem is NP-complete in the general case. In this section, we consider the case in which $G_1$ is a simple path and $G_2$ is a general graph. Since this sub-problem is NP-complete, we consider furthermore that the uninstallation cost of any edge is lower than its weight.

Let us consider now the following algorithm. First, let us define a new function $w'$ on the edges of $G_2$ as:

- $w'(e) = w(e) + \iota(e)$ if $e \notin X_1$, i.e., $e \notin G_1$;
- $w'(e) = w(e) - \delta(e)$ otherwise.

At this step, let $X_2$ be the shortest path in $G_2$ using this new function. Since all the weights are positive, $X_2$ exists and can be found by the classical Dijkstra algorithm in $O(mn\log n)$ time. It is quite easy to see that solution $(X_1 = G_1, X_2)$ is optimal for the *RRCT* problem. Note that, if the weight function $w'$ is negative, the previous problem may be solved using the Bellman-Ford algorithm in $O(n^3)$ . However, this algorithm may not find a solution when there exists an absorbent circuit.

## 4 Conclusion

In this paper, we have shown that the Robust Routing in Changing Topologies problem is NP-complete and have provided a formulation based on 0-1 quadratic program as well as a combinatorial relaxation. However, some questions remain open. Indeed, the problem is polynomial when the transition cost are equals to zero or infinite. On the other hand, we have seen in Theorem 1 that the problem becomes difficult when the transition costs have a large amplitude. It would be of interest to determine the impact of the transition functions on the complexity of the problem, e.g., when these functions are constant.

## References

[Ave04] I. Averbakh. Minmax regret linear resource allocation problems. *Operation Research Letters*, 32:174–180, 2004.

[BGL05a] P. Berthomé, L. Gastal, and A. Lisser. Robust routing in changing topologies. In *INOC 2005*, pages B2–603–608, 2005.

[BGL05b] P. Berthomé, L. Gastal, and A. Lisser. Semidefinite relaxation for solving robust shortest path problem. Manuscript, 2005.

[BS04] D. Bertsimas and M. Sim. The price of robustness. *Operation Research*, 52(1):35–53, January-February 2004.

[BT] A. Ben-Tal. Conic and robust optimization. Lecture Notes.

[BTN00] A. Ben-Tal and A. Nemirovski. Lectures on modern convex optimization: Analysis, algorithms; engineering applications. SIAM-MPS Series in Optimization, 2000.

[BTNEGR00] A. Ben-Tal, A. Nemirovski, L. El Ghaoui, and F. Rendl. *Handbook on Semidefinite Programming*, pages 443–467. Kluwer, Boston, 2000.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and intractablility: A guide to the theory of NP-Completeness*. W.H. Freeman, 1979.

[Goe97] M.X. Goemans. Semidefinite programming in combinatorial optimization. *Mathematical Programming*, 79:143–161, 1997.

[KY97] P. Kouvelis and G. Yu. *Robust Discrete Optimisation and its Applications*. Kluwer Academic Publishers, 1997.

[PKY03] M. Pinar, O. Karasan, and H. Yaman. The robust shortest path problem with interval data, February 2003.

# *Grands graphes & ordonnancement*

Mercredi 12 mai 2005, 08:30 - 10:00

---

# *Could any graph be turned into a small-world?*

Philippe Duchon[1], Nicolas Hanusse[1], Emmanuelle Lebhar[2] and Nicolas Schabanel[2]

[1] *LABRI, Domaine universitaire 351, cours de la libération, 33405 Talence Cedex, France*
[2] *LIP - ENS Lyon, 46 allée d'Italie, 69007 Lyon, France*

In addition to statistical graph properties (diameter, degree, clustering, ...), Kleinberg [Kle00] showed that a small-world can also be seen as a graph in which the routing task can be efficiently and easily done in spite of a lack of global knowledge. More precisely, in a lattice network augmented by extra random edges (but not chosen uniformly), a short path of polylogarithmic expected length can be found using a greedy algorithm with a local knowledge of the nodes. We call such a graph a *navigable small-world* since short paths exist and can be followed with partial knowledge of the network. In this paper, we show that a wide class of graphs can be augmented with one extra edge per node into navigable small-worlds.

**Keywords:** small-world, random graph model, routing algorithm

## 1 Introduction

In the last decade, effective measurements of real interaction networks have revealed specific unexpected properties. Among these, most of these networks present a very small diameter and a high clustering. Furthermore, very short paths can be efficiently found between any pair of nodes without global knowledge of the network, which is known as the small-world phenomenon [Mil67]. Several models have been proposed to explain this phenomenon ( [WS98], [NW99]). However, Kleinberg showed in 2000 [Kle00] that these models lack the essential *navigability* property : in spite of a polylogarithmic diameter, none of the short paths can be computed efficiently without global knowledge of the network ; i.e., routing requires the visit of a polynomial number of nodes (in the size of the network). He introduced an augmented graph model consisting of a grid where each node is given a constant number of random additional directed long range links distributed according to the harmonic distribution, i.e., the probability that a node $\mathbf{v}$ is the $i$-th long range contact of a node $\mathbf{u}$ is proportional to $1/|\mathbf{u}-\mathbf{v}|^s$, where $|\mathbf{u}-\mathbf{v}|$ denotes their distance in the grid and $s>0$ is a parameter of the model. In this model, the local knowledge at each node is the underlying metric of the grid and the positions on the grid of its long range neighbors. Kleinberg considered the greedy routing (decentralized) where each node forwards the message to its neighbor that is the closest to the destination. He proved that this algorithm computes between any pair of nodes a path of polylogarithmic length in the size of the network after visiting a polylogarithmic number of nodes, if and only if the exponent $s$ is equal to the dimension of the grid. Later on, Barrière *et al.* [BFKK01] generalized this result to a regular $n\times\cdots\times n$ torus. Moreover, they showed that the expected number of steps of the greedy algorithm is $\Theta(\log^2 n)$, and that, noticeably, the number of steps is independent of the dimension. This reveals a strong correlation between the underlying grid metric and the additional long range links distribution that turns the grid into a small-world. This statement raises an essential question to capture the small-world phenomenon : are there only specific graph metrics that can be turned into small-worlds by the addition of shortcuts ?

This question on the metric structure can be reinforced by the fact that whenever the exponent $s$ is different from the dimension of the grid, the greedy algorithm follows a path of polynomial length even when the diameter is polylogarithmic (which is the case for $d<s<2d$ [MN04, NM05]). The reader might believe that the navigability property is very specific to the grid topology, but we will show that a wide family of graphs can be turned into navigable small-worlds. In [Kle02], Kleinberg generalized his lattice-based model and showed how to turn into smallworlds tree-based or group-based structures by adding

a polylogarithmic number of long range links per node. [NM05, Fra05] are other recent articles which tackle these questions. In this paper, we attempt to find a class of graph metrics as wide as possible for which the addition of a constant number of random long range links per node gives rise to the small-world phenomenon. Roughly speaking, as soon as the original graph $H$ is homogeneous in terms of ball expansion and as soon as balls centered on each node grow up to slightly more than polynomialy with their radius, $H$ can be augmented to become a navigable small-world. It follows that a wide class of graphs can be turned into navigable small-worlds, including in particular any Cayley graphs known up to now. In a second step, we try to catch the dimensional phenomenon by studying cartesian products of our graphs. We show that if two independent graphs can be augmented into two navigable small-worlds then their cartesian product can also be augmented into a navigable small-world. For instance, as a consequence, any unbalanced torus $C_{n_1} \times C_{n_2} \times \ldots \times C_{n_l}$ can be turned into a small-world in which the greedy algorithm computes paths of length $O(\log^{2+\varepsilon}(\max_i n_i))$, for any $\varepsilon > 0$.
In all the following, we will consider infinite graphs, but our definitions and results apply as well to infinite families of finite graphs (see [DHLS05]).

## 2 Small-worlds and graph metrics

For a given graph $G = (V,E)$, we write $\mathcal{B}_{G,\mathbf{u}}(r)$ for the ball centered on a node $\mathbf{u}$ with radius $r$, and $b_{G,\mathbf{u}}(r)$ for its cardinality. Let $b_G(r) = \max_{\mathbf{u}\in V} b_{G,\mathbf{u}}(r)$. The $G$ subscript will be omitted in case the concerned graph is obvious. We only consider graphs with maximum degree $\Delta$, a fixed constant.

In the following, an *underlying metric* $\delta_H$ of a graph $G$ is the metric given by a spanning connected subgraph $H$ (i.e., $\delta_H(\mathbf{u},\mathbf{v})$ is the distance between $\mathbf{u}$ and $\mathbf{v}$ in $H$). Definitions 1 and 2 are inspired from the work of Kleinberg [Kle00].

**Definition 1** *A* decentralized algorithm using an underlying metric $\delta_H$ *in a graph $G$ is an algorithm that computes a path between any pair of nodes by navigating through the network from the source to the target, using only the knowledge 1) of $\delta_H$ 2) of the nodes it has previously visited as well as their neighbors. But, crucially, 3) it can only visit nodes that are neighbors of previously visited nodes.*

Our definition of a navigable small-world is essentially probabilistic. We consider random graph models in which a fixed "base" graph $H$ is *randomly augmented* by adding random links (called *long range links* below), according to some probability distribution. Routing will then be performed by a decentralized algorithm, using the base metric $\delta_H$ ; our goal is to identify such augmented graph models for which this procedure results in uniformly "fast" routing. Since the augmented graph will have a finite degree, at least some of the $b_{H,\mathbf{u}}(r)$ nodes at distance at most $r$ of $\mathbf{u}$ will remain at distance $\Omega(\log b_{H,\mathbf{u}}(r))$ in the augmented graph This motivates the following definition.

**Definition 2** *An infinite randomly augmented graph $G$, on a (infinite) base graph $H$, is a* navigable small-world *if there exists a decentralized algorithm using the underlying base metric $\delta_H$ that, for any two nodes $\mathbf{u}$ and $\mathbf{v}$, computes a path from $\mathbf{u}$ to $\mathbf{v}$ in $G$ by visiting an expected number of nodes that is polylogarithmic in $b_{H,\mathbf{v}}(\delta_H(\mathbf{u},\mathbf{v}))$.*

## 3 Turning graphs into small-worlds

In this section, we describe a wide class of infinite graphs, or of infinite families of finite graphs, for which we are able to define random augmentation models that result in navigable small-worlds. In all cases, our routing algorithm will be the greedy algorithm, thus the set of visited nodes will coincide with the path computed. Furthermore, even if some algorithms can compute significantly shorter paths [FGP04, MN04, LS04], it has been shown in [MNW04] that no decentralized algorithm can compute a polylogarithmic path between two nodes while visiting asymptotically fewer nodes than the greedy algorithm. All models we will consider add exactly one directed edge[†] leaving each node $\mathbf{u}$. For each node $\mathbf{u}$, there is a function $f_{\mathbf{u}}$ such that each other node $\mathbf{v}$ has probability proportional to $f_{\mathbf{u}}(\delta(\mathbf{u},\mathbf{v}))$ of being the destination $L_{\mathbf{u}}$ ; the normalizing factor is $Z_{\mathbf{u}} = \sum_{\mathbf{v}\in V} f_{\mathbf{u}}(\delta(\mathbf{u},\mathbf{v}))$.

[†] Adding a constant number $k$ of edges instead of one would not significantly alter the results, as will be made clear by the proofs.

**Definition 3** *We say that an infinite graph is* smallworldizable *if there exists, for each* $\mathbf{u}$*, a distribution* $f_{\mathbf{u}}(r)$ *such that we obtain a navigable small-world randomly augmented graph by adding one random long range link to each node* $\mathbf{u}$ *according to* $f_{\mathbf{u}}(r)$ *(each node* $\mathbf{u}$ *is the origin of one long range link whose destination is* $\mathbf{v}$ *with probability proportional to* $f_{\mathbf{u}}(\delta(\mathbf{u},\mathbf{v}))$*.*

The following class of graph is defined for the sake of readability. As shown below, it characterizes a class of smallworldizable graphs.

**Definition 4** *A bounded degree infinite graph* $H$ *is an* $\alpha$-moderate growth graph *if there exists a constant* $\alpha > 0$*, such that the ball size of each node* $\mathbf{u}$ *of* $H$ *can be written as* $b_{\mathbf{u}}(r) = r^{d_{\mathbf{u}}(r)}$*, where* $d_{\mathbf{u}}(r) : [2,\infty) \to \mathbb{R}$ *is* $\mathcal{C}^1$ *and satisfies* $\forall r \geq 2, d'_{\mathbf{u}}(r) \leq \alpha/(r \ln r)$*.*

**Lemma 1** *Let* $G$ *an* $\alpha$*-moderate growth infinite graph. There exists a constant* $C > 0$*, s.t. for all* $\mathbf{u} \in G$*,* $r > 1$*,* $b_{\mathbf{u}}(r) - b_{\mathbf{u}}(r-1) \leq \binom{C \ln\ln(r)}{r} b_{\mathbf{u}}(r)$*.*

**Lemma 2** *Let* $G$ *a moderate growth infinite graph. Then, there exists a constant* $C'$ *such that* $b_{\mathbf{u}}(r) \leq (\ln r)^{-\alpha \ln \beta}/(\beta^{C+C'\alpha})\, b_{\mathbf{u}}(\beta r)$*, for any* $\mathbf{u} \in G$*,* $0 < \beta < 1$ *and* $r \geq 2$*.*

**Theorem 3** *Any moderate growth infinite graph is smallworldizable by the addition of one long range link per node, distributed according to* $f_{\mathbf{u}}(r) = \frac{1}{b_{\mathbf{u}}(r)\log^q r}$*, for any* $q > 1$*. Furthermore, the expected greedy path length between any pair of nodes at distance* $\ell$ *from each other is* $O(\ln^{1+q+\alpha \ln 5} \ell)$*.*

**Proof sketch.** The normalization constants $Z_{\mathbf{u}} = \sum_{\mathbf{v} \in V} f_{\mathbf{u}}(\delta_H(\mathbf{u},\mathbf{v})) = \sum_{r \geq 1} (b_{\mathbf{u}}(r) - b_{\mathbf{u}}(r-1))\, f_{\mathbf{u}}(r)$ are uniformly bounded by Lemma 1. The distribution defining the randomly augmented graph is thus properly defined. Assume that $\mathbf{s}$ and $\mathbf{t}$ are respectively the source and target, we analyze the expected path length computed by the greedy algorithm. Consider some integer $r \geq 2$ and a node $\mathbf{u}$ such that $r/2 < \delta_H(\mathbf{u},\mathbf{t}) \leq r$, and denote by $L_{\mathbf{u}}$ the long range contact of $\mathbf{u}$. We give a lower bound on $\Pr[\delta_H(L_{\mathbf{u}},\mathbf{t}) \leq r/2]$, the probability that the destination node $L_{\mathbf{u}}$ belongs to $\mathcal{B}_{\mathbf{t}}(r/2)$. Since $f_{\mathbf{u}}$ is a decreasing function and $\mathcal{B}_{\mathbf{t}}(r/2) \subseteq \mathcal{B}_{\mathbf{u}}(3r/2)$, each node of $\mathcal{B}_{\mathbf{t}}(r/2)$ has probability at least $f_{\mathbf{u}}(3r/2)/Z$ of being $L_u$. Since, in turn, $\mathcal{B}_{\mathbf{u}}(3r/2) \subseteq \mathcal{B}_{\mathbf{t}}(5r/2)$, we can give a lower bound on $f_{\mathbf{u}}(3r/2)$ in terms of $b_{\mathbf{t}}$ : $f_{\mathbf{u}}(3r/2) \geq \frac{1}{b_{\mathbf{t}}(5r/2)\ln^q(3r/2)}$. Thus, we get a lower bound, depending only on $\mathbf{t}$ and $r$, on the wanted probability :

$$\begin{aligned}\Pr[\delta_H(L_{\mathbf{u}},\mathbf{t}) \leq r/2] &\geq \frac{1}{Z\ln^q(3r/2)}\,\frac{b_{\mathbf{t}}(r/2)}{b_{\mathbf{t}}(5r/2)} \\ &\geq \left(Z 5^{C+C'\alpha} \ln^q(3r/2) \ln^{\alpha\ln 5}(5r/2)\right)^{-1} \geq \left(Z 2^{q+\alpha\ln 5} 5^{C+C'\alpha} \ln^{q+\alpha\ln 5}(r)\right)^{-1}\end{aligned}$$

We partition the whole graph into concentric shells centered on $\mathbf{t}$, where the $k$-th shell consists of all nodes whose $\delta_H$ distance to $\mathbf{t}$ is between $2^{k-1}$ and $2^k$. The previous discussion proves that each node in the $k$-th shell has probability $\Omega(k^{-\gamma})$ of having its long range contact in some $i$-th shell with $i < k$, where $\gamma = q + \alpha \ln 5$. As a result, the expected length of the greedy path from $\mathbf{s}$ to $\mathbf{t}$ is $O(\ln^{1+q+\alpha\ln 5} \ell)$, which is polylogarithmic in $\ell = \delta_H(\mathbf{s},\mathbf{t})$, and *a fortiori* in $b_{\mathbf{t}}(\delta_H(\mathbf{s},\mathbf{t}))$. The argument used here is similar to that of Kleinberg's analysis of its original model ; our changes allow the upper bound to be expressed only in terms of the original metric (and not the total size of the graph). □

The theorem above covers graphs with ball sizes $b(r)$ growing like $r^{\alpha \log\log r}$, $\alpha > 0$, or slower. Note that we get a similar upper bound $O(\ln^{2+\varepsilon} r)$, for any $\varepsilon > 0$, on the expected length of the greedy path between any pair of nodes at distance $r$ from each other. In particular, all known Cayley graphs[‡] are smallwordizable since groups of intermediate ball size, between polynomial and exponential, are still unknown, and it is an open question whether there exists a group with ball size $b(r)$ superpolynomial but less than $e^{\sqrt{r}}$, see for instance [Bar02] for a state of the art.

[‡] A Cayley graph is a graph defined by a group $G$ generated by $g_1,\ldots,g_k$, whose vertices are the elements of $G$ and such that there is an edge between $x$ and $y$ iff there is a generator $g_i \in G$ such that $x = g_i y$.

# 4 Products of small-worldizable graphs

A remarkable fact on the small-world property is its relative independence of the metric dimension in Kleinberg's model. This motivates the study of product of smallworldizable graphs.

**Definition 5** *The* cartesian product $H = F \times G$ *of two undirected graphs* $F$ *and* $G$ *is the graph* $(V_H, E_H)$ *where* $V_H = V_F \times V_G$ *and* $E_H = \{((f,g),(f,g')) : gg' \in E_G, f \in V_F\} \cup \{((f,g),(f',g)) : g \in V_G, ff' \in E_F\}$.

**Theorem 4** *Let* $F$ *and* $G$ *be* $\alpha_1-$ *and* $\alpha_2$*-moderate growth infinite graphs respectively. The cartesian product* $H = F \times G$ *is smallworldizable by the addition of one long range link per node* $\mathbf{u}$ *according to the distribution* $h_{\mathbf{u}}(r) = 1/(b_{H,\mathbf{u}}(r)\ln^{q'} r)$*, for all* $q' > q_0$*, for some constant* $q_0 > 0$*. Furthermore, the expected greedy path length between any pair of nodes at distance* $\ell$ *from each other is* $O(\ln^{1+q'+(\alpha_1+\alpha_2)\ln 10} \ell)$.

Note that this theorem yields another simple method to obtain a generalization of Kleinberg's graph to tori of dimension $d \geq 1$ with arbitrary side sizes, seen as cartesian products of one dimensional Kleinberg graphs of various sizes. It also gives an expected path length $O(\ln^{2+\varepsilon} \ell)$ for cartesian product of uniform growth graphs (i.e. $\alpha = 0$), close to Kleinberg model's path length.

# Références


[Bar02] L. Bartholdi. Groups of intermediate growth, preprint oai :arxiv.org :math/0201293. preprint oai :arXiv.org :math/0201293 (20040622), 2002.

[BFKK01] L. Barrière, P. Fraigniaud, E. Kranakis, and D. Krizanc. Efficient Routing in Networks with Long Range Contacts. In *Proceedings of the 15th International Conference on Distributed Computing (DISC)*, pages 270–284, 2001.

[DHLS05] P. Duchon, N. Hanusse, E. Lebhar, and N. Schabanel. Could any graph be turned into a small world ? *To appear in Theoretical Computer Science special issue on Complex Networks*, 2005. Also available as Research Report LIP-RR2004-62.

[FGP04] P. Fraigniaud, C. Gavoille, and C. Paul. Eclecticism shrinks even small worlds. In *Proceedings of the 23rd ACM Symp. on Principles of Distributed Computing (PODC)*, pages 169–178, 2004.

[Fra05] P. Fraigniaud. Greedy routing in tree-decomposed graphs : a new perspective on the small-world phenomenon. preprint, 2005.

[Kle00] J. Kleinberg. The Small-World Phenomenon : An Algorithmic Perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC)*, pages 163–170, 2000.

[Kle02] J. Kleinberg. Small-world phenomena and the dynamics of information. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems (NIPS) 14*, Cambridge, MA, 2002. MIT Press.

[LS04] E. Lebhar and N. Schabanel. Almost optimal decentralized routing in long-range contact networks. In *LNCS proceedings of 31st International Colloquium on Automata, Languages and Programming (ICALP)*, pages 894–905, 2004.

[Mil67] S. Milgram. The small world problem. *Psychology Today*, 61(1), 1967.

[MN04] C. Martel and V. Nguyen. Analyzing kleinberg's (and other) small-world models. In *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing*, pages 179–188, 2004.

[MNW04] G. S. Manku, M. Naor, and U. Wieder. Know thy neighbor's neighbor : the power of lookahead in randomized p2p networks. In *Proceedings of the 36th ACM Symposium on Theory of Computing (STOC)*, 2004.

[NM05] V. Nguyen and C. Martel. Analyzing and characterizing small-world graphs. To appear in the 2005 ACM-SIAM symposium on Discrete Algorithms (SODA), 2005.

[NW99] M. E. J. Newman and D. J. Watts. Scaling and percolation in the small-world network model. *Phys. Rev.*, 60 :7332–7342, 1999.

[WS98] D. Watts and S. Strogatz. Collective dynamics of small-world networks. *Nature*, 393(440–442), 1998.

# *A basic toolbox for the analysis of dynamics of growing networks*

Clémence Magnien[†], Mahendra Mariadassou[‡] and Camille Roth[§]

*CREA - CNRS - Ecole Polytechnique, 1 rue Descartes, 75005 Paris*

Most complex networks evolve through time, yet up till now their analysis has mainly focused on static snapshots of these networks. Our goal is to provide a set of basic tools for the analysis of the dynamics of networks. These tools should be at the same time very general, in order be applicable to all networks, whatever field they are originated from, and yet be relevent, in order to be able to capture meaningful behaviours. We restricted ourselves to the case of growing networks (*i.e.* networks in which nodes and links appear but are never removed), but our tools can easily be adapted to general dynamic networks. Finally, we have applied our tools to two cases of growing networks, to illustrate their relevence.

**Keywords:** Complex Networks, Dynamic Networks, Network Analysis

## 1 Introduction

The analysis of complex networks has lead since a few years to the introduction of a set of statistical parameters providing information on networks, See for instance [BS03, Bar02]. Some of these parameters played a central role because they are very general: most real-world complex networks display the same behaviour concerning them, which is not captured by classical models (like regular graphs or random graphs).

In many cases, the networks under concern evolve during time: new nodes or links arrive, and some leave. This has been pointed as a central characteristics of real-world complex networks, but until now very little is known on these dynamics. Indeed, dynamic data are difficult to collect and represent, and even if one has such data there are very few, if any, statistical tools for their analysis. The small number of papers addressing this subject, are mostly concerned with answering precise questions, with no goal of generalisation [PHL04, OdC03, Hol03, Sni01, KMCB95, EL03, New01].

Our aim in this paper is to make a first step in this direction: we want to propose a set of simple statistical tools which may be used as a basic toolbox when one deals with a dynamic complex network. We want our tools to be: as simple and general as possible, in order to make them easily usable in various contexts; relevant, in the sense that they indeed capture some non-trivial behaviours of the dynamics; efficiently computable, or at least tractable in large practical cases (real-world networks can be very large, up to several dozens of millions of nodes); and finally they should span well the variety of phenomena one may want to capture, at a basic and general level.

We actually restricted ourselves to a special case: we only considered *growing* networks, *i.e.* networks in which nodes and links arrive during time but are never removed afterwards. Many real-world cases actually fit in this special case. Moreover, most tools we will propose may easily be extended to the general case. New tools should however be introduced to deal with the general case, but this is out of the scope of this paper.

We will now present the formal framework we have considered as well as the two growing networks we have chosen to illustrate our approach, and then we will present the tools we have introduced for the analysis of the dynamics.

[†] magnien@shs.polytechnique.fr
[‡] mahendra.mariadassou@ens.fr
[§] roth@shs.polytechnique.fr

| network | nodes | links | time-grain | density | av. distance | clus. |
|---|---|---|---|---|---|---|
| actor | 869,986 | 51,520,496 | 1 year | $1.361 \times 10^{-4}$ | 3.77 | 0.795 |
| IP exch. | 281,760 | 508,667 | 10 sec. | $1.281 \times 10^{-5}$ | 5.29 | $5.914 \times 10^{-5}$ |

**Table 1:** Number of nodes and links, time grain, density, average distance and clustering coefficient of the growing networks under consideration.

## 2 Formal framework and methodology

There are several ways to describe a growing network. We will use the formalism of *coarse-grained* growing networks, which can be described as follows: a *coarse-grained* growing network is described as a series of sets $(\delta V_i, \delta E_i)$, $i = 1, 2, \ldots$ and a time grain denoted by $\delta t$. The network at time $t = \delta t \cdot i$ (*i.e.* the network at step $i$) is nothing but $G_i = (V_i, E_i)$ with $V_i = \cup_{j \leq i} \delta V_j$ and $E_i = \cup_{j \leq i} \delta E_j$. Notice that, even if a node or a link may appear several times (*i.e.* belong to several $\delta V_i$'s or $\delta E_i$'s), this information is not encoded in the graph at a given time. The coarse-grained formalism naturally induces several notions:

- $\delta V_i^{\top}$ is the set of nodes in $\delta V_i$ which appear at step $i$, *i.e.* $\{v \in \delta V_i$ such that $\nexists j < i,\ v \in \delta V_j\}$. We will call these nodes the *external nodes of step i*.
- $\delta V_i^{\perp}$ is the set of nodes in $\delta V_i$ already present before step $i$, *i.e.* $\{v \in \delta V_i$ such that $\exists j < i,\ v \in \delta V_j\}$. We will call these nodes the *internal nodes of step i*.
- $\delta E_i^{\top}$ is the set of links in $\delta E_i$ between nodes which appear at step $i$, *i.e.* $\delta E_i^{\top} = \delta E_i \cap \delta V_i^{\top} \times \delta V_i^{\top}$. We will call these links the *fully external links of step i*.
- $\delta E_i^{\perp}$ is the set of links in $\delta E_i$ between nodes which were already present before step $i$, *i.e.* $\delta E_i^{\perp} = \delta E_i \cap \delta V_i^{\perp} \times \delta V_i^{\perp}$. We will call these links the *internal links of step i*.
- $\delta E_i^{\overset{\top}{\perp}}$ is the set of links in $\delta E_i$ between a node which appears at step $i$ and a node which was already present, *i.e.* $\delta E_i^{\overset{\top}{\perp}} = \delta E_i \cap \delta V_i^{\perp} \times \delta V_i^{\top}$. We will call these links the *external links of step i*.

Our aim is to introduce a general formalism to describe real-world growth of complex networks. It is therefore essential to use real-world cases to illustrate and evaluate the relevance of our approach and of the parameters we will introduce. We will therefore use two large growing complex networks all along the paper: the *actor* networks, where the nodes are film actors and where a link exists between two actors if they acted together in a film, [Dat], and the *IP exchange* networks, in which the nodes are computers on the Internet and a link exists between two computers if the they echange a packet. For more details, see [Arc].

Table 1 gives the number of nodes and links of these networks, their time grain, as well as the basic statistics for these networks: density, average distance, clustering coefficient. For a definition of these statistics see for instance [GL05]. We have not represented here the degree distributions of these networks. They are, not surprisingly, well approximated by power-laws [GL05].

## 3 Analysis of the dynamics

The first quantity to consider when analysing growing networks is the evolution of the number of nodes and links of each type, presented in Figure 1. Notice that this already allows us to spot different behaviours in the networks we consider.

The statistics we will now present are based on the degrees of the nodes, and their evolution. Since the degree distributions of most real-world networks are heterogeneous, the mean degree is not representative of the network: most nodes' degree is smaller than the mean degree, and some node have a much larger degree.

We have therefore considered separately these two types of nodes. There is a large number of small degree nodes, and they form the bulk of the network. Therefore it makes sense to consider them as a whole, a natural quantity to study being then the overall fraction of small-degree nodes in the network. Figure 2 presents the time-evolution of the fraction of nodes of degree $\leq k$, for different values of $k$. We observe a

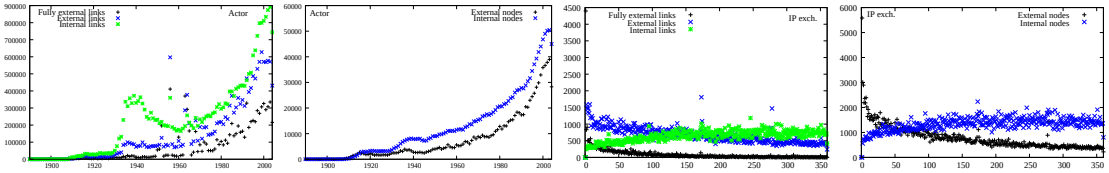



**Figure 1:** Time evolution of the number of fully-external, external and internal links, and external and internal nodes for *actor* and *IP exch.* networks

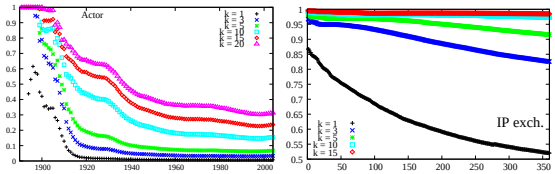



**Figure 2:** Time evolution of the number of small degree nodes for *actor* and *IP exch.* networks. The quantity plotted is the fraction of nodes with degree $\leq k$, for different values of $k$.

difference between our two networks: for the *actor* network, these values for the smallest values of $k$, seem to converge to a constant value. This is not the case for the *IP exch.* network.

High-degree nodes are the exceptional elements of a network, and therefore cannot be considered to form a homogeneous group. We have therefore considered separately the ten nodes that have the highest degree at the end of the network evolution. Once these nodes are identified, we can study the time evolution of their degree throughout the network's evolution. Figure 3 plots the time-evolution of a representative subset of these nodes, compared to the time-evolution of the maximal degree of the network (*i.e.* the degree of the highest degree node of the network at each time step).

The first conclusion is that, for each of the networks under study, the highest-degree nodes have a very *similar behaviour*: the degrees of the *actor* network's nodes all grow with an S-shape, while the growth is linear for the *IP exch.* network. This seems to be a general characteristic of growing networks (confirmed by other experiments not reproduced here). From the observation of the highest-degree nodes, we can also gather specific information on some networks: in the *actor* network, not only do the shape of the plots of the nodes' degree growth have a similar shape, as expected, but, moreover, the time at which most of these nodes' degree starts to grow is almost the same, somewhere close to 1930. This is a much stronger property than the fact that the plots have a similar shape: they could have similar shapes, whith their growth starting at different times. This therefore indicates a peculiar property of the *actor* network.

Second, we observe two different regimes when we compare the behaviour of the highest-degree nodes to the time evolution of the maximal degree of the network. For the *actor* network, the maximal degree is mainly carried by three different nodes: the first one apprears around 1910, and is not among the ten highest degree nodes. The maximal degree is then carried, beween approximatively 1935 and 1950, by the second-highest degree node, and by the highest degree node afterwards. This is very different from what happens for the *IP exch.* network, where the maximal degree is carried by the same node throughout the

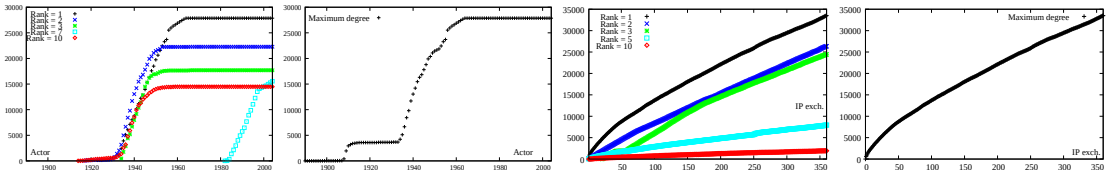



**Figure 3:** Time behaviour of a few high degree nodes for *actor* and *IP exch.* networks, as well as the evolution of the maximum degree for these networks. Rank is the rank of the node, based on his degree, in the final network.

evolution of the network.

The statistics we have introduced rely on the heterogeneous nature of the degree distribution of real-world networks, and seem to capture both properties common to all growing networks, and some specificities of the networks under study. They are therefore relevent for the analysis of networks' dynamics.

# 4 Conclusion

In this paper, we have undertaken the study of the dynamics of networks, restricting ourselves to the case of growing networks. We have introduced a formal framework for this study, then we have introduced some statistics to study the dynamics of networks, mainly concerning the degrees of nodes and their evolution. To illustrate the relevance of our statistics, we have measured them on two growing networks, without trying to give an interpretation of the observed behaviours.

The statistics we have introduced are relevent in the sense that they capture different behaviours for the networks we have studied. Another way to show that a statistic is relevent would be to compare our measurements with what happens for randomly growing networks: if the behaviour of a given network is similar to that of a random network, then the statistic does not capture something specific to this network. If, on the other hand, the two behaviours are different, then the satistics are relevent. We fully expect that this is the case for the statistics we have introduced, but this comparison has yet to be done.

Our tools, though quite general, do not address all simple characteristics of growing networks dynamics. There are in particular two simple, natural question that are not answered here. The first one concerns the degrees of the nodes between which new links appear: do these links tend to appear attached to high degree nodes, or independently of the nodes' degrees? The second question concerns the distance in the network between nodes joined by new links: are such two nodes closer in general than two randomly chosen nodes? These two questions should be addressed in order to yield tools for answering these questions.

Finally, our tools can be easily adapted to deal with the case of general dynamic networks. They do not, however, fully address the questions natural in this context, and new tools might be introduced to answer these questions.

*Acknowledgments.* This work was partly funded by the GAP (Graphes, Algorithmes et Probabilités) project, and by the PERSI (Programme d'Étude des Réseaux Sociaux de l'Internet, `http://www.liafa.jussieu.fr/ persi`) project.

# References

[Arc] The Internet Traffic Archive. http://ita.ee.lbl.gov/.

[Bar02] A.-L. Barabási. *Linked: The New Science of Networks*. Perseus Publishing, 2002.

[BS03] S. Bornholdt and H.G. Schuster, editors. *Handbook of Graphs and Networks*. Wiley-VCH, 2003.

[Dat] The Internet Movie Database. http://www.imdb.com/.

[EL03] Eli Eisenberg and Erez Y. Levanon. Preferential attachment in the protein network evolution. *Physical review Letter*, 91(13), September 2003.

[GL05] J.-L. Guillaume and M. Latapy. Relevance of massively distributed explorations of the internet topology: Simulation results. In *Proceedings of the IEEE 24-th INFOCOM'05 conference*, 2005.

[Hol03] Petter Holme. Networks dynamics of ongoing social relationships. *Europhysics Letters*, 64(3), January 2003.

[KMCB95] David L. Banks Kathleen M. Carley, Ashish Sanil and Dean Behrens. Testing alternative models of network evolution. 1995.

[New01] M.E.J. Newman. Clustering and preferential attachment in growing networks. *Phys. Rev. E*, 64, 2001.

[OdC03] Roberto N. Onody and Paulo A. de Castro. Complex network study of brazilian soccer players. *Physical Review E*, 70, 2003.

[PHL04] Chistofer R. Edling Petter Holme and Fredrik Liljeros. Structure and time-evolution of an internet dating community. *Socical Networks*, 26(2), May 2004.

[Sni01] Tom A.B. Snijders. Statistical evaluation of social networks dynamics. *Social Methodology, Pp 361-395*, 2001.

# *Algorithmes d'ordonnancements bicritères en-lignes (Résumé)*

Fabien Baille, Evripidis Bampis, Christian Laforest, Nicolas Thibault

*LaMI, CNRS / Université d'Evry, Tour Evry 2, 523 place des terrasses, 91000 EVRY France*

Nous proposons dans cet article des algorithmes d'ordonnancement permettant à un opérateur d'un lien haut-débit de gérer des demandes de réservations provenant de ses clients. Nos propositions ont les avantages suivants : (1) elles permettent de gérer les demandes *à la volée*, dès qu'elles arrivent à l'opérateur ; (2) nous prouvons qu'elles permettent d'approcher à de faibles facteurs multiplicatifs près, le profit maximum atteignable et, simultanément, le nombre maximum de clients pouvant être servis ; (3) ces facteurs sont ajustables et permettent à l'opérateur de donner plus d'importance relative à l'un ou l'autre des critères.

**Keywords:** Ordonnancement, algorithme en-ligne, algorithme bicritère

## 1 Introduction

Dans cet article nous étudions la situation suivante. Un lien réseau, composé de sous-canaux identiques, est géré par un *opérateur*. On peut considérer par exemple un lien transatlantique haut débit (fibre optique ou satellite). Un *client* peut demander à utiliser un sous-canal entre deux dates données. Il fait donc une demande de réservation exclusive de ressources (sous-canal) pendant une période choisie. Il peut par exemple demander un canal entre 13h00 et 15h00. Une demande acceptée peut se traduire par la mise à disposition exclusive de ce client du sous-canal numéro 12 entre 13h00 et 15h00.

Chaque demande, si elle est satisfaite, engendre un *profit* pour l'opérateur. Ce dernier doit bien sûr maximiser son profit (bénéfice à court terme). Cependant, il doit aussi veiller à accepter le plus de demandes possibles pour maximiser la satisfaction globale des clients (notion de bien être social, éventuellement imposé par une autorité de régulation). Étant donné que les ressources sont limitées, toutes les demandes ne peuvent (en général) pas être satisfaites. L'opérateur doit alors tenter de maximiser ces *deux* critères (profit et nombre de demandes satisfaites) *simultanément*.

Les (nombreux) travaux de la littérature traitant ce type de problèmes ont en général les caractéristiques suivantes :

- les auteurs considèrent que l'opérateur connaît la totalité des demandes des clients *à l'avance*. Cela peut s'appliquer par exemple à des demandes de réservations de connexions d'un jour sur l'autre : les clients font des réservations la veille pour le lendemain. Cependant, avec de telles méthodes (de type *hors-ligne*), l'opérateur ne peut pas traiter les données à la volée, dès qu'elles arrivent, pour satisfaire (ou pas) immédiatement la demande du client.
- Les auteurs traitent *séparément* le problème de maximisation du profit et celui de maximisation du nombre de clients servis. On notera que les problèmes algorithmiques sous-jacents étant souvent NP-complets, des algorithmes d'approximation sont souvent proposés. Cependant, une solution viable maximisant (ou approchant) les deux critères à la fois n'est en général pas proposée.

Dans cet article nous levons ces deux contraintes majeures, en proposant un algorithme *en-ligne* (traitant les demandes à la volée) et bicritère (i.e. *garantissant* que les deux critères sont approchés simultanément).

Dans ce qui suit, nous décrivons notre modélisation de la situation initiale grâce à des ordonnancements. Nous décrivons dans la section 2 notre algorithme en-ligne bicritère et nous donnons ses garanties. Nous

explorons ensuite dans la section 3 des généralisations de la méthode de base qui permettent, entre autres, de prendre en charge des demandes de réservation dites *dégradables*.

Notons que la place limitée nous interdit de donner tous les détails et toutes les preuves de nos résultats dans cet article.

**Un intervalle. Une machine.** Nous représentons la situation décrite en préambule par le modèle d'ordonnancement suivant. Notre système (i.e. le lien réseau) est composé de *k machines identiques* (i.e. les $k$ sous-canaux). Un *intervalle* (i.e. une demande d'un client) $\sigma_i$ est la donnée des paramètres suivants.

Une *date d'arrivée* $r_i \geq 0$ avant laquelle $\sigma_i$ ne peut pas être ordonnancé (date à laquelle la demande du client arrive et à partir de laquelle la réservation doit débuter). Une *date de fin* $d_i$ à laquelle $\sigma_i$ doit se terminer. Un *poids* $w_i = d_i - r_i$, qui est ici égal à la durée d'exécution de l'intervalle.

**Modèle d'exécution des intervalles sur les machines.** Un intervalle $\sigma_i$ est dit *ordonnancé* lorsqu'il est exécuté sans interruption, entre les dates $r_i$ et $d_i$ (i.e. pendant une durée d'exactement $p_i = d_i - r_i$).

La traduction en termes de réseaux est la suivante. Si la demande $\sigma_i$ est acceptée par l'opérateur, un sous-canal est alors exclusivement dédié à (ou loué par) ce client à partir de la date $r_i$ jusqu'à la date $d_i$. Cette opération génère alors un profit de (ou fonction de) $w_i = d_i - r_i$ pour l'opérateur (même si cela coûte en pratique plus cher au client). Un tel poids, proportionnel à la durée de l'utilisation de la ressource, correspond donc au profit (ou à une partie du profit) de l'opérateur si la demande est satisfaite.

Notons que la préemption n'est pas autorisée dans la mesure où les clients demandent l'utilisation exclusive d'un sous-canal. Notons aussi que l'opérateur peut interrompre un intervalle avant que celui-ci termine son exécution. Dans ce cas, l'intervalle interrompu ne rapporte rien pour les deux critères considérés et il est définitivement perdu.

**Ordonnancement. Poids et taille d'un ordonnancement.** Un *ordonnancement* $O$ d'un ensemble d'intervalles $\sigma$ est un sous-ensemble d'intervalles de $\sigma$ exécutés de telle façon que chaque machine n'exécute qu'au plus un intervalle à chaque instant et que chaque intervalle de $O$ est exécuté sur une seule machine. Par abus de notations, nous confondrons un ordonnancement $O$ avec l'ensemble des intervalles qu'il ordonnance. Nous noterons $W(O) = \sum_{\sigma_i \in O} w_i$ le *poids* d'un ordonnancement $O$ et $N(O) = |\{\sigma_i \in O\}|$ sa *taille*.

**Ordonnancements optimaux.** Soit $\sigma$ un ensemble d'intervalles. Nous noterons $W^*(\sigma) = \max\{W(O) : O \text{ ordonnancement de } \sigma\}$ le *poids optimal* de $\sigma$ et $N^*(\sigma) = \max\{N(O) : O \text{ ordonnancement de } \sigma\}$ la *taille optimale* de $\sigma$.

**Les algorithmes en-lignes.** Dans cet article nous supposons que les intervalles arrivent au fur et à mesure du temps et que l'opérateur *ne connaît pas* à l'avance les intervalles à traiter. Nous supposerons qu'il reçoit ces intervalles un par un, *dans l'ordre des dates de début* ($r_1 \leq r_2 \leq \ldots \leq r_i \leq \ldots$), et qu'une fois *révélé*, un intervalle $\sigma_j$ doit être traité par l'algorithme de l'opérateur. Pour cela, il peut soit *rejeter* $\sigma_j$ soit *accepter* $\sigma_j$ et l'ordonnancer sur une des $k$ machines.

Pour préciser les choses, nous noterons $O_{j-1}$ l'ordonnancement courant avant la révélation de $\sigma_j$. Dans ce contexte, tout algorithme d'ordonnancement en-ligne supprime/*interrompt* dans un premier temps un certain ensemble $S$ (éventuellement vide) d'intervalles déjà ordonnancés dans $O_{j-1}$ dans sa *phase de suppression*. Il exécute ensuite sa *phase d'ordonnancement* qui est réduite à deux possibilités :

- Il ordonnance $\sigma_j$ sur une machine libre. Dans ce cas, le nouvel ordonnancement $O_j$ est $O_j = (O_{j-1} - S) \cup \{\sigma_j\}$.
- Il *rejette* $\sigma_j$. Dans ce cas, le nouvel ordonnancement $O_j$ est $O_j = O_{j-1} - S$.

Il est important de noter ici que si un intervalle $\sigma_l$ a été accepté à un moment donné et qu'il est interrompu ensuite (avant sa date de terminaison $d_l$) alors aucun bénéfice n'est tiré de lui (ni pour la taille ni pour le poids) dans les futurs ordonnancements (un intervalle rejeté ou interrompu est définitivement perdu).

La mesure de qualité d'un algorithme en-ligne $A$ d'ordonnancement pour un critère $C$ (pour nous $C = W$ ou $C = N$) est connu sous le nom de *rapport de compétitivité* (voir [BEY98, FW98]). $A$ est $\rho$-compétitif si

pour toute suite d'intervalles révélée de manière en-ligne, $\sigma_1, \ldots, \sigma_i$

$$\rho C(A(\sigma_1, \ldots, \sigma_i) \geq C^*(\{\sigma_1, \ldots, \sigma_i\})$$

(avec $A(\sigma_1, \ldots, \sigma_i)$ l'ordonnancement construit par $A$ lorsque $\sigma_i$ a été traité).

On trouvera dans [FN95] un algorithme en-ligne, nommé *GOL*, qui résout le problème de la taille de manière optimale, c'est-à-dire avec un rapport de compétitivité de 1. A partir des travaux de [BNCK+99] on peut tirer un algorithme en-ligne, nommé *LR*, nécessitant au moins $k \geq 2$ machines pour son exécution et qui a un rapport de compétitivité de $\frac{2}{1-\frac{2}{k}}$ pour le problème du poids.

## 2 Notre algorithme bicritère en-ligne

Un algorithme est dit bicritère en-ligne s'il est simultanément $\alpha_N$-compétitif pour la taille et $\alpha_W$-compétitif pour le poids. Nous avons déjà proposé dans [BBL03, BBL04b] un algorithme (d'approximation) bicritère pour ce type de mesure. Il nécessite cependant la connaissance préalable totale des intervalles à ordonnancer. Le problème traité ici est donc de *nature* très différente puisque les traitements doivent se faire au "fil de l'eau", dès qu'un intervalle est révélé, sans connaître les intervalles qui seront proposés dans l'avenir. De plus, un intervalle rejeté ou interrompu est définitivement perdu.

Cette section a pour objectif de donner les grandes lignes de notre méthode. L'idée générale est la suivante : on a un système à $k$ machines dites "réelles", sur lesquelles les ordonnancements successifs doivent être produits. Nous allons contrôler la taille de nos ordonnancements successifs en "simulant" GOL et leurs poids en "simulant" LR. Ces deux simulations vont se faire sur des machines que nous qualifions de "virtuelles" car les ordonnancements produits sur celles-ci ne vont servir qu'à prendre des décisions pour les ordonnancements sur les machines réelles. Un paramètre $r$ ($1 \leq r \leq k-3$) choisi au début par l'opérateur va permettre d'attribuer $r$ machines "virtuelles" à GOL et $k-r$ à LR.

Chaque nouvel intervalle révélé $\sigma_i$ est soumis à $GOL^r$ (i.e. GOL appliqué sur $r$ machines) et à $LR^{k-r}$ (i.e. LR appliqué sur $k-r$ machines), qui traitent chacun et indépendamment $\sigma_i$. Comme il a été dit plus haut, $GOL^r$ (resp. $LR^{k-r}$) fait une phase de suppression qui va éliminer un ensemble $S_{GOL}$ (resp. $S_{LR}$) (potentiellement vide) des intervalles déjà ordonnancés sur les $r$ (resp. $k-r$) machines virtuelles, avant l'arrivée de $\sigma_i$. Ensuite, $GOL^r$ (resp. $LR^{k-r}$) décide d'ordonnancer ou pas $\sigma_i$ sur une de ses $k$ machines virtuelles libres.

En fonction des décisions de $GOL^r$ et $LR^{k-r}$, notre algorithme va faire les opérations suivantes :

- Il va supprimer de ses $k$ machines réelles les intervalles qui n'apparaissent plus ni dans l'ordonnancement courant de $GOL^r$ ni dans celui de $LR^{k-r}$. NB : si un intervalle a été supprimé par $GOL^r$ par exemple mais qu'il est encore présent dans l'ordonnancement courant de $LR^{k-r}$, il n'est pas supprimé de l'ordonnancement des machines réelles.
- Si au moins un des deux algorithmes $GOL^r$ ou $LR^{k-r}$ ordonnance $\sigma_i$ alors notre algorithme fait de même.

On peut montrer qu'en faisant ainsi, à chaque étape, si $\sigma_i$ est ordonnancé par $GOL^r$ ou $LR^{k-r}$ alors il existe une machine réelle libre pour l'ordonnancer aussi. On montre de même qu'à chaque étape, l'ensemble des intervalles ordonnancés sur les machines réelles est égal à l'union des intervalles ordonnancés par $GOL^r$ et $LR^{k-r}$.

Grâce à cette propriété, le comportement et les garanties offertes par les deux algorithmes sont conservés. Seul le fait d'utiliser $r$ (resp. $k-r$) machines pour $GOL^r$ (resp. $LR^{k-r}$) au lieu de $k$ machines "dégrade" son rapport de compétitivité. Tout cela peut être contrôlé analytiquement et conduit au résultat suivant.

**Théorème 1** *Notre algorithme appliqué avec le paramètre $r$ est $\frac{k}{r}$ compétitif pour la taille et $\frac{2}{1-\frac{2}{k-r}} \cdot \frac{k}{k-r}$ compétitif pour le poids (pour $k \geq 4$ et $1 \leq r \leq k-3$).*

Remarquons que si on prend par exemple $r = \frac{k-2}{3}$ (si $k-2$ est divisible par 3) on obtient un algorithme $\frac{3}{1-\frac{2}{k}}$-compétitif pour la taille et $\frac{3}{1-\frac{2}{k}}$-compétitif pour le poids. Les rapports sont donc "petits" et ajustables via le paramètre $r$.

# 3 Généralisations

La méthode décrite dans la section 2 peut en fait être étendue. L'objet de cette section est de donner les principaux résultats de ces extensions. Celles ci sont utiles pour prendre en compte d'autres fonctions de profit (section 3.1) et d'autres modes de réservation (section 3.2).

## 3.1 Prise en compte de poids quelconques

Notre façon de "mixer" deux algorithmes monocritères est en fait générale et peut très bien s'appliquer aussi à d'autres fonctions de poids que celles utilisées jusqu'à présent (la maximisation de la taille peut être vue comme une maximisation de poids unitaires). En modifiant la description de l'algorithme on peut arriver, avec les mêmes techniques de preuves, à montrer le résultat plus général suivant.

**Théorème 2** *Soit A (resp. B) un algorithme en-ligne d'ordonnancement ayant un rapport de compétitivité $\rho_A$ (resp. $\rho_B$) pour la maximisation du poids des ordonnancements dans lesquels chaque intervalle $\sigma_i$ a un poids $W_i^A$ (resp. $W_i^B$). Notre algorithme utilisant A sur r machines et B sur $k-r$ machines induit un rapport de compétitivité de $\rho_A \frac{k}{r}$ pour la maximisation des poids $W^A$ et, simultanément, un rapport de $\rho_B \frac{k}{k-r}$ pour la maximisation des poids $W^B$.*

## 3.2 Le cas des intervalles dégradables

Dans le modèle vu au début de l'article, un intervalle est soit totalement ordonnancé (et dans ce cas on en tire un bénéfice) soit il est rejeté ou interrompu avant sa date de fin et dans ce cas aucun bénéfice n'en est tiré. Cette section est consacrée à un modèle intermédiaire dans lequel chaque demande $\sigma_i$ a une date de début $r_i$, une date de fin $d_i$, un poids $w_i = d_i - r_i$ mais aussi une *date de fin minimale* $r_i < q_i \leq d_i$. Un tel intervalle sera considéré comme ordonnancé s'il est exécuté sur une machine entre les dates $r_i$ et $t_i$ avec $q_i \leq t_i \leq d_i$ ; ces intervalles sont dits *dégradables*. Dans ce cas, l'intervalle rapporte un profit égal à son temps d'exécution, c'est-à-dire $t_i - r_i$ et ajoute une unité à la taille de l'ordonnancement courant. Sinon, il ne rapporte rien, ni dans l'unité de poids ni dans celle de taille. Ce modèle dégradable a été traité dans [BBL04a] dans le cas hors-ligne et en considérant les deux objectifs séparément.

L'algorithme décrit dans la section 2 peut dans ce cas être modifié et adapté pour prendre en compte ces intervalles *dégradables*. Le résultat est alors le suivant (les rapports de compétitivité sont calculés par rapport aux ordonnancements optimaux dégradables).

**Théorème 3** *Notre algorithme en-ligne traitant les intervalles dégradables a un rapport de compétitivité de $\frac{k}{r}$ pour la taille et de $4\frac{k}{k-r-2}$ pour le poids (pour $k \geq 4$ et $1 \leq r \leq k-3$).*

# Références


[BBL03] F. Baille, E. Bampis, and C. Laforest. Rapports d'approximation effectifs d'ordonnancements bicritères d'intervalles sur des machines identiques. In *ALGOTEL*, pages 147–154, 2003.

[BBL04a] F. Baille, E. Bampis, and C. Laforest. Maximization of the size and the weight of schedules of degradable intervals. In K.-Y. Chwa and I. Munro, editors, *proceedings of COCOON'04*, LNCS No. 3106, pages 219–228. Springer-Verlag, 2004.

[BBL04b] F. Baille, E. Bampis, and C. Laforest. A note on bicriteria schedules with optimal approximation ratios. *Parallel Processing Letters*, 14:315–323, 2004.

[BEY98] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University press, 1998.

[BNCK+99] Amotz Bar-Noy, Ran Canetti, Shay Kutten, Yishay Mansour, and Baruch Schieber. Bandwidth allocation with preemption. *SIAM J. Comput.*, 28(5):1806–1828, 1999.

[FN95] U. Faigle and M. Nawijn. Note on scheduling intervals on-line. *Discrete Applied Mathematics*, 58:13–17, 1995.

[FW98] A. Fiat and G. J. Woeginger. *Online algorithms: The state of the art*. LNCS no. 1442, Springer, 1998.

# *Détection de structures de communautés dans les grands réseaux d'interactions*

Pascal Pons[†]

*LIAFA (CNRS – Université Paris 7), 2 place Jussieu 75251 Paris Cedex 05*

Les parties localement denses de graphes globalement peu denses (*communautés*) apparaissant dans la plupart des réseaux d'interactions réels jouent un rôle important dans de nombreux contextes. Nous proposons une mesure de la similarité structurelle (au sens des communautés) entre sommets basée sur des marches aléatoires. Nous montrons ensuite comment calculer efficacement cette distance. Nous l'utilisons alors dans un algorithme de clustering hiérarchique qui fonctionne en temps $O(mn^2)$ et en espace $O(n^2)$ dans le pire des cas, cependant il tire profit des caractéristiques des réseaux réaliste, la complexité en temps pour la plupart des cas pratiques est alors de $O(n^2 \log n)$ ($n$ et $m$ sont respectivement le nombre de sommets et d'arêtes du graphe). Les résultats expérimentaux montrent que notre algorithme produit des structures de communautés de meilleures qualité que celles obtenue grâce aux algoritmes existants, tout en restant compétitif du point de vue du temps de calcul.

**Keywords:** communautés, grands réseaux d'interactions, marches aléatoires, clustering

## 1 Introduction

L'existence dans les réseaux d'interactions de zones plus densément connectées que d'autres découle souvent de la présence dans le graphe d'une structure de communautés. Ce type de structure intervient dans de nombreux réseaux d'interactions et apporte de l'information sur leur organisation. Les communautés peuvent avoir des interprétations différentes suivant le type de réseau considéré (réseaux sociaux, réseaux biologiques, réseaux d'information, réseaux de télécommunication, ...).

La notion de communauté dans un graphe est cependant difficile à définir formellement. C'est pourquoi toutes les approches récentes ont utilisé une notion intuitive des communauté : une communauté est alors vue comme un ensemble de sommets dont la densité de connexions internes est plus forte que la densité de connexions vers l'extérieur (sans pour autant définir de seuil formel). Le but est alors de trouver une partition des sommets en communautés vérifiant ce critère (sans savoir a priori le nombre de telles communautés). Le domaine a récemment reçu un vif regain d'intérêt avec l'arrivée de nouveaux algorithmes pouvant être classées en deux grandes familles :

– Les approches séparatives [GN02, NG04, RCC$^+$04] scindent le graphe en plusieurs communautés en retirant progressivement les arêtes reliant deux communautés distinctes. Les arêtes sont retirées une à une, à chaque étape les composantes connexes du graphe obtenu sont identifiées à des communautés. Le processus est répété jusqu'au retrait de toutes les arêtes. On obtient alors une structure hiérarchique de communautés. Les méthodes existantes diffèrent par la façon de choisir les arêtes à retirer.
– Les approches agglomératives [New04, DMn04] utilisent une approche s'apparentant à celle du clustering hiérarchique dans lesquelles les sommets sont regroupés itérativement en communautés. Notre approche en fait partie.

## 2 Evaluation de la similarité et de la différence des sommets grâce à des marches aléatoires courtes

Nous considérons un graphe non orienté $G = (V,E)$ possédant $n = |V|$ sommets et $m = |E|$ arêtes. Afin de pouvoir grouper les sommets du graphe par communautés, nous allons introduire une distance entre

[†]pons@liafa.jussieu.fr

sommets (qui évalue leur proximité structurelle) basée sur des marches aléatoires dans le graphe.

## 2.1 Marches aléatoires dans un graphe

Nous allons nous intéresser à un processus de marche aléatoire (ou de diffusion) dans le graphe $G$. Le temps est discrétisé ($t = 0, 1, 2, \ldots$). À chaque instant, un marcheur est localisé sur un sommet et se déplace à l'instant suivant vers un sommet choisi aléatoirement et uniformément parmi les sommets voisins. La suite des sommets visités est alors une marche aléatoire, et la probabilité de transition du sommet $i$ au sommet $j$ est à chaque étape : $P_{ij} = \frac{A_{ij}}{d(i)}$ (où $A$ est la matrice d'adjacence du graphe et $d(i)$ le degré du sommet $i$).

Ceci définit la matrice de transition $P$ de la chaîne de Markov correspondante. Nous pouvons aussi écrire $P = D^{-1}A$ en introduisant la matrice diagonale des degrés des sommets $D$ ($D_{ii} = d(i)$ et $D_{ij} = 0$ pour $i \neq j$). Nous noterons $P^t_{ij} = (P^t)_{ij}$ la probabilité d'aller du sommet $i$ au sommet $j$ en $t$ étapes.

## 2.2 Communautés et marches aléatoires courtes

Nous allons considérer des marches aléatoires de longueur fixée $t$ suffisamment courte pour ne pas atteindre le régime stationnaire mais suffisamment longue pour collecter des informations globales sur le voisinage du point de départ de la marche (en pratique nous prenons $3 \leq t \leq 6$). Nous supposons donc connaître pour tous sommets $i$ et $j$ les probabilités $P^t_{ij}$ d'aller de $i$ à $j$ en $t$ pas. Chaque probabilité $P^t_{ij}$ apporte de l'information sur $i$ et $j$. Ainsi si nous voulons comparer deux sommets $i$ et $j$ nous utiliserons l'information fournie par les probabilités $P^t_{ik}$, $P^t_{jk}$, $P^t_{ki}$ et $P^t_{kj}$ pour tout sommet $k$. Cependant, il est facile de montrer que $d(i)P^t_{ij} = d(j)P^t_{ji}$, cette relation nous montre que ces deux probabilités apportent exactement la même information.

Pour comparer les sommets $i$ et $j$ nous pourrons donc nous contenter de comparer les vecteurs $P^t_{i\bullet}$ et $P^t_{j\bullet}$ correspondant aux lignes $i$ et $j$ de la matrice $P^t$. Pour celà nous nous appuyons sur le fait que les sommets d'une même communauté ont tendance à "voir" les sommets éloignés de la même façon, ainsi si $i$ et $j$ sont dans la même communauté et $k$ dans une autre communauté il y a de fortes chances que $P^t_{ik} \simeq P^t_{jk}$. Nous devons aussi tenir compte de l'influence du degré $d(j)$ du sommet d'arrivée sur la probabilité $P^t_{ij}$ (cette probabilité est d'ailleurs directement proportionnelle au degré lorsque $t \to \infty$). Dans cette optique, nous introduisons la distance $r_{ij}$ suivante :

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(P^t_{ik} - P^t_{jk})^2}{d(k)}} = \left\| D^{-\frac{1}{2}} P^t_{i\bullet} - D^{-\frac{1}{2}} P^t_{j\bullet} \right\| \tag{1}$$

Nous généralisons naturellement cette distance entre les communautés (ensembles de sommets) en utilisant les probabilités $P_{C\bullet} = \frac{1}{|C|} \sum_{i \in C} P_{i\bullet}$ pour toute communauté $C \subset V$.

**Théorème 1** *La distance $r_{ij}$ s'exprime directement en fonction des vecteurs propres à droite $v_\alpha$ de $P$ associés aux valeurs propres $\lambda_\alpha$.*

$$r^2_{ij} = \sum_{\alpha=2}^{n} \lambda_\alpha^{2t} (v_\alpha(i) - v_\alpha(j))^2 \tag{2}$$

Cette formule établit un lien entre notre distance et les propriétés spectrales de la matrice de transfert, ainsi son comportement est principalement dominé par les vecteurs propres de $P$ liés aux plus grandes valeurs propres. Soulignons que de nombreux travaux ont montré ou constaté que les propriétés structurelles du graphe sont capturées par ces vecteurs propres [SEMS04, DMn04]. Cependant ces approches ont toutes l'inconvénient de devoir calculer explicitement les valeurs propres et les vecteurs propres associés. Notre approche s'appuie sur les mêmes fondements théoriques mais vise à s'affranchir du calcul pénalisant des valeurs propres, comme nous allons le voir dans le paragraphe suivant.

## 2.3 Calcul de la distance

La définition donné par l'équation (1) de $r_{ij}$ permet un calcul en $O(n)$ à partir des probabilités $P_{ik}$ et $P_{jk}$. Le calcul exact de ces différentes probabilités peut être fait en calculant la matrice $P^t$. Les graphes rencontrés en pratique sont généralement peu denses, il est alors judicieux de calculer chaque vecteur $P^t_{i\bullet}$

en multipliant $t$ fois le vecteur $P^0_{i\bullet}$ ($P^0_{i\bullet}(k) = P^0_{ik} = \delta_{ik}$) par $P^T$. Chaque multiplication demande $O(m)$, le calcul de chaque $P^t_{i\bullet}$ demande donc un temps $O(tm)$.

Suivant la ressource mémoire, nous pourrons soit calculer tous les vecteurs probabilités en un temps $O(tnm)$ et les stocker en mémoire pour pouvoir ensuite faire chaque calcul de distance en un temps $O(n)$, ou bien calculer à la volée chaque distance en un temps $O(tm)$.

# 3 Description de l'algorithme

Le but est maintenant de construire une structure de communautés en accord avec notre distance. Nous allons pour ceci proposer une variante de la méthode de clustering hiérarchique de Ward.

## 3.1 Principe de notre algorithme

Nous allons partir d'une partition du graphe en $n$ communautés contenant chacune un seul sommet et fusionner successivement des communautés jusqu'à obtenir une seule communauté correspondant au graphe entier de la façon suivante :

- Choisir deux communautés à fusionner selon la procédure décrite en 3.3.
- Fusionner ces deux communautés en une nouvelle communauté.
- Mettre à jour les distances entre communautés.
- Calculer et mémoriser un paramètre de qualité $Q$ présenté en 3.2

Nous obtenons ainsi une suite de $n$ partitions des sommets en communautés $P_0 \dots P_{n-1}$ parmi lesquelles il va falloir choisir la meilleure (maximisant $Q$).

## 3.2 Evaluation de la qualité d'une partition du graphe en communautés

Nous utilisons pour évaluer la qualité de la partition $P_k$ la modularité $Q(P_k) = Q_k$ introduite dans [NG04, New04] :

$$Q_k = \sum_{C \in P_k} (e_C - a_C^2) \tag{3}$$

où $e_C$ est la fraction d'arêtes internes à la communauté $C$ et $a_C$ est la fraction d'arêtes ayant une extrémité dans la communauté $C$. Cette quantité est calculable en un temps $O(m)$ et peut être mise à jour après chaque fusion en $O(1)$. Nous retenons comme résultat de notre algorithme la partition du graphe possédant la meilleure modularité.

## 3.3 Choix des communautés à fusionner

Pour diminuer le nombre de cas à considérer et ainsi la complexité, nous envisagerons uniquement les fusions de communautés ayant au moins une arête entre elles. Nous utilisons ensuite le principe de l'algorithme de clustering hiérarchique suivant la méthode de Ward [War63]. Cette méthode fusionne les communautés de telle sorte à minimiser à chaque étape la moyenne $\sigma$ de la distance au carré de chaque sommet à sa communauté :

$$\sigma(P_k) = \frac{1}{n} \sum_{C \in P_k} \sum_{i \in C} r_{iC}^2 \tag{4}$$

Nous devons alors connaitre à chaque étape les valeur $\Delta\sigma$ des variations de $\sigma$ après chaque fusion possibles de communautés. Celles-ci peuvent être efficacement obtenues grâce au théorème suivant :

**Théorème 2** *La variation $\Delta\sigma(C_1, C_2)$ de $\sigma$ lors d'une fusion de deux communautés $C_1$ et $C_2$ en une nouvelle communauté $C_1 \cup C_2$ est :*

$$\Delta\sigma(C_1, C_2) = \frac{1}{n} \frac{|C_1||C_2|}{|C_1| + |C_2|} r_{C_1 C_2}^2 \tag{5}$$

### 3.4 Complexité

Le calcul initial des probabilités $P^t_{i\bullet}$ pour tous les sommets $i$ se fait en $O(tmn)$. À chaque étape de fusion, le maintien de la structure de donnée codant les communautés se fait après chaque fusion en temps constant et le calcul de $P^t_{(C_1\cup C_2)\bullet}$ se fait en temps $O(n)$. Ces deux étapes demandent donc pour les $n-1$ étapes un temps total en $O(n^2)$. La partie la plus coûteuse est le calcul des nouvelle valeurs de $\Delta\sigma(C_1\cup C_2, C_3)$. À chaque fusion, il faut recalculer les distances entre la nouvelle communauté créée et chacune de ses communautés voisines en temps $O(n)$.

**Théorème 3** *Le nombre total de calculs de distances effectué lors des fusion est au plus* $2Hm$ *où* $H$ *est la hauteur de la structure arborescente des communautés trouvées par l'algorithme.*

La complexité totale de l'algorithme est donc $O(mnH)$, soit $O(mn^2)$ dans le pire des cas. Cependant les graphes réalistes sont peu denses (n = $(O)(n)$) et leurs communautés s'organisent généralement de manière hiérarchique, de plus l'algorithme de Ward que nous utilisons a tendance à créer des communautés de tailles équilibrées. Ces deux remarques font qu'en pratique la structure arborescente hiérarchique des communautés s'approche du cas favorable où $H = O(\log n)$ qui correspond à une performance de l'algorithme $O(n^2 logn)$.

## 4 Conclusion

Nous avons proposé et implémenté (disponible sur [www]) un algorithme efficace de détection de communautés dans les grands réseaux d'interactions basé sur l'étude des marches aléatoires dans les graphes. Notre approche s'applique à des graphes de grandes tailles qui ne pouvaient pas être traités par la plupart des algorithmes existants. Ainsi, nous avons pu obtenir les structures de communautés de graphes de tailles allant jusqu'à 100 000 sommets. Cette courte présentation ne peut malheureusement contenir ni les nombreuses améliorations et optimisations que nous avons proposées ni les tests de comparaison que nous avons effectués. Ces tests ont souligné la qualité des partitions trouvées par notre algorithme par rapport à celles trouvées par les autres algorithmes existants.

## Références

[DMn04] Luca Donetti and Miguel A. Muñoz. Detecting network communities : a new systematic and efficient algorithm. *arXiv :cond-mat/0404652*, 2004.

[GN02] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *PNAS*, 99(12) :7821–7826, 2002.

[New04] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 69(6) :066133, 2004.

[NG04] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 69(2) :026113, 2004.

[RCC+04] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. Defining and identifying communities in networks. *PNAS*, 101(9) :2658–2663, 2004.

[SEMS04] Ingve Simonsen, Kasper Astrup Eriksen, Sergei Maslov, and Kim Sneppen. Diffusion on complex networks : a way to probe their large-scale topological structures. *Physica A : Statistical Mechanics and its Applications*, 336(1-2) :163–173, May 2004.

[War63] Joe H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301) :236–244, 1963.

[www] `http://www.liafa.jussieu.fr/~pons/`.

# *Pair-à-pair*

Mercredi 12 mai 2005, 10:30 - 12:00

---

# *Clustering in P2P exchanges*

Jean-Loup Guillaume and Stevens Le Blond

LIAFA – CNRS – *Université Paris 7, 2 place Jussieu, 75005 Paris, France.*
*Faculty of Sciences – Vrije Universiteit, De Boelelaan 1081A, 1081 HV Amsterdam, The Netherlands.*

We propose here an analysis of a rich dataset which gives an exhaustive and dynamic view of the exchanges processed in a running eDonkey system. We focus on correlations in terms of data exchanged by sets of peers having provided or queried at least one common data. We introduce a method to capture these correlations (namely the data clustering), and study it in detail. We then use it to propose a very simple and efficient way to group data into clusters and show the impact of this underlying structure on search in typical P2P systems.

**Keywords:** peer-to-peer, clustering, graphs

## 1 Preliminaries

P2P networks such as KaZaA, eDonkey, Gnutella and more recently BitTorrent are nowadays the most bandwidth consuming applications on the Internet, ahead of Web traffic [SGD+02]. Their analysis and optimisation therefore appears as a key issue for computer science research. However, the fully distributed nature of most of these protocols makes it difficult to obtain relevant information on their actual behavior. Since these behaviors have some crucial consequences on the performance of the underlying protocol (both in terms of answer speed and in terms of used bandwidth), it is a challenge of prime interest to collect and analyze such data. The observed properties may then be used to design efficient protocols.

### *Context*

In the last few years, both active and passive measurements have been used to gather information on peers behaviors in P2P networks. These studies gave evidence for a variety of properties which appear as fundamental characteristics of such systems. Among them, let us notice the high ratio of free-riders, the heterogeneous distribution of the number of queries by peer, and recently the presence of semantic clustering in file sharing networks (see [FHKM04, SMZ02, GLB05] and references therein). This last property captures the fact that if two peers are interested in some data, they will probably share interest in other data. Connecting directly such peers, makes it possible to take benefit from this semantic clustering to improve search algorithms and the scalability of the system.

In [FHKM04], the authors propose a protocol based on this idea, which reaches very high performances. It however relies on a static classification which can hardly be maintained up to date. Another similar approach consists in the addition of a link in a P2P overlay between peers exchanging files [SMZ02, VKMvS04]. This has the advantage of being very simple and allows a significant improvement of the search process. In [FHKM04, HKFM04] the authors use traces of a running eDonkey network, obtained by crawling caches of a large number of peers. They study some statistical properties like replication patterns, various distributions, and clustering based on file types and geography, but they also use these data to simulate protocols and to evaluate their performances. The use of actual P2P traces where previous works used models (whose validity is hard to evaluate) is an important step. However, the large number of free-riders, as well as other measurements problems, makes it difficult to evaluate the relevance of such traces. Moreover, these measurements miss the dynamic aspects of the exchanges and the fact that fragment of files are made available by peers *during* the download of the files.

### *Framework and contribution*

Our work lies in this context: we collected traces using a modified eDonkey server, which made it possible to grab accurate information on *all* the exchanges processed by a large number of peers during a significant portion of time. The server handled up to 50 000 users simultaneously and we collected 24 hour traces. The size of a typical trace at various times is given in Figure 1. See [BLG04] and references therein, for details on the measurement procedure, on the protocol and on the properties of our traces.

A natural way to encode the gathered data is to define a bipartite graph $Q = (P, D, E)$, called *query graph*, where $P$ is the set of peers in the network, $D$ is the set of data and $E \subseteq P \times D$ is the set of undirected edges, where $\{p, d\} \in E$ if and only if the peer $p$ is *active* for the data $d$ ($p$ is interested in or is a provider of $d$). Notice that this graph evolves during time.

In order to analyze our data, we will also consider the (weighted) *data graph* $\mathcal{D} = (D, E, w)$ obtained from the query graph $Q$ where $D$ is the set of data, $E \subseteq D \times D$ is the set of undirected edges, where $\{d_1, d_2\} \in E$ if and only if there exists a peer active for both $d_1$ and $d_2$ in $Q$. Finally, $w$ is a weight function over the nodes and the edges such that $w(d)$ is the total number of data exchanged by peers active for $d$, and $w(d_1, d_2)$ is the number of data exchanged by peers active for both $d_1$ and $d_2$.

| | 6h | 12h | 18h | 24h |
|---|---|---|---|---|
| peers | 26187 | 29667 | 43106 | 47245 |
| data | 187731 | 244721 | 323226 | 383163 |
| links in $\mathcal{Q}$ | 811042 | 1081915 | 1571859 | 1804330 |
| links in $\mathcal{D}$ | 12238038 | 20364268 | 31522713 | 38399705 |

**Figure 1:** Time-evolution of the basic statistics for $\mathcal{Q}$ and $\mathcal{D}$

In this paper, we first focus on the *data clustering*, which captures how much the exchanges processed by two sets of peers are similar. We then show that these properties have significant impact on the efficiency of searches in the network and therefore may be used in the design of efficient P2P protocols.

## 2 Data clustering analysis

Our aim is now to analyze similarities between data in terms of exchanges processed by peers active for them. In particular, given two data $u$ and $v$ exchanged by a given peer $p$ we are interested in the number of other common data exchanged by peers active for $u$ or $v$. This can be measured using the following parameter:

$$c(u,v) = \frac{w(u,v)}{w(u) + w(v) - w(u,v)}$$

Indeed, the two data $u$ and $v$ induce an edge $\{u,v\}$ in $D$, the weight $w(u,v)$ is nothing but the number of common data exchanged by peers active for $u$ or $v$, and the expression $w(u) + w(v) - w(u,v)$ gives the total number of data exchanged by peers active for $u$ or $v$. Finally, $c(u,v)$ therefore measures how much these exchanges overlap. Notice that its value is between 0 and 1.

The value of $c(u,v)$ may however be strongly biased: if one of the two nodes has a high weight and the other a low one, then the value would be very low. For example, if a data with an high popularity [†] is connected to an unpopular one, then the clustering will probably be low, even if the few data exchanged by the lowest population are completely included in the large set of data exchanged by the highest one.

In order to capture such dissymmetric behaviors, we will also consider the following statistical parameter:

$$\overline{c}(u,v) = \frac{w(u,v)}{\min(w(u), w(v))}$$

which still lies in $[0;1]$ but is always larger than $c(u,v)$ and does not have this drawback. For instance, in the case described above the obtained value is 1. We will call $c(u,v)$ the *clustering* of $\{u,v\}$ and $\overline{c}(u,v)$ its *min-clustering*.

[†] The popularity of a data is the number of peers active for that data.

Figure 2 shows the time-evolution of the distributions of $c(u,v)$ and $\overline{c}(u,v)$, respectively. First notice that the general shape of these distributions is very stable along time, which indicates that the observations we will derive are not biased by the timescale or date considered.

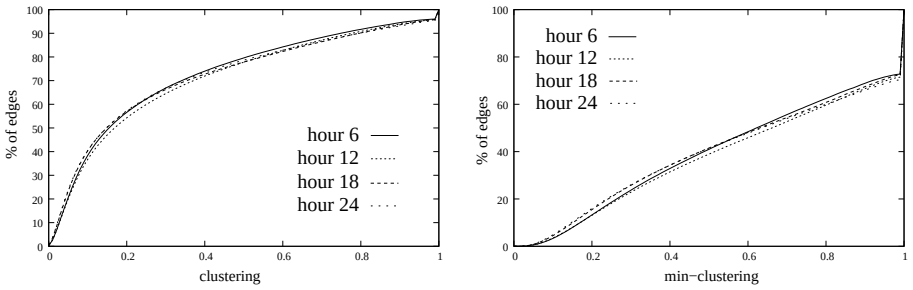



**Figure 2:** Time evolution of the $c(u,v)$ cumulative distribution (left) and of the $\overline{c}(u,v)$ cumulative distribution (right)

Now let us observe (Figure 2 left) that around 60% of the edges always have a clustering lower than 0.2. This may indicate that the overlap of exchanges is not as high as expected. However, this may be a consequence of the fact that both the peer activity and the data popularity are very heterogeneous: there are very active peers while most of them are not, and there are very popular data while most are not. This induces in $\mathcal{D}$ many links between data of very different popularity and a low clustering.

This can be corrected using the distribution of min-clustering (Figure 2 right): only 15% of the edges have a min-clustering lower than 0.2 while for nearly 60% higher or equal to 0.5, which indicates that the overlap is indeed high. For instance, 30% of the overlaps between all exchanges are a complete inclusion.

Such results could denote the presence of a hierarchy among these exchanges. While few popular data form the roots of $\mathcal{D}$, a large number of less popular data have their exchanges highly included into the upward level. Notice that if confirmed, such a property would give a solution to build dynamically a multicast tree from a P2P overlay.

# 3 Consequences on searching

Following several previous works [FHKM04, SMZ02, VKMvS04, HKFM04]), one may wonder if the properties highlighted in previous section may be used to improve search in P2P systems. Suppose that each peer knows the peers active for the same data as itself. Then, when $p$ sends a query for $d$, it first queries these peers. If one of them provides $d$, then it sends it directly to $p$. In this case, the clustering effect has been used and the data was found using only one hop search.

The time-evolution of this hit ratio is plotted in Figure 3 (left). Despite it is quite low in the first few minutes (due to the server bootstrap), the ratio quickly converges to a value close to 45%.

To have a better understanding, the percentage of hits after 24h is correlated with the corresponding percentage of peers, the volume of requests they generated and the replication pattern of the queried data at the Figure 3 (right). The first thing to notice is that nearly 25% of the peers do not find *any* data using the proposed approach. This is quite surprising given the fact that we observed that 50% of *all* the requests are routed successfully with one hop. This can be understood by observing that this 'null hit' population generated only 7% of the requests and then only slighly influenced the high hit rate previously observed. Additionally, the queried data appear to be very slightly replicated at the time they were asked. This low volume of requests together with the low replication pattern constitutes an explanation to the null hit rate: these peers are not active for enough data nor enough replicated ones to find new ones using the one hop protocol.

On the other hand, more than 10% of the peers have a *perfect* success rate. One could think that such a result would imply a prohibitive amount of requests, Figure 3 (right) indicates that it's not the case but the percentage of requests is rather correlated with the number of peers who proceed them. Notice however that data found this way appear to be very highly replicated (the population being active for these data at the time they were asked represents 15% of the peers active for other queried data) which explains the high

success rate. Finally, notice that the average peer's success rate increases to nearly 60% if the 'null hit' population is removed from the calculus.

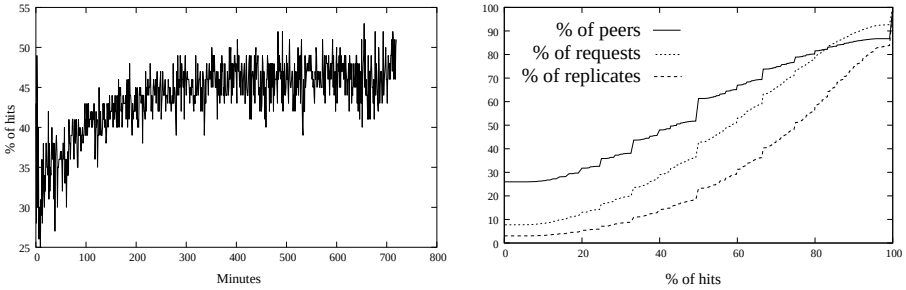



**Figure 3:** Time evolution of the hit % using the one hop protocol (left) and cumulative distribution of the % of peers, the associated % of requests they generated, the % of replication of the queried data and the % of hits they obtained after 24h using the one hop protocol (right)

Maintaining one group knowledge for each data exchanged in the system might be too costly. However, such a knowledge is already used for other issues. In BitTorrent, for instance, some centralized components, known as trackers, periodically distribute sources of peers asking for a given data. One exchange overlay by data is then maintained to guarantee fairness together with optimal transfer rates [Coh03]. Such a mechanism could also be used to propagate the requests of the one hop protocol we just described to provide efficient routing facilities in BitTorrent without more efforts.

# 4 Conclusion

In this contribution, we proposed simple statistical parameters to capture the correlations between the set of peers active for a given data. We used these parameters to confirm that semantic clustering can be used to improve search algorithms. These parameters can also be used to define a very simple and efficient way to compute data clusters, which partially succeed in capturing similarities between data. Notice also that we focused here on *data*, but the same kind of approach might be fruitful with peers.

Finally, let us insist on the fact that the analysis of large real-world traces like the one we presented here is only at its beginning, and that much remains to understand from it. The lack of relevant statistical parameters (concerning for example the dynamics of the trace), and of efficient algorithms to deal with such traces are among the main bottleneck to this, but studies like the one we presented here show that simple methods can already bring much information.

# References

[BLG04] Stevens Le Blond, Matthieu Latapy, and Jean-Loup Guillaume. Statistical analysis of a p2p query graph based on degrees and their time evolution. IWDC'04, 2004.

[Coh03] Bram Cohen. Incentives build robustness in bittorrent, 2003.

[FHKM04] F. Le Fessant, S. Handurukande, A.-M. Kermarrec, and L. Massoulié. Clustering in peer-to-peer file sharing workloads. 3rd International Workshop on Peer-to-Peer Systems (IPTPS), September 2004.

[GLB05] Jean-Loup Guillaume, Matthieu Latapy, and Stevens Le Blond. Clustering in p2p exchanges and consequences on performances. 4th International Workshop on Peer-To-Peer Systems (IPTPS'04), 2005.

[HKFM04] S. Handurukande, A.-M. Kermarrec, F. Le Fessant, and L. Massoulié. Exploiting semantic clustering in the edonkey p2p network. 11th ACM SIGOPS European Workshop (SIGOPS), September 2004.

[SGD+02] S. Saroiu, K. Gummadi, R. Dunn, S. Gribble, and H. Levy. An analysis of internet content delivery systems, 2002.

[SMZ02] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. Internet Measurement Workshop 2002, November 2002.

[VKMvS04] S. Voulgaris, A.-M. Kermarrec, L. Massoulie, and M. van Steen. Exploiting semantic proximity in peer-to-peer content searching. FTDCS, 2004.

# *Exploiter les agrégats et les lois de puissances pour le pair-à-pair*

Philippe Gauron[†]

*LRI, Univ. Paris-Sud, 91405 Orsay cedex, France. courriel : gauron@lri.fr.*

Il a été récemment montré que les centres d'intérêt des utilisateurs de réseaux pair-à-pair ont des propriétés *d'agrégat* que l'on peut utiliser afin de réduire le trafic induit par les méthodes de recherche à base d'inondation. Ces centres d'intérêt ont aussi des propriétés de *lois de puissance* utilisables pour la conception d'algorithmes de recherche basés sur le routage. Dans cet article, nous montrons que l'utilisation conjointe de ces deux propriétés permet la conception d'algorithmes de recherche décentralisés, simples et efficaces. En particulier, nous présentons un algorithme ne nécessitant pas de topologie logique structurée pour connecter les pairs. Des simulations effectuées sur des traces issues d'un réseau réel montrent que les recherches s'effectuent en un nombre d'étapes logarithmique en utilisant cet algorithme.

**Keywords:** Pair-à-pair, Peer-to-peer, P2P, réseaux sans échelle

## Introduction

Cet article s'intéresse aux systèmes pair-à-pair *décentralisés*. Dans ces systèmes, deux méthodes de transfert de requêtes sont utilisables : l'inondation (comme dans Gnutella), et les tables de hachage réparties (THR ou DHT : Distributed HashTables, comme CAN et Chord). Chacune de ces approches a ses inconvénients : le trafic engendré par l'inondation encombre la bande passante, tandis que les protocoles à base de THR nécessitent des connexions inspirées de topologies statiques parfois difficiles à maintenir. Les THR ne permettent pas non plus de recherches complexes, par exemple utilisant des expressions régulières. C'est pourquoi la conception de protocoles de recherche simples renvoyant des réponses rapides et ayant un faible trafic de contrôle reste un problème ouvert.

Notre article propose un algorithme bénéficiant des atouts des deux méthodes : une recherche *simple* dans un système pair-à-pair *non structuré* permettant des recherches complexes. Ce procédé s'appuie sur les propriétés statistiques des requêtes dans les réseaux pair-à-pair réels. En effet, on peut vouloir rapprocher le réseau logique des centres d'intérêts des pairs : si deux pairs ont des intérêts communs, alors ils vont probablement beaucoup échanger, et gagneraient donc à être proches dans le réseau logique. En pratique, les pairs ont tendance à se regrouper en communautés, au sein desquelles ils font beaucoup d'échanges, alors qu'ils ne font que que très peu d'échanges avec l'extérieur. Sur ces bases ont été proposées des améliorations pour des systèmes existants (par exemple [4, 7, 8]). Notre objectif dans cet article est de montrer que ces travaux peuvent être poussés plus avant tout en conservant un système très simple.

## 1 Le graphe des intérêts des pairs

On peut représenter les intérêts des pairs sous forme d'un graphe où deux pairs sont connectés si et seulement si ils ont des intérêts communs. Donner une définition formelle d'« avoir des intérêts communs » est difficile. Plusieurs propositions ont été faites, basées

[†]Cette recherche a reçu le soutien du projet INRIA « Grand Large », et du projet « PairAPair » de l'ACI « Grandes Masses de Données »

sur des mots-clés, les objets en commun, ou des représentations vectorielles. On utilisera ici la définition simple suivante : deux pairs sont connectés dans le graphe des intérêts s'ils ont échangé un objet dans le passé. Remarquons que deux pairs connectés de cette façon peuvent avoir à un moment donné des intérêts très différents, mais leurs intérêts ont au moins été liés à un instant donné.

Il a été montré récemment (voir par exemple [3, 4]) que, comme beaucoup de réseaux sociaux et de systèmes complexes, le graphe des intérêts défini ci-dessus a des propriétés statistiques non-triviales le rendant très différent des graphes aléatoires, en particulier :

- il a une faible densité (le degré moyen est très faible par rapport au nombre de nœuds) ;
- sa distance moyenne est faible (logarithmique par rapport au nombre de nœuds) ;
- il est formé d'agrégats locaux (si la densité globale est faible, la densité locale, elle, est forte) ;
- il est sans échelle (les degrés sont très hétérogènes, la plupart des nœuds ont un degré faible, mais quelques uns ont de forts degrés).

## 2 Utilisation des propriétés des réseaux sans échelle et des propriétés d'agrégats

L'utilisation des lois de puissance des réseaux réels pour concevoir des méthodes de recherches efficaces a été proposée dans plusieurs travaux (comme [1, 2, 6]). Les auteurs de ces articles approximent les distributions de degrés hétérogènes par des lois de puissance, et étudient les propriétés des marches (aléatoires ou déterministes) dans des graphes aléatoires avec de telles distributions de degrés (voir aussi [5]). Ainsi, en passant par les voisins de plus fort degrés, une requête trouve sa cible en un temps linéaire [1]. Par ailleurs, [2] montre par simulation qu'une recherche passant par les nœuds de plus fort degré est plus efficace qu'une marche aléatoire. Elle reste toutefois polynomiale si l'on tient compte des boucles.

Tout comme la nature hétérogène des pairs est visible dans la distribution des degrés, des facteurs culturels et sociaux engendrent une structure d'agrégats sur le graphe des intérêts. Par exemple, si un pair $p_1$ est intéressé par un objet $O$ possédé par un pair $p_2$, il va probablement être intéressé par d'autres objets possédés par $p_2$. En fait, $p_1$ sera probablement intéressé par les objets possédés par les pairs intéressés par $O$. Ainsi (1) les pairs s'organisent en communautés (sous-graphes denses), et (2) deux pairs qui échangent des données sont fortement susceptibles d'échanger à nouveau d'autres données. Différents travaux ont proposé d'utiliser ces propriétés pour améliorer des systèmes existants, et ont mis en évidence que la nature d'agrégats des intérêts peut être utilisée pour créer des systèmes pair-à-pair efficaces [4, 7, 8].

## 3 Notre contribution

La propriété de loi de puissance a été utilisée avec un routage vers le plus fort degré dans les réseaux sans échelle mais n'offrant pas de propriété d'agrégats spécifique. Les algorithmes de recherche obtenus ainsi sont polynomiaux (c'est-à-dire en $n^{\alpha}$, $\alpha > 1$). Par ailleurs, la propriété d'agrégats a été utilisée pour améliorer des protocoles à base d'inondation donc sans routage. La méthode proposée ici consiste à utiliser le routage vers le voisin de plus fort degré directement dans le graphe des intérêts. En effet, la loi de puissance du graphe permet un routage de plus fort degré, tandis que la présence d'agrégats permet à ce routage d'atteindre les objets recherchés en temps logarithmique, par rapport à un temps polynomial permis par les réseaux avec agrégats.

Dans la méthode que nous proposons, *QRE*, le réseau logique est ainsi le graphe des intérêts (défini par les requêtes précédément exécutées) évoluant donc à chaque requête. Dans le réseau logique de *QRE*, un nœud est simplement connecté à tout nœud avec lequel il a déjà échangé des objets. Il tient donc compte de la structure d'agrégat observée dans les réseaux pair-à-pair (les communautés). De plus, il utilise une recherche par les voisins de plus fort degrés, bénéficiant ainsi des résultats observés sur les graphes en loi de puissance.

Une requête est de la forme $\langle @, O, k \rangle$ où @ est l'adresse de la source (par exemple l'adresse IP), $O$ la description de l'objet recherché, et $k \geq 1$ le nombre de copies de $O$ cherchées. Chaque nœud maintient à jour le degré de ses voisins ainsi que la listes des objets qu'ils possèdent. L'algorithme de recherche effectue une recherche en profondeur, guidée par le voisin de plus fort degré, qui s'arrête lorsque $k = 0$. Pour chaque nœud : s'il a $O$, il l'envoie au demandeur @. Si un de ses voisin a l'objet $O$, ce dernier est contacté pour envoyer $O$ à @. Puis la requête est renvoyée au voisin de plus fort degré non visité préalablement. À chaque copie trouvée, le fournisseur et la source sont connectés, et $k$ est décrémenté.

Pour éviter les boucles, les nœuds gardent trace des destinations des requêtes qu'ils ont transmises. Remarquons alors que si au moins $k$ copies de l'objet demandé existent, la requête les trouve (il y a donc exhaustivité), mais s'il n'y a pas assez de copies, la requête va parcourir tout le graphe. Pour gérer cela, on peut borner le temps de vie des requêtes.

# 4 Simulations

Lorsqu'un nœud se connecte, c'est pour récupérer un objet ou en mettre à disposition. Il utilise une passerelle tirée aléatoirement uniformément et lance à travers celle-ci une recherche de l'objet $O$ qu'il cherche ou qu'il possède. Il se connecte ensuite au nœud qui répond à sa requête (qui lui renvoie $O$). En attendant d'avoir de nouveaux voisins, le nœud se connecte temporairement à la passerelle. En pratique, la passerelle est souvent un nœud de la même communeauté que le nouveau nœud, ce qui raccourcit les délais des requêtes.

Afin d'évaluer les performances de notre méthode, nous avons effectué des simulations à partir de *traces réelles*. En effet, aucun modèle ne permet de représenter le comportement des pairs, donc d'évaluer notre méthode en tenant compte à la fois des propriétés d'agrégats et sans échelle, et de la probabilité que deux pairs ayant déjà échangé rééchangent. Les traces que nous avons utilisées sont issues d'*edonkey*, et sont analysées dans [3], et elles représentent 2h 53min de fonctionnement. Nos simulations ont utilisé 46,202 pairs.

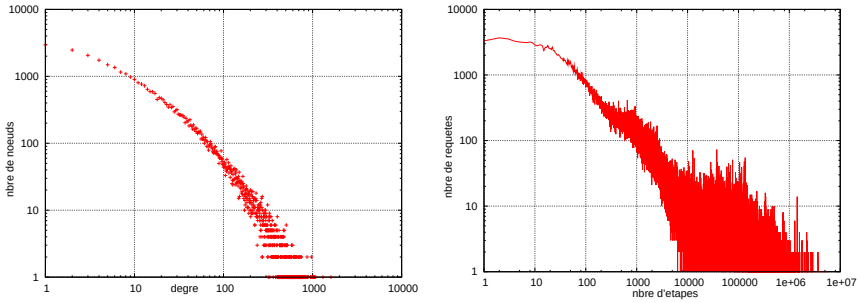



**FIG. 1:** Distribution des degrés (gauche) et nombre de sauts (droite) pour trouver une donnée

La figure 1 (gauche) montre une distribution des degrés à queue lourde, similaire à celles déjà observées pour ce type de réseau. La moyenne des degrés est de 47,9 et seuls 0.25% des nœuds ont un degré supérieur à 300. *QRE* peut donc gérer de grandes quantités de nœuds, ce qui montre qu'il passe à l'échelle.

La figure 1 (droite) représente le nombre de requêtes qui s'effectuent en $k$ sauts. On peut l'approximer par une loi de puissance : la plupart des requêtes s'effectuent en très peu de sauts et très peu de requêtes nécessitent beaucoup de sauts. Les données très rares (ou inaccessibles) nécessitent le parcours complet du réseau.

Le figure 2 montre le nombre d'étapes nécessaires pour trouver un objet en fonction du nombre de nœuds dans le réseau. Une régression linéaire montre que le temps de recherche est linéaire en fonction du logarithme du nombre de nœuds. Les simulations montrent donc que QRE trouve les objets en un nombre de sauts moyen logarithmique. Il est ainsi au moins aussi efficace que toute méthode basée sur une THR (comme Chord).

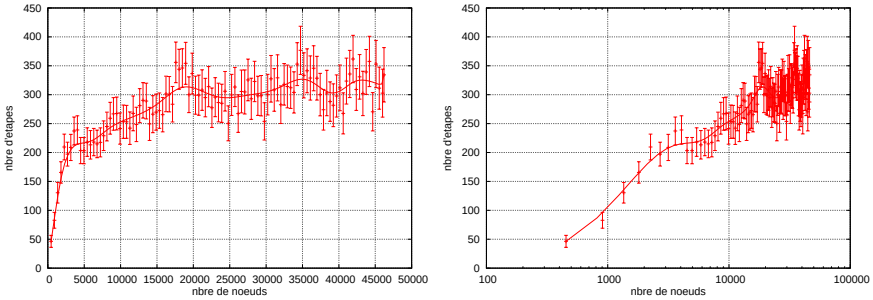



**FIG. 2:** Nombre d'étapes pour trouver un objet

## Conclusion

Des méthodes de recherche plus subtiles et peut-être plus efficaces peuvent sans doute être créées à partir des graphes des intérêts, en utilisant d'autres propritétés du graphe encore à analyser, ou en combinant d'autres approches à celle-ci. En particulier, il n'est pas sûr que l'algorithme de recherche en profondeur d'abord par les voisins de plus fort degrés soit le plus approprié pour ce graphe. Des problèmes peuvent aussi apparaitre si les intérêts des utilisateurs changent au cours du temps, ou si plusieurs personnes utilisent le même nœud. Une solution possible à ce dernier problème serait de permettre plusieurs profils par nœuds, pour éviter l'utilisation de liens inappropriés. Une étude plus poussée serait aussi nécessaire afin de mesurer l'impact d'une limitation des degrés des pairs, ainsi que la robustesse et la sécurité.

**Remerciements :** Cet article est issu d'un article soumis à EuroPar'05 et écrit avec P. Fraigniaud et M. Latapy. Jean-Loup Guillaume a fourni des générateurs de graphe qui ont servi au travail préliminaire.

## Références

[1] L. Adamic, R. Lukose, A. Puniyani, and B. Huberman. Search in power law networks. Physical Review E, vol. 64, 046135, 2001.

[2] B. Kim, C. Yoon, S. Han, and H. Jeong. Path finding strategies in scale-free networks. Physical Review E, vol. 65, 027103, 2002.

[3] S. Le-Blond, M. Latapy, and J.-L. Guillaume. Statistical analysis of a P2P query graph based on degrees and their time evolution. In LNCS vol. 3326, pages 126-137 Springer, proceedings of the 6-th Int. Workshop on Distributed Computing (IWDC), 2004.

[4] F. Le-Fessant, S. Handurukande, A.-M. Kermarrec, and L. Massoulié. Clustering in Peer-to-Peer File Sharing Workloads. In 3rd Int. Workshop on Peer-to-Peer Systems (IPTPS), 2004.

[5] Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker. Search and replication in unstructured peer-to-peer networks. In 6th Int. Conference on Supercomputing, pages 84-95, 2002.

[6] N. Sarshar, P. Boykin, and V. Roychowdhury. Percolation search in power law networks : making unstructured peer-to-peer networks scalable. In 4th Int. Conference on Peer-to-Peer Computing (P2P'04), pages 2-9, 2004.

[7] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), IEEE, 2003.

[8] S. Voulgaris, A.-M. Kermarrec, L. Massoulié, and M. van Steen. Exploiting Semantic Proximity in Peer-to-peer Content Searching. In 10th IEEE Int. Workshop on Future Trends in Distributed Computing Systems (FTDCS), pages 238-243, 2004.

# *Optimisation de la bande passante dans un réseau pair-à-pair : la stratégie BitTorrent face à ses challengers*

Fabien de Montgolfier

*LIAFA, Université Paris 7*
*fm@liafa.jussieu.fr*

Les protocoles d'échanges Pair-à-pair actuels incitent leurs clients à échanger les fichiers grâce à un système d'*échanges de blocs* rappelant l'économie libérale. Cette étude présente des algorithmes génétiques permettant de déterminer «la» meilleure stratégie de négociation des blocs dans un réseau Pair-à-pair de type BitTorrent, c'est-à-dire celle qui maximise la vitesse de téléchargement.

**Keywords:** pair-à-pair, optimisation, protocole BitTorrent, algorithmes génétiques

## 1 Introduction

Les protocoles de téléchargement pair-à-pair ont beaucoup évolué depuis leurs débuts (Napster, Gnutella...) Ils étaient initialement basés sur un concept client/serveur distribué : plusieurs pairs[†] possédaient un fichier qui était téléchargé par d'autres pairs. Mais la quantité de gens «partageant» un fichier se réduisait trop par-rapport à celle des *leechs* (téléchargeurs passifs), ce qui ramenait à la problématique classique de la saturation de bande passante en sortie des serveurs.

### *1.1 Le protocole BitTorrent*

Les protocoles de la génération suivante tentèrent de créer un intérêt du client à partager ses fichiers. Parmi les meilleurs systèmes sont ceux où la générosité des premiers protocoles a été remplacée par un modèle «libéral» : chaque fichier est découpé en *blocs*. Les clients ne possédant qu'une partie du fichier peuvent échanger les blocs qu'ils ont déjà contre d'autres. Ainsi, il suffit que très peu de pairs aient la totalité des blocs pour que le système fonctionne, sans surcharger la bande passante de personne.

Le protocole BitTorrent a été inventé par Cohen dans ce but [Coh03]. Il est basé sur le *generous tit-for-tat* : chaque client donnera les blocs qu'il possède préférentiellement aux pairs qui lui ont fourni le plus de blocs. De plus, afin d'alimenter les nouveaux arrivants ou ceux qui ont un débit très limité, on donne aussi des blocs «généreusement». Il s'établi donc une négociation implicite entre les pairs, chacun essayant par essais-erreurs de trouver ceux qui lui fourniront le plus de blocs, et en même temps alimente ses clients[‡].

### *1.2 Objet de cette étude*

Dans cette étude nous ne nous intéressons pas à la manière dont un pair recherche un fichier ni comment il se connecte à d'autres pairs, mais seulement à son protocole de téléchargement. Nous proposons une méthode à base d'algorithmes génétiques, semblable à ce qui a été fait pour le dilemme du prisonnier itéré [Axe87, BDM96] afin d'identifier les meilleures stratégies. On trouvera donc une description des paramètres des stratégies, du protocole de simulation, et quelques résultats d'expérience.

[†] un *pair* est un utilisateur, situé sur une machine et avec un logiciel, qui «partage» ses fichiers

[‡] on appellera par la suite «*clients*» d'un pair tous les autres pairs avec lesquels il est en relation, peu importe le sens du transfert

# 2 Paramètres d'une stratégie

Examinons maintenant comment écrire une stratégie dans un réseau BitTorrent. Les stratégies considérées doivent être assez simple afin que leur espace puisse être «parcouru» par un algorithme génétique, et suffisamment complexes pour comprendre la stratégie BitTorrent originelle ainsi que la plupart des stratégies simples auxquelles on peut penser. Il me semble que les points suivants doivent être pris en compte :

- **Récompense de la fidélité** Le principe de base est que ceux qui ont le plus envoyé seront ceux qui recevront le plus. Ce renforcement mutuel assure la stabilité des échanges.
- **Générosité** Une stratégie peut avoir intérêt à envoyer des données vers des gens qui n'ont pas de relation avec elle. Loin d'être un acte de pur générosité, il s'agit en fait de se signaler : c'est la seule façon d'entamer les «négociations» avec le pair.
- **Horizon temporel** L'évaluation de la performance d'un client est sa performance (réceptions et émissions) moyenne sur une certaine période. Un petit horizon temporel permet d'être réactif tandis qu'un grand lisse les irrégularités éventuelles de transmission.
- **Durée des choix** Une stratégie peut maintenir ses choix de clients un certain temps : cela permet par exemple d'attirer l'attention des nouveaux clients en leur donnant plus. BitTorrent ne réévalue que toutes les 10 secondes.
- **Auto-limitation** Pour ne pas «gaspiller» la bande passante, on peut majorer la différence *envoyé – reçu*. Par exemple une stratégie *donnant-donnant* majore les émissions par les réceptions, et BitTorrent n'a pas de majorant. Les «roublards» airont un majorant négatif !

En fonction de ces considérations, voici un algorithme qui représente la stratégie universelle.

- Répéter
  - Avec une probabilité `g` envoyer un paquet de `r` blocs à un client aléatoire
  - trier les clients selon leurs `h` dernières actions suivant un certain critère `T`
  - répartir `u` blocs aux `c` meilleurs clients
    - soit proportionnellement à ce qu'ils ont envoyé (mais tel que don – reçu $<$ `l`)
    - soit selon un schéma prédéfini `S` (mais tel que don – reçu $<$ `l`)
  - envoyer les blocs restant, par paquets de `r`, à des pairs choisis au hasard
  - Envoyer de la sorte pendant `d` secondes (`d` tours)

L'algorithme de tri `T` peut être aléatoire (ce qui n'est pas très intéressant), par tailles de téléchargement décroissantes (on privilégie ceux qui ont donné le plus) ou par rapports réception/émission décroissants (on privilégie ceux qui ont le meilleur «rendement»).

# 3 Protocole de simulation

Un tournoi contient un certain nombre de pairs (100 dans cette expérience). Chaque pair dispose de la même bande passante et tous doivent télécharger un fichier de la même taille (autant de blocs ; un bloc est téléchargé atomiquement). A chaque instant, un client peut envoyer un nombre maximal de blocs donnés. Par analogie avec une ligne ADSL, ce nombre est fixé à 16, et il n'y a pas de limite de réception.

Si le schéma de répartition est prédéfini, on donnera toujours $c_1$ blocs au meilleur client, $c_2 \leq c_1$ au second et ainsi de suite. équité. Comme il y a au plus 16 blocs envoyés, cela fait 916 possibilités pour le schéma `S`. le *patrimoine génétique* d'une stratégie est donc `[S,g,r,h,T,u,c,l,d]`

## *3.1 Sélection*

Les meilleures stratégies sont choisies par une méthode darwinienne. La performance d'une stratégie est le temps qu'elle met à compléter ses dix premiers téléchargements (dès qu'un téléchargement est terminé, un pair jouant la même stratégie mais inconnu des clients en place remplace le précédant) Les 90 meilleurs survivent pour le tournoi suivant (en subissant une mutation) et les 10 éliminés sont remplacés par des «enfants» des 10 premiers.

### 3.2 Mutations

Les stratégies survivantes subissent toutes une mutation : chacun des paramètres est remplacé avec probabilité $p_r$ par une valeur aléatoire prise uniformément dans l'intervalle et avec une probabilité $p_g$ par une valeur proche (choisie avec une loi gaussienne de moyenne l'ancienne valeur et d'écart-type 10% de l'intervalle). Nous avons pris $p_r = 1\%$ et $p_g = 3\%$.

### 3.3 Reproduction

La reproduction de deux stratégies suit les lois de Mendel : chaque paramètre de l'enfant a 25% de chances d'être la valeur du «père», 25% d'être la valeur de la «mère» et 50% une moyenne (pour les 7 paramètres numériques). Pour maintenir constante la taille de la population, l'enfant remplace un des deux parents (en plus des 10 enfants des plus forts). Il y a 10 couples qui se reproduisent à chaque nouveau tournoi.

### 3.4 Population initiale

Nous avons fait des expériences avec deux types de population initiales : une population «totalement aléatoire» (chaque paramètre pris uniformément dans son intervalle) et une population fixée comprenant les stratégies suivantes (qui sont assez «naturelles») :

- 5 *méchantes* : ne donnent rien
- 5 *aléatoires* :donnent 1 bloc à 16 clients pris au hasard
- 5 *échos* : donnent autant qu'ils ont reçu aux meilleurs donateurs
- 5 *uniques* : donnent tout (16) au meilleur donateur
- 10 *proportionnelles* : donnent proportionnellement aux 4 meilleurs donateurs
- 10 *proportionnelles réactives* : idem mais avec un horizon de 1 au lieu du maximum
- 40 *BitTorrent classiques* : donne 4 aux 4 meilleurs donneurs, plus proba de 33% de don aléatoire de 4, réévaluation tous les 10 coups
- 10 *roublardes* : comme BitTorrent mais l'émission est majorée par la réception moins 1
- 10 *roublardes sur rendement* : idem mais en triant par rapports reçu/émis décroissants, au lieu de trier par réception décroissantes comme toutes les autres

## 4 Résultats

### 4.1 Comparaison statique des stratégies

Un tournoi (sans algorithme génétiques)[§] donne comme résultat (le temps moyen de téléchargement étant de 7000 blocs / don de 16, soit 437.5)

| | | | | | |
|---|---|---|---|---|---|
| 1er | roublarde | 388.3 | 6ème | proportionnelle réactive | 425.6 |
| 2nd | proportionnelle | 379.4 | 7ème | BitTorrent | 492.0 |
| 3ème | écho | 391.8 | 8ème | aléatoire | 582.4 |
| 4ème | unique | 392.0 | 9ème | méchante | 1868.6 |
| 5ème | roublarde sur rendement | 418.83 | | | |

On peut être surpris de la mauvaise performance de BitTorrent. Cela vient sans doute du fait qu'elle est la seule à réévaluer tous les 10 coups alors que les autres réévaluent à chaque fois. Cependant, comme on va le voir, les algorithmes génétiques vainqueurs sont «à la BitTorrent».

### 4.2 Algorithmes génétiques

La surprise des algorithmes génétiques[¶] est la disparition totale des stratégies proportionnelles. Au bout d'une cinquantaine de tournois, ne survivent que des stratégies à schéma fixé qui trient les clients par nombre de blocs reçus décroissants. Les 10 meilleurs stratégies après un million de tours de simulation (300 cycles

[§] programme disponible sur `http://www.liafa.jussieu.fr/~fm/P2P/SimuStat.tgz`

[¶] programme disponible sur `http://www.liafa.jussieu.fr/~fm/P2P/SimuGen.tgz`

de reproduction) dans une simulation donnée sont données par ce tableau, où la première colonne est le schéma de don (par exemple [4 3 2] indique que l'on donne 4 blocs au meilleur client, 3 au second et 2 au troisième). Si la somme n'est pas 16, les autres sont toujours donné au hasard. Les résultats d'une autre simulation diffèrent très peu si l'on garde les même paramètres (nombre de clients, taux de sélection) mais que l'on change la population initiale.

| schéma | horizon `h` | durée `d` | limitation `l` | taille blocs aléa. `r` | générosité `g` |
|---|---|---|---|---|---|
| [3 3 3 3 2 2 ] | 3 | 17 | +5 | 5 | 4% |
| [3 2 2 2 2 2 ] | 10 | 16 | +7 | 7 | 4% |
| [3 3 3 3 2 2 ] | 14 | 13 | +10 | 8 | 7% |
| [3 3 3 3 2 2 ] | 11 | 11 | +4 | 6 | 5% |
| [4 4 4 4 ] | 6 | 13 | -2 | 9 | 2% |
| [4 4 3 2 2 1 ] | 8 | 15 | +15 | 8 | 4% |
| [3 3 3 3 2 2 ] | 17 | 14 | +11 | 4 | 6% |
| [3 3 3 3 2 2 ] | 18 | 11 | +1 | 4 | 1% |
| [5 4 4 2 1 ] | 17 | 13 | +8 | 7 | 0% |
| [3 3 3 3 2 2 ] | 16 | 13 | +9 | 6 | 6% |
| [ 4 4 4 4] | 20 | 10 | +16 ($= +\infty$) | 4 | 33% |

Il est surprenant de constater la grande similitude de ces stratégies avec la stratégie BitTorrent classique (ce que le tournoi classique ne laissait pas du tout présager d'ailleurs) qui est donnée pour mémoire dans la dernière ligne du tableau. Cela valide le choix d'un horizon temporel long, mais également d'une longue durée de maintien des choix. Les principales différences des stratégies gagnantes sont dans le nombre plus élevé de clients (6 contre 4) et le coefficient de générosité beaucoup plus bas (presque négligeable). Certaines stratégies ont une limitation assez forte (on trouve même des stratégies «roublardes» qui donnent moins que ce qu'elles ont reçu) mais les stratégies à limitation faible semblent très bien survivre aussi : le paramètre n'a pas l'air d'être un facteur évolutif majeur.

# 5 Conclusion

Lorsque j'ai commencé ces expériences, je me demandais (sur une question qui m'a été posée par Laurent Viennot) s'il existait des stratégies meilleures que BitTorrent sur le protocole BitTorrent. Cette étude semble répondre par la négative. Les principales différences sont le nombre de clients (que tous les clients actuels permettent de régler) et le facteur de générosité (qui semble trop grand dans BitTorrent). Le principe originel de BitTorrent est un *generous tit for tat*, mais il semble que le côté «donnant-donnant» prédomine sur le côté «généreux» pour une efficacité maximale. À ce détail près, il est validé.

On peut se demander pourquoi ne pas individualiser les blocs : la rareté des blocs peut être utilisée dans la négociation. Le *bloc rare* [MR04] finit par disparaître du réseau et bloque les téléchargements de tous les utilisateurs à 99%. Des protocoles en développement utilisent un mécanisme de redondance des blocs permettant d'éviter ce problème en échange d'une augmentation minime de la taille du fichier. Cette étude dissocie ces deux facettes d'une stratégie pair-à-pair qui sont assez orthogonales.

# Références

[Axe87] R. Axelrod. *Genetic Algorithms and the Simulated Annealing, L. Davis ed.*, chapter The Evolution of Strategies in the Iterated Prisoner's Dilemma, pages 32–41. Pitman, 1987.

[BDM96] Bruno Beaufils, Jean-Paul Delahaye, and Philippe Mathieu. Our meeting with gradual : A good strategy for the iterated prisoner's dilemma. In *Proceedings of the International Conference on Artificial Life V (ALIFE V)*, pages 159–165, 1996.

[Coh03] Bram Cohen. Incentives build robustness in bittorrent, 2003.

[MR04] Fabien Mathieu and Julien Reynier. File sharing in p2p : Missing block paradigm and upload strategies. Technical Report RR 5193, INRIA, 2004. `http://www.inria.fr/rrrt/rr-5193.html`.

# *A fully distributed peer to peer structure based on 3D Delaunay Triangulation*

Moritz Steiner and Ernst Biersack

*Institut Eurecom, 2229, route des Crêtes, 06904 Sophia-Antipolis, France*
*{moritz.steiner,ernst.biersack}@eurecom.fr*

This paper proposes the 3D Delaunay Triangulation ($\mathcal{DT}$) as a promising solution for constructing scalable p2p networks. The key idea is to maintain for each node a $\mathcal{DT}$ of the neighbour nodes. While demonstrating scalability in a real system is not practical for the current work, we demonstrate the scalability of the 3D $\mathcal{DT}$ using simulation. The results obtained indicate that there are upper bounds on the time needed to join and on the average number of neighbours maintained by a peer. Therefore, the amount of bandwidth and processing requirement for each node is bound, independent of the total number of nodes in the system.

**Keywords:** p2p, overlay network, graph, triangulation, delaunay

## 1 Introduction

A peer to peer (p2p) system is one in which autonomous peers have symmetric roles. P2p systems are constructed by connecting various computers (nodes) in a mesh-like fashion, thus forming a virtual network on top of the physical Internet. The term overlay network is thus often used to describe p2p systems. The whole system is only operable, and the peers can only benefit, if the peers that depend on each other in forwarding information and sharing computer resources. This is why issues of scale and redundancy become much more important than in traditional systems.
This paper proposes a fully-distributed p2p architecture, which attempts to solve the scalability problem based on the mathematical construct of the Delaunay Triangulation in 3D. The main contribution of the paper is to propose a resource efficient solution, which requires no server at all, not even for joining.
Section 2 deals with the partitioning of the virtual world; the geometrical construct Delaunay Triangulation is discussed. Section 3 describes the data structure and some of the algorithms we developed, as well as the main problems of the communication protocol.

## 2 Partitioning the 3D virtual world with the Delaunay Triangulation

### *2.1 The definition*

The Delaunay Triangulation — introduced by Boris Delaunay in 1934 [Del34] — decomposes the convex hull of a point set $S$ into unique cells (triangles in $R^2$ and tetrahedra in $R^3$). There are only two conditions building the $\mathcal{DT}$:

**Definition 2.1.1** *Let $S$ be a point set of n points in $R^2$. A $\mathcal{DT}(S)$ in 2D is a triangulation of $S$ where no point $d$ lies inside the circumcircle $\mathcal{C}_{abc}$ of any triangle $\mathcal{T}_{abc}$.*
*Let $S$ be a point set of n points in $R^3$. A $\mathcal{DT}(S)$ in 3D is a triangulation of $S$ where no point $e$ lies inside the circumsphere $\mathcal{C}_{abcd}$ of any tetrahedra $\mathcal{T}_{abcd}$.*

The circumsphere is the sphere defined by the four vertices *abcd* of a tetrahedron $\mathcal{T}_{abcd}$. This definition implies that the tetrahedra are not flat, otherwise their circumscribing sphere is not defined.

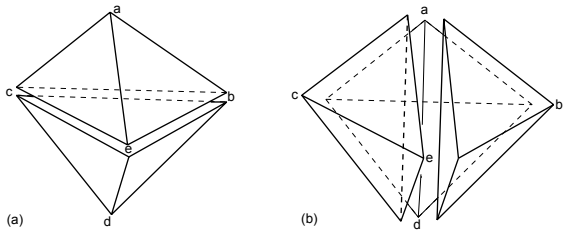



**Fig. 1:** A hexahedra can be separated into two or three tetrahedra [AK00].

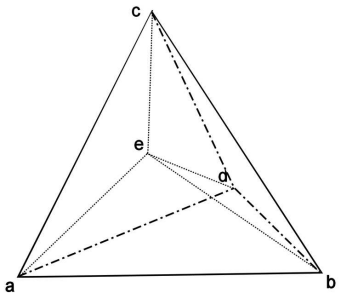



**Fig. 2:** Four new tetrahedra resulting from the split of $\mathcal{T}_{abcd}$ with $e$.

### 2.2 The Delaunay Triangulation construction

The basic component of most algorithms for the construction of the $\mathcal{DT}$ is the *Delaunay diagonal flip*. After a finite number of Delaunay diagonal flips out of any triangulation, the most compact one can be constructed: the $\mathcal{DT}$. For building a p2p infrastructure using $\mathcal{DT}$, a dynamic algorithm is needed. Therefore, in this section, we will focus on an *incremental algorithm*. In the first step the description is only done for the 2D case, then the main problem of switching to 3D is presented.
The addition of a point to the $\mathcal{DT}$ is equal to the problem of computing $\mathcal{DT}_i = \mathcal{DT}(\{p_1,\ldots,p_i\})$ from $\mathcal{DT}_{i-1}$ by inserting $p_i$. Let $\mathcal{T}_{abc}$ be one of the triangles of $\mathcal{DT}_{i-1}$ whose circumcircle $\mathcal{C}_{abc}$ contains the new point $p_i$ and therefore are *in conflict* with $p_i$. It is not longer a Delaunay triangle, according to definition 2.1.1. Let *bc* be the edge that lies inside the quadrilateral $abcp_i$. A *Delaunay diagonal flip* has to be done to replace $\mathcal{T}_{abc}$ and $\mathcal{T}_{bcp_i}$ by $\mathcal{T}_{abp_i}$ and $\mathcal{T}_{acp_i}$. The newly created triangles must be tested as well, with the same process until there are no more triangles in conflict with any points [AK00].
In case no triangle is in conflict with $p_i$, the convex hull of all points from $S = \{p_1,\ldots,p_{i-1}\}$ needs to be enlarged. First $p_i$ is connected to all points from $S$ which it can *see*, that means to which straight lines can be drawn without crossing any edge from $\mathcal{DT}_{i-1}$. All created edges are Delaunay edges to $S \cup p_i$. The opposite edges to $p_i$ must be checked and flipped if necessary.
Liebeherr has used the 2D $\mathcal{DT}$ as structure for a p2p network called Hypercast [LN02]. Hypercast has been tested with up to 10,000 nodes running on up to 100 computers.
In 2D a quadrilateral can only be divided into two triangles, whereas the hexahedra in figure 1 can be separated into two tetrahedra $\mathcal{T}_{abce}$ and $\mathcal{T}_{bcde}$ (Figure 1 a) or into three tetrahedra $\mathcal{T}_{acde}$, $\mathcal{T}_{abde}$ and $\mathcal{T}_{abcd}$ (Figure 1 b). The principal problem in 3D is that there are some sets of points that allow different triangulations, depending on the insertion order of the nodes, therefore the 3D $\mathcal{DT}$ is not unique.
The limited space of this paper does not allow to describe the removal of a point from the $\mathcal{DT}$.

## 3 A distributed algorithm for the Delaunay Triangulation in 3D

Currently no algorithm exists to compute the $\mathcal{DT}$ in a distributed way. By distributed we mean that each point of the triangulation corresponds to one computer, which is aware of its direct neighbours only. The main problem is the consistency: All nodes are aware of their direct neighbours only and not of all the nodes of the triangulation. Thus, it is very difficult to keep the views of all the nodes consistent.
This section describes the data structure, followed by a short presentation of the main method, which adds a point (join from a peer point of view). Finally, some test and validation methods are presented and the main problems of the communication protocol are discussed.

### 3.1 The data structure and the basic methods

We do not use the *Quad-Edge* structure developed by Guibas and Stolfi [GS85] or the simpler structure based on a double connected edge list (DCEL) [BKOS00], but one that allows for easier navigation, even though some information is redundant. There are two main objects in this structure: `Node` and `Tetrahedron`. A `Tetrahedron` is the main object for storing the triangulation. It is composed of its four vertices, stored as `Node`s and its circumsphere stored as `Sphere` that consists of a `Point` for its cen-

ter and its radius. Furthermore it stores its four facets as `Planes` and its four neighbour `Tetrahedron`. A `Node` stores its position in a `Point` and maintains a list of its neighbour `Nodes` and a list of the tetrahedra (`Tetrahedron`) where it belongs to. These references allow to navigate in the triangulation. All the other objects have only a supporting function, e.g. the `Vectors` and `Lines`, which are basically needed for the calculation of the center of the circumsphere of a `Tetrahedron` and to calculate the representation of a `Plane` by its normal vector starting from three `Points` or `Nodes`.

## 3.2 Join the Delaunay Triangulation

The node $nj$ wishing to join needs to know one node belonging to the $\mathcal{DT}$ that executes the join procedure for $nj$. The nearest node *nn* to the desired location of $nj$ is searched by recursively traveling through the $\mathcal{DT}$. For the following pseudo code we assume that the $\mathcal{DT}$ has at least four nodes and the node *nn* has been found.

```
tetra ← neighbours(nn,2) /* all tetrahedra of nn and their respective neighbour tetrahedra */
liesInside ← false /* true if nj lies at least inside one sphere */
for i ← 0 to tetra.length do
    if nj lies inside sphere of tetra[i] then
        newtetras ← null /* variable for the min. 2 and max. 4 tetrahedra resulting from the split */
        for j ← 0 to 3 do
            newtetras[j] ← new
            Tetrahedron(tetra[i](j%4),tetra[i]((j+1)%4),tetra[i](j+2)%4),nn)
            /* tetra[i](j) returns the j.th point of tetra[i] */
            if not nn lies inside tetra[i] then
                if tetra[i]((j+3)%4) lies inside sphere of tetra[i] then newtetras[j] ← null
            end
        end
        Update the neighbour relationship between the 5 nodes (the four nodes of tetra[i] and nn)
        Update the list of tetrahedra of the 5 nodes
        Update the tetrahedra neighbourship relations between the newtetras
        Update the convex hull (in case one facet of tetra[i] was on it)
        liesInside ← true
    end
end
if not liesInside then enlarge the convex hull of DT with nj
```

If the joining node lies inside one tetrahedron (not only inside its sphere) the split creates four new tetrahedra (Figure 2). If the new node lies outside the tetrahedra (but inside its sphere) only two or three new tetrahedra are created (see section 2.2 and figure 1).

## 3.3 Tests and validation

During the execution of the code, whenever a change occurs, all nodes and tetrahedra affected are checked for compliance with definition 2.1.1. Moreover all neighbour relations between the tetrahedra are checked: Two tetrahedra having a common plane must cross-reference each another. Several different scenarios have been tested to validate the algorithm: Randomly distributed nodes; Nodes randomly distributed on the surface of a sphere; Points distributed in smaller and bigger clusters, computed with the Lévy Flight method [Lév37].

As there is no general view of all nodes and all tetrahedra, all nodes write the tetrahedra they know in a common file to gain a global view. This file is then compared with the output of the CGAL $\mathcal{DT}$ algorithm [CGA].

## 3.4 The communication protocol

The protocol itself is not discussed in this paper, but two major problems are described: the consistence of the triangulation during and after the concurrent execution of more than one task and the crash of a peer and the involved reparation of the triangulation.

It is crucial for the coherence of the virtual world, to locally accept only one alteration a time. For example

two peers may join or leave the network at the same time only if they do not alter the same nodes or tetrahedra. Otherwise the two (ore more) peers would create or destroy tetrahedra or peer relationships without knowing about the actions of the other one, which would automatically lead to an inconsistent triangulation. Therefore, all peers involved in a join or leave procedure are locked. Peers can still perform actions, e.g. send messages, but they cannot alter the triangulation. This approach is comparable to the lock mechanisms implemented in database systems to allow transactions conform with the ACID paradigm. A peer that wants to join tries to lock all nodes involved. If it doesn't succeed, it unlocks them and retries after a certain time, to avoid deadlocks.
In the case of a crash of one of the peers, the other peers must detect the crash. Hence heartbeat messages are sent to the neighbours. If one peer has not received the heartbeat message from one of its neighbours, it takes the leadership and locks all the neighbours of the crashed peer and executes the *leaving procedure* for the crashed peer.

### 3.5 Complexity

The task of building the $\mathcal{DT}$ in a static way (all the nodes are known at the beginning of the construction) has a lower bound of $\Omega(n \log n)$ in 2D and $\Omega(n^2)$ in 3D, where $n$ is the number of nodes [YB98].
In our case not the overall time complexity is important, but the complexity for joining one node to the $\mathcal{DT}$. As the simulation results have shown, the number of neighbours of a node is bound, as well as the number of tetrahedra destroyed during the join. These two numbers do not grow with the total number of nodes in the $\mathcal{DT}$. If the nodes are randomly distributed, the average number of neighbour nodes is 15, the maximum is 35. The average number of tetrahedra destroyed during the join of a node is 15, the maximum is 28. The time complexity of the join procedure does not depend on the total number of nodes in the $\mathcal{DT}$, but it only depends on the number of neighbour nodes. This can be seen very clearly, because the main loop of the pseudo code in section 3.2 is executed for all neighbour tetrahedra of degree two, this number directly depends on the number of neighbour nodes. Since the number of neighbour nodes is nearly constant, and does especially not grow with the number of nodes in the $\mathcal{DT}$, the complexity of the join procedure is constant as well.

## 4 Conclusion

This paper presents a solution to the scalability problem in p2p networks, by showing a way to partition the 3D virtual space with the help of the Delaunay Triangulation. More work is needed to improve the performance of the presented algorithm and to allow efficient movements of the nodes without using the join and leave methods.

## References

[AK00] F. Aurenhammer and R. Klein. *Handbook of Computational Geometry*, chapter 18, pages 201–290. Elsevier Science Publishers, 2000.

[BKOS00] M. De Berg, M. Van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2nd edition, 2000.

[CGA] CGAL. The cgal website. http://www.cgal.org.

[Del34] B. Delaunay. Sur la sphère vide. A la mémoire de Georges Voronoi. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskih i Estestvennyh Nauk*, 7:793–800, 1934.

[GS85] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of voronoi. *ACM Transactions on Graphics*, 4(2):74–123, April 1985.

[LN02] J. Liebeherr and M. Nahas. Application-layer multicasting with Delaunay triangulation overlays. *IEEE Journal on Selected Areas in Communications*, 20(8):1472–1488, October 2002.

[Lév37] P. Lévy. *Théorie de l'Addition des Variables Aléatories*. Gauthier-Villars, Paris, 1937.

[YB98] M. Yvinec and J.-D. Boissonnat. *Algorithmic Geometry*. Cambridge University Press, 1998.

# *Radio & 802.11*

Jeudi 13 mai 2005, 09:45 - 10:30

---

# *Performances de IEEE 802.11 pour la communication dans un convoi de véhicules*

Yacine KHALED, Bertrand DUCOURTHIAL et Mohamed SHAWKY

*Heudiasyc, UMR-CNRS 6599, Université de Technologie de Compiègne, B.P 20529 F-60205, Compiègne, France*
*{Yacine.Khaled,Bertrand.Ducourthial,Mohamed.Shawky}@hds.utc.fr*

Les réseaux locaux sans fil basés sur la norme IEEE 802.11 ont constitué ces dernières années une solution de connexion réseau offrant mobilité, flexibilité et faible coût de déploiement et d'utilisation. Cela encourage l'utilisation de ce protocole pour les nouvelles applications nomades. Parmi ces applications, nous pouvons citer les communications entre véhicules qui permettent une meilleure sécurité routière et de nouveaux services aux conducteurs. Un tel réseau à forte dynamique met à l'épreuve les performances de la norme IEEE 802.11. Dans cet article, nous nous sommes intéressés aux performances de la norme IEEE 802.11 pour la communication inter-véhicule à travers des simulations de convois de véhicules.

**Keywords:** Communication inter-véhicule, IEEE 802.11, Network Simulator 2

## 1 Etude des communications inter-véhicules

### *1.1 Objectif et méthodologie*

De nos jours, le domaine de la communication inter-véhicules a vu une grande avancée [Tsu02, LH]. L'existance des équipements pour la norme IEEE 802.11 [LA04, Gas02] avec des prix abordables, encourage son utilisation dans ce type d'application. Ce type de réseaux est caractérisé par la forte mobilité des véhicules, ce qui peut occasionner une topologie très dynamique, un taux de perte important et une très courte durée de communication entre les véhicules. Par exemple, avec un rayon de communication de 250 mètres, deux véhicules qui se croisent à 90 km/h ne peuvent communiquer que durant 10 secondes. Dans ce type d'applications, les mobiles communiquant sont organisés, dans la plupart des cas, en convois circulant dans le même sens (convoi principal). Un autre convoi, qu'on considèrera comme perturbateur, circule dans le même sens ou dans le sens inverse. Les communications sont réalisées par diffusion de messages à plusieurs sauts.

L'étude des performances du IEEE 802.11 dans un convoi de véhicules est faite avec des simulations. Celles-ci sont réalisées avec Network Simulator 2 [ns]. Nous avons fixé plusieurs paramètres à partir des expérimentations sur route. Nous pouvons citer l'utilisation de cartes 802.11b d'une portée de communication de 520 mètres grâce à une antenne externe, le modèle de propagation *two-ray ground*, qui simule une communication avec une seule réflexion sur le sol, et l'utilisation du protocole UDP.

Dans les sections suivantes, nous présentons les différents scénarios et les résultats obtenus à travers notre étude des performances du IEEE 802.11 pour la communication inter-véhicules.

## 2 Convoi de véhicules

L'objectif de ce premier scénario est d'étudier les délais de transmission et les débits de réception dans un convoi de véhicules. Dans ce convoi, le véhicule de tête émet périodiquement des paquets de 1440 octets en broadcast en utilisant le protocole UDP (le débit du broadcast est 1 Mbit/s sans l'utilisation du mécanisme RTS/CTS). Quand les véhicules suiveurs reçoivent le paquet, ils le réémettent une seule fois. Les paramètres sont les suivants :

- Le nombre de véhicule dans un convoi : 4, 8, 12 et 16;
- Les distances inter-véhicules (ie. distance entre deux véhicules voisins) : 50, 100, 200 et 400 mètres;
- Le délai inter-paquet (représente la période d'émission des paquets par le véhicule de tête) : 11, 14, 20, 30 et 50 ms.

Afin d'évaluer les résultats, nous avons retenu les éléments suivants :

- Délai moyen entre deux véhicules voisins;
- Délai moyen de bout en bout : représente le délai de transmission moyen entre le véhicule de tête du convoi et le dernier;
- Délai de bout en bout pour le premier paquet reçu par le dernier véhicule du convoi;
- Taux de perte dans le convoi (ou le débit de réception);
- Nombre de sauts pour atteindre le dernier véhicule.

## 2.1 Délai de bout en bout du premier paquet

D'après la figure 1, nous remarquons que les délais du premier paquet sont très acceptables même avec un long convoi (de 16 véhicules) où le délai de bout en bout avec une distance inter-véhicule de 400 mètres est de 200 ms. On remarque aussi que pour une période inter-paquet de 11 ms, le délai de bout en bout du premier paquet est supérieur à ceux obtenus avec les autres périodes (14, 20, 30 et 50 ms). En effet, le véhicule de tête émet un paquet vers les véhicules suiveurs, et durant le temps de propagation du paquet de véhicule à véhicule, il essaiera d'émettre un autre paquet (après 11 ms). Ceci gênera la réémission de ces paquets intermédiaires par les autres véhicules (Fig. 2).

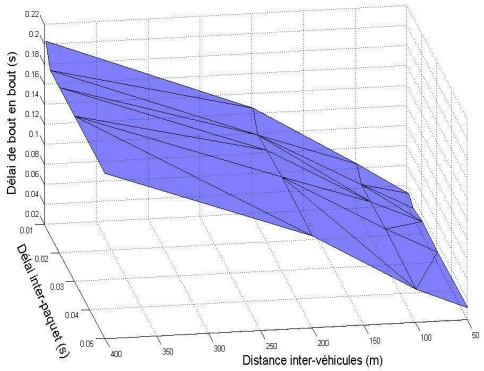



**Fig. 1:** Délai de bout en bout pour le premier paquet dans un convoi de 16 véhicules.

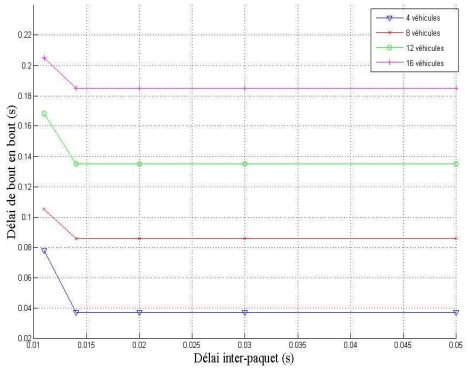



**Fig. 2:** Délai de bout en bout pour le premier paquet avec une distance de 400 m.

## 2.2 Délai moyen de bout en bout

La figure 3 montre que le délai moyen de bout en bout, de l'ordre de plusieurs secondes, est grand par rapport à celui du premier paquet surtout avec des faibles délais inter-paquet (11 et 14 ms) et des courtes distances inter-véhicules (50 et 100 mètres). Or, si nous prenons une distance inter-véhicules de 400 mètres, le véhicule de tête aura un seul voisin. Dans notre scénario, le véhicule de tête émet des paquets de 1440 octets avec une période prédéfinie (e.g. 14 ms), et à la réception du paquet, le véhicule suiveur le réémet une seule fois. Quand le véhicule de tête essaiera d'émettre un deuxième paquet, ce dernier s'arrêtera au niveau de la couche MAC, puisque le canal sera occupé par la réémission du premier paquet par les véhicules intermédiaires, provoquant ainsi un délai plus important.

Nous constatons que la valeur importante du délai moyen de bout en bout est causée par l'attente au niveau de la couche MAC occasionnée par la concurrence des autres véhicules essayant d'accéder au canal en même temps.

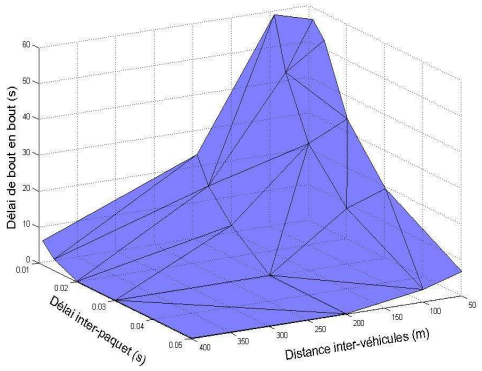



**Fig. 3:** Délai moyen de bout en bout dans un convoi de 16 véhicules.

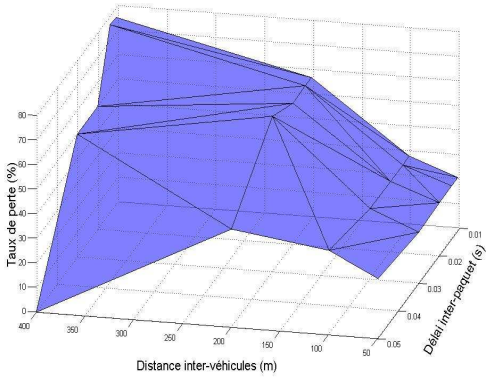



**Fig. 4:** Taux de perte moyen dans un convoi de 16 véhicules.

### *2.3 Taux de perte*

Comme illustré par la figure 4 avec un convoi de 16 véhicules, on remarque que tous les paramètres de notre simulation ont une influence sur le taux de perte. Celui-ci est proportionnel à la distance inter-véhicules, puisque plus la distance est grande plus la détection de transmission en cours s'affaiblit, ce qui peut augmenter le risque de collisions.

D'un autre coté, le taux de perte est inversement proportionnel au délai inter-paquets. Pour un délai élevé, le véhicule de tête permettra aux autres véhicules de réémettre leurs paquets avant d'en émettre un nouveau.

Le nombre de véhicules influence également le taux de perte : plus le nombre de véhicules est grand et plus le taux de perte sera grand. En effet, le nombre de véhicules affecte d'une manière directe la longueur du convoi, ce qui augmente le risque de perte de paquets.

## 3 Influence d'un convoi perturbateur

La section précédente était consacrée à l'étude des communications dans un convoi de véhicule. Dans cette section, nous complétons cette étude en analysant l'influence d'un convoi perturbateur. Nous utilisons donc deux convois dans lesquels ont lieu des communications. L'étude porte sur les communications dans le premier convoi, dit convoi principal. Le second convoi joue le rôle de convoi perturbateur.

### *3.1 Objectif et méthodologie*

L'objectif de ce scénario est d'étudier les délais de transmission et les débits de réception dans le convoi principal. Le convoi principal est paramétré comme suit : le nombre de véhicules est 12, la distance inter-véhicules est de 50 mètres, les paquets sont émis par le véhicules de tête chaque 14 ms (avec un débit de 100 KO/s) et les véhicules se déplacent avec une vitesse de 90 km/h.

Les paramètres du convoi perturbateur varient de la même façon que dans le scénario à un seul convoi de véhicules (cf section 2). Nous ferons les mêmes mesures que dans le premier scénario.

Le convoi perturbateur peut se déplacer avec trois vitesses relatives au convoi principal: 0 km/h, 90 km/h, 180 km/h.

La configuration du convoi principal représente un cas réel puisque la distance de sécurité est de 2 secondes et comme nous avons choisi une vitesse de 90 km/h, cette distance est de 50 mètres. Le choix d'un convoi de 12 véhicules nous permet d'étudier l'influence de la longueur du convoi perturbateur sur la qualité de la communication (sachant que le convoi perturbateur peut avoir 4, 8, 12 et 16 véhicules).

### *3.2 Résultats du convoi principal avec perturbation*

D'après la figure 5 et la figure 6, on constate que la longueur du convoi perturbateur a une grande influence sur la qualité de la communication dans le convoi principal et cela selon la vitesse relative. Si la longueur du

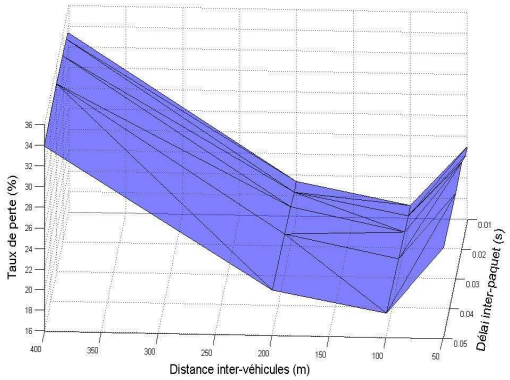



**Fig. 5:** Taux de perte dans le convoi principal avec une vitesse realtive de 0 km/h.

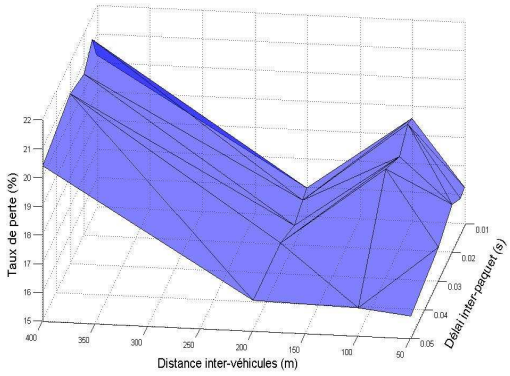



**Fig. 6:** Taux de perte dans le convoi principal avec une vitesse relative de 20 km/h.

convoi perturbateur est égale ou peu supérieure au convoi principal, on aura un meilleur taux de perte mais un délai moyen de bout en bout important. En effet, dans cette configuration, nous assurons une meilleure retransmission des paquets, donc le taux de perte moyen diminue mais le délai moyen de bout en bout augmente.

# 4 Conclusion

Suite aux résultats obtenus avec les simulations sous ns-2, nous constatons que la norme IEEE 802.11 peut être utilisée pour la communication inter-véhicules avec un faible débit. Nous estimons qu'avec l'ajout d'heuristiques appropriées pour la retransmission des paquets dans un convoi de véhicules, nous obtiendrons de meilleurs résulats. Avec de telles améliorations, des débits plus importants pourront être atteints, tout en assurant de faibles délais de transmission.

Actuellement, nous travaillons sur la conception d'algorithmes multi-sauts adaptés à la communication inter-véhicules.

Finalement, suite aux limitations de la couche MAC dans un environnement de forte mobilité, nous planifions d'étudier l'impacte de la norme 802.11g, qui propose un débit plus important que la 802.11b.

# References

[Gas02] M. S. Gast. *"802.11 Wireless Networks: The Definitive Guide"*. O'Reilly Associates, Avril 2002.

[LA04] H. Labiod and H. Afifi. *"De Bluetooth à Wi-fi, Sécurité, qualité de service et aspects pratiques"*. Hermès Science, Février 2004.

[LH] J. Luo and J-P. Hubaux. "A Survey of Inter-Vehicle Communication". *EPFL Technical Report IC/2004/24*.

[ns] Network simulator 2: http://www.isi.edu/nsnam/ns/.

[Tsu02] S. Tsugawa. "Inter-vehicle communications and their applications to intelligent vehicles: an overview". *In proceeding of IV'2002, Versailles, France*, 2002.

# *Efficient Gathering in Radio Grids with Interference*[†]

Jean-Claude Bermond and Joseph Peters

*Projet MASCOTTE, I3S-CNRS/INRIA/Université de Nice-Sophia Antipolis, 2004 Route des Lucioles, BP 93, F-06902 Sophia Antipolis Cedex, France*
*School of Computing Science, Simon Fraser University, Burnaby, British Columbia, V5A 1S6, Canada*

We study the problem of gathering information from the nodes of a radio network into a central destination node. A transmission can be received by a node if it is sent from a distance of at most $d_T$ and there is no interference from other transmissions. One transmission interferes with the reception of another transmission if the sender of the first transmission is at distance $d_I$ or less from the receiver of the second transmission. In this paper we study the case $d_T = 1$ and $d_I > 1$ for two-dimensional grid networks with unit time transmissions. We prove lower bounds on the number of rounds required for any two-dimensional grid and we describe protocols for $n \times n$ grids with $n$ odd that are optimal for odd $d_I$ and near-optimal for even $d_I$.

**Keywords:** radio communication, interference, grids, gathering

## 1 Introduction

In this paper, we study a problem suggested by France Telecom concerning the design of efficient strategies to provide Internet access using wireless devices. Typically, several houses in a village wish to access a gateway (a satellite antenna) to transmit and recieve data on the Internet. To reduce the cost of the transceivers, multi-hop wireless relay routing is used. Information can be transmitted from a node to any node within distance $d_T$. However, a transmission can *interfere* with reception at nodes at distances up to $d_I > d_T$ from the transmitter. If two calls are mutually non-interfering, we say that they are *compatible*. The goal is to enable access to the gateway by the users with the constraint that the number of different transmissions that can be done is limited by the number of channels and time slots available. This depends on the technology used (see [Bia00, Gal04, Müh02], for example). We will use the term *round* to mean a time slot on a given channel during which we can have only compatible transmissions or *calls*. We are interested in schedules that minimize the number of rounds.

We consider the case in which the nodes are at the points of the 2-dimensional Euclidean plane. This is the situation when the streets in the village form a grid pattern. We assume that $d_T = 1$, so a node at position $(x,y)$ on the plane can transmit to nodes $(x,y\pm 1)$ and $(x\pm 1,y)$. This choice of $d_T$ minimizes the cost of the transceivers. We study the problem of *gathering* one piece of information from each node into a central gateway node $v_0$ for transmission over the Internet. The inverse problem of gathering, in which each node receives a personalized message from the central node, is called *distribution* or *scattering* and can be solved by reversing the order and directions of the transmissions in a gathering protocol. We assume that all pieces of information are of the same size, such as the size of a packet, so all transmissions take one time unit. The gathering problem then becomes one of organizing the transmissions into rounds of compatible calls so that the number of rounds is minimized.

We represent the wireless network as a graph $G = (V,E)$ in which the vertices represent wireless network nodes and there is an edge between each pair of vertices that are at distance $d_T$ or less from each other.

[†]Research supported by the CRC CORSO with France Telecom, the Equipe Associée RESEAUXCOM of INRIA, and the European Project FET CRESCCO.

When $d_T = 1$, the graph is a 2-dimensional grid. We assume that $G$ is a square grid with $N = n^2$ vertices. We will concentrate on the case when $n = 2p+1$ is odd and the vertices are arranged symmetrically around the central vertex $v_0 = (0,0)$ with $p$ columns of vertices on either side of the vertical axis through $v_0$ and $p$ rows above and below the horizontal axis through $v_0$. If $n$ is even, $v_0$ will be slightly off-centre. Minor modifications of the techniques described in this paper will work for $n$ even.

In a radio network, a transmission can be received by all nodes at distance $d_T$ or less, but in the gathering problem, only one node will forward the message towards the gateway, so we can assume that each call involves a pair $(s,r)$ where $s$ is the sender and $r$ the receiver of the message. When the distance $d(s_i,r_j)$ between the sender of call $(s_i,r_i)$ and the receiver of call $(s_j,r_j)$ is such that $d_T < d(s_i,r_j) \leq d_I$, then the call $(s_i,r_i)$ is too weak to be received by $r_j$, but it is strong enough to interfere with the reception of call $(s_j,r_j)$ by $r_j$. In this paper, we use rectilinear distance (i.e., distance on the grid) as an approximation to Euclidean distance, mainly because it simplifies the analysis. Rectilinear distance is a good approximation to Euclidean distance when the ratio between $d_I$ and $d_T$ is small, and this is usually the case in practice.

Several examples are shown in Figure 1. In Figure 1(a), the calls $(s_1,r_1)$ and $(s_3,r_3)$ are compatible when $d_I = 3$ and so are the calls $(s_3,r_3)$ and $(s_4,r_4)$. All other pairs of calls are incompatible. For example, the call $(s_1,r_1)$ does not interfere with reception at $r_2$, but $(s_2,r_2)$ interferes with reception at $r_1$, so these calls are incompatible. Figure 1(b) shows the calls around $v_0$, which is represented by a large circle, when $d_I = 3$. None of the calls in the shaded *interference zone* are compatible with each other, so at most one of these calls can be done at any given time. The situation is slightly different when $d_I$ is even. In Figure 1(c), all of the calls in the shaded interference zone interfere with each other as in the odd case, but the ways in which information can enter vertices on the boundary of the interference zone are more restricted. The only subset of four compatible calls from vertices on the dashed square to vertices on the boundary of the interference zone is shown with solid arrows. All other such calls can only be done two or three at a time.

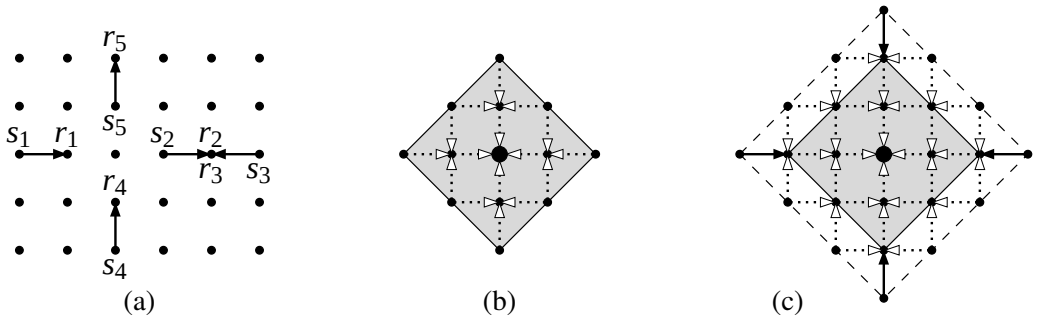



**Fig. 1:** Interference for $d_T = 1$ and (a) $d_I = 3$, (b) $d_I = 3$, (c) $d_I = 4$.

The one-dimensional version of the problem studied in this paper, that is, gathering into a designated node of a path, is solved in [BCY05]. In [KMP04], the case where there is a continuing demand is considered and systolic algorithms are given. In [BKMP05], general lower bounds and protocols are given for $d_T \geq 1$ for various networks such as trees and stars. For the gathering problem, the case $d_T = d_I = 1$ is the *half-duplex* 1-*port* model which has been widely studied in the literature (e.g., [BGRV98]). The broadcasting and gossiping problems in radio networks with $d_T = d_I = 1$ are studied in [GM03] and [GP02] respectively.

## 2 Lower Bounds

We say that the calls entering $v_0$ are *level* 1 calls, the calls entering neighbours of $v_0$ are level 2 calls, and so on. A call $(s,r)$ is a level $\ell$ call if $d(s,v_0) = \ell$ and $d(r,v_0) = \ell - 1$. Note that if $\ell + \ell' \leq d_I + 1$ then a level $\ell$ call interferes with a level $\ell'$ call. In particular, when $d_I = 2k-1$ is odd, all calls at level $k$ or less interfere with each other (see Figure 1(b)). When $d_I = 2k$ is even, all calls at level $k$ or less interfere with each other and the subsets of compatible level $k+1$ calls are restricted. In particular, there is only one subset of four compatible calls at level $k+1$ (see Figure 1(c)), and this subset is incompatible with all calls in the interference zone.

Note that information at a vertex at distance $d$ from $v_0$ will use at least one level $\ell$ call for each $1 \leq \ell \leq d$ to reach $v_0$. We will use $N_d$ to denote the number of vertices of the square grid (with $N = n^2$ vertices, $n = 2p+1$) that are at distance exactly $d$ from $v_0$. We have that $N_0 = 1$, $N_d = 4d$ for $d \leq p$, and $N_d = 4(2p+1-d)$ for $d > p$.

**Theorem 1** *The number of rounds to gather in the $n \times n$ grid is at least $\sum_{i=1}^{k} iN_i + k(N - \sum_{i=0}^{k} N_i)$ if $d_I = 2k-1$ is odd, and $\sum_{i=1}^{k} iN_i + (k+\frac{1}{4})(N - \sum_{i=0}^{k} N_i)$ if $d_I = 2k$ is even.*

**Proof Outline**:
*Case $d_I = 2k-1$ is odd:* Information from vertices at distance $d > k$ from $v_0$ must use $k$ calls inside the interference zone, all of them pairwise interfering, to reach $v_0$. Information from vertices at distance $d \leq k$ from $v_0$ must use $d$ calls inside the interference zone. So, the total number of rounds is at least $\sum_{i=1}^{k} iN_i + k(N - \sum_{i=0}^{k} N_i) = k(N-1) - \sum_{i=1}^{k}(k-i)N_i$. For example, for $d_I = 3$ (and $p \geq 2$), we need $2N-6$ rounds.

*Case $d_I = 2k$ is even:* The lower bound on the number of calls inside the interference zone is the same as for the odd case: $\sum_{i=1}^{k} iN_i + k(N - \sum_{i=0}^{k} N_i)$. The difference from the odd case is that the level $k+1$ calls are also restricted, so that the information from vertices at levels $k+1$ and greater requires $\frac{1}{4}(N - \sum_{i=0}^{k} N_i)$ additional rounds for a total of $\sum_{i=1}^{k} iN_i + (k+\frac{1}{4})(N - \sum_{i=0}^{k} N_i)$ rounds. □

# 3 Gathering Protocol

In this section, we describe protocols that achieve or come close to the lower bounds of Theorem 1. We say that a vertex is *active* if it has information that needs to be sent or forwarded to $v_0$. A vertex that is not active and is not on any path that will be used to forward information in future rounds is *dormant*.

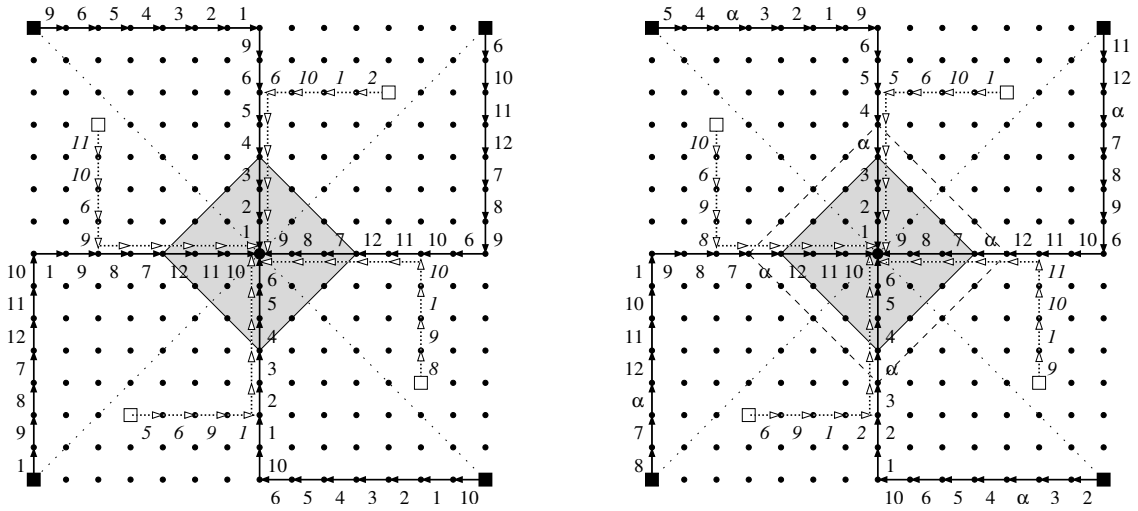
**Theorem 2** *Suppose that $n = 2p+1$ and $d_I = 2k-1$ are odd. Then gathering in the $n \times n$ grid can be completed in $\sum_{i=1}^{k} iN_i + k(N - \sum_{i=0}^{k} N_i)$ rounds and this is optimal.*

**Proof Outline**: The general idea is to organize the calls into stages of $4k$ rounds. In each stage, four active vertices that are symmetrically arranged around $v_0$ are chosen. Information is forwarded along paths from each of these vertices to $v_0$ for $4k$ rounds. At the end of the stage, the four vertices become dormant, all other vertices on the four paths have sent one piece of information and recieved another, and $v_0$ has received four more pieces of information. The paths are chosen in such a way that the calls in each round are compatible. We iterate this procedure until the only remaining active vertices are inside the interference zone around $v_0$. It takes $\sum_{i=1}^{k} iN_i$ sequential calls inside the interference zone to move the remaining information into $v_0$. □

Figures 2(a) shows two examples of stages, one with solid arrows and the other with dotted arrows. The numbers indicate the rounds during which the calls are made. Stages are executed sequentially, so that at any given time, only one set of paths is being used. It is not hard to verify that the calls on the solid paths are compatible in each round. Similarly, the calls of the dotted paths are compatible. In general, paths in the upper right quadrant go towards the closest axis and then along the axis to $v_0$. The paths in the three other quadrants are obtained by rotation.

**Theorem 3** *Suppose that $n = 2p+1$ and $d_I = 2k$ is even. Then gathering in the $n \times n$ grid can be completed in $\sum_{i=1}^{k} iN_i + (k+\frac{1}{4})(N - \sum_{i=0}^{k} N_i) + k - 1$ rounds.*

**Proof Outline**: If $d_I$ is even, the paths are similar to the odd case except for the level $k+1$ calls which enter vertices on the boundary of the shaded interference zone from vertices on the dashed square (see Figure 1(c)). These calls require an extra round in each stage. The most efficient way to do the level $k+1$ calls is to use the four calls labelled $\alpha$ in the example in Figure 2(b). This works for all paths starting at level $k+2$ or greater. Most of the paths that start at the $4k+4$ vertices on the dashed box can only be done two at a time without interference. Careful use of the calls labelled $\alpha$ allows us to do twelve of these calls three at a time for a total of $k(4k+4)+2k$ rounds to move the information of these $4k+4$ vertices to $v_0$. □



(a) (b)

**Fig. 2:** Gathering stages for $d_T = 1$ and (a) $d_I = 5$, (b) $d_I = 6$.

## Addendum

We have recently improved the lower bound for the case $d_I$ even to match the upper bound of Theorem 3. The proof of this lower bound and also the complete proofs of Theorems 2 and 3 are considerably more complicated than the proof outlines presented here. All details will appear in the full version of this paper.

## Acknowledgements

We thank Jérôme Galtier of France Telecom for suggesting this problem to us, and Jérôme Galtier, Ralf Klasing, Nelson Morales, and Joseph Yu for helpful discussions.

## References

[BCY05] J.-C. Bermond, R. Corrêa, and J. Yu. Optimal gathering schemes on paths under interference constraints. Manuscript, March 2005.

[BGRV98] J.-C. Bermond, L. Gargano, A. Rescigno, and U. Vaccaro. Fast gossiping by short messages. *SIAM J. Computing*, 27:4:917–941, 1998.

[Bia00] G. Bianchi. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE J. Selected Areas of Communication*, 18:535–547, 2000.

[BKMP05] J.-C. Bermond, R. Klasing, N. Morales, and S. Pérennes. Gathering unit messages in known radio networks. Manuscript, January 2005.

[Gal04] J. Galtier. Optimizing the IEEE 802.11b performance using slow congestion window decrease. In *Proc. 16th ITC Specialist Seminar on Performance Evaluation of Wireless and Mobile Systems*, pages 165–176, 2004.

[GM03] I. Gaber and Y. Mansour. Centralized broadcast in multihop radio networks. *J. Algorithms*, 46:1:1–20, 2003.

[GP02] L. Gasieniec and I. Potapov. Gossiping with unit messages in known radio networks. In *Proc. IFIP 17th World Computer Congress - TC1 Stream / 2nd IFIP International Conference on Theoretical Computer Science*, pages 193–205, 2002.

[KMP04] R. Klasing, N. Morales, and S. Pérennes. On the complexity of bandwidth allocation in radio networks with steady traffic demands. Technical Report 5432, INRIA, December 2004.

[Müh02] P. Mühlethaler. *802.11 et les réseaux sans fil*. Eyrolles, 2002.

# *Internet : métrologie passive de l'Internet*

Jeudi 13 mai 2005, 11:00 - 12:30

---

# *Génération de Topologies Réalistes pour la Simulation du Routage Interdomaine*

J.M. Fourneau et H. Yahiaoui

*Laboratoire PRiSM, UVSQ. 45, av. des États-Unis. 78035, Versailles - FRANCE*

La simulation du routage interdomaine nécessite l'utilisation de topologies se rapprochant le plus possible de la taille et de la conformation du réseau réel. L'utilisation de topologies à granularité plus fine que le graphe de domaines de routage, permet d'observer d'avantage d'instabilités. Nous avons élaboré un algorithme de génération de topologies de sessions BGP pour la simulation réaliste de BGP et l'avons implémenté dans un outil général de manipulation de topologies interdomaine. Cet algorithme infère les topologies de sessions BGP en utilisant des topologies, édifiées à partir de données bruts de l'Internet, et d'informations de connectivité des domaines de routages, au niveau routeur.

**Mots Clé:** Routage Interdomaine, Simulation de BGP, Topologie Internet

## 1 Introduction

Durant plus de dix ans, BGP (Border Gateway Protocol) a été le protocole *officiel* de routage interdomaine. Cependant, des études ont montré le caractère instable du routage par BGP. Pour tenter de comprendre et de corriger ces instabilités, plusieurs simulateurs ont déja été conçus [5][6][7]. Nous avons imaginé un simulateur à large échelle du protocole, qui a pour ambition de reproduire les instabilités de façon réaliste. Une composante importante d'un tel simulateur est la topologie utilisée. En effet, reproduire les instabilités observées sur Internet exige l'utilisation de topologies les plus réalistes possible. La majorité des simulateurs existant fonctionnent avec la représentation classique de la topologie interdomaine : un graphe d'ASs (Autonomous Systems) interconnectés par des liens BGP. Chaque AS est abstrait et simplifié pour constituer l'entité communiquante de base. Des études récentes [4] ont montré qu'une partie importante de l'instabilité constatée de BGP était dûe, en partie, à la complexité interne des ASs et à la multiplicité des liens existant entre certains couples d'ASs. Il apparait, donc, que la simulation réaliste de BGP à grande échelle nécessite des topologies concervant cette complexité.

C'est pour répondre à ce besoin qu'ont été imaginées les topologies de sessions BGP. Basés sur une représentation plus fine du réseau interdomaine, ces topologies permetteront de générer les comportements d'instabilité dûs à certaines politiques de routage alliées à la complexité interne des ASs. Cependant, la pauvereté des informations à propos de la structure interne des ASs, qui transpire à l'exterieur ne permet pas de construire une topologie fiable de l'Internet. Il a, donc, fallu mettre au point une méthode de génération, permettant, à partir de la topologie des ASs et d'informations complémentaires, de créer des topologies de sessions BGP qui, si elle ne correspondent pas exactement à la réalité, possèdent au moins un caractère de réalisme suffisant pour produire les instabilités de BGP qui nous intéressent.

Le présent article se poursuit en décrivant dans la section 2 les topologies d'AS, les moyens de les obtenir, ainsi que leurs limitations pour la simulation à large échelle. Puis, la section 3 introduit la notion de topologie de sessions BGP. Cette partie expose un algorithme de génération de topologies de sessions BGP à partir d'information de connectivité aux niveau AS et de connaissances complémentaires. Enfin, la section 4 expose l'implémentation de cet algorithme et son paramétrage.

## 2 Topologies de Systèmes Autonomes

BGP permet de hiérarchiser le routage sur Internet en considérant la connectivité d'entités complexes : les systèmes autonomes. Chaque système autonome est un domaine de routage indépendant, administré indépendamment des autres AS, en utilisant des protocoles et des métriques de routage, uniquement définis à l'intérieur du domaine. Il apparaît alors, que du point de vue de BGP, l'Internet en entiers se comporte comme un graphe, dont les nœuds sont les ASs et dont les arcs sont schématisés par les sessions BGP établies entre ASs. C'est, justement, cette vision de l'Internet qui est majoritairement utilisé pour la simulation du protocole interdomaine. En effet, l'adoption des topologies de simulation basées sur le graphe de connectivité des ASs de l'Internet permet de répondre aux impératifs d'abstraction exigés dans la simulation. De plus, l'extraction d'informations de topologies à partir de données disponibles sur Internet permet de produire assez facilement des topologies d'AS. Alors qu'il est pratiquement impossible d'obtenir une vision plus détaillée de l'intérieur des AS (une vision au niveau *speaker* BGP). L'obtention de telles topologies utilise, dans la majorité des simulateurs ou des projets de cartographie de l'Internet, la même méthode. Celle-ci se base sur la présence, sur Internet, de certains speakers spéciaux. Ces speakers, souvent dévolues à l'observation du routage interdomaine, procèdent régulièrement à une sauvegarde de leurs tables de routage. Ces *logs* peuvent, alors être utilisés pour extrapoler une topologie de l'Internet actuel. Par exemple, d'un fragment de *log* comme celui-ci (extrait des logs BGP du projet RouteViews [2]) :

```
   Network  Next Hop        Metric LocPrf Weight Path
*> 4.0.0.0  134.24.127.3                       0 1740 1 i
*           194.68.130.254                     0 5459 5413 1 i
```

On pourra déduire la présence d'au moins un lien BGP (une sessions BGP établie entre un speaker de chaque AS) entre chacun des couples (5459, 5413), (5413, 1), . . .L'analyse et la synthèse de telles informations, récupérées à partir de différents point de l'Internet à différents instants, produira une carte presque complète des liens entre ASs.

Si ce type de topologie a été largement exploité, il nous est apparu comme incomplet pour le type de simulations que l'on veux réaliser. En effet, l'étude du volume et des types de messages BGP échangés sur Internet a montré que certaines instabilités du routage interdomaine étaient dûes à des causes internes aux ASs [4]. Par exemple, certaines politiques de routage, dites de *mapping MED-IGP*, attachent aux annonces de routes des attributs MED (Multi_Exit_Descriminator) dont les valeurs correspondent aux métriques IGP interne aux ASs. Ces politiques peuvent,donc, produire des instabilités sur le routage interdomaine à partir de variations internes aux ASs. En simulant des topologies aux ASs simplifiés, il est impossible de reproduire de tels comportements. Ce qui diminue le réalisme des instabilités que l'on voudrait obtenir. De plus, les topologies d'AS simulées possèdent une limitation dûe à leur structure interne : un AS de ces topologies ne peut être relié que par un seul lien vers un autre AS. Cette disposition ne reflète pas du tout la réalité où les ASs, surtout de grandes tailles, nécessitent souvent l'établissement de plusieurs sessions BGP entre eux pour ne pas provoquer de surcharge de traffic sur certains liens. En conséquence de quoi, il peut arriver que différents speakers d'un même AS choisissent des routes différentes pour les mêmes destinations. Pour produire des topologies d'ASs à partir de tels réseaux, la solution classique consiste à séparer les ASs arborant de tels comportements en **ASs logiques**. Cette méthode résoud une partie uniquement des problèmes de simulation liés à la compléxité intra-AS. Les difficultés évoquées plus haut n'y trouve aucune solution.

Pour pallier à ces manques, nous avons décidé d'utiliser des **Topologie de Sessions BGP**. Dans ces topologies, le nœud élémentaire n'est plus l'AS, mais le **speaker** ; un AS pouvant être composé de plusieurs speakers interconnectés. Bien que moins abstraites que les topologies d'ASs, donc plus difficiles à simuler pour de grandes tailles de topologies, ces topologies nous permetterons de reproduire au plus près la hiérarchie de connectivité du routage interdomaine. Ainsi, la présence de sessions multiples entre ASs devient évidente à produire. Et l'impact des différents timers et délais est mis en valeur par la présence de plusieurs speakers, communiquant à des rythmes différents

dans un même AS. Il devient aussi possible d'observer les conséquences de certaines politiques sur le routage interdomaine, ainsi, que les répercussions de la transmission d'instabilités **internes aux ASs** vers l'exterieur de ces ASs.

# 3 Topologies de Sessions BGP inter-Speakers

Les topologies de simulation que nous avons proposés sont, donc, des **topologies de sessions BGP**. Moins abstraites que les topologies d'ASs, ces topologies prennent en compte la complexité des phénomènes au sein des ASs et permettent de prendre en compte les cas de multiplicité des connexions entre ASs, qui est une configuration plus que fréquent sur l'Internet d'aujourd'hui.

Les nœuds de ces topologies sont les speakers BGP des ASs. L'adoption d'une granularité plus fine que l'AS a pour but d'observer des phénomènes d'instabilité dont la source est, par exemple, l'intérieur des AS, mais dont les effets se voient sur la structure global du routage interdomaine. Les arcs reliant les nœuds sont les sessions BGP établies entre les speakers des différents ASs. Etant donné la multiplicité des speakers appartenant à un AS, il devient possible d'établir plusieurs sessions entre chaque paire d'AS. Il suffira pour cela de sélectionner des paires de speakers différents, tel que chaque speaker appartient à l'un des AS, et d'établir une sessions EBGP entre eux. Les liaisons entre les speakers d'un même AS deviennent indispensables. En effet, l'unicité des speakers au sein des ASs des topologies d'AS permettait d'ignorer la transmission des routes interdomaines entre speakers d'un même AS. En adoptant un topologie plus complexe, nous devons permettre aux speakers d'un même AS de se communiquer leurs routes respectives, afin de conserver la cohérence du routage. La connectivité que nous avons adopté pour cela est de deux type : les ASs relativemt "petits" (petit nombre de speakers) disposeront d'un graphe en clique de sessions IBGP (BGP adapté pour la transmission intra-AS). Les ASs plus "grands" seront élaborés en disposants des réflecteurs de routes pour diminuer le nombre de sessions IBGP attaché à chaque speaker de l'AS.

Les topologies de Sessions BGP répondent aux exigences de la simulation à grande échelle du routage interdomaine. Cependant, ce type de topologies ne peux être directement produit comme le sont les topologies d'AS. En effet, la structure interne des ASs (nombre de speakers, leur disposition et les sessions qu'ils établissent) est rarement connue de l'exterieur de l'AS. Il est, donc, indispensable de disposer d'une méthode d'**inférence** de topologies de sessions BGP à partir des topologies d'ASs. L'algorithme d'inférence que nous proposons utilise une topologie d'ASs **pondérée** pour générer la topologie de sessions BGP. La pondération est une opération qui consiste à assigner des poids aux liens entre ASs à partir d'informations de niveau routeur. La méthode la plus simple consiste à pondérer le lien par le nombre de lien routeur-routeur reliant chaque paire d'ASs. Puis, l'algorithme opère par :

1. La réduction des poids des liens en les divisant par un facteur commun : Pour tenter d'estimer le nombre de sessions BGP existantes entre chaque paire d'AS, chaque poids de le topologie est divisé par un facteur $factor$ :

$$factor = \frac{\sum_i PoidsLien_i}{NbrLiens \div (1 - PourcentageLiensManquants)}$$

   où , $NbrLiens$ est le nombre de liens dans le graph des ASs et $PourcentageLiensManquant$ représente une estimation de la fraction de liens BGP qui n'apparaissent pas dans le graphe des ASs.
2. La création, dans chaque AS, d'un nombre de speakers égale au poids maximum des liens cet AS. À chaque lien de l'AS est associé un compteur initialisé avec le poids du lien.
3. La sélection dans l'AS d'un speaker n'ayant aucune session. Pour chaque compteur de lien non nul, on associe au speaker une session avant de décrementer ce compteur. Chaque session sera aussi associée à l'un des speakers de l'AS voisin ne possédant pas de sessions avec un speakers de l'AS courant. Cette opération est répetée tant que subsiste des compteurs non nuls.
4. Lorsque toutes les sessions sont construites, créer les sessions IBGP au sein des ASs.

## 4 Implémentation

L'algorithme d'inférence de topologies de sessions BGP décrit ci-dessus a été implementé dans un outil de manipulation de topologies, associé au projet de dévellopment du simulateur des instabilités de BGP. L'algorithme exige en entrée une topologie d'AS (figure 1-a), des informations de niveau routeur (nombre de liens routeur-routeur existant entre chaque couple d'ASs ayant établie au moins une sessions BGP entre eux) et une valeur numérique représentant la proportion de liens supplémentaires désirée. La sortie de cet algorithme est une topologie de Sessions BGP disposant d'un nombre de sessions correspondant à la proportion indiquée (figure 1-b pour une proportion=30%, figure 1-c pour une proportion=50%).

Malheureusement, nous ne disposons pas encore d'information fiable de niveau routeur pour la totalité d'Internet, ni de méthode pouvant facilement fournir ces détails (l'exploration par *traceroute* étant trop contraignante pour le réseau global). L'algorithme est, donc pour l'instant, approvisionné en information de niveau routeur grâce à un générateur aléatoire. Ce générateur attribue des poids aux arêtes de la topologie d'AS avant qu'elle ne soit introduite dans le générateur.

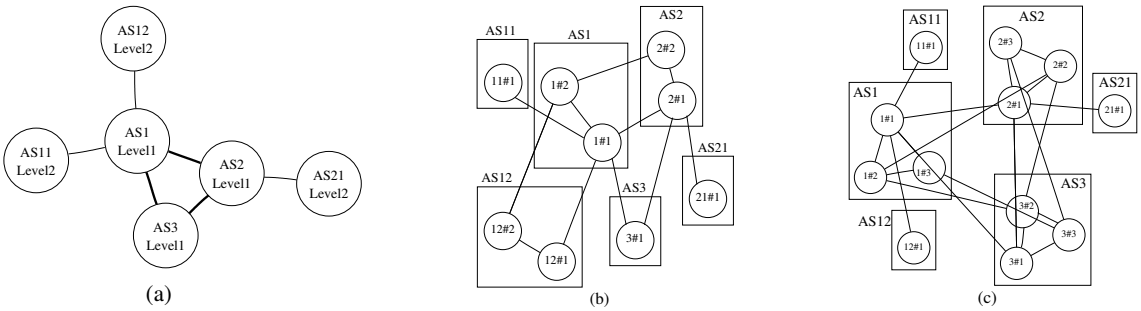



**Fig. 1:** Génération de topologies de Sessions BGP

Les topologies ainsi produites, peuvent différer des topologies réelles de sessions BGP (les topologies réelles complètes sont difficilement obtenus par cartographie). Cependant, elles possèdent le degré de réalisme nécessaire à la simulation des instabilités de BGP à large échelle.

## Références

[1] Y. Rekhter et T. Li : *A Border Gateway Protocol (BGP-4)*. RFC 1771, Mars 1995.

[2] D. Meyer : *Route Views Project*. http ://antc.uoregon.edu/route-views/ University of Oregon.

[3] W. Li : *Inter-Domain Routing : Problems and Solutions*. http ://citeseer.ist.psu.edu/li03interdomain.html, 2003.

[4] C. Labovitz, R. Malan et F. Jahanian : *Origins of Internet Routing Instability*. Proceedings of IEEE INFOCOMM '99 Conference, March 1999.

[5] B. Quoitin : *C-BGP User's Guide*. CSE Departement, UCL, Belgique, Avril 2004.

[6] Hao et Koppol : *An Internet Scale Simulation Setup for BGP*. ACM SIGCOMM, Volume 33, Numéro 3, Juillet 2003.

[7] B.J. Premore : *An Analysis of Convergence Properties of BGP using Discrete Events Simulation*. Dartmouth College, Hznover, New Hamshire, Mai 2003.

[8] S. Agarwal : *Characterizing the Internet Hierarchy from Multiple Vantage Points*. http ://www.cs.berkeley.edu/ sagarwal/research/BGP-hierarchy/.

[9] B. Cheswick et H. Burch : *The Internet Mapping Project*. http ://research.lumeta.com/ches/map/.

[10] H. Chang and S. Jamin and W. Willinger : *Inferring AS-level Internet topology from router-level path traces*. Proceedings of SPIE, Vol 4526, Juillet 2001.

# *Caractérisation de l'autosimilarité de trafics WEB et FTP par un outil de simulation*

H.HASSAN (1) JM. GARCIA (1) O. BRUN (1) D. GAUCHARD (1)

*(1) LAAS-CNRS, 7 avenue du Colonel Roche, 31077 Toulouse, France.*

**Résumé**

L'autosimilarité est une caractéristique importante du trafic Internet. Les études effectuées sur les effets de l'autosimilarité du trafic sur les réseaux ont montré une dégradation importante de la performance des réseaux par rapport aux trafics à courte dépendance (i.e Poisson). Cet article est consacré à l'étude de l'influence des différents paramètres du trafic HTTP/FTP sur la génération d'un trafic autosimilaire. Un éditeur générique des Sources de Trafic a été développé afin de reproduire le comportement de populations d'utilisateurs et d'un ensemble d'applications : sources aléatoires quelconques, sources audio et vidéo avec la plupart des codecs, et des sources HTTP et FTP . Les simulations sont effectuées avec un outil de simulation hybride DHS (Distributed Hybrid Simulation) développé au LAAS-CNRS.

**Keywords:** Sources de trafic,Autosimilarité, HTTP, FTP, simulation événementielle.

# 1 Introduction

Les différentes études effectuées sur des réseaux LAN ou WAN, ont mis en évidence le caractère autosimilaire et LRD (long range dependency) de leur trafic. Les traces étudiées représentent le nombre de paquets ou de bits entrant dans le réseau en fonction du temps. La propriété d'autosimilarité du trafic signifie qu'une trace générée a le même comportement statistique , quel que soit l'échelle de temps choisie. L'autosimilarité (et la longue dépendance) est un caractère important du trafic. Les études sur les effets de l'autosimilarité sur les réseaux ont montré [2] que cette propriété cause la dégradation de la performance des réseaux par rapport aux trafics à courte dépendance (i.e Poisson).

Dans cette étude nous rappelons la définition de l'autosimilarité et sa mesure par le paramètre de Hurst, ainsi que la méthode R/S plot pour l'évaluation de ce dernier. Nous présentons brièvement l'Editeur des Sources de Trafic et les modèles étudiés. Nous décrivons ensuite la génération des traces Web, basées sur des modèles conçus avec l'Editeur de Source de Trafic développé à cet effet, et simulés avec l'outil de simulation DHS (Distributed Hybrid Simulation). Dans un premier temps, nous étudions les paramètres du modèle HTTP : distribution des tailles des fichiers et périodes de lectures et leur influence sur l'autosimilarité. Ensuite nous nous intéressons à l'influence des paramètres réseaux : bande passante et débit des sources. Enfin nous évaluons l'impact des gros transferts de fichiers via FTP sur l'autosimilarité du trafic résultant.

# 2 La propriété d'autosimilarité

**Définition** : Soit ( $Xn, n \in \mathrm{N}$ ) un processus stationnaire. On note $(X_k^m)_{k\in N}$ son processus agrégé d'ordre m défini comme suit :

$$X_k^{(m)} = \frac{1}{m}(X_{(k-1)m+1} + ... + X_{km}) \tag{1}$$

Le processus $(X_n)_{n\geq 0}$ est dit autosimilaire de paramètre H si $\forall m$ entier :

$$(X_k^{(m)} = m^{1-H}(X_{(k-1)m+1} + ... + X_{km}))_{k\in N} \overset{\ell}{=} (X_n)_{n\in N} \tag{2}$$

Cette égalité est au sens de la distribution. Le paramètre de Hurst H associé à un processus autosimilaire, représente intuitivement le « Zoom ». La corrélation d'un processus autosimilaire s'exprime aussi par rapport au paramètre de Hurst. En effet, la fonction d'autocorrélation associée à un processus autosimilaire $(X_n)_{n\geq 0}$ de paramètre H est de la forme :

$$\rho_{X_n}(k) \approx_{k\to\infty} H(2H-1)k^{2H-2} \tag{3}$$

On en déduit qu'un processus $(X_n)_{n\geq 0}$ est à dépendance longue si 0.5<H<1. Plus le paramètre de Hurst est grand, plus le processus est autosimilaire. Plusieurs méthodes statistiques existent pour estimer le paramètre de Hurst. Hurst [4] a proposé

une méthode appelée analyse des étendues normalisées, ou *Rescaled Range Analysis* (*R/S Analysis*), pour évaluer la valeur du paramètre H.
Soit une série temporelle de *N* nombres, notés *x(t).* La méthode préconisée par Hurst consiste à prendre en considération une série de subdivisions indépendantes, d'effectif τ. Pratiquement, on part de séries d'effectif 10, puis 11, 12, 13, jusqu'à l'effectif le plus élevé permettant de distinguer deux subdivisions (*N*/2 ou (*N*/2)-1). Pour chaque subdivision considérée, la moyenne des τ données est :

$$E(x)_\tau = \frac{1}{\tau}\sum_{t=1}^{\tau} x(t) \tag{4}$$

Pour chaque subdivision, on centralise les données, en leur soustrayant la moyenne locale. Puis on établit une série de valeurs cumulées à l'intérieur de chaque période: on ajoute à chaque donnée la somme des valeurs centrées qui la précèdent:

$$X(t,\tau) = \sum_{u=1}^{t}\{x(u) - E(x)_\tau\} \tag{5}$$

Donc pour chaque valeur de t ($1 \le t \le \tau$), on a une valeur de X(t, τ). X(t, τ) est une distribution intégrée. On peut noter que si le signal original est un bruit blanc, X(t, τ) est un mouvement brownien. L'étendue *R* (*range*) est la différence entre le minimum et le maximum de *X(t, τ) :*

$$R = \max_{1\le t\le\tau} X(t,\tau) - \min_{1\le t\le\tau} X(t,\tau) \tag{6}$$

Cette étendue est ensuite normalisée au moyen d'une division par l'écart-type local (S(t,τ)). Enfin on procède à la pondération moyenne, par niveau d'effectif, des étendues normalisées R/S. Hurst montre que l'étendue normalisée R/S, où S représente l'écart type de la série des x(t) est liée à la grandeur de l'intervalle considéré par la relation suivante:

$$R/S = (a\tau)^H \tag{7}$$

Où « a » est une constante et H est l'exposant de Hurst. H peut être estimé par la méthode des moindres carrés, ou par l'estimation de la pente de la régression log/log de R/S sur τ.

# 3 Editeur des Sources de Trafic et DHS

La modélisation fiable des flux Internet est confrontée à la diversité des applications déployées sur Internet. Dans le but de modéliser ces nouvelles applications nous avons développé une bibliothèque de sources de trafic qui permet une représentation hiérarchique et générique des sources de trafic. L'Editeur des Sources de Trafic généralise le modèle ON-OFF pour décrire les périodes d'activité et inactivité des applications. La modélisation se fait en trois niveaux : Session, Application et Paquets, avec un niveau Application qui peut comporter des sous niveaux Applications pour représenter l'activité quelque soit son degré de complexité. L'Editeur permet aussi de représenter plusieurs flux en parallèle pour modéliser le trafic des protocoles de signalisation liés aux applications multimédia.
L'Editeur des Sources de Trafic génère des modèles destinés à l'outil de simulation DHS. DHS est un simulateur de réseaux IP/MPLS, basé sur la théorie de du trafic différentiel [3] et de la simulation hybride. Il permet de combiner des modèles continus avec des modèles discrets événementiels.

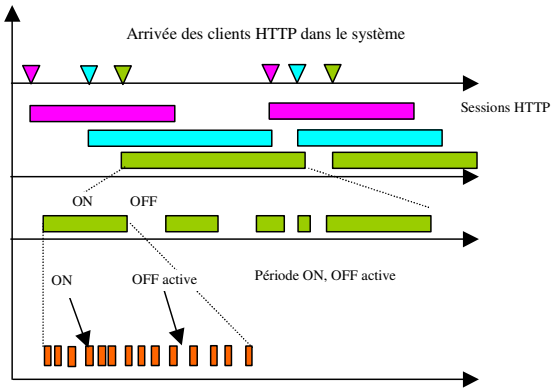



**Fig. 1 –** *Modèle HTTP*

Le même noyau de simulation permet de simuler tout un réseau en mode analytique, événementiel ou hybride. Dans ce dernier mode certains flux sont analytiques et d'autres sont événementiels. Dans l'ensemble des tests qui suit, nous avons utilisé DHS dans le mode événementiel. Les sources HTTP utilisées dans nos simulations comportent un niveau ON-OFF. Une période ON qui représente le téléchargement d'une page web, avec une période OFF qui représente le temps de la lecture passé par l'utilisateur. Plusieurs modèles de sources HTTP, notés W1, W2 et W3 ont été créés avec les paramètres suivants :

| **Modèle de source HTTP** | **Distribution du nombre de pages** | **Moyenne nombre pages** | **Variance nombre de pages** | **Distribution de la taille d'une page** | **Taille Moyenne d'une page** | **Variance taille page** | **Distribution période OFF** | **Moyenne OFF** | **Variance OFF** |
|---|---|---|---|---|---|---|---|---|---|
| W1 | Normale | 10 | 5 | Pareto | 15 Ko | 60 Ko | Pareto | 30 sec | 60 sec |
| W2 | Normale | 10 | 5 | Normale | 15 Ko | 60 Ko | Normale | 30 sec | 60 sec |
| W3 | Normale | 10 | 5 | Exponentielle | 15 Ko | 15 Ko | Exponentielle | 30 sec | 30 sec |

Le modèle de source utilisé pour FTP est plus simple et comporte une seule période ON qui représente le téléchargement d'un fichier. Plusieurs sources FTP, notées F1 et F2 ont été créées avec les paramètre suivants :

| Modèle de source FTP | Distribution de la taille du fichier | Moyenne | Variance |
|---|---|---|---|
| F1 | Pareto | 2 Mo | 4 Mo |
| F2 | Constant | 1 Mo | 0 |

- Les distributions de probabilité associées aux tailles de fichiers de FTP, ou du nombre de pages de HTTP ou de la taille des pages HTTP, ou des périodes OFF peuvent être choisies parmi de nombreuses lois : Normale, Lognormale, Gamma, Pareto Weibull, Exponentielle et constante.
- Le transport des paquets se fait par TCP new Reno implémenté de bout en bout dans le simulateur. Le débit max est limité à 1024 Kbps par connexion TCP, sauf indication contraire.
- La distribution de la taille des paquets peut aussi être paramétrée de manière quelconque. Les distributions utilisées dans la simulation sont généralement des distributions tronquées avec une valeur inférieure et supérieure ceci afin de refléter plus précisément le trafic IP. Dans nos simulations nous avons généré des paquets TCP de taille nominale de 984 octets et des ack d'une taille de 40 octets.

# 4 Expérimentations

Le réseau étudié est composé de deux routeurs, un routeur Source et un routeur Destination. Les deux routeurs sont reliés par un lien ayant un débit nominal de 10 Mb/sec, et un délai nul. La taille du buffer de sortie du routeur Source est de 64 paquets. L'activité des utilisateurs (clients) est représentée par des sessions WEB ou FTP selon les cas. Ces sessions démarrent au cours du temps suivant un processus de Poisson (processus de naissance de la session utilisateur). Chaque session se termine lorsque l'ensemble des données liées à cette session a été transmise (ensemble de pages http ou fichier transmis par FTP. L'ensemble des pages et la taille des pages http est aléatoire comme cela a été décrit précédemment et la taille des fichiers sous FTP est elle aussi aléatoire.
Les simulations sont réalisées sur 12000 secondes avec une granularité de 10 ms pour avoir des échantillons représentatifs du trafic.

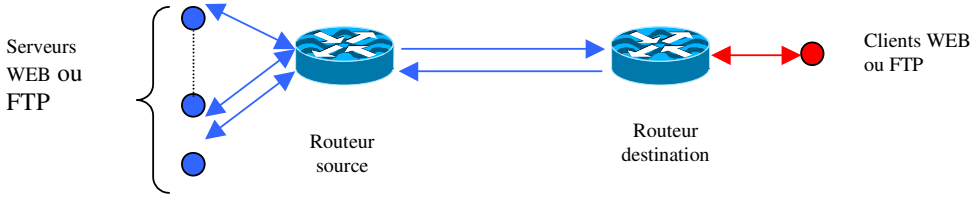



**Fig. 2 –** *Réseau de simulation*

**Sources HTTP**.
Nous avons généré les traces correspondantes aux trois modèles W1, W2, W3 séparément, avec une moyenne de 40 sessions simultanées. Comme le montre le tableau ci-contre, le modèle W1 (ON-OFF de type Pareto) a généré un trafic autosimilaire. Ce résultat est cohérent avec les études qui mettent en cause les distributions à queue lourde de taille de page [1]. Toutefois, les résultats de la simulation montrent aussi que les autres types de distribution génèrent un trafic autosimilaire bien que la taille des pages ne soit plus à queue lourde. La distribution exponentielle génère un trafic corrélé, mais moins autosimilaire que les autres distributions. Il est clair que la nature ON-OFF du trafic WEB est une caractéristique essentielle qui provoque la longue dépendance du trafic, bien plus que la distribution de taille de page ou de la période de lecture. Le réseau dans le cas précèdent est très peu chargé et les pertes sont nulles. Nous allons étudier dans la suite l'influence des pertes sur l'autosimilarité du trafic.

| Modèle | Débit moyen du trafic agrégé | Nombre moyen de session | Taux d'arrivée des sessions | Durée moyenne des sessions | Perte | Hurst |
|---|---|---|---|---|---|---|
| W1 | 131 Kbps | 40 | 0.107 sessions/sec | 372.2 sec | 0% | 0.87 |
| W2 | 128 Kbps | 40 | 0.105 sessions/sec | 378.5 sec | 0% | 0.85 |
| W3 | 129 Kbps | 40 | 0.092 sessions/sec | 434.5 sec | 0% | 0.74 |

**Sources FTP.**
1) Pour analyser l'influence des pertes et du mécanisme de retransmission de TCP sur l'autosimilarité, nous avons généré la trace de la source F2 avec une moyenne d'une session ; le débit de la source TCP est de 1024 Kb/sec. Le trafic généré est très peu corrélé avec une perte nulle. Pour évaluer l'impact de la perte, nous avons réduit la bande passante du lien entre les deux routeurs (de 0.1 à 1 Mbps). Le courbe 3 trace l'évolution du paramètre de Hurst en fonction de la bande passante du lien. Nous observons une augmentation de la corrélation avec l'augmentation des pertes. Ce résultat a été vérifié également par l'augmentation du nombre moyen de session pour une bande passante fixe. Les pertes mesurées ne dépassent pas les 4% dans

le cas d'une bande passante de 0.1 Mb/sec (H=0.98), mais la corrélation du trafic devient très importante. Ce résultat montre bien que le mécanisme de retransmission de TCP et d'adaptation du débit est un facteur prédominant dans la longue dépendance du trafic.

2) Nous avons évalué l'influence des gros transferts de fichiers par FTP. Nous avons généré une source de type F2 avec une moyenne d'une session. Le trafic généré est très peu corrélé avec une perte nulle. Nous avons additionné à ce trafic l'activité de la source F1 qui représente un gros transfert de fichier. Nous avons calculé le paramètre H en augmentant le nombre moyen de session F1. La figure 4 montre que la corrélation du trafic augmente avec l'augmentation de l'activité des gros transferts. L'activité de FTP ressemble à un énorme trafic de type ON-OFF. Ce trafic influence rapidement les propriétés statistiques du trafic global pour le rendre de plus en plus autosimilaire.

Pour minimiser l'effet des gros transferts, Nous avons baissé le débit max de TCP pour les sources F1 (de 1 Mb/sec à 128 Kb/sec). On constate alors que la corrélation du trafic baisse avec la diminution du débit max. Cette corrélation reste tout de même plus importante que la corrélation du trafic sans les sources F1. Cette limitation différenciée des débits permet de limiter l'influence des gros transferts de fichier sur l'autosimilarité du trafic.

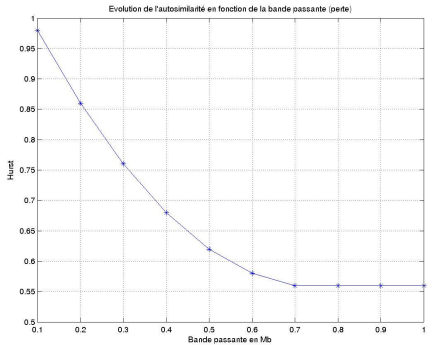



**Fig. 3 –** *Evolution du paramètre de Hurst avec la bande passante du lien entre les deux routeurs*

3) Nous avons appliqué cette technique aux petits flux homogènes de type W1 générés précédemment. On différencie deux cas. La limitation du débit n'a aucune influence sur l'autosimilarité dans le cas où les pertes sont nulles (les files d'attente ne se remplissent pas). Par contre, en cas de congestion (les files d'attente se remplissent), la diminution du débit des sources réduit l'autosimilarité ainsi que les pertes de paquets.

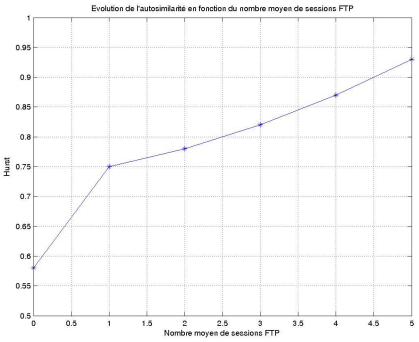



**Fig. 4 –** *Influence des gros transferts sur l'autosimilarité du trafic*

# 5 Conclusion

Nous avons étudié dans cet article l'autosimilarité du trafic Internet sur des traces générées à partir des modèles HTTP et FTP. Nous avons pu vérifier que le modèle ON-OFF avec le transport TCP produit un trafic autosimilaire. Ce résultat est valable pour toutes les distributions et non seulement pour celles à queue lourde. Le mécanisme de crédit et de retransmission de TCP génère des rafales de données qui vont être à l'origine de la dépendance longue du trafic. Dans le cas des gros transferts par FTP, nous avons montré que la diminution du débit maximum de la source TCP réduit considérablement le caractère autosimilaire du trafic. Ce résultat s'applique aussi au cas des petits flux, mais uniquement en cas de pertes. Il s'agit ici d'une diminution constante imposée à la source avant que TCP n'ai pu modifier le débit. L'autosimilarité du trafic est une caractéristique importante à prendre en compte avec la charge du réseau. Toutefois, un trafic autosimilaire peut exister sur un réseau peu chargé, sans que cela n'ait de conséquence néfaste sur les performances. Par contre la performance d'un réseau déjà chargé peut être fortement dégradée par un trafic autosimilaire. Dans ce cas, la diminution du débit des gros flux permettra de maintenir une bonne performance. Une autre alternative pourrait consister à créer plusieurs classes de flux (petits flux, moyens, gros etc.) affectées respectivement sur plusieurs files d'attente DiffServ ordonnancées par WFQ avec des taux de service appropriés.

# Références

[1] S.TAQQU, W WILLINGER and R SHERMAN. Proof of a fundamental Result in Self-Similar Traffic Modeling. ACM SIGCOMM Computer Communication Review Volume 27, Issue 2. Pages: 5 – 23. 1997

[2] A. ADAS and A. MUKHERJEE. On resource management and QoS guarantees for long range dependent traffic. In Proc. IEEE INFOCOM '95, pages 779–787, 1995.

[3] J.M.GARCIA, D.GAUCHARD, O.BRUN, P.BACQUET, J.SEXTON et E.LAWLESS Modélisation différentielle du trafic et simulation hybride distribuée. Rapport LAAS No01660, 2001

[4] HURST, H.E. *Long-term storage: An experimental study*. London: Constable. 1965.

[5] D. STAEHLE, K. LEIBNITZ and P. TRAN-GIAN, Source Traffic Wireless Applications, Report N°261, June 2000, University of Würzburg

[6] R. ADDIE, M. ZUKERMAN, and T. NEAME. Fractal traffic: measurements, modelling and performance evaluation. In Proc. IEEE INFOCOM '95, pages 977–984, 1995

[7] Ian KAPLAN, Estimation of the Hurst Exponent. May 2003. http://www.bearcave.com/misl/misl_tech/wavelets/hurst/

# *Contrôle de Routes par des Appareils de Surveillance (CRAS)*

O. Cogis, B. Darties, S. Durand, J.-C. König, J. Palaysi

*LIRMM, 161 rue Ada, 34392 Montpellier Cedex 5*

Nous nous intéressons au placement d'un minimum d'appareils de surveillance sur les liens d'un réseau de sorte que tout son trafic soit surveillé. Nous montrons que ce problème est NP-difficile et non-APX, même restreint aux grilles avec des chaînes les plus courtes possibles. Nous donnons des algorithmes polynomiaux pour les réseaux linéaires, les anneaux ou encore les étoiles à condition pour ces dernières qu'elles soient orientées. Dans le cas ou les liens d'une étoile ne sont pas orientés, le problème est NP-Difficile mais nous donnons une 2-approximation valable dans un cas plus général comprenant celui des arbres et celui des grilles lorsque les chaînes sont de type ligne-colonne)[†].

**Keywords:** réseaux, surveillance passive, complexité, approximation

## 1 Introduction

La surveillance du trafic dans les réseaux peut se dire active ou passive [NT04, CFGL04]. Le problème de surveillance passive présenté dans [CFGL04] consiste à placer des appareils de surveillance sur le plus petit nombre possible de liens d'un réseau pour contrôler une proportion fixée du trafic. Étant donné un rationnel $k$ appartenant à $]0,1]$, on peut formaliser le problème comme suit :

**Problème 1** *$PPM_k$ (Partial Passive Monitoring)*

**Donnée :** *Un graphe[‡] $G$, un entier positif $h$, un ensemble de chaînes $R$ dans $G$ et une fonction de coût $C$ de $R$ dans $\mathbb{Q}^+$, qui associe un coût à chaque chaîne de $R$.*

**Question :** *Existe t-il un sous ensemble $F \subseteq E(G)$ tel que $|F| \leq h$ et $\frac{\sum_{P \in R_F} C(P)}{\sum_{P \in R} C(P)} \geq k$ (où $R_F = \{P : P \in R \text{ et } E(P) \cap F \neq \emptyset\}$) ?*

Après avoir montré dans un premier temps que $PPM_k$ est NP-complet pour tout $k$ appartenant à $]0,1]$, nous nous intéressons au cas particulier $k = 1$ ($PPM_1$ a déjà été montré NP-complet dans [CFGL04]), problème que nous reformulons comme suit :

**Problème 2** *CRAS (Couverture de Routes par des Appareils de Surveillance)*

**Donnée :** *Un graphe connexe non orienté $G$, un ensemble $R$ de chaînes dans $G$ et un entier positif $h$.*

**Question :** *Existe t-il un sous-ensemble d'arêtes $F \subseteq E(G)$ avec $|F| \leq h$ et tel que si $P$ est une chaîne de $R$ alors $E(P) \cap F \neq \emptyset$ ?*

Par la suite, *minCRAS* désigne le problème d'optimisation «naturellement» associé au problème *CRAS* (i.e. pour $G$ et $R$ donnés, déterminer le plus petit $h$ pour lequel le problème *CRAS* admet une réponse positive) et, pour tout problème, nous utiliserons une notation similaire pour distinguer sa version décision de sa version optimisation.

Sauf mention contraire, les graphes sont des graphes simples (non orientés, sans boucle ni arête multiple). Une grille $M_{n \times m}$ est un graphe dont les sommets $(i, j)$ sont définis par

[†] Une version longue de cette soumission, avec les preuves, existe sous la forme d'un rapport de recherche, de même nom, du LIRMM.

[‡] On notera $E(G)$ l'ensemble des arêtes de $G$ et $V(G)$ l'ensemble de ses sommets.

un numéro de ligne $i$, $1 \leq i \leq n$, et un numéro de colonne $j$, $1 \leq j \leq m$, et où deux sommets sont adjacents ssi ils ne diffèrent que sur une seule de leurs composantes et alors d'une valeur de 1.

Les résultats présentés sont alors les suivants :

- *minCRAS* est NP-difficile et non-APX[§] ; le résultat est encore vrai si on restreint le problème aux grilles et à des plus courtes chaînes ;
- *minCRAS* restreint aux arbres est NP-difficile mais APX ; il en va de même si la restriction est faite aux chaînes ligne-colonne dans les grilles (une chaîne est dite **ligne-colonne** (ou L-C) lorsqu'elle possède un sommet $(i, j)$ tel que tous ses autres sommets appartiennent à la ligne $i$ ou à la colonne $j$ ; un tel sommet $(i, j)$ est appelé **sommet coin**, qu'on choisit de plus de $i$ et de $j$ minimum lorsqu'il en existe plusieurs) ;
- *minCRAS* restreint aux chaînes, aux anneaux et aux étoiles orientées est polynômial.

## 2 NP-complétude de $PPM_k$

Pour montrer que pour tout nombre rationnel $k \in ]0, 1]$, le problème $PPM_k$ est NP-Complet, nous considérons le problème suivant :

**Problème 3** *$CE_k$ (Couverture d'Ensemble dans une proportion au moins k).*

**Donnée :** *Un ensemble fini d'éléments $X = \{x_1, x_2, ..., x_n\}$, un sous-ensemble $E = \{E_1, E_2, ..., E_m\}$ de parties de $X$, un entier positif $h \leq m$.*

**Question :** *Existe-t-il un sous-ensemble $F \subseteq E$ tel que*

$$|F| \leq h \text{ et } \left| \bigcup_{E_i \in F} E_i \right| \geq k \times |X|$$

Ce problème s'apparente au problème de Couverture Partielle d'Ensemble (*CPE*) introduit par M. Kearns [Kea90] mais dans *CPE* la proportion $k$ est une donnée du problème. Le problème *CPE* est bien sûr NP-Complet puisque le problème de Couverture d'Ensemble (CE) en est un cas particulier ($k = 1$) connu pour être NP-Complet (par exemple voir *set cover* dans [ACG+03]). Nous montrons ici que quelle que soit la valeur de $k$, le problème $CE_k$ reste NP-Complet.

**Proposition 1** *Pour tout rationnel $0 < k \leq 1$, le problème $CE_k$ est NP-Complet.*

**Théorème 1** *Le problème $PPM_k$ est NP-Complet pour tout $0 < k \leq 1$.*

## 3 non-APproXimabilité de *minCRAS*

Réduire le problème *CRAS* au problème *CE* est immédiat : les chaînes deviennent les éléments, les arêtes deviennent les sous-ensembles (une arête «couvre» les chaînes qui l'empruntent). Compte tenu de résultats connus pour le problème *CE* [¶], on en déduit que le problème *minCRAS* est approximable avec un facteur d'approximation de $1 + \ln|R|$. Mais une construction «réciproque» est également possible, même en restreignant le problème *CRAS* à des chaînes les plus courtes dans les grilles (elle est illustrée en figure 1). D'où :

**Théorème 2** *Le problème minCRAS est NP-difficile et non-APX. Il est encore NP-difficile et non-APX lorsqu'il est restreint aux grilles et à des chaînes les plus courtes.*

---

[§] APX signifie qu'il existe un algorithme polynômial approché résolvant *minCRAS* à un facteur constant près. non-APX signifie le contraire, sauf si P=NP.

[¶] Le problème *CE* appliqué à un ensemble de cardinal $n$ est non-APX et $1 + \ln n$-approximable (voir par exemple [ACG+03]).

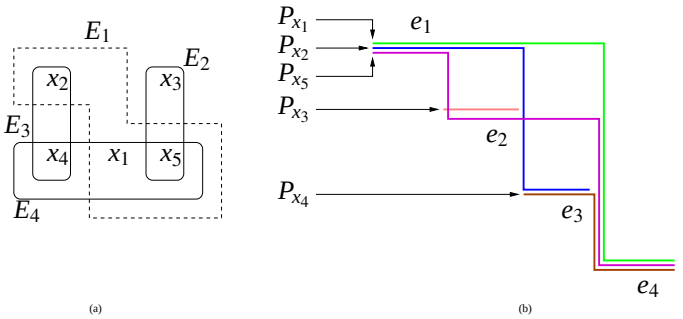



**FIG. 1:** Une instance $(\{x_1,x_2,x_3,x_4,x_5\},\{E_1 = \{x_1,x_2,x_5\},E_2 = \{x_3,x_5\},E_3 = \{X_2,X_4\},E_4 = \{x_1,x_4,x_5\}\})$ du problème CE et sa réduction à une instance du problème *CRAS* dans une grille avec des plus courtes chaînes.

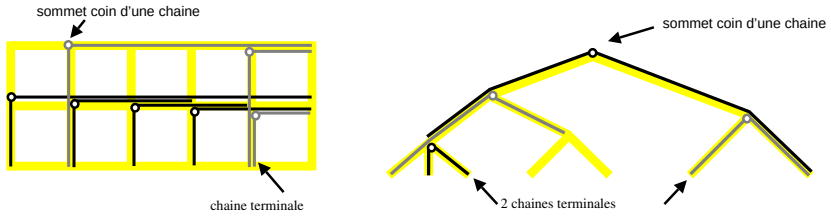



**FIG. 2:** Exemples de sommets coin et de chaînes terminales dans les grilles et les arbres. On peut vérifier que l'instance de *minCRAS* dans la grille n'est pas *tree representable*.

## 4 Cas NP-difficiles mais approximables pour *minCRAS*

Si $G$ est un arbre, *minCRAS* est le problème bien connu de *multicut*. *CRAS* est donc NP-complet, même restreint aux arbres de hauteur 1 [GVY97]. Il est aussi NP-complet dans les grilles avec des routages L-C[‖].

Dans les arbres, il existe une 2-approximation [GVY97], que les auteurs généralisent à toute instance de *CE* représentable par des chaînes dans un arbre (instance *tree-representable*). À condition d'enraciner les arbres en un sommet arbitraire et de définir comme sommet coin d'une chaîne d'un arbre son sommet le plus proche de la racine, l'algorithme *AminCRAS* (voir encadré) est un algorithme glouton, pour le problème *minCRAS*, avec garantie de performance aussi bien pour les arbres que pour les routages L-C dans les grilles, dont on notera qu'ils ne sont pas *tree-representable* (voir par exemple la figure 2).

Dans les deux cas, les **arêtes-coin** d'une chaîne $P$ sont les arêtes de $P$ incidentes à son sommet coin (leur nombre est donc égal à 1 ou 2). $P$ est une **chaîne terminale** dans $R$ si pour toute autre chaîne $Q \in R$ l'intersection de $P$ et de $Q$ est vide ou contient au moins une des arêtes-coins de $P$ (voir la figure 2 pour des exemples).

Enfin on partitionne en quatre classes les chaînes L-C suivant que, si $(i,j)$ est le coin d'une chaîne L-C, pour tous ses sommets $(k,l)$ on a $k \leq i$ ou $k \geq i$, et $l \leq j$ ou $l \geq j$.

**Proposition 2** *L'algorithme AminCRAS donne une 2-approximation dans les arbres et dans les grilles avec des chaînes L-C d'une même classe.*

**Théorème 3** *minCRAS restreint aux chaînes L-C dans les grilles est APX.*

[‖] Communication personnelle de Guillaume Bagan

**Données** : Un ensemble $R$ de chaînes (resp. de chaînes L-C d'une même classe) dans un arbre enraciné (resp. une grille)
**Résultat** : Un ensemble d'arêtes $S$ avec au moins une arête de chaque chaîne de $R$

$S \leftarrow \emptyset; R' \leftarrow R$ ;
**tant que** $R' \neq \emptyset$ **faire**
  Choisir une chaîne $P$ terminale dans $R'$ ;
  Ajouter à $S$ la ou les arêtes-coins de $P$;
  Retirer de $R'$ toutes ses chaînes passant par une arête-coin de $P$;
**fin**

**Algorithme 1** : L'algorithme AminCRAS.

# 5 Cas polynômiaux

Quand l'arbre est une chaîne, en enracinant l'arbre à une de ses extrémités, l'algorithme AminCRAS donne un résultat optimal. Dans les étoiles orientées[**], toute instance de *minCRAS* peut se ramener, en temps polynômial, à une instance de *minCAS* dans un biparti[††], problème connu pour être polynômial (voir par exemple [CR04]).

**Théorème 4** *Le problème minCRAS est polynômial s'il est restreint aux chaînes, aux anneaux ou aux étoiles orientées.*

# 6 Conclusion et Perspectives

Le problème *minCRAS* est NP-difficile et non-APX dans le cas général. Pour certaines topologies, il reste NP-difficile mais devient APX (arbres et grilles avec des chaînes L-C). En revanche, en restreignant la topologie aux chaînes ou aux anneaux, on a montré que des démarches gloutonnes étaient optimales.

Améliorer le facteur d'approximation de *minCRAS* restreint aux arbres, donc en particulier aux arbres de hauteur 1, améliorererait du même coup les meilleures approximations connues pour le problème *vertex cover*[‡‡] [GVY97]. En revanche des bornes inférieures restent à établir pour *minCRAS* restreint aux routages L-C dans les grilles, et le résultat obtenu pour les étoiles orientées invite à poursuivre l'étude du contrôle dans les réseaux orientés.

# Références

[ACG+03] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation*. Springer, 2003.

[CFGL04] C. Chaudet, É. Fleury, and I. Guérin Lassous. Positionnement optimal de sondes pour la surveillance active et passive de réseaux. CFIP 2005, nov 2004.

[CR04] O. Cogis and C. Robert. *Théorie des Graphes*. Vuibert, 2004.

[GVY97] N. Garg, V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18 :3–20, 1997.

[Kea90] M. Kearns. *The Computational Complexity of Machines Learning*. MIT Press, Cambridge MA, 1990.

[NT04] H. X. Nguyen and P. Thiran. proceedings of active measurement for multiple link failures diagnosis in ip networks (pam2004). In *Passive and Active Measurement Workshop*, Antibes - Juan-lès-Pins, France, Avril 2004. Springer-Verlag. Lecture Notes of Computer Science (volume 3015).

[**] Une étoile orientée est un arbre de hauteur 1 dans lequel chaque arête $\{i, j\}$ est remplacée par un arc $(i, j)$ et/ou un arc $(j, i)$.

[††] S'il n'y a pas de chemins de longueur 1 ; ce que nous pouvons supposer sans perte de généralité.

[‡‡] Pour une présentation de *vertex cover*, voir par exemple [ACG+03]

# *Surveillance passive dans l'Internet*

C. Chaudet[1], E. Fleury[2], I. Guérin-Lassous[2] et H. Rivano[3], M.-E. Voge[3]

[1] *ENST - 46, rue Barrault - 75634 Paris Cedex 13, France - Claude.Chaudet@enst.fr*
[2] *INRIA Ares - CITI, INSA de Lyon, 69621 Villeurbanne, France - Prénom.Nom@insa-lyon.fr*
[3] *I3S/INRIA Mascotte - INRIA Sophia Antipolis, BP 93, 06902 Sophia-Antipolis - Prénom.Nom@sophia.inria.fr*

Afin d'obtenir les informations nécessaires à une bonne gestion des ressources de leur réseau, les opérateurs placent des sondes passives sur les liens de leurs points de présence. Dans cet article, nous donnons des écritures en programmes linéaires mixtes des problèmes de placement de sondes simples ou avec échantillonnage, et donnons une stratégie pour la maintenance de la surveillance partielle de trafics dynamiques dans un point de présence. Ces formulations améliorent les résultats de deux articles récents de la littérature.

**Keywords:** surveillance passive, positionnement de sondes, métrologie, problème de flot, simulations

## 1 Introduction

Le problème de la surveillance passive dans l'Internet consiste à placer sur les liens d'un réseau des équipements spécifiques (appelés *sondes*) analysant le trafic transitant sur les liens. Cette surveillance permet d'évaluer dynamiquement la sécurité, la connectivité et les différentes utilisations faites du réseau, et donc de gérer efficacement l'infrastructure et les ressources du réseau. Le positionnement des sondes dans le réseau est un problème clé pour la surveillance dans l'Internet, et suscite beaucoup d'intérêt dans la communauté "Réseaux". Ces équipements sont très coûteux et présentent de nombreuses contraintes physiques comme l'espace mémoire, l'alimentation, etc. Des articles très récents se sont intéressés au positionnement optimal de sondes [1, 2]. Les approches et problèmes résolus dans ces deux articles sont très similaires. Dans [2], les auteurs introduisent des notions de coûts de déploiement et de maintenance pour les sondes, alors que dans [1] les sondes ont un coût unitaire. En revanche, dans [1], les auteurs s'intéressent à la résolution optimale du positionnement par le biais de la programmation linéaire en nombres entiers, tandis que dans [2], les auteurs utilisent des algorithmes d'approximation basés sur une approche gloutonne. Enfin, dans [2], les sondes sont capables d'échantillonner le trafic, *i.e.* de contrôler le nombre de paquets à analyser.

Dans cet article, nous présentons le problème ainsi que le cadre de l'étude en section 2, puis nous améliorons et étendons les résultats donnés dans [1] et [2]. Notamment, nous présentons dans la section 3 une amélioration de la formulation de [1] : nous modifions le programme linéaire initial en un autre programme linéaire répondant toujours à notre problème mais nécessitant un temps de résolution bien plus court. Enfin, dans la section 4, nous étendons cette modélisation pour prendre en compte la notion d'échantillonnage introduite dans [2], et obtenir un algorithme rapide de gestion de la surveillance du réseau lorsque le trafic est dynamique.

## 2 Présentation du problème

Un point de présence (POP) permet de diriger les connexions des utilisateurs, connectés aux *routeurs d'accès*, vers le cœur du réseau de l'opérateur. La surveillance du trafic passant à travers un POP est un enjeu majeur, notamment pour donner aux opérateurs les moyens de négocier au mieux les accords de service avec d'autres opérateurs. À ces fins, un opérateur installe des sondes sur les interfaces des routeurs du

réseau. Chacune de ces sondes surveille une partie du trafic, fixe ou adaptable (sonde avec *échantillonnage*). L'objectif d'un opérateur n'est pas forcément de surveiller la totalité du trafic, mais une certaine proportion $k$. En effet l'expérience montre que le surcoût nécessaire à la couverture des derniers pourcentages de trafic est très important, alors que le gain pour la maintenance du réseau ne l'est pas.

**Routage** Le routage des flux obéit à des règles propres à l'opérateur, pouvant aller d'un calcul des plus courts chemins type OSPF à des algorithmes d'équilibrage de charge. Nous supposerons donc que les chemins peuvent être arbitraires, mais qu'il n'y en a qu'un "petit" nombre par flux. Par exemple, le protocole de routage OSPF garde moins d'une dizaine de chemins différents pour chaque flux.

**Modélisation** Les routeurs et liens de communication d'un POP seront représentés par un graphe $G = (V,E)$. Les flux sont transportés sur des chemins $p \in \mathcal{P}$. On agrège dans un même trafic tous les flux passant sur un ensemble $\mathcal{P}_{s,t}$ de chemins allant d'un client/POP source $s$ à un client/POP destination $t$. Chaque chemin $p$ transporte une quantité de trafic $v_p$. Le problème qui nous intéresse est donc de sélectionner certaines arêtes du graphe et, le cas échéant, de régler la fréquence d'échantillonnage sur chaque arête, afin de mesurer au moins $k \cdot \sum_{\mathcal{P}} v_p$ unités de trafic ($k \in [0,1]$).

Dans le cas, théorique, de sondes sans échantillonnage mesurant l'intégralité du trafic qu'elles voient passer, si l'on désire mesurer 100% du trafic, ce problème s'écrit comme un *Minimum Set Cover* dans le graphe d'incidence des chemins sur les arêtes[†].

Pour toute valeur de $k \in [0,1]$, le problème se modélise assez directement par un *Minimum Edge Cost Flow* (MECF) dans un graphe auxiliaire. Le MECF est un problème de flot dans lequel le coût de certaines arêtes est binaire, nul quand aucun flot ne passe par l'arête et 1 quand une quantité quelconque passe l'arête. Ce problème est $\mathcal{NP}$-difficile et non approximable [3]. Étant donnée la structure du graphe auxiliaire, il est possible de combiner les équations de flot pour obtenir les formulations en programmes linéaires mixtes compactes présentées dans les sections suivantes.

## 3 Amélioration de la modélisation initiale

**Programme linéaire 1** (PPM(k))**.**

$$
\begin{array}{rrcll}
\textit{Minimiser} & \sum_{e\in E} x(e) & & \textit{le nombre de sondes posées} \\
\textit{t.q.} & \sum_{e\in p} x(e) & \geq & \delta(p) \quad \forall p \in \mathcal{P} \\
& \sum_{p\in\mathcal{P}} \delta(p)\cdot v_p & \geq & k \cdot \sum_{p\in\mathcal{P}} v_p \\
& \delta(p) & \in & [0,1] \quad \forall p \in \mathcal{P} \\
& x(e) & \in & \{0,1\} \quad \forall e \in E
\end{array}
$$

Dans le programme linéaire PPM(k), $x(e)$ indique si une sonde est placée sur l'arc $e$ (si oui $x(e)$ est alors égal à 1) et $\delta(p)$ indique si le trafic passant par $p$ est surveillé ou non. Dans la modélisation initiale de [1], $\delta(p)$ est entier et vaut 0 ou 1. La première contrainte détermine les trafics qui seront surveillés par les sondes, alors que la deuxième permet de s'assurer que sufisamment de trafic est surveillé. Il est possible de considérer $\delta(p)$ comme une variable décimale dans $[0,1]$, ce qui ne change rien à la solution et accélère

[†] Le graphe d'incidence des chemins sur les arêtes est le graphe biparti avec un sommet par chemin dans une partie, un sommet par arête dans l'autre, et un arc entre deux sommets si le chemin correspondant au premier sommet passe sur l'arête correspondante au deuxième. Il est à noter que cette réduction prouve la $\mathcal{NP}$-difficulté et la non-approximabilité du problème : c'est le cas de *Minimum Set Cover* dans les graphes bipartis, et que tout graphe biparti est le graphe d'incidence de chemins sur les arcs d'une grille.

grandement sa résolution. En effet, dès qu'on a une solution avec $0 < \delta(p) < 1$, on peut arrondir $\delta(p)$ à 1, les $x(e)$ restent inchangés,et la valeur de $\sum_{p\in\mathcal{P}} \delta(p)v_p$ augmente, comme recherché.

Nous avons comparé les temps de calcul obtenus avec la modélisation initiale et celle proposée ici. Les POPs et les trafics considérés sont les mêmes que ceux présentés dans [1], en revanche les programmes linéaires ont été résolus par la bibliothèque CPLEX[4]. La table 1 compare les différents temps de calcul pour des POPs de 10, 15 et 29 nœuds et un taux de surveillance de 80%.

| $\lvert V\rvert - \lvert\mathcal{P}\rvert$ | version modifiée | version initiale |
|---|---|---|
| 10 – 132 | $0,02\ s$ | $0,05\ s$ |
| 15 – 1980 | $1,9\ s$ | $30,6\ s$ |
| 29 –11130 | $34,2\ s$ | $873\ s$ |

**TAB. 1:** Temps de calcul moyen pour 40 exécutions sur des POPs de 10, 15 et 29 nœuds

# 4 Surveillance avec échantillonnage

Certaines sondes sont en mesure de ne capturer qu'une partie du trafic, grâce à un échantillonnage des paquets. Sur des liens très haut débit atteignant des débits de plusieurs Gb/s (OC-48, OC-192, OC-255), ne pas avoir à stocker et analyser tous les paquets, mais seulement une certaine proportion, apporte un gain significatif dans les coûts d'exploitation des sondes. Cette proportion de trafic échantillonné va dépendre du coût d'exploitation des sondes placées, et donc du coût d'échantillonnage d'un paquet sur une sonde, ce coût pouvant varier d'une sonde à l'autre en fonction de la vitesse du lien surveillé. On souhaite donc minimiser le coût d'exploitation des sondes tout en garantissant toujours une proportion globale $k$ de trafic surveillé. Si l'opérateur du POP souhaite avoir une vision de tous les trafics, sans pour autant devoir surveiller tous les chemins, on garantit une couverture minimum $h$ sur chaque trafic. On notera que $h \leq k$ étant donné qu'il s'agit d'un minimum de couverture de chaque trafic alors que $k$ est un minimum sur la totalité des trafics.

## *4.1 Écriture en programmation linéaire mixte*

Nous allons modéliser le coût d'installation d'une sonde sur un lien $e$ par $cost_i(e)$ et son coût d'exploitation par $cost_e(e)$. Ces fonctions de coût peuvent être quelconques sans que cela n'ait d'impact sur la formulation du problème linéaire 2, présenté ci-dessous. Néanmoins, le coût d'exploitation est souvent une fonction croissante concave [2], ce qui permet de prendre en compte le facteur d'échelle.

**Programme linéaire 2** (PPME(h,k))**.**

$$\begin{aligned}
\textit{Minimiser} \quad & \sum_{e\in E} (cost_i(e)\cdot x(e) + cost_e(e)\cdot f(e)) && \textit{le coût d'installation et d'exploitation}\\
\textit{t.q.} \quad & \sum_{e\in p} f(e) \geq \delta(p) \quad \forall p\\
& x(e) \geq f(e) \quad \forall e\in E\\
& \sum_{p\in\mathcal{P}_{s,t}} \delta(p)\cdot v_p \geq h\cdot \sum_{p\in\mathcal{P}_{s,t}} v_p \quad \text{pour tout trafic } (s,t)\\
& \sum_{p\in\mathcal{P}} \delta(p)\cdot v_p \geq k\cdot \sum_{p\in\mathcal{P}} v_p\\
& \delta(p), f(e) \in [0,1] \quad \forall p\in\mathcal{P}, \forall e\in E\\
& x(e) \in \{0,1\} \quad \forall e\in E
\end{aligned}$$

Comme pour le programme linéaire 1, $x(e)$ indique si une sonde est placée sur l'arc $e$. Par contre $\delta(p)$ indique quelle proportion du trafic passant par $p$ est surveillée. On introduit ici les variables $f(e)$ qui

modélisent le taux d'échantillonnage de la sonde placée sur $e$, et les contraintes qui indiquent tout naturellement que pour échantillonner du trafic sur un arc $e$, il est nécessaire d'y avoir installé une sonde au préalable. Les contraintes suivantes indiquent qu'une proportion minimale de $h$ de chaque trafic doit être surveillé et que l'on doit aussi avoir une proportion de couverture de trafic globale d'au moins $k$.

### 4.2 Trafic dynamique et adaptation de l'échantillonnage

Le trafic transporté par un POP est intrinsèquement dynamique et fluctue selon les périodes d'activité de la journée. Un changement des débits des flux et /ou des routes peut dégrader fortement la pertinence d'une surveillance avec échantillonnage. S'il n'est pas envisageable pour un opérateur de modifier la position des sondes à chaque changement de trafic, il reste tout à fait possible d'adapter les taux d'échantillonnage des sondes déjà installées. Il s'agit alors de trouver une solution au programme linéaire $PPME(h,k)$ en fixant a priori les $x(e)$, puisque les sondes sont déjà mises en œuvre. On note $PPME^*(x,h,k)$ ce problème.

$PPME^*(x,h,k)$ s'écrit comme le programme linéaire 2 en considérant les $x(e)$ comme des constantes. Il n'y a donc plus de variables binaires, et il est alors possible de trouver une solution optimale en temps polynomial. En fait, il s'agit même d'un calcul de flot de coût minimum qui peut s'effectuer rapidement, sans recours à la programmation linéaire.

Si un opérateur se donne un niveau de surveillance minimal par trafic $h$, un niveau global $k$ et un seuil de tolérance $T < k$, une stratégie de maintien de la surveillance dans un POP pourrait être :

1. Tant que $\sum_{p\in\mathcal{P}} \delta(p)\cdot v_p \geq T\cdot\sum_{p\in\mathcal{P}} v_p$, attendre ;
2. Dès que $\sum_{p\in\mathcal{P}} \delta(p)\cdot v_p < T\cdot\sum_{p\in\mathcal{P}} v_p$, calculer $PPME^*(x,h,k)$, mettre à jour les taux d'échantillonnage ;
3. Goto 1.

La résolution de $PPME$ devient donc une phase initiale de la conception du POP pour laquelle la complexité n'est pas cruciale. Par contre, lors de l'adaptation de l'échantillonnage au trafic, le temps de calcul est très important car il s'agit de réaction à des situations potentiellement fortement dynamiques. Le calcul de $PPME^*$ est rapide et, s'agissant d'un calcul de flot, nécessite peu de ressources.

## 5 Conclusion

Cet article donne une modélisation des divers problèmes de placement de sondes dans un réseau afin de mettre en œuvre une surveillance passive. Il donne des solutions efficaces et améliore les résultats précédents grâce à une modélisation par flot qui permet aussi de déduire les différents résultats de $\mathcal{NP}$-complétude et de non-approximabilité.

Les extensions possibles de cette thématique concernent la prise en compte totale des multi-chemins, problème difficile. Le but est d'arriver à une modélisation qui s'affranchit du facteur multiplicatif entre le nombre de chemins employés et le nombre de flux considérés.

Pour finir, on peut s'interroger sur l'impact que peuvent avoir la topologie des POPs, qui est le plus souvent multi-étages, et les protocoles de routage de plus court chemin (OSPF) employés, sur le placement des sondes au sein d'un POP.

## Références

[1] C. Chaudet, E. Fleury, and I. Guérin Lassous. Positionnement optimal de sondes pour la surveillance active et passive de réseaux. In *CFIP 2005*, Bordeaux, France, March 2005.

[2] K. Suh, Y. Guo, J. Kurose, and D. Towsley. Locating network monitors : complexity, heuristics, and coverage. In *Infocom 2005*, Miami, FL, USA, March 2005.

[3] G. Even, G. Kortsarz, and W. Slany. On network design problems : fixed cost flows and the Covering Steiner Problem. *Transactions on Algorithms*. à paraître.

[4] ILOG CPLEX. http ://www.ilog.com/products/cplex/index.cfm. (v7.5).

# *Liste des auteurs*