Université Assane Seck de Ziguinchor

UFR Sciences et Technologies

Département Informatique

Mémoire Master 2

Présenté par Avéwé Bassène

Encadré par Dr Youssou Dieng

Etat de l'art sur le Routage Compact

Date de soutenance : 11 janvier 2017

Etat de l'art sur le Routage Compact

Résumé: Savoir comment transmettre une information est fondamental dans un réseau. Il est essentiel que chaque entité du réseau soit capable de décider localement, avec sa vue du réseau, du chemin par lequel l'information doit passer. Ainsi, il est souvent utile d'étudier la topologie du réseau, modélisée par un graphe, pour répondre à ces exigences.

Nous nous intéressons dans un premier temps, à une présentation des généralités sur les graphes. En effet, comme dans beaucoup de problèmes de graphes, pour étudier la topologie des réseaux afin d'exploiter les propriétés structurelles qui en découlent il est souvent préférable de connaitre la notion de graphe et l'ensemble de ses propriétés.

Ensuite, nous avons fait un état de l'art sur le routage compact. En effet, la croissance de taille des réseaux est supérieure à celle des capacités des routeurs actuels. La problématique du routage compact consiste à stocker des tables ayant moins d'entrées que les routeurs du réseau mais garantissant l'acheminement de messages par des routes proches des plus courts chemins. Dans cette partie, nous présentons les principes d'implémentation de trois types de routage compact et un état de l'art de chacun d'eux; le routage par intervalle, le routage de plus courts chemins et le routage de plus courts chemins avec facteur d'étirement.

Dans la dernière partie, nous abordons le problème du routage compact dans les graphes de halin non valués. Nous nous sommes intéressés aux schémas de routage de plus courts chemins utilisant des adresses, des tables de routage de tailles optimales de $O(\log n)$ bits, où n est le nombre de sommets du graphe. Nous n'avons pas pu trouver un tel schéma de routage pour les graphes de halin mais, nous proposons un schéma de routage compact avec facteur d'étirement d'au plus 2.

Dicipline: Informatique

Mots clefs :Routage compact,
graphe planaire,
graphe planaire-extérieur, graphe de halin
Algorithme distribué, Algorithmique

Université Assane Seck - Ziguichor-Sénégal

State of the art on compact routing

Abstract: In a network, it is crucial to know how to construct an efficient routing scheme. It is fundamental for each entity with its local knowledge of the network, to be able to decide on which link to forward messages. Thus, it is important to study the underlying network topology in order to design routing schemes.

In the first part of this paper, we are interested in a presentation of the generalities on graphs. Indeed, as in many graph problems, the study of the topology of the networks in order to exploit the structural properties that stem from it is often better to know the notion of graph and all of its properties.

Then, we made a state of the art on compact routing, indeed, the growth of network sizes is superior to that of current routers' capabilities. The problem of compact routing consists in storing tables with fewer entries than the network routers but guaranteeing the routing of messages by roads close to the shortest paths. In this part, we present the wise of implementation of three types of compact routing and a state of the art of each of them; the interval routing, the shortest paths routing and the shortest paths routing with stretch factor.

At last, we treat the problem of compact routing scheme for no weighted halin graphs. More precisely, we are interested in shortest-path routing schemes that use $O(\log n)$ bits for addresses, headers and routing tables, where n is the number of vertices in the graph. We could not find such a routing scheme for no weighted halin graphs but, we propose a shortest paths routing scheme with stretch factor at most 2.

Dicipline: Computer-Science

Keywords :Compact routing,
planar graph,
outerplanar-graph, halin graph
distributed Algorithm,
Algorithmic

Université Assane Seck - Ziguichor-Sénégal

TABLE DES MATIÈRES

TABLE DES FIGURES

CHAPITRE 1

INTRODUCTION

1.1 Introduction

Avant toute chose, il est indispensable de répondre à la question suivante: qu'est-ce qu'un réseau ? Un réseau peut être défini comme un ensemble d'entités interconnecter (entretenant des relations). Ces entités peuvent représenter des personnes, des villes, des routes, etc..., et suivant l'entité, on parlera de réseaux routiers, réseaux de neurones, réseaux sociaux, réseaux d'eau ou de gaz (canalisations), réseaux informatique, etc...

Ainsi, par définition, Un réseau informatique est un ensemble de périphériques et matériels (que l'on pourra qualifier d'hôtes) reliés entre-eux, au moyen d'une seule technologie, pour communiquer. Deux ordinateurs reliés entre-eux suffisent à former un réseau.

Un réseau informatique permet:

- La communication de plusieurs ordinateurs ou périphériques entre eux
- Le partage de documents (fichiers, dossiers) mais aussi de périphériques (imprimantes, disques durs en réseau : NAS).
- Le jeu à plusieurs
- L'unicité de l'information (il est bien plus facile de partager la même information en provenance d'un réseau que sur des ordinateurs cloisonnés).
- Une organisation plus efficace et donc une meilleure productivité pour effectuer les tâches classiques de bureautique collaborative.

Les besoins de faire communiquer des ordinateurs et périphériques entre eux remontent aux années 60. A l'époque, la communication n'était possible que sur

une portée limitée (quelques dizaines de mètres). Les réseaux filaires tel qu'on les connait sont apparus dans les années 1970. Les réseaux sans fil Wifi (très utilisés aujourd'hui) sont apparus en 1997.

Il existe différents moyens de relier des infrastructures ou périphériques entre-elles. Ainsi, en fonction de la portée géographique des équipements à relier, on parlera de:

- Réseau local ou LAN (Local Area Network): il permet de relier des ordinateurs et périphériques situés dans la même pièce, voire dans le même bâtiment.
- Réseau sans fil (WIFI) ou WLAN (Wireless Local Area Network): il s'agit d'un réseau utilisant la technologie WIFI, pouvant couvrir plusieurs dizaines de mètres.
- Le réseau Métropolitain ou MAN (Metropolitan Area Network): il couvre une ville entière.
- Réseau étendu ou WAN (Wide Area Network): Il peut couvrir un pays, un continent, ou le monde tout entier. Internet est un WAN.

Il existe également d'autres méthodes de découpage. On peut classer les réseaux en fonction de leur type d'utilisation.

- Intranet: il s'agit d'un réseau interne à une entreprise ou organisation, n'étant pas accessible de l'extérieur par le public.
- Extranet: c'est un réseau externe. On parle aussi de réseau extranet en désignant la partie visible du réseau d'une entreprise ou organisation.
- Internet: le regroupement de tous les réseaux!

L'arrangement physique, c'est-à-dire la configuration spatiale du réseau est appelée topologie physique. On distingue généralement les topologies suivantes: topologie en bus, topologie en étoile, topologie en anneau, topologie en arbre, topologie maillée, etc... . Toutes ces topologies sont basées sur l'un ou l'autre des types suivants:

- Les réseaux **Peer to Peer** (P2P) : Il s'agit d'une communication simple entre deux ordinateurs.
- Les réseaux **Client/Serveur**. Ils sont organisés avec des postes serveurs qui fournissent l'information au client (comme internet par exemple).

Et chacune de ces topologies à ses avantages et inconvénients;

Topologie linéaire: La topologie linéaire consiste à relier les ordinateurs les uns à la suite des autres, en formant une ligne virtuelle. Offrant des avantages dans certains types d'utilisation (notamment dans la sécurité du réseau), son problème principal est sa faible tolérance de panne. Un seul ordinateur qui défaille et l'ensemble du réseau (ou presque) peut se mettre à ne plus fonctionner. La réparation est alors impérative.

Topologie en Bus: C'est en quelque sorte une topologie linéaire améliorée. Les ordinateurs sont reliés les uns à la suite des autres, mais la panne de l'un d'entre eux ne perturbe pas le fonctionnement du réseau. On bouche généralement les extrémités d'un réseau utilisant la topologie en bus par des bouchons, pour éviter les réflexions du signal.

Topologie en étoile: C'est la plus utilisée. Un concentrateur (Hub, Switch, Routeur) représente un équipement du réseau. Les ordinateurs ou périphériques sont reliés à lui. Toute panne d'un périphérique n'entraine pas de panne du réseau. En revanche, la panne d'un concentrateur perturbera la communication de tous les périphériques et ordinateurs qui en dépendent.

Topologie en anneau: Les ordinateurs ou périphériques sont reliés les uns aux autres, il n'y a pas d'extrémités contrairement à un réseau linéaire. La défaillance d'un élément n'entraîne pas de panne du réseau.

Topologie en arbre: Aussi appelée topologie hiérarchique, elle consiste à relier les éléments à la manière d'une pyramide. En haut se trouve un élément, qui sera lui-même relié à d'autres éléments, qui eux même seront à leur tour reliés à d'autres éléments, et ainsi de suite.

Dans un réseau, pour permettre une communication harmonieuse entre plusieurs ordinateurs ou périphériques reliés, il y à un ensemble de règles ou une série d'étapes à suivre. Cette série de règles est appelée protocole. Les protocoles sont classés en deux catégories:

- Les protocoles où les machines s'envoient des accusés de réception (pour permettre par exemple, une gestion des erreurs au moment de l'envoi des données). Ce sont les protocoles dits orientés connexion.
- Les autres protocoles qui n'avertissent pas la machine qui va recevoir les données sont les protocoles dits non orientés connexion.

Les protocoles sont hiérarchisés en couches, et suivant le nombre et l'agencement des couches on peut directement identifier ce que l'on appelle des modèles. Un modèle est un standard de communication, en réseau, de tous les systèmes informatiques. On en distingue principalement deux;

Le modèle TCP/IP (Transmission Control Protocol/Internet Protocol) hiérarchisé en quatre couches, et le modèle OSI (Open Standard Interconnection) en sept couches.

Chaque couche s'occupe d'apporter un plus permettant la transmission de données, et de fournir des éléments avec les couches de niveau supérieur. Plus on monte dans le niveau des couches, plus la différence entre les données physiques (signal électrique) et logiques (langage de programmation) est accentuée.

Le réseau Internet est un ensemble de protocoles regroupés sous le terme "TCP-IP" (Transmission Control Protocol/InternetProtocol). Voici une liste non exhaustive des différents protocoles qui peuvent êtres utilisés:

- HTTP (Hyper Texte Transfert Protocol): C'est celui que l'on utilise pour consulter les pages web.
- FTP (File Transfert Protocol): C'est un protocole utilisé pour transférer des fichiers.
- SMTP (Simple Mail Transfert Protocol): C'est le protocole utilisé pour envoyer des mails.
- POP: C'est le protocole utilisé pour recevoir des mails.
- Telnet: Utilisé surtout pour commander des applications côté serveur en lignes de commande.
- IP: (internet Protocol): L'adresse IP vous attribue une adresse lors de votre connexion à un serveur.

Une adresse IP (avec IP pour Internet Protocol) est un numéro d'identification qui est attribué de façon permanente ou provisoire à chaque appareil connecté à un réseau informatique utilisant l'Internet Protocol. L'adresse IP est à la base du système d'acheminement (le routage) des messages sur Internet. Et, chacun des messages envoyés dans un réseau dispose d'un ensemble d'informations servant à faciliter son expédition vers la destinataire. Cette ensemble d'informations est très souvent regroupé sous le terme "d'en-tête du message".

1.2 Notion de routage

Typiquement lorsqu'on est dans un réseau, l'une des idées qui nous viennent en tête c'est de vouloir envoyer des données (messages) d'une machine A, vers une autre machine B. Les deux machines peuvent être dans le même réseau comme ils peuvent être dans des réseaux différents. Dans tous les cas, entre ses deux machines, il y a un ensemble de nerouteurs qui doivent "décider" des chemins par lesquels les données doivent être envoyées. Ce principe est appelé routage.

Un routeur est un élément intermédiaire dans un réseau informatique assurant le routage des données. Son rôle est de faire transiter des paquets d'une interface réseau vers une autre, au mieux, selon un ensemble de règles.

Pour permettre alors à un routeur de "décider", de façon autonome de la manière dont les messages doivent être envoyés dans un réseau, il y a un ensemble d'information qu'il faut mettre à sa disposition. Cet ensemble d'information constitue ce que l'on appelle un schéma de routage.
Les composants d'un schéma de routage sont les tables de routages, les en-têtes

de message, et un algorithme de routage assurant la prise de décision en se basant sur les informations de la table de routage. Cette dernière spécifie pour chaque ordinateur du réseau, les informations permettant de le joindre.

Bien que les routeurs soient des matériaux incontournables dans les réseaux informatiques, ils disposent cependant d'un espace de stockage (capacité mémoire) limité et d'un coût relativement proportionnel à cette capacité. Il devient alors important de chercher à réduire la quantité d'information à mettre à leurs dispositions.

Dans le routage, la longueur des routes utilisées pour envoyer les messages d'un équipement A vers un autre équipement B est un critère de qualité. En outre cette grandeur physique, l'espace requis par les tables de routage, la taille des en-têtes de message, les adresses sources et destination du message et le temps de latence (temps de prise de décision) de chaque routeur le long du chemin entre A et B, sont autant de facteurs à prendre en compte.

Ainsi, cette recherche de politique ayant pour but de réduire au mieux l'ensemble des informations nécessaires aux routeurs pour accomplir leurs tâches est appelée routage compact.

La conception d'un schéma de routage dans les réseaux informatiques passe par une étude structurelle de la topologie sous-jacente de celui-ci. Alors pour mettre en place un schéma de routage d'un réseau, on utilise l'outil graphe pour faire une représentation théorique de celui-ci afin de faciliter sont études. Un graphe est un objet des mathématiques combinatoires généralisant le concept de relation binaire et pouvant être représenté par un schéma reliant des sommets par des arcs ou des arêtes. L'outil graphe et ses propriétés ont été présentées dans le chapitre 2 de ce mémoire.

Notre travail consiste à déterminer un schéma de routage compact de plus court chemin dans les graphes dits de halin. Beaucoup d'efforts ont été consacrés à la conception de schémas de routage compact en termes de longueur des routes, qui équilibre la charge de chaque routeur en termes de quantité d'information.

Dans les chapitres 3 et 4 de ce travail, nous avons expliqué les principes de base de quelques types de routage compact et présenté des résultats phares obtenus sur l'étude des réseaux basés sur ces types de routage; le ≪routage par intervalle≫ au chapitre 3 et le ≪ routage compact de plus courts chemins ≫ dans le chapitre 4.

Du point de vue théorique, il semble toujours possible de répartir l'information de routage équitablement sur tous les nœuds du réseau, tout en gardant la même qualité sur la longueur des routes.

Cependant, la conception de schémas de routage comporte souvent un compromis taille des tables/longueur des routes. Il arrive souvent que des longueurs de route plus longues que le plus court chemin soient induites par le schéma de routage dans le souci de réduire la taille des tables de routage. On parle dans ces cas de schémas de routage avec facteur d'étirement. Le facteur d'étirement d'un schéma de routage est le maximum des rapports, entre la longueur de la route induite par l'algorithme pour envoyer un message entre deux nœuds du réseau, sur la longueur du plus court chemin entre ces deux nœuds.

Par exemple, un but rechercher pour les réseaux de halin, est de déterminer un schéma de routage compact de plus court chemin. Malheureusement, il n'est pas toujours aisé de trouver un tel schéma. Cependant, nous avons pu obtenir un schéma de routage compact avec facteur d'étirement d'au plus 2. Notre résultat est présenté au chapitre 5 de ce mémoire. L'approche adoptée face à ce problème est en partie basée sur les travaux de P. Fraigniaud et C. Gavoille dans [19] et indépendamment Thorup et Zwick [21]. Elle s'appuie de plus sur le routage compact dans les graphes planaire extérieure présentée dans [20] par Y. Dieng et C. Gavoille.

Ainsi nous obtenons que tout graphe de halin non valué à n sommets admet un schéma de routage compact avec facteur d'étirement d'au plus 2.

CHAPITRE 2

GÉNÉRALITÉS

2.1 Introduction

L'envoi de message d'un ordinateur à un autre à travers les réseaux est devenu un mécanisme courant dans notre vie de tous les jours. Un réseau informatique se définit comme un ensemble d'équipements interconnectés entre eux et qui échange des informations. L'une des fonctionnalités fondamentales d'un réseau informatique est le routage.
Le routage dans un réseau informatique est l'action de déterminer la route à suivre par un message pour aller d'un équipement(Machine) émetteur vers un ou plusieurs équipements récepteurs. Cette route est définie à l'aide d'un ensemble d'information tout aussi indispensable les uns que les autres.
Un schéma de routage est le mécanisme opérant en mode distribué et permettant de transmettre des messages d'une source à une destination dans le réseau. Les composants d'un schéma de routage sont les tables de routage, les adresses, les en-tête, et un algorithme de routage.
La fonction de routage est un algorithme stocké sur chaque routeur. Il prend comme paramètre l'en-tête du message arrivant et qui, en fonction de la table de routage du sommet courant, détermine le numéro de port sur lequel le message doit être retransmis.
Un routeur est un composant du réseau informatique qui assure le routage des messages d'une machine à une autre. Il est composé d'une ou de plusieurs interfaces d'entrées et de sorties apellées port, qui joue le rôle d'envoi et de réception de message dans le réseau. Chaque routeur est équipé d'un espace de stockage dans lequel sont stockées les informations de routage(Fonction de routage, table de routage).
Une table de routage est une structure de données stockées sur chaque routeur du réseau et permettant à la fonction de routage de retourner le numéro local

du port par lequel un message d'adresse de destination i doit être transmis.
Tout message circulant dans le réseau porte un en-tête, et l'adresse de destination du message se trouve dans cet en-tête. L'en-tête du message est généré par la machine source et reste souvent fixe tout au long du trajet. Il est généré à partir de la table de routage de l'emetteur et de l'adresse de destination du message. Dans beaucoup de schémas, l'en-tête se résume à l'adresse de destination.
A la réception d'un message par un routeur, l'algorithme de routage prend comme paramètre l'en-tête du message et en fonction de la table de routage locale, détermine le numéro de port par lequel le message doit être retransmis.
Comme on peut le constater, le routeur est un composant important dans le routage. Mais bien qu'il soit un élément incontournable, force est de signaler que les routeurs ont une taille en espace mémoire limité et leur fabrication est indépendante de la taille du réseau dans lequel ils doivent être utilisés. Une briève présentation du principe de routage est la suivante:

Soit un réseau donné, construire sur chaque nœud (routeur) de celui-ci un schéma de routage efficace qui détermine, pour tout message entrant du nœud, le port par lequel il doit être retransmis. Ce calcul est possible grâce aux informations sur l'en-tête du message, sur l'identifiant du lien d'entré du message et sur la table de routage locale. Le schéma de routage peut aussi dépendre du type de routage. Nous verrons plus loin quelques types de routages dans les graphes mais avant, définissons les quelques propriétés de la théorie des graphes dont nous aurions besoin dans la suite du document.

2.2 Graphe et propriètés

Un graphe G est un couple (V, E) où

- V est un ensemble (fini) d'objets. Les éléments de V sont appelés les sommets du graphe.
- E est un sous-ensemble de $V \times V$. Les éléments de E sont appelés les arêtes du graphe.

Une arête e d'un graphe G est une paire $e = (x, y)$ de sommet de ce graphe.
Les sommets x et y d'un graphe G sont les extrémités de l'arête e de G.
On appelle ordre d'un graphe G le nombre de sommet de celui-ci.
Deux sommets x et y sont adjacents si il existe l'arête (x, y) dans E de G. Les sommets x et y sont alors dits voisins.
Une arête e est incidente à un sommet x si x est l'une des extrémités de e. Le degré d'un sommet x de G est le nombre d'arêtes incidentes à x. Il est noté $deg(x)$. Pour un graphe G, $deg(x)$ correspond également au nombre de sommets adjacents à x. Pour un graphe G d'ordre n, le degré d'un sommet est un entier compris entre 0 et $n - 1$. Un sommet x est dit isolé s'il est de degré 0; c'est à dire relié à aucun autre sommet.
Dans un graphe, il est naturel de vouloir se déplacer d'un sommet à un autre

en suivant les arêtes, une telle marche est appelée une chaîne ou un chemin. Un chemin est une liste $p = \{x_1, ..., x_k\}$ de sommets telle qu'il existe dans le graphe une arête entre chaque paire de sommets successifs. Le terme chemin s'emploie en propre pour les graphes orientés(défini plus bas), pour les graphes non orientés, on parle de chaîne. Cependant la définition formelle est exactement la même pour chemin et chaîne, seule change la structure (graphe orienté ou non) sur laquelle ils sont définis. Une chaîne peut être fermée, c'est-à-dire que les sommets de départ et de fin sont les mêmes et dans ce cas on l'appelle cycle (fig 1.1).

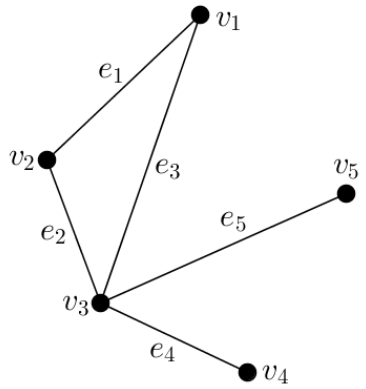


Figure 2.1: G=(V,E) ou $(v_1, e_1, v_2, e_2, v_3, e_3, v_1)$ est un cycle.

On appelle distance entre deux sommets x et y d'un graphe G, la longueur de la plus petite chaîne reliant x et y. Le diamètre d'un graphe G est la plus longue des distances entre deux sommets. Le graphe est une représentation symbolique d'un réseau dans différents domaines(physique, informatique, mathématique, cartographie, réseau routier,etc..). Un graphe peut être utilisé pour représenter n'importe quelle situation de la vie courante dans laquelle on doit passer par plusieurs étapes pour atteindre un but. Les sommets peuvent représenter les pièces d'une maison, les coins de rues dans une ville, des ordinateurs sur un réseau, des personnes dans un entourage ou des décisions dans un algorithme.

Un graphe peut être orienté ou non. Un graphe orienté a des flèches sur ses arêtes indiquant qu'il est possible de le parcourir suivant un sens, comme passer d'un sommet A à un sommet B, mais pas de B à A. La relation B à A est par contre possible pour un graphe non orienté. Dans ce type de graphe, la communication entre les sommets du graphe est bidirectionnelle. Une arête peut également avoir un coût, c'est à dire un nombre associé à l'arête. Ce coût peut représenter un coût monétaire, une distance ou n'importe quel autre concept selon le graphe, on parle dans ce cas de graphe valué.

Dans un graphe non orienté $G = (V, E)$, un couple de sommets $\{s_i, s_j\}$ est appellé arête et est représenté graphiquement par $s_i - s_j$. On dit que s_i et s_j sont des sommets adjacents ou voisins.

Dans un graphe orienté $G = (V, E)$, un couple de sommets $\{s_i, s_j\}$ est appellé arc et est représenté graphiquement par $s_i \rightarrow s_j$, où s_i est le sommet initial ou origine, et s_j le sommet final ou extrémité.

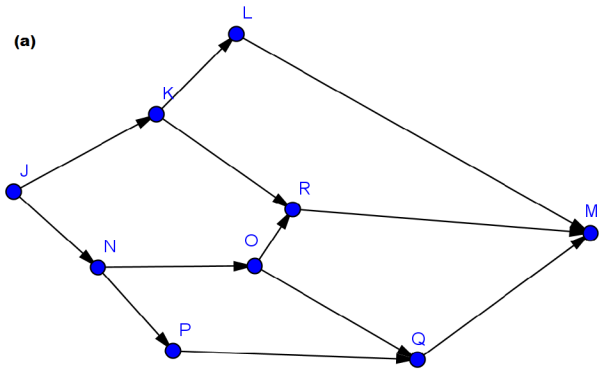


Figure 2.2: Graphe G=(V,E) orienté et non valué.

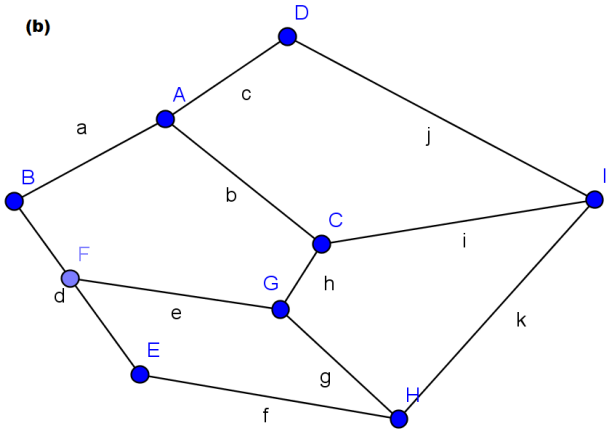
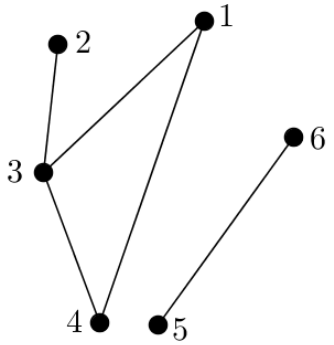


Figure 2.3: Graphe G'=(V',E') non orienté et valué.

2.2.1 Quelques types de graphes

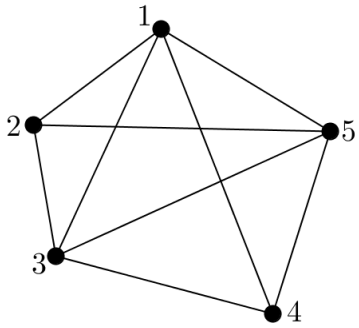
Un graphe connexe: Un graphe G est connexe s'il est possible, à partir de n'importe quel sommet, de rejoindre tous les autres en suivant les arêtes. Un graphe non connexe se décompose en composantes connexes. Sur le graphe ci-dessous, les composantes connexes sont $\{1, 2, 3, 4\}$ et $\{5, 6\}$.



Graphe non connexe

$V = \{1, 2, 3, 4, 5, 6\}$
$E = \{\{1,3\}, \{1,4\}, \{2,3\}, \{3,4\}, \{5,6\}\}$

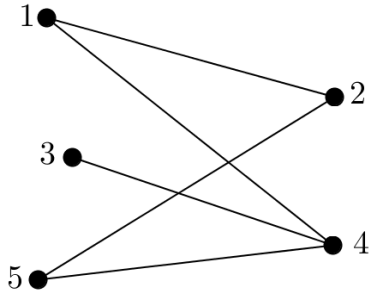
Un graphe complet: Un graphe G est dit complet si chacun des sommets de G est voisin de tous les autres sommets.



Graphe complet K_5

$V = \{1, 2, 3, 4, 5, 6\}$
$E = \{\{1,2\}, \{1,3\}, \{1,4\}, \{1,5\}, \{2,3\}, \{2,4\}, \{2,5\}, \{3,4\}, \{3,5\}, \{4,5\}$

Un graphe biparti: Un graphe est biparti si ses sommets peuvent être divisés en deux ensembles X et Y, de sorte que toutes les arêtes du graphe relient un sommet dans X à un sommet dans Y (dans l'exemple ci-dessous, on a $X = \{1, 3, 5\}$ et $Y = \{2, 4\}$, ou vice versa).

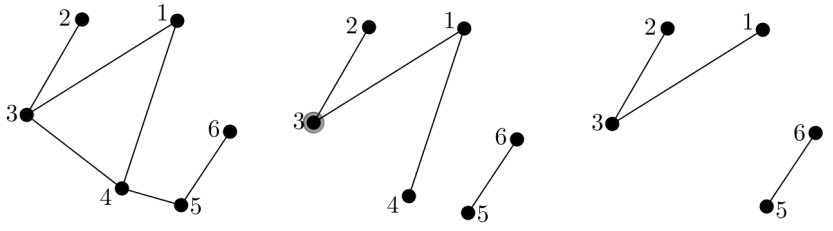


Graphe biparti

$V = \{1, 2, 3, 4, 5\}$
$E = \{\{1,2\}, \{1,4\}, \{2,5\}, \{3,4\}, \{4,5\}\}$

Graphe partiel et sous-graphe: Soit $G = (V, E)$ un graphe. Le graphe $G' = (V, E')$ est un graphe partiel de G, si $E' \subseteq E$. Autrement dit, on obtient

G' en enlevant une ou plusieurs arêtes au graphe G. Pour un sous-ensemble de sommets $A \subseteq V$, le sous-graphe de G induit par A est le graphe $G = (A, E(A))$ dont l'ensemble des sommets est A et l'ensemble des arêtes $E(A)$ est formé de toutes les arêtes de G ayant leurs deux extrémités dans A. Autrement dit, on obtient G' en enlevant un ou plusieurs sommets au graphe G, ainsi que toutes les arêtes incidentes à ces sommets.



Graphe G

$V = \{1, 2, 3, 4, 5, 6\}$

$E = \{\{1, 3\}, \{1, 4\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{5, 6\}\}$

Graphe partiel de G

$V = \{1, 2, 3, 4, 5, 6\}$

$E = \{\{1, 3\}, \{1, 4\}, \{2, 3\}, \{5, 6\}\}$

Sous-graphe de G

$V = \{1, 2, 3, 4, 5, 6\}$

$E = \{\{1, 3\}, \{2, 3\}, \{5, 6\}\}$

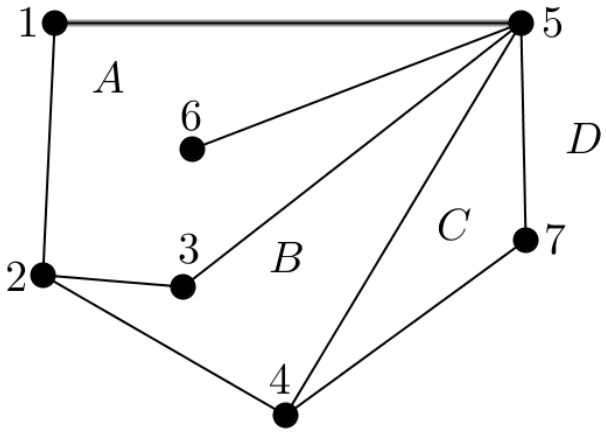
Graphes eulériens: On appelle cycle eulérien d'un graphe G un cycle passant une et une seule fois par chacune des arêtes de G. Un graphe est dit eulérien s'il possède un cycle eulérien. On appelle chaîne eulérienne d'un graphe G une chaîne passant une et une seule fois par chacune des arêtes de G. Un graphe ne possédant que des chaînes eulériennes est semi-eulérien. Plus simplement, on peut dire qu'un graphe est eulérien(ou semi-eulérien) s'il est possible de dessiner le graphe sans lever le crayon et sans passer deux fois sur la même arête.

Graphes hamiltoniens: On appelle cycle hamiltonien d'un graphe G un cycle passant une et une seule fois par chacun des sommets de G. Un graphe est dit hamiltonien s'il possède un cycle hamiltonien. On appelle chaîne hamiltonienne d'un graphe G une chaîne passant une et une seule fois par chacun des sommets de G. Un graphe ne possédant que des chaînes hamiltoniennes est semi-hamiltonien. Contrairement aux graphes eulériens, il n'existe pas de caractérisation simple des graphes (semi-)hamiltoniens. On peut énoncer quelques propriétés et conditions suffisantes:

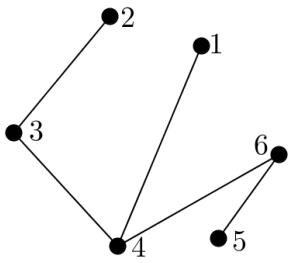
- Un graphe possédant un sommet de degré[1] 1 ne peut pas être hamiltonien;
- Si un sommet dans un graphe est de degré 2, alors les deux arêtes incidentes à ce sommet doivent faire partie du cycle hamiltonien;
- Les graphes complets K_n sont hamiltoniens.

[1]Le degré (ou valence) d'un sommet d'un graphe est le nombre de liens (arêtes ou arcs) reliant ce sommet. Le degré d'un sommet s est noté $deg(s)$.

Graphes planaires: On dit qu'un graphe est planaire si on peut le dessiner dans le plan de sorte que ses arêtes ne se croisent pas. Rappelons que les arêtes ne sont pas forcément rectilignes. Une carte, ou graphe planaire topologique, est une représentation particulière d'un multigraphe planaire fini. On dit qu'une carte est connexe si son graphe l'est. Une carte divise le plan en plusieurs régions. Par exemple, la carte ci-dessous, avec sept sommets et neuf arêtes, divise le plan en quatre régions (A, B, C, D). Trois régions sont limitées alors que la quatrième (D), extérieure au diagramme, ne l'est pas. Ces régions sont appelées faces.

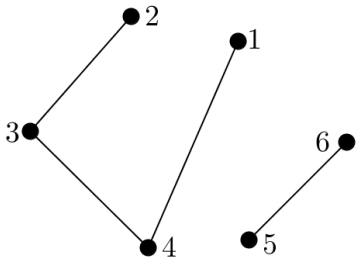


Arbres: On appelle arbre tout graphe connexe sans cycle. Un graphe sans cycle et non connexe est appelé une forêt. Une feuille ou sommet pendant est un sommet de degré 1.



Arbre

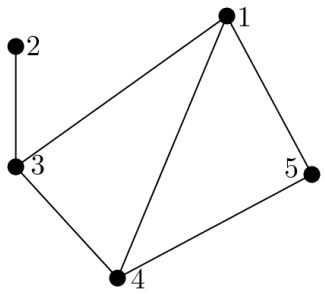
Les sommets 1, 2 et 5 sont les feuilles



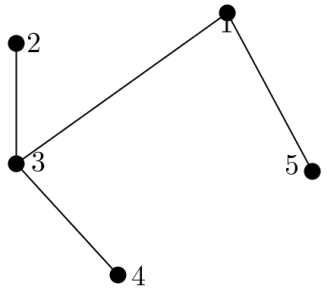
Forêt

Les sommets 1, 2, 5 et 6 sont les feuilles

Arbres couvrants: Un arbre couvrant (aussi appelé arbre maximal) est un graphe partiel qui est aussi un arbre.



Graphe G



Un arbre couvrant

On notera entre autre les relations père-fils ou petit-fils entre les sommets de l'arbre; par exemple dans le graphe à droite on dira que le sommet 3 est un descendant du sommet 1 ou plus particulièrement un fils (enfant) du sommet 1 et 1 un ancêtre de 3 ou encore parent du sommet 3.

2.2.2 Opérations sur les graphes

Calcul de plus court chemin: Algorithme de Dijkstra

Edgser Wybe Dijkstra (1930-2002) a proposé en 1959 un algorithme qui permet de calculer le plus court chemin entre un sommet particulier et tous les autres d'un graphe. Le résultat est une arborescence, c'est-à-dire un arbre avec un sommet particulier appelé racine. Numérotons les sommets du graphe $G = (V, E)$ de 1 à n. Supposons que l'on s'intéresse aux chemins partant du sommet 1. On construit un vecteur $\lambda = \lambda(1); \lambda(2); ...; \lambda(n)$ ayant n composantes tel que $\lambda(j)$ soit égal à la longueur du plus court chemin allant de 1 au sommet j. On initialise ce vecteur à $c_{1,j}$, c'est-à-dire à la première ligne de la matrice des coûts du graphe, définie comme indiqué ci-dessous:

$$\text{f(x)}= \begin{cases} 0 & \text{si } i = j \\ \infty & \text{si } i \neq j \text{ et } (i,j) \notin E \\ \delta(i,j) & \text{si } i \neq j \text{ et } (i,j) \in E \end{cases}$$

Où $\delta(i, j) > 0$ est le poids de l'arc (i, j).

On construit un autre vecteur p pour mémoriser le chemin pour aller du sommet 1 au sommet voulu. La valeur $p(i)$ donne le sommet qui précède i dans le chemin. On considère ensuite deux ensembles de sommets, S initialisé à $\{1\}$ et T initialisé à $\{2, 3, ..., n\}$. À chaque pas de l'algorithme, on ajoute à S un sommet jusqu'à ce que $S = V$ de telle sorte que le vecteur λ donne à chaque étape la longueur minimale des chemins de 1 aux sommets de S.

Parcours en largeur (Breadth First Search = BFS)

Le parcours en largeur est obtenu en gérant la liste d'attente au coloriage comme une file d'attente (FIFO= First In First Out). Autrement dit, on enlève à chaque fois le plus vieux sommet gris dans la file d'attente, et on introduit tous les successeurs blancs de ce sommet dans la file d'attente, en les coloriant en gris. Structures de données utilisées:

- On utilise une file F, pour laquelle on suppose définies les opérations $initFile(F)$ qui initialise la file F à vide, $ajouteFinFile(F, s)$ qui ajoute le sommet s à la fin de la file F, $estVide(F)$ qui retourne vrai si la file F est vide et faux sinon, et $enleveDebutFile(F, s)$ qui enlève le sommet s au début de la file F.
- On utilise un tableau Π qui associe à chaque sommet le sommet qui l'a fait entrer dans la file, et un tableau *couleur* qui associe à chaque sommet sa couleur (blanc, gris ou noir).
- On va en plus utiliser un tableau Π qui associe à chaque sommet son niveau de profondeur par rapport au sommet de départ s_0 (autrement dit, $d[s_i]$ est la longueur du chemin dans l'arborescence Π de la racine s_0 jusque s_i). L'algorithme ParcoursLargeur() présente la méthode du parcours d'un graphe G en largeur en partant d'un sommet initial s.

```
ParcoursLargeur(Graphe G, Sommet s):
    f = CreerFile();
    f.enfiler(s);
    marquer(s);
    tant que la file est non vide
        s = f.defiler();
        afficher(s);
        pour tout voisin t de s dans G
            si t non marqué
                f.enfiler(t);
                marquer(t);
    Fin tant que;
Fin;
```

Parcours en profondeur (Depth First Search = DFS)

Le parcours en profondeur est obtenu en gérant la liste d'attente au coloriage en noir comme une pile (LIFO = Last In First Out). Autrement dit, on considère à chaque fois le dernier sommet gris entré dans la pile, et on introduit devant lui tous ses successeurs blancs. Structures de données utilisées :

- On utilise une pile P, pour laquelle on suppose dénies les opérations $initPile(P)$ qui initialise la pile P à vide, $empile(P, s)$ qui ajoute s au sommet de la pile P, $estVide(P)$ qui retourne vrai si la pile P est vide et faux sinon, $sommet(P)$ qui retourne le sommet s au sommet de la pile P, et $depile(P, s)$ qui enlève s du sommet de la pile P.
- On utilise, comme pour le parcours en largeur, un tableau Π qui associe à chaque sommet le sommet qui l'a fait entrer dans la pile, et un tableau *couleur* qui associe à chaque sommet sa couleur (blanc, gris ou noir).
- On va en plus mémoriser pour chaque sommet si :
  - dec[si] = date de découverte de si (passage en gris).
  - fin[si] = date de n de traitement de si (passage en noir) où l'unité de temps est une itération. La date courante est mémorisée dans la variable *tps*. L'algorithme Explorer présente la méthode du parcours en profondeur dans un graphe G en partant d'un sommet initial s:

```
Explorer(graphe G, sommet s)
    marquer le sommet s
    pour tout sommet t voisin du sommet s
        si t n'est pas marqué alors
            explorer(G, t);
    Fin pour;
Fin;
```

Arbre de niveau (Layering-tree)

Soit $G = (V; E)$ un graphe avec un sommet distingué s, Partitionnons $V(G)$ en niveaux:

Pour tout entier $i \geq 0$, un niveau $L_i = \{u \in V(G) | d_G(s, u) = i\}$. Alors chaque niveau L_i est partitionné en $L_1^i, \ldots, L_{pi}^i$, tels que deux sommets appartiennent à une même partie si et seulement s'ils sont connectés par un chemin visitant seulement les sommets à distance au moins i de s. Un arbre de niveau de G ou

Layering tree est le graphe dont l'ensemble des sommets est une collection de toutes les parties L_j^i.

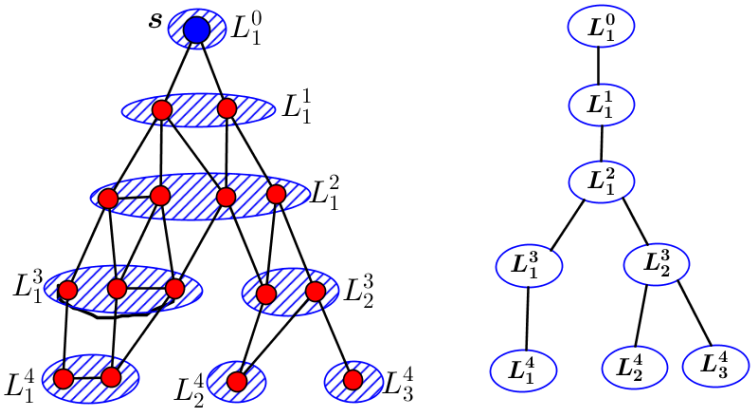


Figure 2.4: Un graphe G et un arbre de niveau de G

Tout arbre T admet un sommet n appelé *médian*, tel que chaque composant connecté à $T \setminus \{u\}$ a au plus $1/2|V(T)|$ sommets.

Un arbre d'hiérarchie de T est un arbre enraciné H définie comme suit:

La racine de H est la médian u, et ses enfants sont les racines des arbres d'hiérarchie des composants connectés de $T \setminus \{u\}$.

2.3 Généralité sur le routage

Un routage sur un réseau modélisé par un graphe est la donnée pour tout couple (x, y) de sommets du graphe d'une chaîne reliant x à y (appelée aussi route). Pour la communication réelle de messages à travers un réseau chaque sommet doit calculer d'une manière locale la route d'un message qui transite par ce sommet. Ainsi le schéma se construit de manière distribuée en suivant, sommets après sommets, les chemins du routage. Nous allons expliquer ce mécanisme à l'aide du schéma de la figure 3.

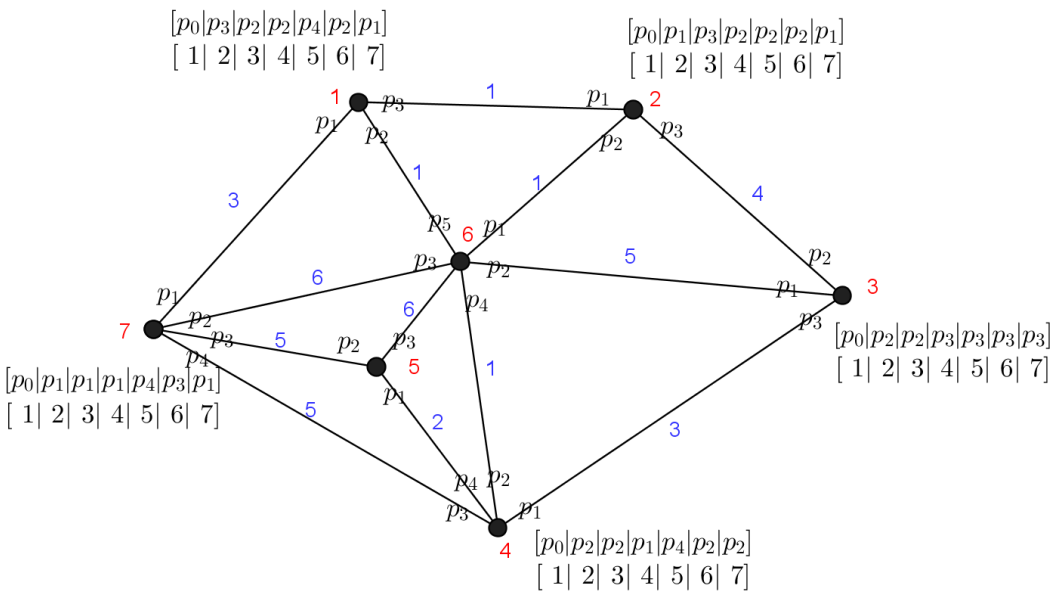


Figure 2.5: Graphe G=(V,E) est numéroté(ports et sommets).

Soit G un réseau à n nœuds(ici $n = 7$) numérotés par des entiers de 1 à 7, et T une table de routage complet de G. Il s'agit ensuite de stocker T dans chaque nœud du réseau, en spécifiant pour chaque case numéro i de la table, le port de sortie permettant de router vers la destination i sur un plus court chemin. La table T est construit de telle sorte que tout nœud u, connaissant l'adresse de destination v du message, puisse à l'aide de la fonction de routage, décider du port de sortie par lequel le message doit être retransmis pour atteindre v suivant un plus court chemin. Cette décision est prise en local et est basé sur la table de routage du nœud courant et sur l'adresse de destination du message.
Considérons un message d'adresse de destination 3 arrivant au nœud d'identifiant 7.
L'algorithme de routage vérifie d'abord que le message n'est pas destiné au nœud courant; il récupère l'adresse de destination dans l'en-tête du message et la compare à l'adresse du nœud courant qui est ici 7 ($7 \neq 3$). Et comme il y a une différence entre ces deux adresses alors il passe à l'étape suivante.
L'algorithme de routage va vérifier dans la table de routage, le numéro de port contenu dans la colonne du tableau ayant le numéro d'indexe 3. Le port numéroté p_1 est alors choisi et le message est retransmis à l'arête reliée à ce port. Le message atteindra alors le nœud d'identifiant 1.
En ce nœud, l'algorithme de routage vérifie que $1 \neq 3$ et que le numéro de port contenu dans la colonne du tableau ayant le numéro d'indexe 3 est p_2, ce qui veut dire que le nœud courant n'est pas destinataire du message et le message pourrait atteindre sa destination en étant transmis sur l'arête reliée au port p_2.

Le message est alors retransmis au nœud d'identifiant 6.
Au nœud d'identifiant 6, le même processus se répéte; une fois encore l'algorithme s'assure que $6 \neq 3$ et que la table de routage du nœud courant indique le port p_4 comme étant le port de l'arête par lequel le message doit être retransmis.
Le message atteint le nœud 4 qui à son tour, à l'aide de l'algorithme de routage indique p_1 comme port de sortie pour atteindre le nœud 3.
A son arrivée au nœud d'identifiant 3, l'adresse de destination du message est comparée à l'identifiant du nœud courant. L'algorithme vérifie l'égalité 3=3 et conclut alors que le message est bien à destination et s'arrête.

Le chemin emprunté par l'algorithme pour router le message du nœud 7 au nœud 3 est l'ensemble des arêtes $\{(7,1),(1,6),(6,4),(4,3)\}$. Le total de tous les coûts des arêtes sur ce chemin est de 8. Il est facile de vérifier que cette valeur est la distance minimale parmi tous les chemins reliant 7 à 3. Donc le routage s'est bien effectué suivant un plus court chemin, ce qui est un facteur éssentiel dans un routage.
Pendant son exécution, l'algorithme de routage vérifie d'abord en un temps t_1 que l'adresse de destination est bien différente de l'adresse du nœud courant; En effet, il s'agit de faire un test d'égalité entre 2 entiers ce qui se fait effectivement en un temps constant, donc t_1 est bien une constante. Puis en un temps t_2, pour identifier le port adéquat via lequel le message doit être retransmis; Pour trouver ce port il suffit juste de regarder la case dont le numéro(index) est égal à l'identifiant du nœud destination. Cette case du tableau contient en faite le port recherché. Et ceci correspond donc à lire l'élément qui se trouve à la case i du tableau, ce qui se fait par conséquent en un temps constant. D'où t_2 une constante. On conclut donc que cette vérification se fait en un temps constant $t = t_1 + t_2$. Ce qui est aussi un facteur important dans le routage.
Sur chaque nœud du réseau, la table de routage stockée est à n entrées, soit au total un espace mémoire de $\tilde{O}(n)$ (le tilde indique qu'on travail aux facteurs logarithmiques près). En tenant compte du fait qu'il faut $\log n$ bits pour stocker un entier n, on se retrouve avec une table de taille $(n \log n)$ bits pour chaque nœud du réseau. Cette valeur peut devenir plus conséquente pour des réseaux de grande taille car, plus le réseau est grand plus il y a de nœuds, et plus il y a de nœuds, plus l'espace mémoire requis pour le stockage est important. Il faut comprendre que les routeurs sont des matériaux informatiques avec un espace de stockage limité et par conséquent il est primordiale de chercher à réduire la taille des informations de routage afin qu'elle puisse utiliser le minimum d'espace possible. C'est la problématique à laquelle s'intéresse le routage compact.

Le problème du routage compact consiste à trouver une description succincte et, de la fonction de routage qui calcule la route à suivre en chaque sommet en fonction de la destination du message, et, des informations nécessaire à cette fonction pour router convenablement.

CHAPITRE 3

LE ROUTAGE PAR INTERVALLE

L'un des nombreux types de routage compact est le routage par intervalle. Il a surtout pour but d'optimiser la taille des informations de routage. Le routage par intervalle est une façon d'implémenter un schéma de routage dans un réseau. Il est basé sur une représentation compacte de la table de routage stockée sur chaque nœud du réseau, en regroupant sous forme d'intervalle d'adresses consécutives, l'ensemble des adresses de destination utilisant le même port.

Pour illustrer ce schéma, nous allons utilisé un graphe à n sommets non orienté représentant la topologie sous-jacent du réseau. Les nœuds sont numérotés par des entiers de 1 à n. Sur chaque nœud du réseau, la table de routage consistera à regrouper dans une entrée l'ensemble des destinations passant par le même port, sous forme d'intervalles d'entiers consécutifs. Le routage est exécuté de façon distribuée, c'est-à-dire décentralisée sur chacun des nœuds du réseau. Nous allons voir comment la table de routage est construite dans le routage par intervalle.

3.1 Principe

Dans le schéma de routage par intervalle, on a une table de routage telle qu'à l'intérieur de chaque case on regroupe l'ensemble des destinations qui utilisent le même port. Donc la taille de la table est proportionnelle au nombre de ports du nœud courant. Maintenant ce qu'il y a c'est qu'à l'intérieur de chaque case, toutes les destinations qui utilisent le même port sont compactées sous forme d'intervalle d'entiers consécutifs. Prenons l'exemple de la figure 3: Sur la table de routage du nœud numéro 1, on peut vérifier que les nœuds $3, 4, 5$ et 6 passent par le même port p_2 donc on peut les representer sous forme d'intervalles d'entiers consécutifs comme suit $[3, 6]$. Et ainsi de suite pour les autres destinations; [2] pour p_3 et [7] pour p_1. De ce fait il est possible d'avoir plusieurs

intervalles dans une seule case du tableau. Alors maintenant l'enjeu va être de trouver un moyen de faire de telle sorte que le nombre d'intervalle dans chaque case de ce tableau soit le plus petit possible. Cela va forcément dépendre de la manière dont les sommets sont numérotés. Et par conséquent l'enjeu revient donc à trouver la fonction de numérotation permettant de rendre compact le nombre d'intervalle dans chaque case. Si pour un graphe donné on trouve une numérotation qui permet de faire un schéma de routage de telle sorte que chaque case de notre tableau contienne au plus k d'intervalle, on dira qu'on a un routage par k-intervalle en anglais le $k-interval$ routing. Avec donc k intervalles dans chaque case, le nombre total d'intervalle dans un nœud x va être inférieur ou égal à $deg(x) \times k$. Le schéma de routage va être de taille $O(deg(x) \times k)$.

L'explication du principe du routage par intervalle sur un graphe G est la suivante:
En chaque nœud $x \in G$, la fonction de routage s'arrête si l'adresse de destination y de l'en-tête du message correspond à x, sinon le message est retransmis suivant une arête étiquetée par l'intervalle $I = [a, b]$ tel que $y \in I$. Vérifier si $y \in [a, b]$ revient à tester si y est compris entre a et b.

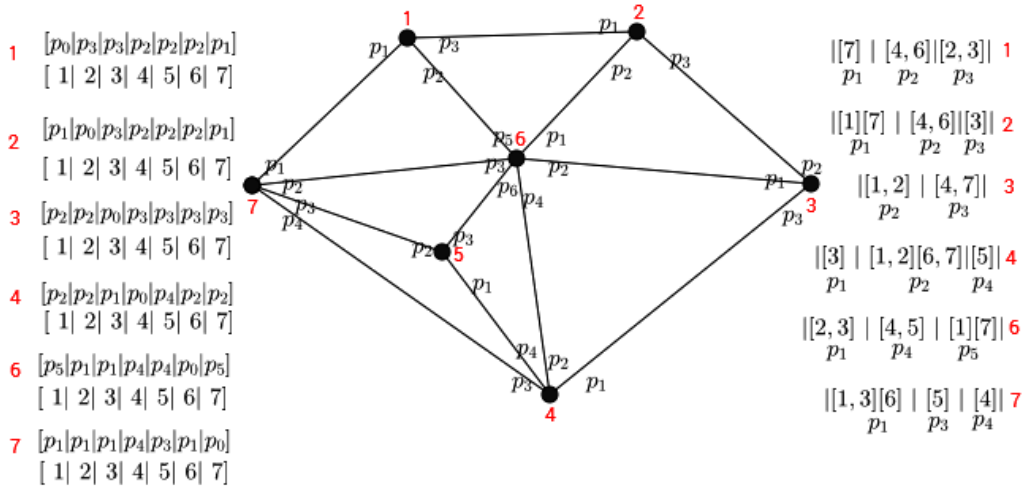


Figure 3.1: Graphe G=(V,E) et Principe du routage par intervalle.

Voyons maintenant comment un message d'adresse de destination 3, arrivant au nœud d'identifiant 7 est-il acheminé. Au nœud 7, l'algorithme de routage récupère l'adresse de destination dans l'en-tête du message et s'assure que celle-ci est bien différente de l'identifiant du nœud courant ($3 \neq 7$), Puis il cherchera à trouver, dans la table de routage du nœud courant, l'arête étiquettée par

l'intervalle contenant l'identifiant 3. L'intervalle $[2, 4]$ est alors choisi car celui-ci est tel que $2 \leq 3 \leq 4$. Le message est donc retransmis par cette arête et va atteindre le nœud d'identifiant 1. L'algorithme de routage vérifie que $3 \neq 1$ et que 3 n'est pas élément des intervalles [7] et [2] mais plutôt élément de $[3, 6]$ avec l'inégalité $3 \leq 3 \leq 6$ et, le message est donc retransmis via l'arête portant l'étiquette $[3, 6]$. Au nœud 1 le même processus est répété et le message est retransmis vers le nœud 6, puis vers le 4, car bien que le nœud 3 soit adjacent au 6, l'arête reliant ces deux sommets n'est pas un plus court chemin entre 6 et 3 dans le sens où le coût de l'arête les reliant est de 5 tandis que le total des coûts des arêtes $(6, 4)$ et $(4, 3)$ n'est que de 4, donc l'arête utilisée est celle reliant les nœuds 6 et 4. Au nœud 4 l'algorithme vérifie que l'arête d'étiquette [3] retransmet bien le message au nœud 3, celui-ci s'arrête en étant sûr que le message est bien à sa destination. Le routage par intervalle s'est bien éxecuté suivant un plus court chemin avec un chemin de longueur 8. Cette valeur est la plus petite des longueurs parmis toutes celles reliant les nœud 7 et 3. En effet, il est possible dans le routage par intervalle qu'il y ait un nœud tel que l'ensemble des destinations pouvant être atteint depuis celui-ci soit de ports de sorties différents, par conséquent chaque arête de ce nœuds est étiquetté par un intervalle contenant un seul élément, dans ce cas on se retrouve avec un intervalle d'un entier dans chaque case de la table. Si toutes les arêtes d'un réseau peuvent être étiquetée par au plus 1 intervalle, on parle alors d'un schéma de routage par 1-intervalle(ou *1-interval routing schema*).

Comme introduit dans [1] par N. Santoro, R. Khatib, le schéma de routage par intervalle nécessite que chaque ensemble de destination soit représenté en un seul intervalle. Une étude générale sur le routage par intervalle a été réalisée par Cyril Gavoille [2]. Il donne une définition formelle du routage par intervalle et la présentation de quelques notions de base. Il est cependant important de préciser que le routage par intervalle ne marche pas souvent pour tous les types de graphe, et d'ailleurs Gavoille dans [2] donne des familles de graphes qui admettent ce type de routage. Dans son article étude sur le routage par intervalle, Gavoille définit un schéma de routage par intervalle comme suit:

Définition 1 *Soit $G = (V, E)$ un graphe, une paire (L, S) est un schéma de routage par intervalle dans G si les conditions suivantes sont satisfaites:*

1. *L est une fonction de numérotation des sommets de $G : L : V \to \{1, ..., |V|\}$;*
2. *S est une fonction de numérotation des arêtes de G avec $S : E \to 2^{L(V)}$ tel que:*
   - *Pour tout x de V, $\{S(x, y)|(x, y) \in E\} \bigcup L(x) = \{1, \ldots, |V|\}$;*
   - *Pour tous arcs distinct (x, y) et $(x, z) \in E$, $S(x, y) \bigcap S(x, z) = \emptyset$;*
3. *Pour tout $x, y \in V$, $x \neq y$, il existe une séquence de nœuds $(u_0, ..., u_t)$ tel que $u_0 = x$ et $u_t = y$, et pour tout $i \in \{1, \ldots, t\}$, $L(y) \in S(u_{i-1}, u_i)$.*

L'auteur définit un schéma de routage par intervalle ou IRS(Interval Routing Schema) dans G comme une fonction de routage dans G qui retourne pour toute paire de nœuds source-destination (x, y), un chemin de x à y définit par la séquence de nœuds de la condition 3. Un tel chemin est appelé chemin de routage. Si la paire de fonctions (L, S) présentée par Gavoille satisfait les conditions 1 et 2 de la définition 1, alors elle est appelée un schéma de numérotation par intervalle, ou encore un ILS(Interval labeling schema). Notons qu'un ILS dans un graphe G n'est pas forcément un IRS dans G, par exemple dans le cas où le chemin entre une paire de nœuds contient une boucle, il est clair que la condition 3 de la définition 1 serait fausse. Un IRS dans un graphe G est un ILS valide, c'est un ILS qui vérifie la condition 3. Dans la problématique du routage par intervalle, l'auteur s'intéresse seulement aux ILS valident. Dans [3], J. van Leeuwen, R.B. Tan ont montré que la validité d'un ILS peut être vérifiée en temps $O(n^2)$ avec n le nombre de nœud du graphe.

3.2 Notions de bases sur le routage par intervalle

3.2.1 Compacité, linéarité et précision du routage par intervalle

Les notions de compacité, de linéarité et de précision sont des termes connexes au routage par intervalle. Les deux premières notions sont relatives à la façon de coder les étiquettes des arcs d'un graphe, alors que la notion de précision est relative à la construction du schéma de routage par intervalle (IRS) lui-même.

Définition 2 *Soit $R = (L, S)$ un IRS d'un graphe G, la compacité de R est le nombre d'intervalles maximum de $S(x, y)$ entre tous les arcs (x, y) de G.*

Les intervalles linéaires sont définis de façon similaire avec le nombre maximum d'intervalle linéaire parmi ceux de tous les arcs de G. La compacité et la compacité linéaire diffèrent par tout au plus 1: Un intervalle cyclique est égal à deux intervalles linéaires. Un k-IRS qui a aussi une compacité linéaire k est appelé k-linéaire schéma de routage par intervalle (k-$LIRS$). On dénote par k-IRS tout IRS de compacité k. De plus un IRS est qualifié de stricte, noté SIRS si tout arc (x, y) de G vérifie $L(x) \notin S(x, y)$.

3.2.2 Dilatation et facteur d'étirement

Bien que la compacité, la linéarité et la précision(strictness) soient des paramètres qualificatifs de l'étiquettage, il existe plusieurs autres critères pour mesurer la qualité du routage induite par le schéma de routage par intervalle. Ici le graphe considéré a des liens de poids uniformes.

Définition 3 *Soit R, un IRS d'un graphe G. La dilatation de R noté par dilatation(R), est la longueur du plus long chemin de routage induit par R. Un*

k-dilatation de G, noté k-dilatation(G), est le minimum, parmi tous les k-IRS R dans G, de la dilatation R.

Définition 4 *Soit R, un IRS d'un graphe G. Le facteur d'étirement de R, noté stretch(R), est le plus petit réel s tel que pour tout paire de noeuds (x, y), la longueur du chemin de routage induit par R de x à y est s fois plus grand que la distance dans G entre x et y. Le k-stretch de G, noté k-stretch(G), est le minimum, parmi tous les k-IRS R dans G, du facteur d'étirement de R.*

3.3 Etat de l'art sur le routage par intervalle

Le premier article présentant le routage par intervalle est celui de N. Santoro, R. Khatib [1]. Dans ce papier les auteurs ont montré qu'il est possible de construire un schéma de routage efficace pour des réseaux arbitraires. Ce mécanisme est basé sur une numérotation adéquate des nœuds du réseau, et en s'inspirant du Théorème 2 de Gavoille [2], ils ont montrés que tout graphe sans cycle admet un schéma de routage par intervalle avec 1 intervalle pour chaque entrée de la table de routage, ou encore 1-IRS (1-interval routing schema). Le schéma des auteurs est basé sur un algorithme de numérotation utilisant le parcours en profondeur (DFS) sur les arbres couvrant.

Théorème 1 *(C. Gavoille* [2]*) Tout arbre à n nœuds a un* 1-$SIRS$ *qui peut être implémenté avec* $O(\sqrt{n})$ *bits sur chaque nœud.*

Signalons que tout graphe peut avoir un schéma de routage par intervalle en numérotant arbitrairement les nœuds et en utilisant n'importe qu'elle fonction de routage avec des chemins simples et générant des étiquettes sur chaque arc. Mais comme le routage par intervalle a pour but de rendre plus compact les informations de routage, ce qui nous intéresse ici c'est de trouver un schéma de routage par intervalle avec une plus petite compacité possible.
Ainsi avec une numérotation des nœuds du graphe basée sur le parcours en profondeur (DFS), nous obtenons les résultats des auteurs suivants:

Théorème 2 *(Santoro et khatib* [1] *) Tout graphe acyclique a un* 1-$SIRS$.

Les arbres sont sans cycle donc admettent un 1-SIRS, et selon Gavoille [2], en tenant compte du théorème 1 et du théorème 2 ci-dessus (qui prend en compte que les graphes non orientés et sans cycles), et en utilisant la notion d'arbre couvrant, tout graphe a un 1-$SIRS$. Ce résultat de Gavoille a été d'ailleurs étendu par van Leeuwen et Tan [4].

Théorème 3 *(J. van Leeuwen et Tan* [4]*) tout graphe possède un* 1-$SIRS$ *telle qu'aucun des ces arcs n'aient une étiquette vide.*

Fraigniaud et Gavoille dans [5] se sont aussi prononcé sur les graphes ayant un 1-$LIRS$.

Théorème 4 *(Fraigniaud et Gavoille* [5]*)*

- *Un graphe à un* 1-$LIRS$ *si et seulement si il n'est pas un graphe lithium.*
- *Un graphe a un* 1-$SLIRS$ *si et seulement si il n'est pas un faible graphe lithium.*

Il dérive de cette article ([5]) un algorithme en temps $O(n^2)$ pour étiqueter les graphes supportant un 1-$LIRS$ et 1-$SLIRS$. Un simple algorithme basé sur le DFS est aussi mentionné dans [6].
Notons que parmi tous les algorithmes listés ci-dessus, aucun ne mentionne la longueur du chemin de routage.
Gavoille [2] définit la notion de plus court chemin comme suit:

Définition 1 *Soit R un IRS d'un graphe G. R est un IRS de plus court chemin si les chemins de routage induit par R dans G sont seulement composés de plus court chemin dans G.*

Gavoille ajoute alors qu'un IRS de plus court chemin est un IRS de facteur d'étirement 1.
La notion de plus court chemin sera étudiée plus loin dans les prochains chapitres.

Comme vu plus haut dans ce papier, le but premier du routage par intervalle est de rendre les informations de routage plus compactes. Des études à cet égard ont été réalisées en se basant sur la compacité du schéma de routage par intervalle.
Il existe des preuves constructives sur une compacité au pire des cas d'au plus $n/12$ [7], avec n le nombre de nœuds du graphe. Une autre preuve utilisant une borne minimale de $\Theta(n)$ [8] montre que $\Theta(n)$ intervalles pourraient être nécessaires simultanément sur $\Theta(n)$ arêtes.
Parallèlement à Kranakis, D. Krizanc [8], Gavoille et Perennnes [9] donnent le théorème suivant dans [2]

Théorème 5 *Pour tout n assez grand, il existe un graphe cubique G avec au plus n nœuds tels que $IRS(G) = \Theta(n)$. De plus $\Theta(n)$ intervalle nécessite $\Theta(n)$ arêtes.*

Ainsi en se basant sur le théorème 5, Gavoille dans [2] affirme qu'il existe des graphes avec $(1+\epsilon)n$ arêtes pour tout $0 \leq \epsilon \leq 1/2$, et de compacité $\Theta(\epsilon n)$. Il poursuit qu'ainsi, il suffit de choisir un graphe qui satisfait le théorème 5 avec un nombre de nœuds de $2\epsilon n$ et de connecter $(1-2\epsilon)n$ nœuds de degré 1 à quelques nœuds de ce graphe. Il devient claire que ce nouveau graphe sera de n nœuds, $3\epsilon n + (1–2\epsilon)n = (1+\epsilon)n$ arêtes et de compacité $\Theta(\epsilon n)$.

Dans le [10], il apparait la construction d'un schéma de routage à n nœuds de degré borné par d, avec $3 \leq d \leq \epsilon n$ pour $\epsilon < 1$, de mémoire locale $\Theta(n \log d)$.

Nous listons ici quelques résultats basés sur l'architecture des graphes utilisés dans le routage par intervalle.

Familles de graphes	**Résultats**	**Auteurs**
Trees	$SIRS = 1$	Santoro et Khatib [1]
Rings	$SIRS = 1$	Santoro et Khatib [1]
Paths	$SLIRS = 1$	van Leeuwen et Tan [4]
Complete graphs	$SLIRS = 1$	van Leeuwen et Tan [4]
Complete bipartite graphs	$SLIRS = 1$	van Leeuwen et Tan [4]
Butterfly	$IRS = \Omega(\sqrt{n/\log n})$	R. Kralovic et al. [12]
Star graph	$SLIRS = \Omega(n(\log\log n/\log n)^5)$	R. Kralovic et al. [12]
Generalized hypercubes	$SLIRS = 1$	Kranakis et al. [11] et Fraigniaud et Gavoille [5]

Greg N. Frederickson et Ravi Janardan dans [15], affirment que la numérotation des nœuds d'un réseau permet à chaque arête de chaque nœud d'être étiqueter avec zéro ou plusieurs intervalles d'entiers représentant les identifiants de tous les nœuds atteignable via un plus court chemin à partir de cette arête. En partant de la classe des réseaux planaires extérieurs, les auteurs établissent une hiérarchie naturelle des réseaux basée sur le nombre d'intervalles nécessaires pour étiqueter chaque arête. Ils ont montré que les réseaux planaires extérieurs sont précisément des réseaux qui nécessitent juste un intervalle par arc. Les auteurs ont aussi donné un algorithme optimal qui sert à étiqueter les arêtes d'un réseau planaire extérieur. Dans le schéma des auteurs, les nœuds sont numérotés de 1 à n et les terminaisons de chaque arête $\{v, w\}$ incident à tout sommet v, par un sous-intervalle de l'intervalle $[1, n]$. L'intervalle représente l'ensemble $R_{v,w}$ des nœuds tels qu'il ait un plus court chemin entre v et chaque sommet dans $R_{v,w}$. Les auteurs ont proposé un schéma de numérotation par intervalle pour les graphes planaires extérieurs ayant des arêtes de poids arbitrairement positifs. Ce schéma stocke juste d entrées d'informations de routage sur chaque sommet v, de degré d du réseau. Ainsi $\Theta(n)$ entrées d'informations de routage sont stockées au totale. Pour des arêtes de coûts positifs, les auteurs ont montré que les graphes planaires sont précisément les graphes qui admettent une propriété d'intervalle. De plus, les auteurs ont donné un algorithme optimal

pour déterminer les étiquettes des arêtes. Dans l'étiquetage des arêtes du graphe planaire extérieur, les auteurs ont procédé comme suit; pour tout sommet v de degré d, soit w_1, w_2, ..., w_d des voisins de v indexés suivant le sens des aiguilles d'une montre et bordant la face extérieur du graphe en partant de v. Chaque arête incidente à v est étiqueté par un intervalle. Les intervalles de toutes les arêtes incidentes à v forment une partition de l'intervalle $[1, n] - v$. Le recouvrement d'intervalle est autorisé. Par exemple, l'intervalle $[i, j), i > j$, contient $i, i + 1, \ldots, n, 1, \ldots, j - 1$. En supposant que l'on étiquette l'arête $\{v, w_i\}$ par l'intervalle $[l_i, l_{i+1})$. Les valeurs $l_i, i = 1, 2, \ldots, d$, sont stockées dans une table au sommet v, chacune avec un pointeur pour associer l'arête $\{v, w\}$. Quand le message arrive au sommet v, si sa destination u n'est pas égale à v alors dans la table est recherché l'entrée l_i tel que $l_i \leq u \leq l_{i+1}$. Le message est envoyé suivant l'arête $\{v, w_i\}$. Et puisque les valeurs l_i, $i = 1, 2, \ldots, d + 1$ forment une liste circulaire voir [16], [17], l'entrée peut être recherchée en temps $O(\log d)$ en utilisant une recherche binaire modifiée. Dans la numérotation des sommets du graphe planaire extérieur, les auteurs ont assigné des entiers consécutifs de 1 à n aux sommets du graphe en choisissant comme début un sommet arbitraire et en se déplaçant dans le sens des aiguilles d'une montre suivant la bordure extérieure du graphe. Si un sommet v est visité plus d'une foi dans ce déplacement, ceci implique que v est un point d'articulation du graphe G. Ils appellent cette numérotation des sommets une numérotation dans le sens des aiguilles d'une montre. En anglais *a clockwise node naming*. Un graphe planaire avec une numérotation dit du clockwise est présenté sur le graphe à gauche de la figure 2.2. Comme montrés par les auteurs, pour toute affectation de poids sur les arêtes du graphe, chaque terminaison de celles-ci peut être étiquetée par un intervalle tel que tout message soit acheminé suivant un plus court chemin. Un tel label des arêtes est fait sur le schéma du graphe planaire à gauche présenté dans celui de droite de la figure 2.2.

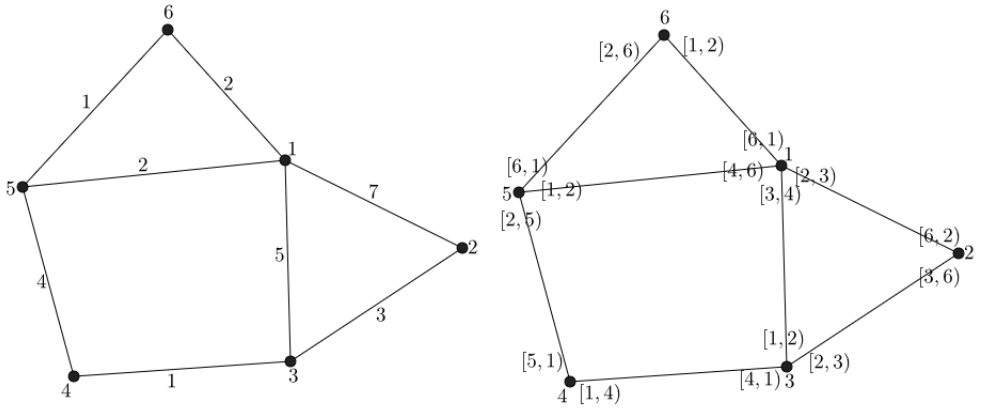


Figure 3.2:
Gauche: Graphe planaire extérieur et numérotation de ses sommets basée sur le clockwise.
Droite: Etiquetage des arêtes du graphe planaire extérieur (Gauche) par intervalle de plus court chemin.

Dans la section 4 du [15], les auteurs définissent les graphes admettant la propriété de 1-intervalle. Il s'agit de caractériser la classe de graphes pour laquelle il existe un étiquetage par intervalle des arêtes($LIRS$). Ils considèrent deux critères pour identifier cette classe de graphes. D'abord les sommets du graphe doivent être numérotés convenablement quelque soit la numérotation choisie. Ensuite, les poids des arêtes du graphe sont supposés fixes avant la numérotation des sommets de celui-ci.

Les auteurs affirment qu'un graphe à n sommets a une propriété de 1-intervalle s'il existe une numérotation de ces sommets avec des entiers de 1 à n tel que pour toute affectation des poids sur ses arêtes, en chaque sommet, toute terminaison d'arête puisse être étiquetée par un sous-intervalle de $[1, n]$ tel que l'arête en question soit la première arête sur un plus court chemin, du sommet courant vers n'importe quel autre sommet du sous-intervalle. Pour caractériser les graphes ayant cette propriété, les auteurs montrent le théorème suivant.

Théorème 6 *Un graphe a une propriété de* 1*-intervalle (*1*-interval property) si et seulement s'il est planaire extérieur.*

Lorsqu'il est nécessaire d'étiqueter par au plus k intervalle par arête, on parle d'une propriété de k-intervalle. Un 2-intevalle schéma de routage est donné dans [18] pour les grilles à coût uniforme. Et un schéma de routage par k-intervalle est aussi présenté dans ce même papier ou chaque arête est étiquetée par au plus k intervalles.

Si la numérotation des sommets peut être choisie après qu'on ait connaissance des coûts des arêtes, alors la classe de graphe pour laquelle chaque terminaison d'arête pouvait être étiquetée par un seul intervalle serait légèrement plus grande que la classe des graphes planaires extérieurs. Les auteurs précisent qu'un graphe a une propriété de 1-intervalle léger (*weak* 1-*interval property*) si pour tout poids de chaque arête, il y ait un identifiant à son sommet tel que chaque terminaison d'arête de tout sommet puisse être étiquetée par un intervalle codant un plus court chemin.

Théorème 7 *Un graphe a une propriété de* 1-*intervalle léger (weak* 1-*interval property) si et seulement si ces composantes bi-connexes sont soit planaire extérieur ou* K_4.

Selon les auteurs, la propriété qui permet à tout graphe planaire extérieur G de supporter le routage par intervalle est qu'il existe une disposition des sommets tels que tous les sommets soient sur la bordure d'une même face. En tout sommet v de G, toute arête $\{v, w\}$ étiquetée par un intervlle I, exige un ensemble de sommets consécutif sur cette face.

Si un graphe planaire G à une disposition dans laquelle deux faces contiennent tous les sommets, alors pour tout sommet v' de G, toute arête $\{v', w'\}$ serait étiquetée par un ensemble de sommet pouvant être subdivisé en deux sous ensembles dont chaque sous ensemble représente les sommets de chaque face. Il faut d'abord numéroter les sommets autour d'une face, et puis autour de l'autre face. Chaque arête exigera au plus un intervalle sur chaque face, sauf que pour chaque face il peut y avoir d'autres arêtes qui exigent des sommets d'identifiant plus petit et plus grand. Ces numéros ne seront pas consécutifs, même avec une couverture d'intervalle. Et donc ces arêtes nécessitent deux intervalles. C'est l'exemple montré sur la figure 2.3(a), dans laquelle les arêtes $\{1, 6\}, \{4, 5\}, \{5, 12\}, \{12, 13\}, \{13, 7\}, \{7, 8\}, \{3, 9\}, \{9, 10\}$, et $\{10, 11\}$

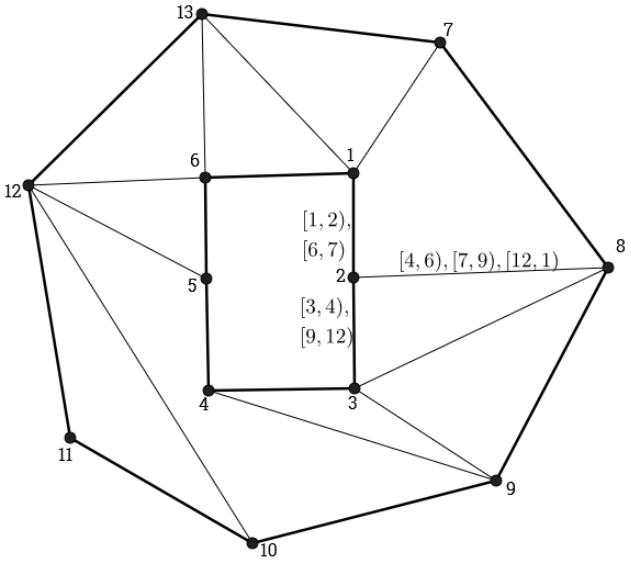


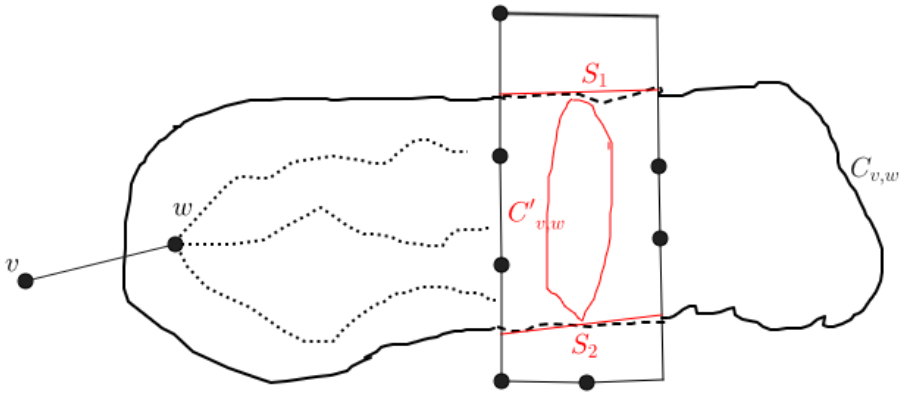
Figure 3.3: Un graphe avec une couverture de face de deux faces avec des labels assignés aux terminaisons des arêtes au noeud 2.

ont un coût de 1, et toutes les autres arêtes ont un coût de 6. L'étiquetage d'arête est donné pour des arêtes incidentes au sommet 2, ou par exemple l'arête $\{2, 8\}$ incidente au sommet 2 est étiquetée avec trois intervalles. Ainsi il est possible de générer un étiquetage multi-intervalle pour les arêtes, dans un graphe dans lequel il y a juste quelques faces qui contiennent tous les sommets. Les auteurs exploitent maintenant le potentiel des graphes admettant des faces qui n'ont pas de sommet en commun. Pour cela ils définissent un graphe $G = (V, E)$ non orienté avec des arêtes de poids positives. Soit $R_{v;w}$ l'ensemble des sommets pour lesquels il existe un plus court chemin depuis v contenant l'arête $\{v, w\}$. Les auteurs supposent que G est planaire. Ils considèrent un plongement $G_e = (V, E, F)$ de G, ou F est l'ensemble des faces. Soit $C_{v,w}$, une courbe simple fermée sur le plan qui sépare $R_{v;w}$ et G-$R_{v;w}$ et qui ne coupe aucune arête à deux reprise. Soit un segment de $C_{v,w}$ étant une portion maximale continue de $C_{v,w}$ contenu sur une face. Les auteurs montrent alors par le lemme 1 que G contient au plus un segment de $C_{v,w}$.

Lemme 1 *Au plus un segment de $C_{v,w}$ est contenu dans n'importe laquelle des faces de G.*

Le lemme 1 est en fait démontrer de la façon suivante: Puisque G est connexe et v est dans G–$R_{v,w}$,alors G–$R_{v,w}$ est connexe. Et comme $C_{v,w}$ est construit de telle sorte qu'elle ne croise aucune arête deux fois, donc toute arête croisée par $C_{v,w}$ contient une terminaison dans $R_{v,w}$ et l'autre pas. Supposons qu'il ait une face f dans G_e telle que deux segments continus S_1 et S_2 de $C_{v,w}$ soient

contenus dans celle-ci, il est alors possible de tracer une courbe fermée $C'_{v,w}$ dans cette face telle que $C'_{v,w}$ passe par le point intérieur de S_1 et par le point intérieur de S_2 telle que $C'_{v,w}$ soit contenu dans f. Ce qui signifie que $C'_{v,w}$ ne coupe aucune arête de $R_{v,w}$. Alors dans ce cas $R_{v,w}$ n'est pas connexe car il n'y a aucun sous-ensembles vides de $R_{v,w}$ dans les régions créées par $C'_{v,w}$ dans $C_{v,w}$. Et si $R_{v,w}$ n'est pas connexe alors dans ce cas G aussi ne l'est pas puisque $R_{v,w}$ est un sous graphe de G, ce qui contredit la propriété de connexité de G. Le schéma 2.4 illustre les explication ci-dessus.



.

Figure 3.4:

Pour calculer le nombre d'intervalle nécessaire pour étiqueter n'importe quelle arête du graphe planaire G, les auteurs introduisent les notions suivantes:

`Face triviale`: C'est une face dont la bordure comprend un seul sommet. Ainsi les auteurs incluent la possibilité d'avoir zéro ou plusieurs sommets, et permettent à chacun d'eux de créer un intervalle soit même.
`Une couverture de face (face covering):` une couverture de face F' des sommets de G_e est un ensemble de faces, tel que tout sommet dans G est dans l'une des faces de F'. Une couverture de face F' est disjoint si aucun sommet n'est partagé par deux faces. L'assignation des identifiants des nœuds est faite par des entiers consécutifs bordant chaque face.

Théorème 8 *Soit G un plongement planaire avec une couverture de face disjoint de p faces non triviales et q faces triviales. Le nombre d'intervalle nécessaire pour étiqueter n'importe quelle arête est au plus* $[(3p+q)/2]$.

Pour comprendre ce théorème, il est important de comprendre l'énoncé du lemme 1: Soit une face triviale f avec les sommets i_f,$i_{f+1}, \ldots, j_f$. Puisqu'il est montré au lemme 1 que f ne pouvait contenir qu'un seul segment de $C_{v,w}$,

alors $R_{v,w}$ ne contient soit aucun sommet de f, soit contient les sommets de l'intervalle $[i', j']$ de f, soit contient les sommets des deux intervalles $[i_f, i']$ et $[j', j_f]$ avec $i_f <= j'$ et $i' + 1 < j' <= j$. Le schéma 2.5 suivant montre les trois cas possibles.

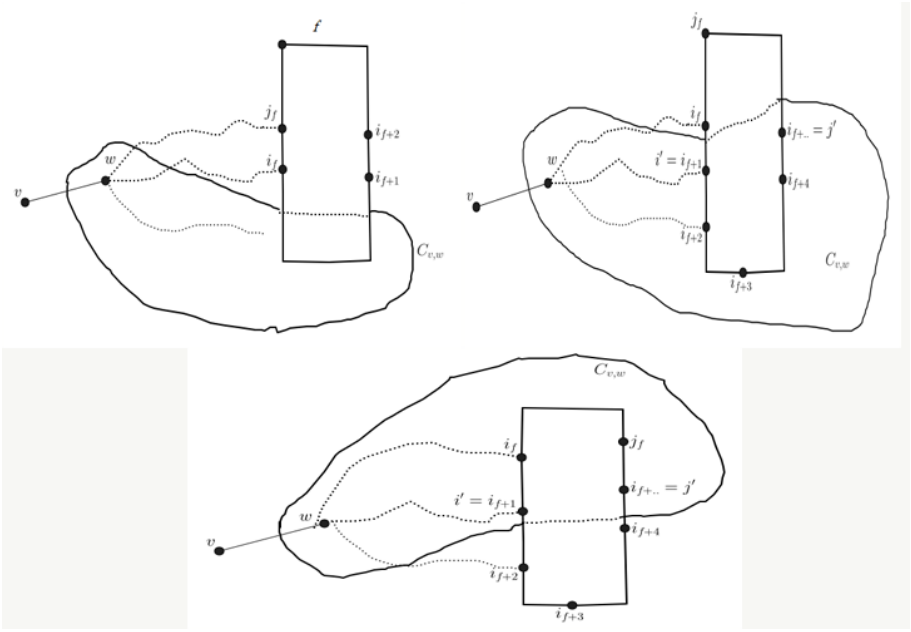


Figure 3.5:

Représentons $R'_{v,w}$ sur les indices de 1 à $3p + q$ en fonction de $R_{v,w}$ qui est représenté sur les indices de 1 à n. Chacune des faces non triviale f avec $f = 1, 2, \ldots, p$ est contractée en face triangulaire dont les sommets sont numérotés sur les indices $3f - 2$, $3f - 1$ et $3f$.
Alors pour toute face f:
Si $f = 1$ on a les sommets $1, 2, 3$ pour la face f_1.
Si $f = 2$ on a les sommets $4, 5, 6$ pour la face f_2.
Si $f = 3$ on a les sommets $7, 8, 9$ pour la face f_3.
Ce qui signifie que toute face dans $R_{v,w}$ est représentée dans $R'_{v,w}$ par une face à trois sommets comme suit:

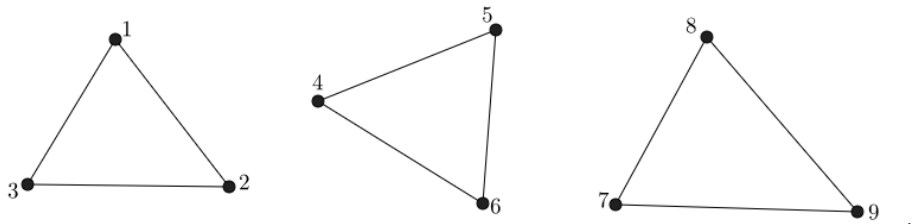


Figure 3.6:

$R'_{v,w}$ est construit sur les conditions suivantes:
$R'_{v,w}$ contient $3f-2$ si $R_{v,w}$ contient i_f.
$R'_{v,w}$ contient $3f$ si $R_{v,w}$ contient i_f.
$R'_{v,w}$ contient $3f-1$ si $R_{v,w} \cap [i_{f+1}, j_{f-1}]$.

En effet, le problème a été réduit pour faciliter l'étude. Toute face dans $R_{v,w}$ est représenté dans $R'_{v,w}$ en une face de trois sommets comme montrer sur la figure 2.6. Puisque l'on a p faces triviales de trois sommets chacune alors il est facile de voir, au pire des cas, que chaque sommet peut représenter un intervalle. Et comme on a p faces triviales de trois sommets chacune, on peut avoir au plus $3p$ intervalles consécutifs dès l'instant où les sommets des p faces sont ordonnés. On sait que pour des entiers de $1, 2, ..., 10$, le nombre d'intervalle non adjacent est d'au plus $10/2$, alors pour les $3p$ sommets des f faces non triviales dans $R'_{v,w}$ le nombre d'intervalles non adjacents est d'au plus $3p/2$. Avec les q faces triviales dans $R'_{v,w}$, il est donc possible d'avoir avec ces q faces au plus $q/2$ intervalles non adjacents. Alors pour p faces non triviales et q faces triviale, le nombre d'intervalle nécessaire pour numéroter toute arête de G_e est d'au plus $3p/2 + q/2$ qui nous fait $(3p+q)/2$, ce qui vérifie le théorème 8.

Pour borner le nombre d'intervalle nécessaire pour étiqueter toute arête incidente au sommet v de degré d, les auteurs définissent un nouveau type de graphe appelé graphe mimicking. Le graphe mimicking est défini de la façon suivante et permet de déterminer le nombre $r(d)$ de croisement que fait C_i avec les bordures des faces contenu dans la couverture faciale F'. Et à partir de ce nombre, on peut déduire le nombre d'intervalle nécessaire pour étiqueter tout sommet v de degré d du graphe G_e. Soit v un sommet de G_e, Il s'agit de générer à partir du plongement planaire G_e, de la couverture faciale F_v de G_e et de la courbe fermé C_i, un autre plongement planaire G_e' appelé graphe mimicking avec des propriétés suivantes: On définit un point de croisement comme un point où la courbe simple C_v croise une bordure d'une face non triviale f dans la couverture faciale F_i. Un sommet v est appelé point de croisement si v coïncide avec un point de croisement x entre la C_v et une face f de F_i dans G_e. Si v est un sommet d'une face triviale dans G_e, alors v est ajouté à G_e' et $\delta = 1$, sinon $\delta = 0$. Avec ce nouveau graphe, les auteurs ont pu déterminer la valeur de $r(d)$;

$r(d) = 2p + d - 2$.

Lemme 1 *Soit v un sommet de degré d. Le nombre de croisement que fait C_i avec l'ensemble des p faces disjoint non triviales d'une couverture faciale F' dans G_e est d'au plus $r(d) = 2p + q - 2$.*

En effet, d'après le lemme 1, le nombre de croisement entre les bordures des faces triviales et C_i est de $r(d)$. Chaque paire de croisements consécutifs d'une face peut être représentée en un intervalle et au pire des cas en deux intervalles. Il y a aussi le fait que v coïncide avec un point de croisement ou que v soit le sommet d'une face triviale, dans ces deux cas, v est inclus dans G_e' comme mentionner plus haut. Et comme le nombre de sommet d'une face non triviale f est d'au plus trois(3), Ainsi il y a au plus $3p + d - 2$ intervalles induit par les faces non triviales. Ceux-ci ajoutés au q faces triviales correspondant au pire des cas à un nombre q d'intervalles, on obtient le résultat du théorème 9.

Théorème 9 *Soit G_e un plongement planaire avec une couverture de face disjoint de p faces non triviales et q faces triviales. Le nombre d'intervalles pour étiqueter toute terminaison d'arête d'un sommet de degré d est d'au plus $3p + q + d - 2$.*

Dans l'étude effectuée précédemment, la configuration du graphe initiale supposée est celle dans laquelle les faces sont disjointes, aucun contact entre les faces du graphe. Cf figure 2.7 (a). Il serait aussi normal d'étudier le cas où les faces du plongement G_e sont telles qu'elles partagent un ou plusieurs sommets comme indiquer dans le graphe de la figure 2.7 (b).

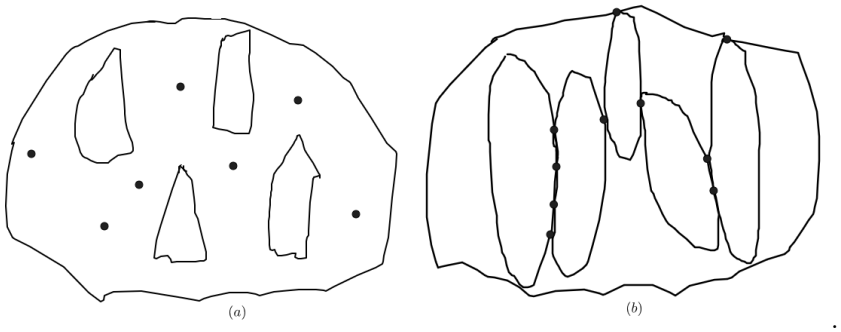


Figure 3.7: (a):Plongement planaire avec des faces disjointes, (b):Plongement planaire avec des faces liées par un ou plusieurs sommets

Pour ce dernier cas, l'étude des auteurs donne les résultats des théorèmes 10 et 11.

Théorème 10 *Soit G_e un plongement planaire avec une couverture faciale de p faces non triviales formant une s face composante, et q faces triviales. Le nombre d'intervalles nécessaire pour étiqueter n'importe quelle arête est d'au plus $[(3p + s + q)/2]$.*

Une s face composante est une composante dans laquelle l'ensemble des faces non triviale qui la constitue sont connectées entre eux par s sommets. Soit une composante c de p faces non triviales contenant les sommets i_c, $i_{c+1}, ..., j_c$, pour $c = 1, 2, ..., s$. Comme dans la preuve du théorème 9, représentons $R_{v,w}$ à l'image de $R'_{v,w}$ sur les indices de 1 à $2p + s + q$. Etant donné qu'il y aura $2p + s + q$ sommets dans $R'_{v,w}$ d'où un même nombre d'intervalles consécutifs, car chacun de ces sommets peut au pire des cas représenter un intervalle. Et donc $(2p + s + q)/2$ intervalles non adjacents pour les $2p + s + q$ sommets.

Théorème 11 *Soit G_e un plongement planaire avec une couverture faciale de p faces non triviales formant une s face composante, et q faces triviales. Le nombre d'intervalles nécessaire pour étiqueter toutes terminaisons d'arête d'un sommet v de degré d est d'au plus $2p + s + q + d - 2$.*

Les auteurs ont étendu leur résultat sur l'étiquetage des arêtes basées sur le genre du graphe. Le genre d'une surface est le nombre maximum de courbes fermées simples disjointes pouvant être tracées à l'intérieur de cette surface sans la déconnecter, autrement dit la réunion de ces courbes reste connexe. Dans cette étude les auteurs ont pu montré que le nombre d'intervalle nécessaire pour étiqueter une arête d'un prolongement augmente par 2 pour chaque augmentation de 1 sur le genre de surface dans laquelle il est plongé.

Pour le cas des graphes planaires étudiés précédemment, ils sont plongés sur les surfaces de genre 0. Les auteurs ont trouvé une généralisation du nombre d'intervalle nécessaire pour étiqueter toute arête d'un graphe G en fonction du genre de sa surface.

Le théorème suivant illustre les résultats obtenus par les auteurs dans l'extention de leur recherche.

Théorème 12 *Soit G_e un plongement planaire de G sur une surface de genre g. Soit une couverture faciale de G_e contenant p face non triviales, formant une s face composante et q faces triviales. Le nombre d'intervalles étiquetant toute terminaison d'arête en un sommet v de degré d est d'au plus $[(2p+s+q+4g)/2]$.*

La question qui nous vient à l'esprit est, est-ce qu'une étude plus fine sur l'étiquetage des arêtes basées sur les surfaces de genre 1 ou 2 ne donnerait-il pas de résultats plus significatifs.

La liste n'est pas exhaustive, il faut surtout comprendre que l'étude sur le routage par intervalle est ouverte. Beaucoup de questions restent sans réponse et peuvent faire l'objet d'autres études approfondies dans le domaine du routage compact, que ce soit de plus court chemin ou le routage compact dans d'autres types de graphes(planaires, planaires extérieurs, les constellations,...).

Pour conclure, nous pensons que le routage par intervalle est certainement un modèle de routage suffisant pour l'étude théorique de la structure et de l'information implicite des réseaux distribués. Cette méthode de routage est implémentée en pratique dans les routeurs INMOS C104 [13] et dans les routeurs RCube [14].

Questions Ouvertes:

- Quel est la borne générale pour un k-étirement(G), avec $k > 1$?
- Quelles sont les limites sur la dilatation moyenne et le facteur d'étirement moyen d'un k-IRS pour les graphes arbitraires?
- Est-ce que la caractérisation des graphes supportant des plus court chemins avec 1-$SLIRS$ (et ses variantes) est NP-complet si on se limite qu'aux graphes planaires ?
- Est-ce que le problème suivant reste NP-Difficile: Trouver le plus petit k telle que le k-$étirement(G)$, pour $s \geq 3/2$?
- Y a-t-il une limite pour l'espace mémoire dans les travaux de R. Kralovic et al. [12]:?
  $IRS(IRS(Butterfly) = \Omega(\sqrt{n/\log n})$
  $SLIRS(Stargraph) = \Omega(n(\log\log n/\log n)5)$
- Y a-t-il des anneaux cordeau de compacité $\Omega(\sqrt{n})$?
- Caractériser les anneaux cordeau de compacité k, avec un étiquetage cyclique, pour $k \geq 2$.
- Est-ce que tous les graphes à n nœuds ont une compacité proche de 1?

CHAPITRE 4

LE ROUTAGE COMPACT DE PLUS COURT CHEMIN

Les problèmes de cheminement sont des problèmes classiques de la théorie des graphes. L'objectif est de calculer une route entre des sommets d'un graphe qui minimise ou maximise une certaine fonction économique. Le problème le plus classique consiste à chercher le chemin qui minimise la somme des valuations des arêtes traversées. Il existe des algorithmes de complexité en temps polynomiale pour le résoudre, comme l'algorithme de Dijkstra. En théorie des graphes, l'algorithme de Dijkstra sert à résoudre le problème du plus court chemin. Il permet, par exemple, de déterminer un plus court chemin pour se rendre d'une ville à une autre connaissant le réseau routier d'une région. Plus précisément, il calcule des plus courts chemins à partir d'une source dans un graphe orienté pondéré par des réels positifs. On peut aussi l'utiliser pour calculer un plus court chemin entre un sommet source et un sommet destination.

4.1 Etat de l'art sur routage compact de plus courts chemins

De nombreuses études basées sur le routage compact de plus court chemin ont été faites. Parmi lesquelles on invoque celles de P. Fraigniaud et C. Gavoille [19]. Dans leur papier, les auteurs traitent du routage de plus court chemin dans les réseaux en arbre. L'idée est de router un message suivant un plus court chemin dans ce réseau en utilisant une structure de donnée compact distribuée. Les auteurs ont prouvé que les réseaux en arbre comportant n nœuds supportent des schémas de routage de taille O(log n) bits pour les en-têtes de message, l'espace mémoire et l'adressage des nœuds, avec un temps de calcul constant. Ce résultat améliore le meilleur schéma de routage connu($O(\log^2 n)$) d'un facteur de O(log n) en terme d'exigence mémoire et de temps.

Le problème est le suivant: Pour un graphe G donné et pour tout sommet x de G, il s'agit de trouver une fonction de calcul efficace qui détermine pour tout message entrant, l'arête de sortie par laquelle le message doit être retransmis. Le terme efficace regroupe un ensemble de facteurs désirables parmi lesquels, des routes de longueur raisonnable, un temps de calcul local faible, des structures de donnée compacte. Les auteurs insistent sur le fait que le résultat de la complexité dans un routage dépend fortement du modèle choisi. C'est-à-dire la possibilité de modifier les adresses des nœuds, la possibilité de modifier l'en-tête du message, ... Les auteurs signalent que les résultats diffèrent aussi selon les variations des paramètres tels que la taille des en-têtes, la longueur des routes, etc....

Dans leur étude, P. Fraigniaud et C. Gavoille considèrent un graphe $G = (V, E)$ connexe non orienté. Le schéma de routage utilise les adresses et un algorithme de routage distribué dont le rôle est de router les messages. Chaque sommet est identifié par une adresse unique dans le réseau. L'adresse d'un sommet u est noté $l(u)$ et l'identifiant d'un sommet u est noté $id(u)$. Les liens d'entrés et de sorties sur chaque nœud sont numérotés par des entiers de 1 à $deg(u)$, avec $deg(u)$ le degré du sommet u. Les points de terminaison de ses liens sont appelés port. Cette numérotation est locale et peut ne pas être identique entre les voisins; Une arête u, v peut être numéroté i du coté de u et j du coté de v. Tout nœud est supposé comporter un lien qui la relie à sa machine locale et le port relié à ce lien est numéroté 0. Le principe est le suivant: à la réception du message, le nœud courant peut modifier son en-tête. Cette modification dépend uniquement de l'adresse de destination et de la structure de donnée locale. Un schéma de routage dans lequel l'en-tête du message reste inchangé de la source à la destination est appelé direct. Dans leur papier, Fraigniaud et Gavoille traitent seulement du routage de plus court chemin et leur schéma de routage est direct. Autrement, le routage est basé uniquement sur l'adresse de destination et l'en-tête du message reste inchangé durant tout le trajet. Selon la numérotation des ports, les auteurs considèrent deux variations majeures: Celle où la numérotations des ports est fixée depuis la conception du routeur (Modèle de port fixe) et celle où le choix de la numérotation est laissé libre, c'est-à-dire le choix de la numérotation des ports est fait par le concepteur du schéma de routage.

L'un des schémas de routage le plus connu pour les graphes arbitraires est la table de routage. Il consiste à numéroter les sommets du graphe par des entiers de 1 à n et de stocker sur chaque sommet une table à n entrées telle que la i-ième entrée de la table retourne le numéro de port adéquat par lequel le message d'adresse de destination y doit être retransmis. La table de routage est direct et utilise $O(n \log d)$ bits en espace mémoire sur chaque sommet de degré d dans le modèle de port fixe et s'exécute en temps $O(1)$. Il est apparu dans les travaux de Gavoille et Pérénnès [5] que $\Omega(n \log d)$ est la meilleure borne connue pour le schéma de routage de plus court chemin utilisant des adresses de taille $c \log n$

bits pour toute constante $c \geq 1$. Selon ces résultats il est difficile d'exiger un plus petit espace mémoire pour les arbres. Dans les arbres, l'une des stratégies consistent à fixer les adresses par une numérotation suivant un DFS utilisant le routage par intervalle de SANTORO et KHATIB [1] et LEEUWEN et TAN [4]. Ces derniers considèrent pour tout nœud interne u de l'arbre, l'arrêt qui mène à un enfant v de u est associé à un intervalle contenant l'ensemble des adresses des nœuds du sous arbre enraciné en v et le lien qui mène à la racine est associé à un intervalle complémentaire. L'espace mémoire requis est de $O(n \log d)$ bits sur chaque nœud pour un temps de routage de $O(\log d)$ dans le modèle de port fixe. Ce coût total est faible comparer à celui de la table de routage qui est d'ordre $O(n \log n)$ bits. Dans leurs travaux, P. Fraigniaud et C. Gavoille ont proposé un schéma de routage direct de plus court chemin dans les arbres utilisant $5 \log n$ bits pour coder les adresses, $3 \log n$ bits pour l'espace mémoire et un temps de routage constante.

Le schéma de routage de Fraigniaud et Gavoilles avec $O(\log n)$ bits d'adresse est présenté de la façon suivante:

Soit T un arbre à n sommets arbitraire, et soit r un sommet quelconque de T, T est supposé enraciner en r. Pour tout nœud u de T, on note T_u le sous-arbre de T enraciné en u et induisant u et ses descendants. Soit $\omega(u)$, le nombre de sommet du sous-arbre T_u aussi appelé poids de u et $\omega_1(u)$, le poids du plus "lourd" enfant du sous-arbre T_u. Soit $id(u)$, la numérotation des sommets de T par des entiers consécutifs de l'intervalle $[1, n]$ obtenu par le DFS sur T. Si x est un sommet enfant du sommet y, alors le rang de y noté $rank(x, y)$ est définie comme le nombre d'enfant de x dont le label dans le DFS sont au plus $id(y)$. Pour tout couple de sommets u, v, la fonction port (u, v) est utilisée pour la numérotation de l'arête $\{u, v\}$ en u. Cette fonction est définie comme suit: Pour tout enfant v de la racine r, $port(r, v) = rank(r, v)$ et pour tout sommet $u \neq r$,

$$\text{port(u,v)} = \begin{cases} 1 & \text{si } v \text{ est parent de } u \\ 1+\text{rank(u,v)} & \text{sinon} \end{cases}$$

Pour tout nœud $u \neq r$, les auteurs définissent le $path(u)$ comme la séquence de rang rencontrée sur le chemin de r à u. Le *clean-path* de u noté $cpath(u)$ est définie comme la séquence obtenue après élimination des rangs du $path$ de valeur égale à 1. Ainsi les différents paramètres du schéma de routage des auteurs sont composés comme suit:
L'adresse $l(u)$ est composé de, $\quad l(u) = \langle id(u), cpath(u) \rangle$.

La structure de donnée stockée en chaque sommet u est:
$table(u) = \langle id(u), w(u), w1(u), |cpath(u)| \rangle$.

Un exemple de numérotation est présenté sur le schéma de la figure 3.1.

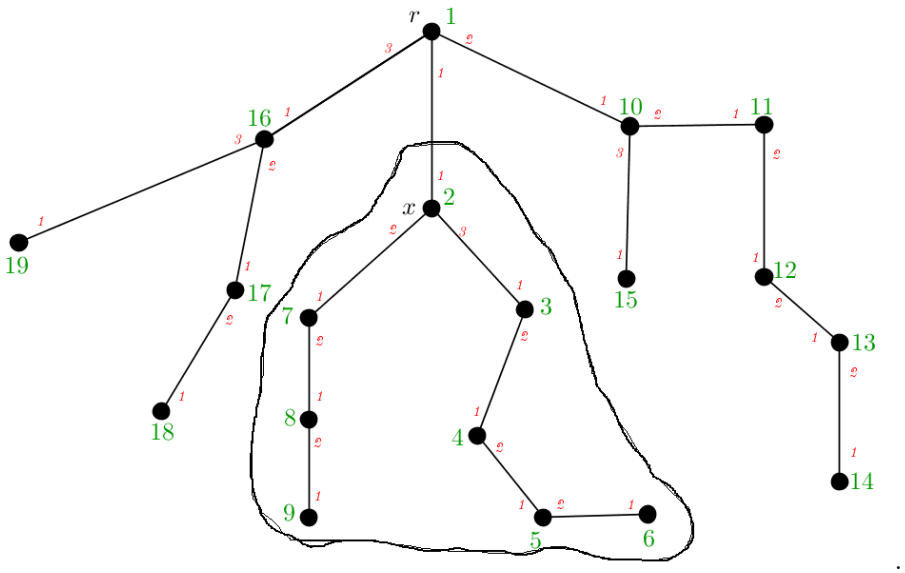


Figure 4.1: Arbre T enraciner en r. En vert numérotation suivant le DFS, en rouge numérotation des ports sur chaque sommet de T

Dans la nécessité de calculer la complexité du schéma de routage en terme d'espace mémoire, les auteurs proposent le lemme 2:

Lemme 2

- *Si u est le parent de v, alors* $rank(u, v) \leq w(u)/w(v)$
- *Si* $cpath(v) = (p_1, p_2, \ldots, p_k)$, *alors* $\prod_{i=1}^{k} p_i \leq n/(\omega(v))$,

  et $k \leq \log(n/\omega(v))$

Pour router les messages dans le réseau, les auteurs ont proposé un algorithme de routage direct, c'est-à-dire l'en-tête du message n'est autre que l'adresse de destination et cette adresse reste inchangée tout au long de la transmission du message. Soit deux sommets x et y de T supposons que le sommet x reçoit un message d'en-tête $h = l(y)$, la décision de routage est décrite par la fonction $ROUTE$ de la façon suivante.

Notons $ROUTE(x, l(y))$ par $f(x, l(y))$ ou tout simplement f.

$$f = \begin{cases} 0 & \text{si } id(y) = id(x) \\ 1 & \text{si } id(y) < id(x) \\ 1\text{+b} & \text{si } id(x) < id(y) \leq id(x) + \omega_1(x) \text{ ou } id(y) \geq id(x) + \omega(x) \\ \text{p+b} & \text{sinon,où } p = (|cpath(x)| + 1)\text{-}i\grave{e}me \text{ élément de } cpath(y) \end{cases}$$

Pour s'assurer que la fonction $ROUTE$ route tout message suivant un plus court chemin, les auteurs démontrent le lemme 3.

Lemme 3 *Pour toute paire de sommets x, y, l'algorithme de routage décrit par la fonction $ROUTE$ route n'importe quel message de x à y suivant un plus court chemin.*

En effet, à la réception d'un message par un sommet x de T, l'algorithme $ROUTE$ décrit trois possibilités de retransmission selon le contenu de l'en-tête du message.

Au sommet x la fonction de routage vérifie l'en-tête du message pour déterminer sa destination. Elle trouvera que le message est soit destiné à x lui même, soit destiné à ses parents, ou soit destiné à ses enfants. Si le message est destiné à x, alors il l'achemine suivant le port numéro 0, c'est-à-dire celui qui relie x à son réseau locale. Si le message n'est pas destiné à x, alors il est destiné à ses parents ou à ses enfants. Dans le cas où le message est destiné aux parents de x, il est alors acheminé suivant le port 1, le port de l'arête qui relie x à son père. Si les deux cas précèdents ne sont pas vérifiés, alors le message est destiné aux enfants de x et ce qui nous laisse deux possibilités: Soit le message est destiné à un sommet qui appartient au plus lourd fils de x ou à un sommet qui appartient aux autres fils de x. Ainsi, la fonction de routage retourne la réponse $1 + b$, c'est-à-dire le port numéroté 1 si $id(x) = 1$ et 2 sinon (cas où le sommet destination appartient au plus lourd fils de x). Sinon le message est destiné au $(i + 1)$-$i\grave{e}me$ élément de $cpath(y)$, avec i le nombre d'élément de $path(x)$. Donc pour acheminer le message, il est envoyé à un fils z de x tel que $path(z) = (path(x), rank(x, z))$. Et puisque y est un descendant de x et par conséquent $cpath(x)$ est un préfixe du $cpath(y)$ et la fonction de routage retourne la position $p + b$ de la destination. Avec $p = (|cpath(x)| + 1)$ et $b = 0$ si $id(x) = 1$(i.e $x = r$) et 1 sinon.

Afin d'implémenter l'espace mémoire nécessaire sur chaque sommet pour le stockage des informations de routage, les auteurs démontrent le lemme 4. Pour tout sommet u du graphe G, ce lemme est basé sur le calcul de l'espace mémoire de stockage de l'adresse de u noté $l(u)$ et de la table de routage du sommet u noté $table(u)$.

Lemme 4 *Pour tout sommet u, l'adresse de u noté $l(u)$ est de taille plus petit que $5 \log n + 1$ bits, et sa structure de donnée locale $table(u)$ est de taille plus petit que $3 \log + \log \log n + 4$ bits*

En effet, rappelons que $l(u) = \langle id(u), cpath(u)\rangle$. $id(u)$ est un entier donc il est représenté par une chaine binaire de taille $\lceil \log n \rceil$ qui est strictement inférieur à $(\log n) + 1$ bits. A l'aide de la preuve du lemme 2, il est facile de montrer que le $cpath(u) = (p1, \ldots, pk)$ peut être codé sur $4 \log n$ bits. Alors par conséquent le champ addresse de tout nœud u noté $l(u)$ peut donc être codé sur au moins $5 \log n + 1$ bits d'espace mémoire. Et pour finir, la table de routage stocker en tout nœud u noté $table(u) =< id(u), w(u), w_1(u), k >$ avec $k = |cpath(u)|$ peut être codé par une chaine binaire de longueur au plus $3\lceil \log n \rceil + \lceil \log k \rceil$. Cette chaine est visiblement plus petite que $(3 \log n+3)+(\log k+1)$ d'où plus petit que $3 \log n + \log k + 4$. Or, d'après le lemme 2, $k \leq \log n$ alors, la formule précédente devient $3 \log n+\log \log n+4$. Cette valeur est en bits l'espace mémoire nécessaire pour stocker la table de routage $table(u)$ sur tout nœud u de G. Ce qui termine l'implémentation du schéma de routage des auteurs.

Au même moment où P. Fraigniaud et C. Gavoille [19] traitent cette article, Thorup et al [21] indépendamment ont eux aussi étudié le routage de plus court chemin, leur recherche est basé sur la question suivante: Quels sont les familles de graphe qui supportent le routage de plus court chemin?. Dans leurs travaux, ils montrent comme dans [19] que tout arbre à n sommets possède un schéma de routage de plus court chemin utilisant des adresses, des en-têtes et des tables de routage de $O(\log n)$ bits. Y. Dieng et C. Gavoille [20] ont étendu les résultats obtenus par Freignaud et al. indépendamment Thorup et al. aux (k, r)-constellations. Ils ont d'abord montré que tout graphe planaire-extérieur connexe non valué à n sommets admet un schéma de routage de plus courts chemins avec des adresses, des en-têtes et des tables de routage de taille $O(\log n)$ bits. En se servant de leur résultat, ils ont par la suite généralisé ces résultats aux familles de graphe plus large appelé les (k, r)-constellations. Pour comprendre la définition d'une (k, r)-constellation, il est important de définir d'abord les notions de mineur d'un graphe, composante biconnexe et graphe planaire extérieur.

* On appelle mineur d'un graphe G, le graphe H obtenu en contractant des arêtes d'un sous graphe de G. Un mineur d'un graphe est dit $K_{x,y}$ s'il est composé de x sommets de degré y reliés à y autres sommets de degré x. Voir figure 3.2.

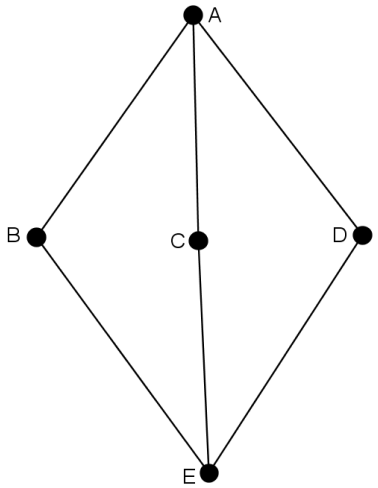


Figure 4.2: Exemple, mineur $K_{2,3}$

* Une composante bi-connexe est un ensemble maximal d'arêtes (non orientées) qui vérifie la propriété suivante:
  - Soient a et b deux arêtes distinctes, alors il existe un cycle simple (qui ne passe pas deux fois par le même sommet) qui passe par a et b.
* Un graphe planaire-extéreur est un graphe que l'on peut déssiner sur le plan de telle sorte que tous les sommets bordent la face extérieure.

Un graphe est une (k, r)-constellation s'il est sans mineur $K_{(2,r)}$ et si chacune de ses composants bi-connexes peut être rendue planaire-extérieur par la suppréssion de k sommets.

Dans un premier temps, Dieng et Gavoille ont montré que les graphes planaires extérieur tout comme les arbres présentés par (Fraigniaud et al.,2001;Thorup et al.,2001) supportent un schéma de routage de plus courts chemins. Les théorèmes 13 et 14 mettent en évidence les résultats de leurs travaux.

Théorème 13 *Tout graphe planaire-extérieur connexe non valué à n sommets admet un schéma de routage de plus courts chemins avec des adresses, des entêtes et des tables de routage de* $O(\log n)$ *bits. La latence est constante.*

Notons que si un graphe est planaire-extérieur, alors il vérifie le théorème 13. La validation du résultat du théorème 13 a permis aux auteurs d'étendre leurs études aux graphes (k, r)-constellations du fait que chacune des composantes bi-connexes d'une (k, r)-constellations peut être rendue planaire-extérieure par

la suppression de k sommets. Pour exemple, les graphes planaire-extérieurs sont les $(0,3)$-constellations, car ils sont sans mineur $K_{(2,3)}$. Le chiffre 0 indique qu'il n'est nécessaire de supprimer aucun sommet pour le rendre planaire-extérieur. Le graphe de la figure 3.3 est une $(2,6)$-constellation car il suffit de supprimer les sommets en bleu pour le rendre planaire-extérieur et, il est facile de voir qu'il est sans mineur $K_{(2,6)}$.

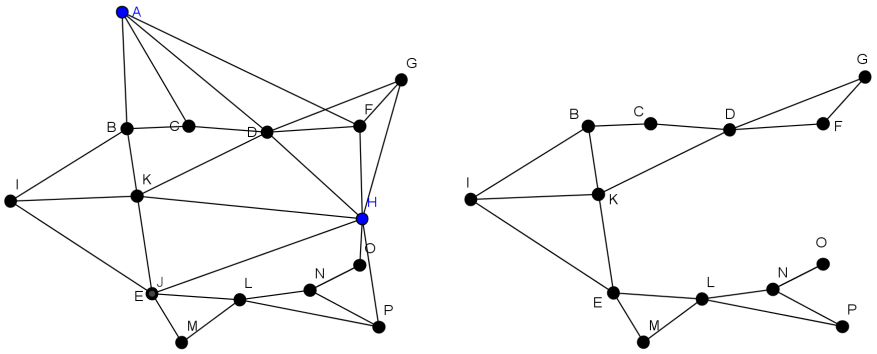


Figure 4.3: A gauche une $(2,6)$-constellation rendue planaire-extérieur après suppression des sommets en bleu.

Théorème 14 *Toute (k,r)-constellation connexe non valué à n sommets admet un schéma de routage de plus courts chemins avec des adresses et des en-têtes de $O((k+1)\log n)$ bits et des tables de routage de $O((kr+1)\log n)$ bits. La latence est de $O(\log k + \log r)$.*

En effet, pour construire un schéma de routage pour les planaire-extérieurs, les auteurs ont considéré un graphe planaire-extérieur G dont les sommets sont numérotés suivant un parcours bordant la face extérieure par des entiers de 1 à n. Ils ont par la suite considérer un arbre T couvrant G de plus courts chemins séparé en deux(2) parties; pour tout sommet x de T, τ_x est le sous-arbre de T enraciné en x et constitué des sommets descendants de x. Cette ensemble constitue la première partie. L'autre partie est l'ensemble des sommets de T non descendant de x couvert par un sous-arbre H. Pour ce faire, les auteurs affectent à chaque partie une des tables de routage **Table**-τ et **Table**-H comme suit:

Table-τ permet à tout sommet x de router vers toute destination y appartenant à ses descendants. La table **Table**-H permet à partir de x, d'atteindre toute destination $y \notin \tau_x$, c'est-à-dire n'étant pas un descendant de x. Alors pour envoyer un message, il suffit de trouver dans laquelle des parties se trouve le sommet destination et quelle table de routage utilisée.

Le schéma de routage de P. Fraigniaud et C. Gavoille est de plus courts chemins car les tables **Table**-τ et **Table**-H sont construites à partie des sous-arbres τ et H de l'arbre couvrant T de plus courts chemin dans G.

D'après Fraigniaud et Gavoille [19], l'espace mémoire nécessaire pour router dans un arbre est de $O(\log n)$ bits.

Pour faire le routage dans le graphe planaire-extérieur, Y. Dieng et C. Gavoille ont proposé l'algorithme $ROUTE$.

ROUTE de x vers y :

1. si $y \in \tau_x$, alors retourner **Table**-$\tau_x[y]$
2. sinon, retourner **Table**-$H_x[y]$

L'adresse de tout sommet x de G est constituée par son numéro issu de la numérotation dans G, et de la numérotation suivant un DFS sur τ. Donc l'adresse de chaque sommet est codée sur $2\log n + o(\log n)$ bits. Puisque les en-têtes des messages sont constituées de l'adresse du sommet de destination du message, elles sont alors de même taille que les adresses.

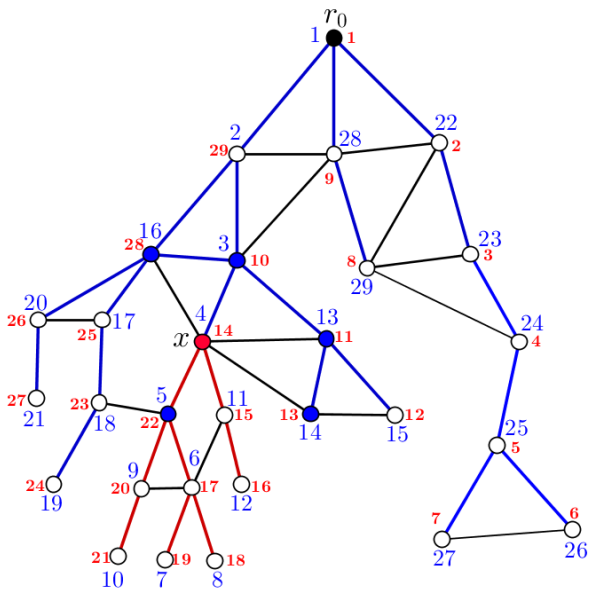


Figure 4.4: Plongement planaire-extérieur d'un graphe G avec un arbre couvrant τ enraciné en r_0 et un sommet x distingué avec son sous arbre τ_x. En bleu nous représentons les sommets de τ et en rouge ceux du plongement planaire-extérieur de G.

La généralisation du schéma de routage dans les graphes planaire-extérieur est très liée à celui des arbres car les arbres ne sont pas très éloignés des graphes planaire-extérieur. Ce schéma de routage est principalement basé sur le fait que les planaire-extérieur exclus les mineurs $K_{2,3}$.

Par ailleurs, on peut remarquer que le graphe de halin n'est rien d'autre qu'un arbre où on a relié les feuilles par un cycle. Ce qui veut dire en fait que les graphes de halin ne sont pas eux non plus très éloignés des arbres.

Rappelons que les graphes (k,r)-*constellations* sont définis comme étant des graphes sans mineur $K_{2,r}$ et que chacune de leurs composantes bi-connexes peuvent être rendue planaire-extérieur en supprimant au plus k sommets. Y. Dieng et C. Gavoille ont exploité cette propriété pour faire une étude plus générale dans les graphes (k,r)-*constellation* en donnant le théorème suivant:

Théorème 15 *Toute (k,r)-constellation connexe non valuée à n sommets admet un schéma de routage de plus courts chemins avec des adresses et des entêtes de $O((k+1)\log n)$ bits, et des tables de routage de $O((kr+1)\log n)$ bits. La latence est $O(logk + logr)$.*

Dans leurs schéma de routage, les auteurs considèrent une (k, r)-constellation G connexe à n sommets. Soit T un arbre couvrant G de plus courts chemin et une numérotation de ses sommets suivant un parcours du DFS.

Chacune des composantes bi-connexes de G est rendu planaire-extérieur par suppression d'un nombre k de sommets.

Ainsi pour router dans le graphe G, il suffit de construire une table Table1 de routage permettant de router entre deux sommets d'une même composante bi-connexe de G, une table Table2 pour router entre deux sommets de G n'étant pas dans la même composante bi-connexe et pour finir, une table Table3 permettant de router dans l'arbre couvrant de plus courts chemins T.

4.2 Etat de l'art sur routage compact de plus courts chemins avec facteur d'étirement

Un routage compact peut aussi ne pas être de plus court chemin, son efficacité est souvent mesuré en fonction d'un paramètre très important du schéma de routage appelé le facteur d'étirement. Un facteur d'étirement d'un schéma de routage est le ratio maximal entre le coût de la route utiliser par le schéma de routage pour router le message d'un sommet x à un sommet y sur celui du plus court chemin entre ces deux sommets. Autrement dit, c'est le rapport entre la longueur de la route utilisé par le schéma de routage sur celui du plus court chemin entre deux sommets donnés. Dans le papier [22], les auteurs A. Bar-NOY, N. LINIAL, B. AWERBUCH, D. PELEG présentent plusieurs familles de schéma de routage simple dont principalement deux que sont: Les hierarchical covering pivot (HCP_k) et les hierarchical balanced (HB_k) pour tout $k \geq 1$.

On a vu que pour envoyer un message entre deux sommets quelconque d'un graphe, plusieurs méthodes de routage sont possible, parmi lesquelles comme vu précédemment le routage direct et le routage par inondation. Le problème avec ces schémas de routage c'est que pour le routage direct chaque sommet doit contenir une large table de routage de taille $\Omega(n)$ bits. Et cette taille peut augmenter avec le nombre de machine du réseau. Pour le schéma de routage par inondation, l'un des inconvénients majeurs est que le coût de la communication est très élevé car, il utilise tous les liens du réseau et par conséquent le facteur d'étirement n'est pas borné. Maintenant, une question naturelle à se poser est comment concevoir un schéma de routage avec en même temps une faible nécessité en espace mémoire et un faible coût de communication.

Le problème est donc étudier en premier par L. KLEINROCK et F. Kamoun dans [24] en s'assurant que le réseau respecte certaines propriétés. Dans [23], D. Peleg and E. Upfal ont étudié le problème sur des réseaux généralisés. Le schéma de routage utilisé dans [23] garantit un facteur d'étirement de $12k + 3$

avec une nécessité mémoire d'au totale $O(n^{1+1/k})$ bits. Le schéma de routage présenté par D. Peleg and E. Upfal est déficitaire dans le sens où, entre autres, il ne marche qu'avec des arêtes de coût uniforme et qu'aussi, l'espace mémoire globale est connu mais pas celui de chaque sommet pris individuellement.

A. Bar-NOY et al. ont présenté dans [22] un schéma de routage qui tente de surmonter les problèmes précédent. Dans leur papier, Le schéma de routage HCP_k garantit un facteur d'étirement de $2k-1$ et nécessite un espace mémoire de $O(k.n^{1+1/k}\log^2 n)$ bits dans le réseau. Le schéma HB_k nécessite $O(k.n^{1+1/k}\log n)$ bits au totale avec un facteur d'étirement de $2.3k$–1.

Bien que le schéma HCP_k des auteurs soit moins efficace que celui dans [23], elle est plus performant pour des valeurs de k plus faible. Par exemple pour $k = 2$, ce schéma utilise $O(n^3/2\log^2 n)$ bits en espace mémoire avec un facteur d'étirement d'au plus 3 au lieu de 27 dans [23]. Par contre, le problème majeur de cette méthode est qu'il comporte des sommets qui nécessitent beaucoup plus d'espace mémoire que la normale (i.e. il n'est pas équilibré). Pour régler ce problème, les auteurs utilisent le schéma HB_k. Pour ce schéma, les auteurs ont proposé un algorithme efficace pour configurer les tables de sorte qu'elles utilisent la même borne en espace mémoire sur chacun des sommets du graphe.

Dans la réalisation de leur schéma de routage, B. Auwerbuch, A. Bar-noy, N. linial et Peleg se sont basés sur le schéma de routage de Peleg et Upfal dans [23]. Le principal composant du schéma de routage dans [23] est l'arbre d'intervalle.

Soit $G = (V, E)$ un graphe et r un sommet particulier de G, avec $r \in V$. Le schéma de routage dans l'arbre d'intervalle est noté $ITR^*(G, r)$ et est construit en quatre (4) étapes;

Il s'agit de construire un arbre T couvrant G de plus court chemin enraciné en r, puis numéroter les sommets de V suivant un DFS sur T. A chaque sommet v de T, associé un intervalle sous forme de paire d'entiers de sorte que ces entiers bornes, au minimum et au maximum, les numéros des sommets du sous-arbre enraciné en v. (voir figure 3.6)

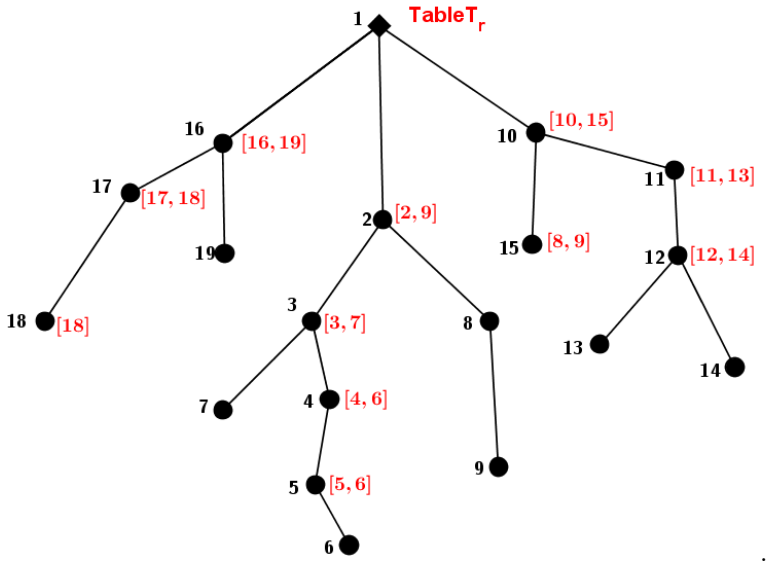


Figure 4.5: Présentation de la configuration d'un arbre d'intervalle $ITR*(G, r)$, avec la table de translation $TableT_r$ stoquée à la racine r.

Et à la racine r de l'arbre d'intervalle $ITR^*(G, r)$, stocker une table TableTr qui spécifie pour tout sommet v de V l'intervalle auquel il appartient dans l'arbre. Cette table est appelée table de translation.

Ainsi dans le schéma du [23], délivrer un message d'un sommet u à un sommet w, avec $u, w \in V$, revient à envoyer le message le plus haut possible dans l'arbre jusqu'à la racine r, et puis avec l'aide de la table de translation $TableT_r$, retransmettre le message vers l'intervalle $y = [int(a), int(b)]$ tel que $w \in y$.

La complexité de ce schéma de routage est d'au totale $O(|V| \log n)$ bits.

On constate que le schéma de routage à arbre d'intervalle de Upfal et Peleg est compact mais n'est pas un routage de plus court chemin puisqu'il faut toujours envoyer le message jusqu'à la racine pour ensuite le retransmettre au destinataire. Ce qui éclaire un peu l'intitulé de ce chapitre.

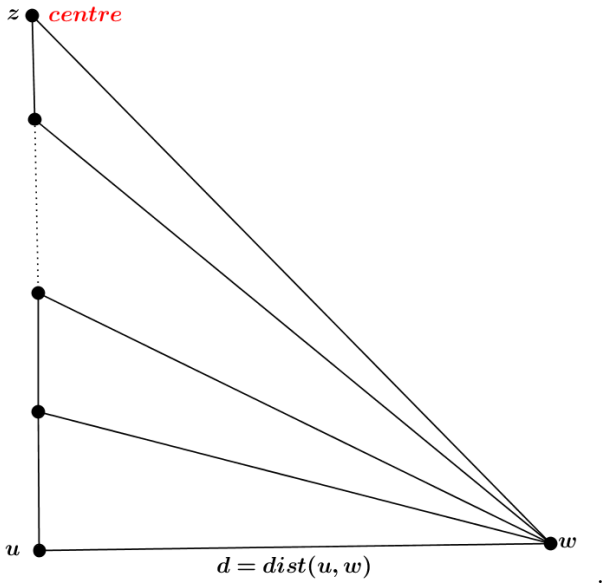


Figure 4.6: Description schématique du routage de u à v dans un HCP_k.

Pour comprendre le schéma de routage des auteurs de ce papier, nous allons définir les notions de rayon et de centre d'un graphe G.

- Le rayon local en un sommet x d'un graphe G est:

$$r_x(G) = max_{v \in V}\{(dist(x, v)\}$$

- Le centre d'un graphe G est tout sommet c tel que pour autre sommet $v \in V$,
$$r_c(G) \leq r_v(G)$$

Le rayon d'un graphe G est le rayon local $r_c(G)$ de quelque centre c de G.

Dans leur schéma de routage, les auteurs sélectionnent quelques centres de G qu'ils appellent pivot. En ces centres, ils construisent des arbres d'intervalle couvrant la totalité du graphe. Et pour tout autre sommet v diffèrent de c, ils choisissent son voisin et en ce sommet, ils construisent un arbre d'intervalle.(voir figure 3.7)

Ainsi pour envoyer un message d'un sommet u à un sommet w de G, u envoie le message directement à l'arbre d'intervalle enraciné en u noté R_u, sinon u retransmet le message à son pivot R_c qui à son tour va tenter de joindre w. Ce processus se réitère jusqu'à ce que l'intervalle contenant w soit trouvé et le message y est ensuite envoyé.

La complexité en espace mémoire de ce schéma est d'ordre $O(kn^{1+1/k}\log^2 n)$ bits.

Dans [25], Yon Dourisboure utilise la notion d'arbre de niveau (Layering-tree) (introduit en premier par V. D. Chepoi et F. F. Dragan dans [26]) pour décrire un schéma de routage efficace sur des classes de graphes appelées graphes chordaux. Parmi ces graphes on note les graphes k-chordaux et les graphes de longueur arborescent δ.

Un graphe k-chordal est un graphe qui ne contient aucun cycle de longueur supérieur à k. Un graphe est dit de longueur arborescent δ s'il comporte une décomposition arborescent dans laquelle le diamètre de n'importe quel sac est d'au plus δ.

Le schéma de routage de Dourisboure utilise des adresses et des espaces mémoires de taille $O(\log^2 n)$ bits par sommet avec une déviation d'au plus $k+1$ pour un graphe k-chordal, et, des adresses et espaces mémoires de $O(\delta \log^2 n)$ bits par sommet et une déviation de $6\delta + 2$ pour des graphes de longueur arborescent δ.

Selon Dourisboure, l'efficacité d'un schéma de routage est mesurée en fonction de sa déviation ou de son facteur d'étirement. Il définit la déviation d'un schéma de routage par le facteur d si le schéma garantit que la longueur de la route entre toute paire de sommets ne dépasse jamais la distance entre ces sommets ajoutée de d.

Le meilleur schéma de routage connu sur les graphes k-chordaux admet une déviation de $2\lfloor k/2 \rfloor$ avec des adresses et des espaces mémoires de $O(\log^3 n/\log\log n)$ bits par sommet [27]. Dans ce papier, Dourisboure montre qu'en relaxant cette déviation d'un bit, c'est-à-dire $k+1$ au lieu de $2\lfloor k/2 \rfloor$, il est possible d'obtenir un schéma de routage avec des adresses et des espaces mémoires de $O(\log^2 n)$ bits par sommet.

Le schéma de routage de Y. Dourisboure est décrit comme suit:

Soit G un graphe arbitraire et LT l'arbre de niveau de G. Soit H l'arbre d'hiérarchie de LT et S l'arbre couvrant G de plus court chemin enraciné à la source s de LT.
Chaque sommet u de G contient les informations pour router dans S.

L'idée ici est d'utiliser, pour toute paire de sommets $x, y \in G$, le plus proche ancêtre qui leur est commun pour router entre eux. Cet ancêtre est déterminé grâce à l'arbre d'hiérarchie H. Ainsi, pour envoyer un message de tout sommet u vers tout sommet v de G, la solution est de trouver, dans H, le plus proche

ancêtre commun à u et v. Il suffit alors d'envoyer le message à cet ancêtre en utilisant l'arbre de plus court chemin S, puis de là, le message est retransmis à v en utilisant toujours un plus court chemin dans S comme l'indique la figure 3.7.

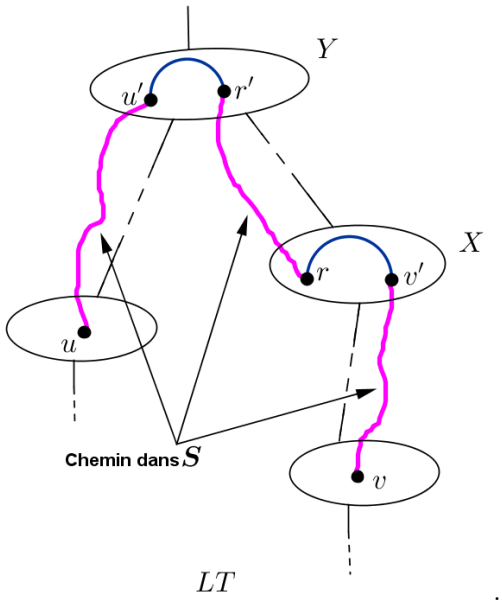


Figure 4.7: Exemple de route induit par le schéma de Dourisboure

Questions Ouvertes:

Est-ce qu'on pourrait généraliser le schéma de routage de Fraigniaud et al. Indépendamment Thorup et al. aux graphes sans mineur $K_{2,r}$?

Est-ce que le schéma de routage dans les arbres ne peut pas être généralisé aux graphes de halin?

Comment concevoir un schéma de routage ayant en même temps une faible nécessité en espace mémoire et un faible coût de communication?

CHAPITRE 5

UN SCHÉMA DE ROUTAGE COMPACT AVEC FACTEUR D'ÉTIREMENT DANS LES GRAPHES DE HALIN

5.1 Définition

5.1.1 Graphe de halin

En théorie des graphes, un graphe de halin est un graphe planaire construit à partir d'un arbre T en reliant toutes ses feuilles dans un cycle qui fait le tour de l'arbre de telle façon que T reste planaire. On exige de plus que l'arbre comporte au moins quatre sommets et ne comporte pas de sommets de degré 2.

5.1.2 Excentricité

Etant donné un arbre T et un sommet $v \in T$, on définit "excentricité" de v, la longueur du plus long chemin entre v et l'un des autres sommets de T.
Un sommet de T d'excentricité minimum est appelé le centre de T.

En effet, ce centre est obtenu à partir d'un algorithme efficace qui s'exécute comme suit:

Soit un arbre T:

1- Enlever toutes les feuilles de T. Soit T_1 l'arbre restant.

2- Enlever toutes les feuilles de T_1. Soit T_2 l'arbre restant.

3- Répéter l'opération enlevée comme suit: Enlever toutes les feuilles de T_i. Soit T_{i+1} l'arbre restant.

4- Lorsque l'arbre restant n'a plus qu'un ou deux sommets, arrêter. Supposons que cet arbre est T_k.

5- Si T_k contient un seul sommet, ce sommet est le centre de T. L'excentricité du sommet et k.

6- Si T_k contient deux sommets, l'un ou l'autre est un centre pour T. L'excentricité est Alors de $k+1$.

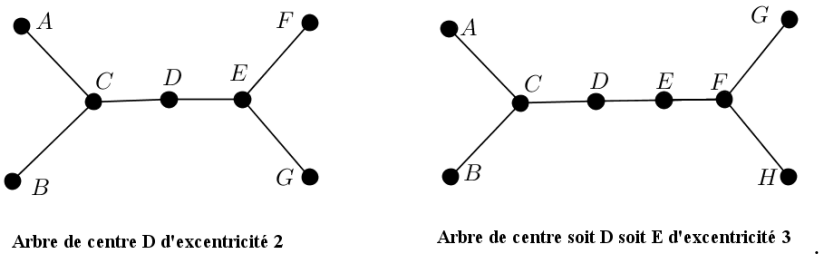


Figure 5.1: Exemples de centre pour un arbre

5.1.3 Rayon d'un graphe

Le rayon local r_x d'un graphe $G = (V, E)$ en un sommet x est le maximum des distances entre x et tout autre sommet $v \in V$.

$$r_x = max_{v \in V}\{dist(x, v)\}.$$

5.2 Notre contribution

5.2.1 Schéma de routage compact avec facteur d'étirement d'au plus 2

Dans notre travail, nous avons porté notre réflexion sur la détermination d'un schéma de routage compact pour les graphes de halin. Nous avons pensé qu'il serait possible de faire un routage compact dans un graphe de halin avec un facteur d'étirement d'au plus 2 en se basant sur les travaux de Cyril Gavoille et Pierre Fraigniaud [19]. Notre travail nous laisse voir que lorsque l'on veut router entre deux sommets quelconque x et y d'un graphe G de halin, il y a quatres cas de dispositions possibles pour x et y dans G. Notre approche est expliquée dans les paragraphes suivants.

Soit $G = (V, E)$ un graphe de halin. Soit X le graphe constitué de l'ensemble des sommets bordant la face extérieur de G et soit T_c l'arbre de G enraciné au centre c de T et couvrant les sommets de $G \backslash X$. on notera r_c, le rayon en c de T.

Numérotons les sommets de X par des entiers de 1 à $|X|$ puis ceux de T_c suivant un DFS sur T_c, par les entiers de $|X|+1$ à n. Chaque sommet u de G contient dans son adresse, noté $l(u)$, les informations sur l'ensemble de ses voisins et de plus, autre que l'identifiant et le *cpath*[1], l'adresse de chaque sommet contient une valeur booléen indiquant si le sommet est voisin direct d'un sommet de X ou pas. L'adresse de tout sommet de G contient aussi une table à deux entrées nommée *ValInt* et stockant le minimum et le maximum de ces descendants dans X.

L'en-tête de chaque message contient un ensemble d'information dont entre autres, l'adresse de destination du message.

Supposons qu'on veuille transmettre un message d'un sommet quelconque x de G vers un autre sommet y: les quatres cas possibles sont décrits par l'algorithme ROUTE de la manière suivante:

A la réception du message, l'algorithme ROUTE récupère dans l'en-tête du message, l'adresse du sommet destinataire et celle du sommet courant. Ainsi, il détient les identifiants des deux sommets x et y, les informations sur le voisinage (y ou x est un voisinage de X ou pas) et les valeurs dans y.ValInt[i] et x.ValInt[i] avec $i \in [1,2]$.

L'algorithme de routage de G:

ROUTE(l(x),l(y))

si $x, y \in X$ alors execute(**Cas 1**)
si $x \in X$ et $y \notin X$ alors execute(**Cas 2**)
si $x \notin X$ et $y \in X$ alors execute(**Cas 3**)
si $x, y \notin X$ alors execute(**Cas 4**)

L'algorithme de routage pour G présenté ci-dessus est élémentaire, l'originalité vient de son explication.

[1]Pour tout sommet u d'un arbre T enraciné en un somet r, $path(u)$ est défini comme la séquence de rang rencontrée sur le chemin de r à u. Le *clean-path* de u noté $cpath(u)$ est définie comme la séquence obtenue après élimination des rangs du $path(u)$ de valeur égale à 1. (voir [19]).

5.3 Explications

5.3.1 Exécution de l'algorithme ROUTE

Notre Algorithme ROUTE dans son exécution fait appel à la fonction *EstVoisinageX* dans chacun des cas précèdants. *EstVoisinageX* est une fonction qui prend en paramètre l'adresse d'un sommet x de G et retourne vrai ou faux selon que le sommet est voisin direct ou non d'un sommet de X. L'exécution de chaque cas est gérée par la fonction *execute*. Cette fonction exécute les cas selon les positions de x et de y dans G. *ValInt* est la structure de donnée contenant les valeurs de l'intervalle contenue dans l'adresse de chaque sommet.

Cas 1: **Si** x et $y \in X$ et $id(x) > id(y)$
si $2r_c \geq min(|X| - x + y, x - y)$
router$X(y)$
sinon router$T_c(y)$

Sinon
si x et $y \in X$ et $id(x) < id(y)$
si $2r_c \geq min(|X| + x - y, |x - y|)$
router$X(y)$
sinon router$T_c(y)$
FinSi;

Si x et $y \in$ à X, l'idée est de trouver le plus court chemin permettant d'atteindre y depuis x. Puisque $x \in X$ et que X est un cycle, alors, l'astuce revient à trouver d'abord le minimum ν des longueurs entre x et y en passant de part et d'autre des arêtes incidentes à x et appartenant à X. Ensuite comparer ν à la longueur du chemin entre x et y dans T_c. La valeur ν est la longueur du plus court chemin dans X entre x et y. ν est trouvé par $min(|X| - x + y; x - y)$ si $id(x) > id(y)$ et $min(|X| + x - y; |x - y|)$ sinon. Ainsi si ν est plus petite ou égale à la longueur dans T_c (voir algorithme cas1), alors le message est envoyé par X et sinon il est envoyé par le chemin entre x et y dans dans l'arbre T_c qui est d'au plus $2r_c$.

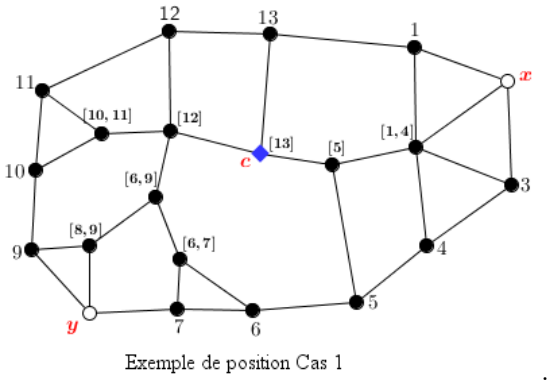


Figure 5.2: Exemple de disposition de x et de y dans le graphe G pour le cas 1 de l'algorithme `ROUTE`.

Cas 2: **Si** x $\in X$ et $y \notin X$

si EstVoisinage$X(y)$

si y.ValInt[0] $> x$

ν=min(y.ValInt[0]-x; $|X|$-x+y.ValInt[1])

sinon

ν=min(|y.ValInt[1]-x|; $|X|$-x+y.ValInt[0])

si $\nu > 2r_c$-1 alors router$T_c(y)$

sinon router$X(y)$

si NON(EstVoisinage$X(y)$)

si y.ValInt[0] $> x$

ν=min(y.ValInt[0]-x; $|X|$+x-y.ValInt[1])

sinon

ν=min(|y.ValInt[1]-x|; $|X|$-x+y.ValInt[0])

sinon si y.ValInt[0] $\leq x \leq$ y.ValInt[1]

ν=min(y.ValInt[1] $- x$;$x -$ y.ValInt[0])

router$T_c(y)$

si $\nu > 2r_c$-2 alors router$T_c(y)$

sinon router$X(y)$

FinSi;

Si $x \in X$ et $y \notin X$, en récupérant l'en-tête du message, la fonction de routage dispose des informations sur les valeurs de y.ValInt[i] qui constitues le minimum et le maximum des descendants de y dans X. L'idée est de trouver le minimum des longueurs des chemins entre x et le plus petit descendant de y dans X

(y.ValInt[0]) et x et le plus grand descendant de y dans X (y.ValInt[1]). Cette valeur est stockée dans ν. Puisque $y \in T_c$ alors il peut être atteint depuis tout sommet $x \in X$: soit en passant par le chemin ν relié à une des deux arrêtes adjacents à x et appartenant à X, soit par le chemin relié à l'arrête adjacente à x et appartenant à T_c que nous noterons ρ. De plus, le fait que $y \in T_c$ nous donne deux positions possibles pour y: soit y est voisin direct d'un des sommets de X, soit il ne l'est pas. Ceci est vérifié par la fonction *EstVoisinageX* appelée par l'algorithme ROUTE dans son execution. Ainsi, l'idée revient à déterminer par lequel des deux chemins (ν ou celle passant par T_c) le message doit être retransmis pour atteindre y. La valeur ν contient la longueur minimale du chemin entre x et un des descendants de y dans X. Cette valeur est obtenue en tenant compte du fait que y soit voisin de X ou pas et celui que les descendants de y dans X aient des identifiants plus grands ou plus petits que celui de x.

Ainsi, pour router dans G, il suffit de comparer ν à $2r_c-1$ (si $\nu > 2r_c-1$) pour le cas ou y est voisin direct de X, sinon comparer ν à $2r_c-2$ (si $\nu > 2r_c-2$). Dans tous les deux positions possibles de y, on constate qu'en routant par ρ(c'est-à-dire dans l'arbre) on route par un plus court chemin puisque dans le cas où y est un voisin direct de X, aucune longueur ne peut dépasser $2r_c - 1$ et, dans le cas où y n'est pas voisin direct de X, il en est de même qu'aucun chemin passant par T_c ne peut dépasser $2r_c - 2$. Dans le cas où $\nu \leq 2r_c - 1$ ou ($\nu \leq 2r_c - 2$), le message est retransmis à y par un chemin passant par X. Ce chemin n'est pas forcément un plus court chemin. On verra dans la correction que notre schéma de routage présente un facteur d'étirement η à déterminer par la suite.

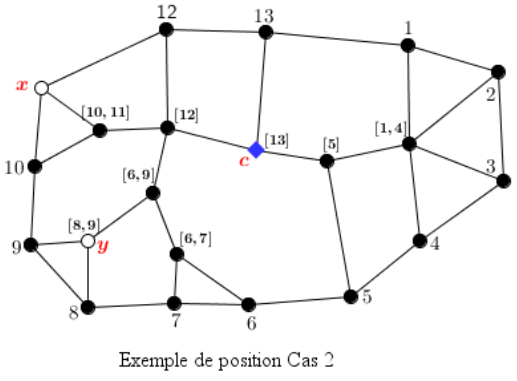


Figure 5.3: Exemple de disposition de x et de y dans le graphe G pour le cas 2 de l'algorithme `ROUTE`.

Le cas 3 est aussi similaire au cas 2 expliqué ci-dessus, au facteur prés que cette fois ci, c'est $y \in X$ et x pas.

Cas 3: **Si** x $\notin X$ et $y \in X$

si EstVoisinage$X(x)$

si x.ValInt[0] $> y$

ν=min(x.ValInt[0]-y; $|X|$+y-x.ValInt[1])

sinon

ν=min(y-x.ValInt[1]; $|X|$-x.ValInt[0]+y)

si $\nu > 2r_c$-1 alors router$T_c(y)$

sinon router$X(y)$

si NON(EstVoisinage$X(y)$)

si x.ValInt[0] $> y$

ν=min(x.ValInt[0]-y; $|X|$+y-x.ValInt[1])

sinon

ν=min(y-x.ValInt[1]; $|X|$+x.ValInt[0]-y)

si $\nu > 2r_c$-2 alors router$T_c(y)$

sinon router$X(y)$

FinSi;

Cas 4: **Si** x, y $\notin X$

si y.ValInt[0] >x.ValInt[1]

ν=min(y.ValInt[0]-x.ValInt[1]; $|X|$-y.ValInt[1]+x.ValInt[0])

sinon

ν=min(x.ValInt[0]-y.ValInt[1]; $|X|$-x.ValInt[1]+y.ValInt[0])

si EstVoisinage$X(x)$ et EstVoisinage$X(y)$

si $\nu > 2r_c$-2 alors router$T_c(y)$

sinon router$X(y)$

si EstVoisinage$X(x)$! = EstVoisinage$X(y)$

si $\nu > 2r_c$-3 alors router$T_c(y)$

sinon router$X(y)$

si NON(EstVoisinage$X(x)$) et NON(EstVoisinage$X(y)$)

si $\nu > 2r_c$-4 alors router$T_c(y)$

sinon router$X(y)$

FinSi;

Si $x, y \notin X$, c'est-à-dire que les deux sommets appartiennent à T_c. Alors pour atteindre y depuis x, on route le message soit dans T_c, soit en empruntant les sommets de X. ν contient la valeur minimale de la longueur du chemin dans X entre les descendants de x et ceux de y sachant que l'identifiant du plus grand descendant de y (y.ValInt[1]) est soit plus grand ou soit plus petit que celui du plus grand descendant de x (x.ValInt[1]) dans X.

Pour router un message de x à y, trois positions possibles sont à prendre en compte: Soit les deux sommets sont tous voisins de X, soit l'un est voisin et l'autre pas ou soit ils ne sont tous les deux pas voisin de X. Ce qui revient alors à comparer ν à $2rc - 3$ ($\nu > 2r_c - 3$) si x ou y est voisin direct de X, sinon comparer ν à $2r_c - 2$ ($\nu > 2r_c - 2$) si x et y sont des voisins direct de X et sinon, enfin comparer ν à $2r_c - 4$ si aucun des deux sommets n'est voisin direct de X. Si toutes ces conditions sont vérifiées, router en passant par ρ route par le plus court chemin puisqu'aucun chemin dans T_c ne dépasse $2r_c - 3$, ou $2r_c - 2$ ou $2r_c - 4$ selon que x ou y soit voisinage ou non d'un sommet de X. Ainsi, pour chacun des cas, y est à une distance d'au plus ces trois valeurs, et comme ν est de longueur plus grande que ces valeurs, alors $routeT_c$ d'écrit le plus court chemin.

Dans les conditions où ν est inférieur ou égal soit à $2r_c - 2$, soit à $2r_c - 3$, ou soit à $2r_c - 4$ selon les positions dans T_c entre x et y, le message est retransmis à y par un chemin passant par X d'écrit par $routeX$. Ce chemin n'est pas forcément un plus court. On montrera dans la correction que notre schéma de routage présente un facteur d'étirement η à déterminer par la suite.

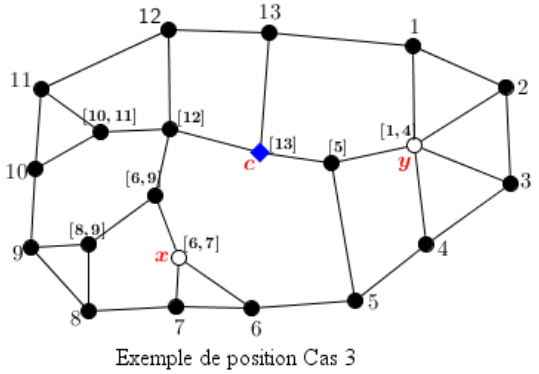


Figure 5.4: Exemple de disposition de x et de y dans le graphe G pour le cas 4 de l'algorithme `ROUTE`.

5.4 Corrections

Lemme 2 *Soit G un graphe de halin et soit X le cycle extérieur de G, alors pour tout $x, y \in G$ l'algorithme ROUTE permet de router le message sur un plus court chemin avec un facteur d'étirement d'au plus $\eta \leq 2$.*

Preuve: Pour tout $x, y \in X$, l'algorithme ROUTE retourne le port de sortie permettant de router sur un plus court chemin entre x et y dans G.

Pour tout $x \in X$ et $y \notin X$, en routant dans routeT_c, on route le message suivant le plus court chemin de x à y. En effet, pour le (cas 2), le chemin de x à y passant par T_c est le plus court car r_c étant le rayon de G, si $y \in T$ et est voisin d'un sommet de X, il n'y a pas de chemin permettant d'atteindre y dans l'arbre plus long que $2r_c - 1$. Alors, router dans T_c permet d'atteindre y par un chemin de longueur au plus $2r_c - 1$, et si $\nu > 2r_c - 1$ (voir algorithme) alors ν est de longueur plus grande que la longueur de tout chemin passant par T_c. Ainsi la table route$T_c(y)$ est choisie et nous permet de router dans l'arbre sur le plus court chemin. Le principe est le même pour les cas 3, 4 avec respectivement des chemins longueur au plus $2r_c - 2$, $2r_c - 4$.

Sinon, c'est-à-dire si $\nu \leq 2r_c - 1$, soit ρ le chemin entre x et y dans T_c, on sait que $|\rho| \leq 2r_c - 1$ puisque y est un sommet périphérique et $x \in X$. Alors si $\nu \leq 2r_c - 1$(voir ROUTE), le message est envoyé par le chemin passant par les sommets de X. Ainsi, le rapport des longueurs du chemin emprunté pour envoyer le message de x à y sur le plus court chemin entre x et y est appelé le ratio et est noté içi η avec $\eta = |\nu|/|\rho|$. Ce ratio constitue un facteur d'étirement pour les cas 1 avec $2r_c - 1/2r_c - 1 \leq 1$, cas 2 avec $2r_c - 2/2r_c - 2 \leq 1$ si $\nu \leq 2r_c - 2$, et $2r_c - 3/2r_c - 3 \leq 1$ ou $2r_c - 4/2r_c - 4 \leq 1$ pour le cas 4 si $\nu \leq 2r_c - 3$ ou si $\nu \leq 2r_c - 4$.

Un cas particulier est à signaler, celui ou $x \notin X$ et $y \in X$ et x compris entre y.ValInt[0] et y.ValInt[1]. On sait que $\nu \leq |X|/2$ et T_c est l'arbre enraciné en c de rayon r_c. Si y.ValInt[0] $\geq x \geq$y.ValInt[1] alors x est descendant de y dans T_c donc la longueur ρ du chemin entre x et y dans T_c est d'au plus $\rho \leq r_c$. Soit λ la longueur du chemin dans X entre x et y. λ est au plus égale à $\nu + r_c$ avec $\nu \leq |X|/2$, d'où $\lambda \leq |X|/2 + r_c$. Ainsi, dans l'éventualité où x est compris entre y.ValInt[0] et y.ValInt[1] en routant dans T_c on route sur un chemin avec un facteur d'étirement d'au plus $\eta = \rho/\lambda < 1$.

Le cas 1 requiert toute notre attention. Posons:

$$k = \begin{cases} \text{min(|X|-x+y, x-y)} & \text{si } id(x) > id(y) \\ \text{min(|X|+x-y, |x-y|)} & \text{sinon} \end{cases}$$

En routant par la table tableX, on route sur le plus court chemin entre x et y car si $2r_c \geq k$, alors $y \notin T_c$ puisqu'aucune longueur ne dépasse $2r_c$; r_c étant le rayon de G. Et si $2r_c < k$, en routant dans T_c, la longueur pouvant être atteint est d'au plus $2r_c - 1$. Comme chaque sommet de G est 3-connexe, alors chaque sommet sur les $2r_c - 1$ admet au moins un descendant dans X. Ainsi pour les $2r_c - 1$ sommets, on dispose d'au moins du même nombre de sommet dans X. Or dans X, la longueur maximale entre x et y est d'au plus $X/2$. Ce qui veut dire $2r_c - 1/2$ pour un nombre maximal de $2r_c - 1$ sommets (avec $X = 2r_c - 1$). D'où un facteur d'étirement d'au plus $2(2r_c - 1)/2r_c - 1 = \eta = 2$.

En conclusion à notre réflexion, nous dirons que tout graphe de halin $G = (V, E)$, admet un schéma de routage avec un facteur d'étirement d'au plus 2.

CHAPITRE 6

CONCLUSION ET PERSPECTIVES

Dans cette partie, nous allons succinctement faire un rappel de l'ensemble des résultats obtenus dans le routage compact que ce soit de plus court chemin ou avec facteur d'étirement, et ensuite faire une briève synthèse de notre résultat sur le routage dans les graphes de halin pour enfin présenter les perspectives de recherche relatives à notre résultat.

6.1 Conclusion

La première partie de ce mémoire se subdivise en quatres chapitres. Le premier chapitre est constitué des généralités sur les graphes. Dans le deuxième chapitre, nous avons présenté les travaux sur le routage par intervalle introduit par N. Santoro et R.Khatib [1]. Une technique standard pour réduire la complexité mémoire des fonctions de routage est d'utiliser le routage par intervalles. Des résultats originaux ont été obtenus comme ceux concernant le nombre minimal d'intervalles à utiliser pour garantir que les chemins de routage soient au plus de longueur 1.5 fois le diamètre du graphe, ou bien pour que chaque chemin de routage soit au plus 5 fois plus long que le plus court (on parle d'étirement au plus 5). Il a été montré aussi qu'il est possible de trouver pour tout graphe à n sommets un routage avec un intervalle par arc ayant un étirement moyen proche de $\log n$.

Dans le troixième chapitre, nous avons présenté les résultats sur les travaux faits sur le routage compact de plus court chemin. De nombreuses études basées sur le routage compact de plus court chemin ont été faites. Parmi celles-ci on évoque celui de P. Fraigniaud et C. Gavoille dans [19]. Ils ont prouvé que les réseaux en arbre comportant n nœuds supportent des schémas de routage de taille $O(\log n)$ bits pour les en-têtes de message, l'espace mémoire et l'adressage des nœuds. Le cas particulier des arbres est un problème important puisque de nombreux schémas de routage décrits pour les graphes généraux sont basés sur

une couverture hiérarchique du graphe en régions où le mécanisme de routage est réalisé via un arbre couvrant la région. Fraigniaud et Gavoille dans [19] ont aussi montré que si les numéros de port des arêtes peuvent être permutés, des tables de seulement $O(\log n)$ bits par sommet suffisent à router un message dans un arbre à n sommets selon les plus courts chemins. Si les numéros de port sont fixés et ne peuvent être changer, ils ont cependant construit des tables de $O(\log^2 n/\log\log n)$ bits et ils ont montré que cette taille était optimale. La meilleure borne connue pour le schéma de routage de plus court chemin utilisant des adresses de taille $c\log n$ bits pour toute constante $c \geq 1$ est de $O(\log n)$ bits. Il a été présenté par Gavoille et Pérénnès [5]. Ces derniers affirment qu'ils serait difficile d'exiger un plus petit espace mémoire pour les arbres. Ce qui se confirme par les travaux de SANTORO et KHATIB [1] et LEEUWEN et TAN [4]. Ces derniers considèrent pour tout nœud interne u d'un arbre T, l'arrêt qui mène à un enfant v de u est associé à un intervalle contenant l'ensemble des adresses des nœuds du sous arbre enraciné en v et le lien qui mène à la racine est associé à un intervalle complémentaire. L'espace mémoire requis est de $O(n\log d)$ bits sur chaque nœud. Les travaux de Y. Dieng et C Gavoille, à partir du théorème 13, ont montré que les graphes planaires extérieur tout comme les arbres présentés par (Fraigniaud et al.,2001;Thorup et al.,2001) supportent un schéma de routage de plus courts chemins avec des adresses, des en-têtes et des tables de routage de $O(\log n)$ bits. Ce résultat fut remarquable puisqu'il leur permettra d'étendre leurs travaux sur une nouvelle classe de graphe appelé $(k;r)$-constellation en démontrant par le théorème 2 du [20] que toute (k,r)-constellation connexe non valuée à n sommets admet un schéma de routage de plus courts chemins avec des adresses et des en-têtes de $O((k+1)\log n)$ bits, et des tables de routage de $O((kr+1)\log n)$ bits.

Dans le quatrième chapitre, nous présentons un état de l'art sur le routage compact avec facteur d'étirement. Nous avons vu que le routage compact peut aussi ne pas être de plus court chemin. Son efficacité est souvent mesuré en fonction de son facteur d'étirement. Nous avons montré dans ce chapitre que tout graphe de halin admet un schéma de routage avec facteur d'étirement au plus 2. Pour tout x, y de G, notre schéma de routage prend en compte toutes les positions possibles que peut prendre chacun des sommets de G qu'il soit récepteur ou transmetteur du message. La détermination du centre et du rayon du graphe fut d'une aide importante dans l'élaboration de notre schéma de routage. Ce résultat est assez simple mais son importance est de donner une idée plus claire sur la démarche qui est utilisée.

6.2 perspectives

Dans cette partie, nous allons poser sous forme de questions l'ensemble des points ouverts dans ce mémoire, et qui seront l'objet de réflexion dans un futur proche.

Question 1:
Dans [19] Gavoille et Fraigniaud définissent pour tout arbre T de racine r, le *path(u)* de tout sommet u comme la séquence de rangs rencontrée sur le chemin de r à u. Le *clean-path* est définit par les auteurs comme la séquence obtenue après élimination des rangs du *path* de valeur égal à 1. En redéfinissant le *cpath* comme étant la séquence de rangs du *path* (sans élimination de valeur), alors $|cpath(u)|$ nous donne le nombre exact d'élément du *path(u)*, et comme le graphe n'est pas valué donc toutes les arêtes sont de poids uniforme et par conséquent $|cpath(u)|$ donne la valeur de la longueur du chemin entre r et u dans T. Est-ce que cette redéfinition du *cpath* ne permettrait pas d'obtenir un schéma de routage plus compact que celui obtenu dans ce mémoire?

Question 2:
Il a été établit que tout arbre à n sommets possède un schéma de routage de plus court chemin utilisant des adresses, des en-têtes et des tables de routages de $O(\log n)$ bits. Sachant que les graphes de halin ont une structure trés proche de celle des arbres, on se pose la question de savoir si le graphe de halin ne pouvait pas être rendu planaire-exterieur aprés suppression d'un nombre k de sommets à déterminer, et puis utiliser le routage dans les planaires-extérieurs de Dieng et Gavoille [20], basé sur celui des arbres, pour router dans notre graphe suivant le plus court chemin et avec un espace mémoire d'au plus $O(\log n)$ bits?

Question 3:
Nous constatons dans nos recherches qu'on est toujours appelé à comparer c à la longueur maximale de l'arbre à chaque fois que nous désirons router dans G. Si nous pouvons trouver une relation entre le nombre d'arrêtes dans X noté $|X|$ et le rayon r_c de G, nous nous posons la question de savoir s'il ne serait pas possible de faire un routage plus compact et par un plus court chemin dans G.

Question 4:
Est-il possible de trouver un étiquetage compact des sommets d'un graphe de halin de tel sorte que la distance entre tout paire de sommets du graphe puisse être déduit en examinant seulement leurs étiquettes?

BIBLIOGRAPHIE

[1] N. Santoro, R. Khatib, Labelling and implicit routing in networks, Comp. J. 28 (1985) 5–8.

[2] Cyril Gavoille, A survey on interval routing, Theoretical Computer Science 245 (2000) 217–253,

[3] J. van Leeuwen and R. B. Tan. Routing with Compact Routing Tables, Tech. Rep. R UU-CS-83-16, Dept. of Computer Science, Utrecht University (1983).

[4] J. van Leeuwen, R.B. Tan, Interval routing, Comput. J. 30 (1987) 298–307.

[5] P. Fraigniaud, C. Gavoille, Interval routing schemes, Algorithmica 21 (1988) 155–182.

[6] T. Eilam, S. Moran, S. Zaks, A simple DFS-based algorithm for linear interval routing, in: M. Mavronicolas, P. Tsigas (Eds.), 11th Internat. Workshop on Distributed Algorithms (WDAG), Lecture Notes in Computer Science, vol. 1320, Springer, Berlin, September 1997, pp. 37–51.

[7] C. Gavoille, E. Guevremont, Worst case bounds For shortest path interval routing, J. Algorithms 27 (1988) 1–25.

[8] E. Kranakis, D. Krizanc, Lower bounds for compact routing, in: C. Puech, R. Reischuk (Eds.), 13th Ann. Symp. on Theoretical Aspects of Computer Science (STACS), Lecture Notes in Computer Science, vol. 1046, Springer, Berlin, February 1996. (1988) 1–25.

[9] C. Gavoille, S. Perennes, Lower bounds for interval routing on 3-regular networks, in: N. Santoro, P. Spirakis (Eds.), 3rd Internat. Coll. on Structural Information Communication Complexity (SIROCCO), Carleton University Press, June 1996, pp. 88–103.

[10] C. Gavoille, S. Perennes, Memory requirement for routing in distributed networks, in: 15th Ann. ACM Symp. on Principles of Distributed Computing (PODC), ACM Press, New York, 1996, pp. 125–133.

[11] E. Kranakis, D. Krizanc, S.S. Ravi, On multi-label linear interval routing schemes, in: 19th Internat. Workshop on Graph – Theoretic Concepts in Computer Science – Distributed Algorithms (WG), Lecture Notes in Computer Science, vol. 790, Springer, Berlin, June 1993, pp. 338–349.

[12] R. Kralovic,P. Ruzicka,D. Stefankovic, The complexity of shortest path and dilation bounded interval routing, in: C. Lengaur, M. Griebl, S. Gorlatch (Eds.), 3rd Internat. Euro-Par Conf., Lectures Notes in Computer Science, vol. 1300, Springer, Berlin, August 1997, pp. 258–265.

[13] D. May, P. Thompson, Transputers and Routers: Components for concurrent machines, INMOS Ltd., 1990.

[14] B. Zerrouk, V. Reibaldi, F. Potter, A. Greiner, D. Anne, RCube: a gigabit serial links low latency adaptive router, in the Records of the IEEE Hot Interconnects IV, Palo Alto CA, U.S.A, August 1996.

[15] Greg N. Frederickson and Ravi Janardan, Designing Networks with Compact Routing Tables, Algorithmica (1988) 3; 171-190.

[16] J. I. Munro and H. Suwanda, Implicit data structures for fast search and update, journal of computer and system sciences.Vol 21, No. 2, October 1980, pp 236-250.

[17] G. N. Frederickson, Implicit data structures for the dictionary problems, journal of the Association for Computing Machinery. Vol 30, No. 1, January 1983, pp. 80-94.

[18] J. van Leeuwen, R.B. Tan, Interval routing,Technical report RUU-CS-85-16, Department of Computer Science, University of Utrecht. The Netherlands.

[19] Pierre Fraigniaud, Cyril Gavoille, Routing in tree, Research report RR-1252-01, January 2001.

[20] Youssou Dieng, Cyril Gavoille, Routage compact optimal dans les (k, r)-Constellations, LaBRI, Université de Bordeaux, RSTI-TSI. Volume 30-n 5/2001, pages 485 à 513.

[21] Thorup M.,Zwick U., Compact Routing Schemes, 13^th *Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, ACM Press,p.1-10, july, 2001.

[22] A. Bar-NOY, N. LINIAL, B. AWERBUCH, D. PELEG, Improved Routine Stratégies With Succinct Tables, JOURNAL OF ALGORITHMS 11, 307-341 (1990).

[23] D. Peleg and E. Upfa, A tradeoff between size and efficiency for routing tables, JAVM 36 (1989), 510-530.

[24] L. KLEINROCK and F. Kamoun, hiérarchical routing for large network; Performance evaluation and optimisation, Comput. Networks 1 (1977), 155-174.

[25] Y. Dourisboure, Compact Routing Schemes for Generalised Chordal Graphs, Journal of Graph Algorithms an Application http://jgaa.info/ vol. 9, no. 2, pp. 277-297 (2005).

[26] V. D. Chepoi and F. F. Dragan, A note on distance approximating tree in graphs, Europ. J. Combinatorics, 21 :761, 2000.

[27] F. F. dragan, C. Yan, and I.Lomonosov, Collective tree spanners of graphs, In T. Hagerup and J. Katajainen, editors, 9th Scandinavian workshop on Algorithme theory (SWAT), volume 3111 of Lecture Notes in Compunter Science, page 64- 76. Springer, 2004.