Problèmes algorithmiques et de complexité dans les réseaux sans fil

Benoit Darties

Académie de Montpellier
Université Montpellier II
— Sciences et Techniques du Languedoc —

Thèse

présentée au Laboratoire d'Informatique de Robotique
et de Micro-électronique de Montpellier

Spécialité	:	**Informatique**
Formation Doctorale	:	**Informatique**
École Doctorale	:	**Information, Structures, Systèmes**

PROBLÈMES ALGORITHMIQUES ET DE COMPLEXITÉ DANS LES RÉSEAUX SANS FIL

par

Benoît Darties

Soutenue le 14 décembre 2007, devant un jury composé de :

M. Olivier Cogis, Président
M. Jean-Claude König, Directeur de Thèse
M. Sylvain Durand, co-Directeur de Thèse
M. Pascal Berthomé, Rapporteur
M. Dominique Barth, Rapporteur
M. David Coudert, Examinateur

À Charlotte et mes parents.

Remerciements :

En premier lieu, j'adresse mes sincères remerciements à Messieurs Jean-Claude König et Sylvain Durand, respectivement directeur et co-encadrant de cette thèse. Cette dernière n'auvait pas vu le jour sans la confiance, la patience, et la générosité qu'ils ont su m'accorder. Ce fut un réel plaisir que d'œuvrer à leur côtés.

Je remercie Monsieur Olivier Cogis , qui m'a fait l'honneur d'avoir accepté la présidence de mon jury de thèse. J'associe à ces remerciements Messieurs Dominique Barth et Pascal Berthomé pour avoir accepté de juger ce travail en qualité de rapporteurs.

Je désire remercier les membres de l'équipe Algorithmique et Performances des Réseau du LIRMM à Montpellier, au sein de laquelle j'ai effectué cette thèse. Je ne pouvais y espérer un meilleur accueil que celui qui m'a été octroyé. Je tiens à remercier tout particulièrement Monsieur Jérôme Palaysi, pour sa collaboration sur une grande partie des résultats de cette thèse, et son implication dans mes travaux.

Je tiens également à saluer l'ensemble des doctorants et stagiaires avec qui j'ai eu le plaisir de partager mes trois années de thèse. Travailler à leurs cotés fut une expérience enrichissante et les nombreux échanges scientifiques que nous avons eu m'ont permis d'approfondir mes connaissances dans de nombreux domaines. Un merci particulier à Gilles, Binh-Minh, Christophe, Julien, Florent, Clément et Olivier.

À titre plus personnel je remercie toute ma famille à commencer par mes parents, pour leur soutien sans limite durant toute ma scolarité et plus particulièrement ces trois dernières années. Je tiens à exprimer ma profonde gratitude envers Charlotte, pour tous les encouragements et l'amour qu'elle me porte au quotidien.

Je conclurai en remerciant l'ensemble de mes amis pour leur soutien, et les moments de détente que j'ai pu partager avec eux entre deux séances de travail, et particulièrement : Thomas, Sylvain, Renaud, Sébastien H., Nicolas, Émilie, Fabrice, Olivier G., Olivier R, Céline, Mo, Cécile, Guillaume, Alice, Sébastien, Yannick, François, Éric, Jessica, Nathalie, Silje, Mary -Angel, Frédéric, Laetitia, Matthieu, Alex, Pauline, Florent, Frédéric, Claire-Bénédicte, Lise, Valentine et bien d'autres encore.
.

Table des matières

Sigles utilisés dans ce mémoire :

DBMDSpT	:	DEGREE-BOUNDED MINIMUM DIAMETER SPANNING TREE
DCMStT	:	DEGREE-CONSTRAINED MINIMUM STEINER TREE
DCSpT	:	DEGREE-CONSTRAINED SPANNING TREE
ECF	:	EDGE-COST FLOW
ESC	:	EXACT-SET-COVER
HP	:	HAMILTONIAN PATH
ICECF	:	INFINITE CAPACITY EDGE-COST FLOW
KSC	:	K-SET-COVER
SC	:	SET-COVER
MDSpT	:	MINIMUM DIAMETER SPANNING TREE
MSC	:	MINIMUM-SET-COVER
UVDP	:	UNDIRECTED VERTEX-DISJOINT PATHS

Introduction générale

Les techniques de communication ont connu dès la fin du XXe siècle un essor sans précédent. De nouveaux outils de communication ont été développés et sont devenus accessibles au particulier en l'espace de quelques années seulement. Internet est sans aucun doute l'exemple le plus significatif pour illustrer cette démocratisation. Cet outil devenu incontournable aujourd'hui était encore inconnu du grand public quinze ans plus tôt. L'utilisation de réseaux informatiques permet aujourd'hui de relier entre eux des utilisateurs, des machines, des services et des besoins par l'intermédiaire de structures de communications adaptées.

Établir un lien entre deux entités quelconques d'un réseau suppose l'existence soit d'un lien direct entre les deux, soit d'une succession de relais consécutivement liés entre ces deux entités. L'information circule alors de proche en proche et permet la communication.

Le câble est le premier et le plus répandu des supports de communications utilisés afin d'établir un lien direct entre deux nœuds. Son apparition dans les réseaux informatiques est une conséquence directe de son utilisation bien connue dans les réseaux de téléphonie analogique. De nombreuses avancées ont permis d'augmenter considérablement le volume d'information transporté par ces câbles, par exemple par l'utilisation de la fibre optique. Les réseaux filaires ont cependant un coût de déploiement élevé. De plus, la mise en place de grands réseaux haut-débit s'est faite sans vraiment tenir compte du passage à l'échelle et de l'évolution de la demande. Un cas d'école souvent cité est celui d'une métropole américaine, qui dès la fin du XXe siècle avait déployé des réseaux en fibre optique pour couvrir la ville entière. Les capacités des liens étaient surdimensionnées afin de pouvoir faire transiter un volume d'information de l'ordre de dix fois la demande de l'époque. Mais aucune réflexion n'ayant été envisagée sur l'adaptabilité du réseau, ce dernier s'est retrouvé obsolète lorsque la demande actuelle a connu une croissance bien plus forte que prévu, au point d'atteindre cent fois la demande de l'époque.

L'utilisation de la technologie sans-fil inscrit les réseaux de communication dans une nouvelle ère. Deux entités peuvent communiquer directement sans lien physique, à la condition qu'elles soient proches l'une de l'autre. Cette technologie, qui offre des possibilités sans précédent en terme de mobilité, trouve sa principale application dans la téléphonie mobile. La popularité de ces réseaux sans fil vient également du fait de leur facilité de déploiement. Aussi, le nombre de réseaux domestiques wi-fi a augmenté de manière exponentielle en quelques années seulement.

Que ce soient pour les réseaux domestiques wi-fi[1] ou la téléphonie mobile, la technologie sans fil n'est employée qu'aux extrémités terminales du réseau de communication, c'est à dire pour réaliser une communication entre une borne sans fil et l'utilisateur final. Le reste du réseau repose la plupart du temps sur une infrastructure filaire, qui relie physiquement les bornes entre elles. D'autres réseaux

[1] en mode infrastructure

utilisent exclusivement la technologie sans-fil comme support de communication (cas notamment des réseaux ad-hoc, et des réseaux de capteurs). Ces réseaux sont utilisés dans un grand nombre d'applications (militaires, scientifiques, médicales, amateurs, ...) et peuvent même s'étendre sur plusieurs kilomètres grâce à l'utilisation d'antennes directionnelles. Ces réseaux occupent désormais une place primordiale dans notre société, et leur déploiement devrait continuer à se démocratiser dans les prochaines années.

Si la facilité de déploiement est un atout indéniable des réseaux sans fil, se posent en contrepartie de nouveaux problèmes liés à leur fonctionnement. Le contenu d'une transmission radio-électrique peut être dégradée à cause de phénomènes d'interférences. Dans les réseaux sans fil où les nœuds utilisent la même fréquence, l'établissement d'une communication nécessite un accès exclusif au média pour prévenir ces interférences. Par ailleurs les ondes hertziennes sont difficiles à confiner dans une surface géographique restreinte. Toute émission générée par une antenne classique[2] se propage dans plusieurs directions. Les phénomènes d'amortissement d'ondes radio-électriques limitent cette propagation à une zone géographiquement proche du point d'émission. Ces caractéristiques compliquent énormément la communication entre entités du réseau, puisqu'un nœud ne peut pas recevoir simultanément plusieurs transmissions, à l'inverse de ce qui s'opère dans les réseaux filaires. La norme 802.11 a été introduite afin de définir les couches basses du modèle OSI nécessaires à l'établissement de liaisons sans fil. Mais les performances obtenues par l'utilisation de cette norme s'effondrent lorsque le nombre de transmissions concurrentes augmente considérablement, générant de multiples interférences. L'utilisation d'accusés de réception permet de garantir l'acheminement du message, mais entretient la saturation du média radio.

Dans cette thèse, nous étudions la résolution algorithmique et la complexité de trois familles de problèmes inspirés des contraintes de fonctionnement des réseaux sans fil. Nous étudions dans une première partie le problème de la diffusion d'un message depuis une source unique vers l'ensemble des nœuds d'un réseau sans fil. Réaliser une diffusion efficace est un point clé dans le fonctionnement de nombreux protocoles de routage. Ces derniers utilisent par exemple une diffusion pour découvrir des routes de communication pertinentes entre une source et une destination. L'objet de nos travaux consiste ici à réaliser une diffusion minimisant les coûts en temps ou en nombre d'émissions. Nous définissons à ce titre le problème de la diffusion dans différents modèles de communication et selon différentes fonctions objectif. L'analyse de ces problèmes nous amène à définir différentes stratégies de résolution et à étudier ces dernières. Une partie de nos travaux est dédiée à la résolution de ces problèmes dans des topologies particulières comme la grille ou le tore.

Nous abordons dans une seconde partie le problème de satisfaction d'un ensemble de requêtes de communications dans un réseau sans fil multi-sauts synchrone. Les émissions se déroulent par étapes, et le coût d'une solution à ce problème est égal au nombre d'étapes nécessaires pour satisfaire toutes les requêtes. L'étude que nous proposons vise à déterminer dans quelles circonstances la recherche d'une solution optimale est difficile. Plus précisément, nous étudions l'impact de plusieurs paramètres sur la complexité de ce problème (topologie, nombre maximum d'étapes autorisé, le nombre de requêtes à satisfaire).

La dernière partie de ce mémoire nous amène au problème de conception d'un réseau sans fil multi-sauts, initialement proposé par une entreprise de télécommunications. L'objectif du problème est d'assurer une distribution de flux depuis des nœuds sources vers des nœuds clients, en minimisant le coût de l'infrastructure déployée. Les communications directes entre nœuds sont réalisées à

[2] qui n'est pas directionnelle ou parabolique

l'aide d'antennes directionnelles. L'utilisation de cette technologie permet de minimiser l'impact des phénomènes d'interférences, au point d'ignorer ces derniers dans notre modélisation. La difficulté du problème réside ici dans la satisfaction de contraintes de déploiement (nombre d'antennes limité par nœud, résistance aux pannes, ...). Nous étudions la complexité de ce problème, et proposons plusieurs méthodes de résolution exacte et approchée pour des instances de taille raisonnable. Nous implémentons ces méthodes afin de résoudre le problème pour des instances réelles proposées par l'entreprise.

Chapitre 1

Prérequis et Notations

1.1 Notations de la théorie des ensembles

La lecture de ce document requiert les notions élémentaires de la théorie des ensembles. Nous précisons dans cette première section les notations ensemblistes utilisées tout au long de ce mémoire.

Le cardinal d'un ensemble A est noté $|A|$. La notation $\emptyset$ désigne l'ensemble vide.

On note $\mathbb{N}$ l'ensemble des entiers naturels, et $\mathbb{N}^*$ l'ensemble des entiers naturels strictement positifs. On note $\mathbb{R}$ l'ensemble des réels, et $\mathbb{R}^+$ l'ensemble des réels positifs (respectivement $\mathbb{R}^*$ et $\mathbb{R}^{*+}$ lorsque 0 est exclu de ces ensembles).

La notation $[a, b]$ désigne l'ensemble des entiers compris entre a et b.

On note $\{x|p(x)\}$ l'ensemble formé des éléments x qui vérifient la propriété $p(x)$. Lorsque les éléments sont indicés, nous préférons la notation $\{x_i\}_{i\in[1,k]}$ pour désigner l'ensemble comprenant les éléments $x_1, x_2, \ldots, x_k$.

Nous utilisons les trois principales opérations ensemblistes :

- L'***union*** de deux ensembles A et B, notée $A \cup B$, est l'ensemble des éléments qui appartiennent à A ***ou*** à B.
- L'***intersection*** de deux ensembles A et B, notée $A \cap B$, est l'ensemble des éléments qui appartiennent à A ***et*** à B.
- La ***différence*** de deux ensembles A et B, notée $A - B$, est l'ensemble des éléments de A qui ne sont pas éléments de B.

La notation $\bigcup_{i\in[1,k]} A_i$ désigne l'ensemble des éléments qui appartiennent à au moins un des ensembles $A_1, A_2, \ldots, A_k$.

Soient A et B deux ensembles. Leur ***produit cartésien*** est l'ensemble $\{(a, b)|a \in A, b \in B\}$, et se note $A \times B$. Le produit $A \times A$ est noté A^2, $A \times A \times A$ est noté A^3 et ainsi de suite pour tous les entiers naturels supérieurs à 3. Nous utilisons l'abus de notation usuel $x_1, x_2, \ldots, x_k \in X$ pour $(x_1, x_2, \ldots, x_k) \in X^k$.

L'ensemble des parties de A est noté $\mathcal{P}(A)$. L'ensemble des parties de A à k éléments est noté $\mathcal{P}_k(A)$.

1.2 Éléments de la théorie des graphes

La théorie des graphes est probablement née alors que Leonhard Euler écrivait son article sur les sept ponts de Königsberg en 1736. La théorie des graphes constitue un domaine des mathématiques qui, historiquement, s'est aussi développé au sein de disciplines diverses telles que la chimie (modélisation de structures), la biologie (génome), les sciences sociales (modélisation des relations) ou en vue d'applications industrielles.

Un graphe permet de représenter simplement la structure et les connexions d'un ensemble d'entités, en exprimant les relations ou les dépendances entre ses éléments (réseau de communication, réseaux ferroviaire ou routier, arbre généalogique, ...). En plus de son existence purement mathématique, le graphe est aussi une structure de données puissante pour l'informatique.

Dans les années 1960, Claude Berge a posé les bases de la théorie moderne des graphes. La première définition formelle d'un graphe n'est plus usitée à cause de sa lourdeur. De nombreux éléments de la théorie des graphes peuvent être définis de différentes manières. Ceci peut avoir comme conséquence l'apparition de difficultés de compréhension. Nous proposons dans cette section les principales définitions de la théorie des graphes afin de prévenir toute ambiguïté. Ces définitions prévalent dans ce manuscrit sur les notations couramment utilisées.

1.2.1 Graphes non orientés

Vocabulaire élémentaire :

Définition 1 (un graphe, un graphe non orienté) *:*
Un graphe (ou graphe non orienté) $G = (V, E)$ est la donnée d'un ensemble fini non vide de sommets V, et d'un ensemble E constitué de certaines paires non ordonnées de sommets distincts appelées arêtes.

Nous utilisons parfois la notation fonctionnelle $V(G)$ pour désigner l'ensemble des sommets d'un graphe G, et $E(G)$ l'ensemble de ses arêtes. Sauf précision nous désignons par n et m les cardinaux respectifs de $V(G)$ et $E(G)$. Pour un graphe G donné, nous définissons le vocabulaire suivant :

- Si $\{a, b\}$ est une arête de G, on dit que $\{a, b\}$ est ***incidente*** aux sommets a et b.
- Deux sommets a et b sont ***adjacents*** dans G si l'arête $\{a, b\}$ appartient à $E(G)$.
- Pour un sommet a, l'***ensemble des voisins*** de a, noté $N_G(a)$, est l'ensemble des sommets qui lui sont adjacents.
- Le degré d'un sommet a de G, noté $d_G(a)$, est le nombre de sommets qui lui sont adjacents, c'est à dire $|N_G(a)|$.
- Nous notons respectivement $\delta(G)$ et $\Delta(G)$ le plus petit et le plus grand degré parmi les sommets de G.

En l'absence d'ambiguïté, le G disparaît des notations : V, E, $N(a)$, Δ, δ, ...

Définition 2 (Un sous-graphe et un sous-graphe induit) *:*
Soit $G = (V, E)$ un graphe.

*Un **sous-graphe** de G est un graphe $G' = (V', E')$ tel que :*
- $V' \subseteq V$
- $E' \subseteq E \cap \{\{a, b\} | a, b \in V'\}$

*Si $E' = E \cap \{\{a,b\}|a,b \in V'\}$, alors G' est le **sous-graphe** de G **induit** par V'.*

Nous utilisons parfois la notation $G_{[S]}$ pour désigner le sous-graphe de G induit par l'ensemble S.

Définition 3 (Un graphe partiel) *:*
Soit $G = (V, E)$ un graphe.

*Un **graphe partiel** de G est un graphe $G' = (V, E')$ avec $E' \subseteq E$.*

Définition 4 (Une chaîne) *:*
*Une **chaîne** C est un graphe dans lequel :*
- *$V(C) = \{x_0, x_1, x_2, \ldots, x_k\}$ avec $k \geq 0$,*
- *$E(C) = \big\{\{x_i, x_{i+1}\}|i \in [0, k-1]\big\}$.*

Une chaîne d'un graphe G est un sous-graphe de G qui est une chaîne. Pour une chaîne C donnée :
- Les ***extrémités*** de C sont les sommets x_0 et x_k, et nous notons $ext(C) = \{x_0, x_k\}$. Nous dirons que C ***relie les sommets*** x_0 et x_k.
- Les sommets x_1 à x_{k-1} sont les ***maillons internes*** de C.
- La ***longueur*** de C est égale à son nombre d'arêtes (ici k).

Pour un graphe quelconque G donné (sans valuation sur les arêtes) :
- La **distance entre deux sommets** a et b de $V(G)$, notée $dist_G(a,b)$, est la longueur de la plus courte chaîne d'extrémités a et b.
- Le **diamètre** d'un graphe G est la plus grande des distances entre deux sommets. Nous le notons $diam(G) = \max_{a,b \in V(G)}\{dist_G(a,b)\}$.

Définition 5 (Un parcours dans un graphe G) *:*
*Un **parcours** dans un graphe G est une liste ordonnée de sommets de G telle que 2 sommets consécutifs dans la liste sont adjacents dans le graphe.*

Nous considérons seulement des **parcours simples** pour lesquels le nombre d'occurrences d'un sommet dans la liste est au plus 1. La **longueur d'un parcours** est égale à son nombre de sommets moins 1.

Définition 6 (un cycle) *:*
Un cycle P est un graphe dans lequel :
- *$V(P) = \{x_0, x_1, x_2, \ldots, x_k\}$, avec $k > 0$,*
- *$E(P) = \big\{\{x_i, x_{i+1}\}|i \in [0, k-1]\big\} \cup \big\{\{x_k, x_0\}\big\}$.*

Un cycle dans un graphe G est un sous-graphe de G qui est un cycle. La longueur d'un cycle est égale à son nombre d'arêtes (ici $k+1$).

Définition 7 (une corde) *:*
Soient $G = (V, E)$ un graphe, et $P = (V', E')$ un cycle dans G.

L'arête $\{a, b\}$ avec $a, b \in V'$ est une corde de P si et seulement si $\{a, b\} \in E - E'$.

Vocabulaire lié à la connexité :

Définition 8 (un graphe connexe, une composante connexe) *:*
*Un graphe $G = (V, E)$ est **connexe** si et seulement s'il existe une chaîne d'extrémités a et b pour toute paire de sommets $\{a, b\}|a, b \in V$.*

Une composante connexe de G est un sous-ensemble maximal de sommets $S \subseteq V$ tel qu'il existe une chaîne d'extrémités a et b pour toute paire de sommets $\{a, b\}|a, b \in S$.

Un graphe connexe possède une seule composante connexe. Un graphe possédant deux ou plusieurs composantes connexes n'est pas connexe.

Définition 9 (un ensemble dominant, un ensemble dominant connexe) *:*
Soit $G = (V, E)$ un graphe connexe.

L'ensemble $D \subseteq V$ est un ensemble dominant pour G si et seulement si tout sommet de x est contenu dans cet ensemble, ou adjacent dans G à un sommet de cette ensemble. Formellement, $\forall x \in V - D, \exists y \in D|\{x, y\} \in E$.

*L'ensemble D est un **ensemble dominant connexe** pour G si et seulement s'il est dominant pour G, et que $G_{[D]}$ est connexe.*

Définition 10 (un point d'articulation, un séparateur) *:*
Soit $G = (V, E)$ un graphe connexe.

Un sommet a de V est un point d'articulation dans G si le graphe de G induit par $V - \{a\}$ n'est pas connexe.

*Un ensemble de sommets $S \subseteq V$ est un **séparateur** si le graphe de G induit par $V - S$ n'est pas connexe.*

Définition 11 (graphes sommet-disjoints, arête-disjoints) *:*
*Deux graphes $G = (V, E)$ et $H = (V', E')$ sont **sommet-disjoints** (resp. **arête-disjoints**) si et seulement si $V \cap V' = \emptyset$ (resp. $E \cap E' = \emptyset$). Nous dirons qu'un ensemble de graphes est **sommet-disjoints** (resp. **arête-disjoints**) si et seulement si ces graphes sont deux à deux sommet-disjoints (resp. arête-disjoints).*

Définition 12 (un graphe k-connexe) *:*
Un graphe $G = (V, E)$ est k-connexe s'il reste connexe après suppression d'un ensemble quelconque de $k-1$ sommets de E. Autrement dit, G est k-connexe si et seulement s'il existe au moins k chaînes sommet-disjointes pour toute paire de sommets $\{a, b\}|a, b \in V(G)$.

Quelques topologies utilisées dans ce mémoire :

Définition 13 (un arbre) *:*
*Un **arbre** est un graphe connexe sans cycle.*

La figure 1.3(a) montre un arbre de 15 sommets.

Définition 14 (un graphe planaire) *:*
Un ***graphe planaire*** *est un graphe qui peut se représenter sur un plan euclidien sans qu'aucune arête n'en croise une autre.*

Définition 15 (un graphe triangulé) *:*
Un graphe triangulé, parfois noté G_Δ, est un graphe dans lequel tout cycle de longueur strictement supérieure à 3 contient une corde.

Définition 16 (une grille $G_{M\times N}$, un tore $T_{M\times N}$) *:*
Une grille $G_{M\times N}$ est un graphe dans lequel :
- *les sommets sont les couples (a, b) pour tout $a \in [0, M-1]$ et tout $b \in [0, N-1]$,*
- *il existe une arête entre toute paire de sommets $\{(a, b), (c, d)\}|a, c \in [0, M-1], b, d \in [0, N-1]$ si et seulement si $|a-c| + |b-d| = 1$.*

Un tore $T_{M\times N}$ est une grille $G_{M\times N}$ à laquelle on a rajouté les ensembles d' arêtes $\{(a, 1), (a, N-1)\}|a \in [1, M-1]$ et $\{(1, b), (M-1, b)\}|b \in [1, N-1]$

Dans la suite, nous appelons M et N les dimensions de la grille $G_{M\times N}$. Une grille de dimensions quelconques est toujours un graphe planaire, à l'inverse d'un tore de mêmes dimensions, dont une représentation en 3 dimensions est donnée en figure 1.1. Les représentations adoptées dans ce mémoire pour les grilles et les tores sont montrées dans les figures 1.2(a) et 1.2(b). Dans une grille ou un tore, nous dirons qu'un sommet (a, b) a comme coordonnées $[a, b]$. Une ***ligne*** (resp. ***une colonne***) est alors l'ensemble des sommets de même ordonnée (resp. abscisse). Cette terminologie doit s'appliquer sur une représentation planaire d'un tore, et non sur le tore lui-même.

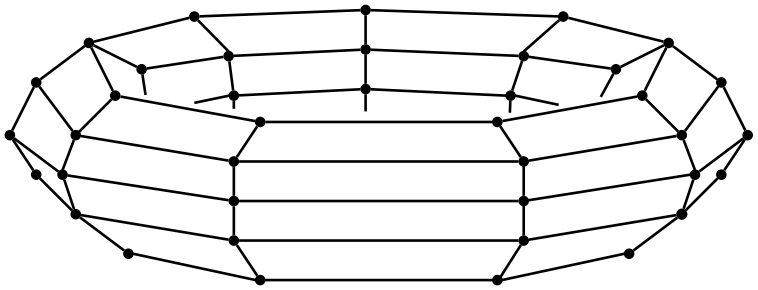


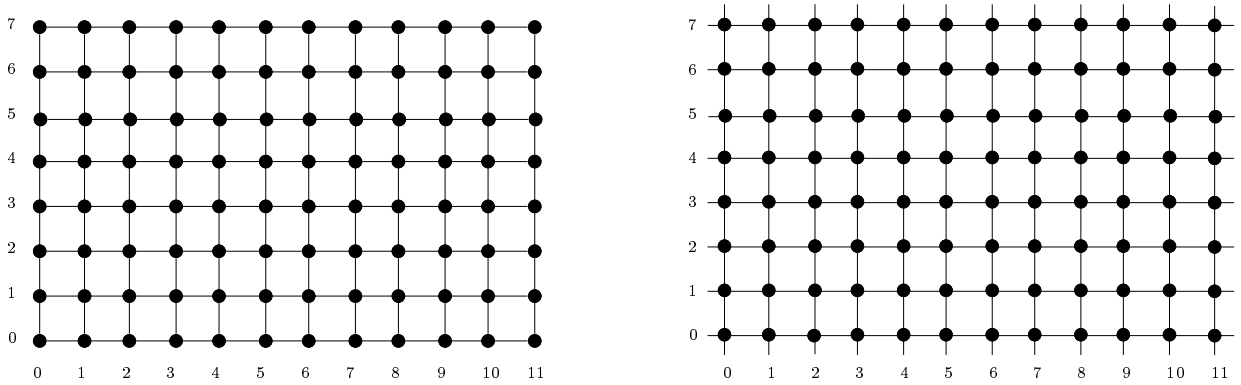
FIG. 1.1 – Une représentation en 3 dimensions d'un tore

1.2.2 Vocabulaire propre aux graphes orientés

Définition 17 (un graphe orienté) *:*
Un graphe orienté $G = (V, E)$ est la donnée d'un ensemble fini non vide de sommets V, et d'une relation binaire E définie sur V. Les éléments de E sont appelés des ***arcs****.*

La notation fonctionnelle $V(G)$ désigne l'ensemble des sommets d'un graphe orienté G, et $E(G)$ l'ensemble de ses arcs. À tout graphe orienté correspond un graphe non orienté, construit en remplaçant chaque arc par l'arête correspondante.

Définition 18 (un chemin) *:*
Un ***chemin*** *P est un graphe orienté dans lequel :*



(a) Représentation planaire d'une grille $G_{12\times 8}$ (b) Représentation planaire d'un tore $T_{12\times 8}$

FIG. 1.2 – Représentations planaires d'une grille et d'un tore

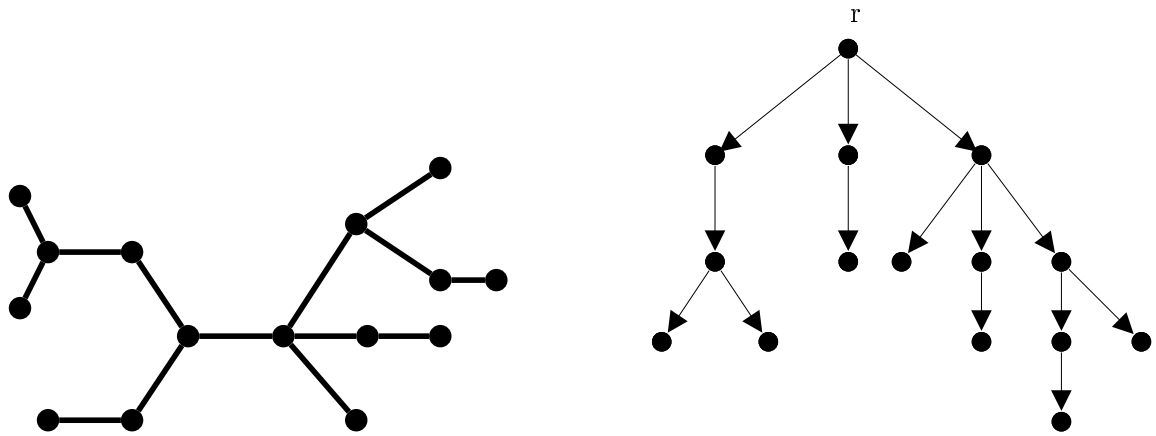
- $V(P) = \{x_0, x_1, x_2, \ldots, x_k\}$ *avec* $k \geq 0$,
- $E(P) = \big\{(x_i, x_{i+1}) | i \in [0, k-1]\big\}$.

Un chemin d'un graphe G est un sous-graphe de G qui est un chemin. Pour un chemin P donné :
- L'***origine*** et la ***destination*** de P, notées $ori(P)$ et $dest(P)$, sont respectivement les sommets x_0 et x_k. Ces termes s'étendent naturellement aux arcs, qui représentent des chemins à un arc.
- Les sommets x_1 à x_{k-1} sont les ***maillons internes*** de P.
- La ***longueur*** de P est égale à son nombre de d'arcs (ici k).

Définition 19 (une arborescence) *:*
Une arborescence, ou arbre enraciné, est un arbre dans lequel l'un de ses sommets se distingue des autres. Ce sommet s'appelle la ***racine*** *de l'arborescence. Les arcs sont alors orientés de la racine vers les feuilles*[1].

Un sommet d'une arborescence est souvent appelé un nœud[2]. Pour une arborescence donnée T de racine r :
- un nœud b quelconque sur l'unique chemin de r à a est appelé ***ancêtre*** de a. Nous dirons parfois que a et b sont liés par une ***relation de descendance***.
- Si b est un ancêtre de a, alors a est un **descendant** de b.
- Si le nœud b est ancêtre de a et lui est adjacent, alors b est le ***père*** de a (nous notons $pere(a) = b$), et a est un ***fils*** de b.
- La racine r est le seul nœud de T sans père.
- Deux nœuds a et b ayant le même père sont des **frères**.
- Un nœud sans fils est une **feuille**.
- Un nœud qui n'est pas une feuille est un ***nœud interne***.
- La **profondeur** d'un nœud a dans une arborescence de source r est la longueur du chemin allant de r à a, c'est à dire son nombre d'ancêtres. Nous la notons $profondeur_T(a)$.

[1] Le sommet le plus proche de la racine est l'origine de l'arc, le plus éloigné sa destination.
[2] Terme également utilisé pour désigner un participant dans un réseau !



(a) Un arbre (b) Une arborescence de racine r

FIG. 1.3 – Un arbre, et une arborescence (arbre enraciné)

– La **hauteur** de T, notée $hauteur(T)$, est la plus grande des profondeurs de ses feuilles.

La figure 1.3(b) montre une arborescence définie en enracinant l'arbre de la figure 1.3(a) sur le sommet r.

Définition 20 (un parcours en profondeur) *:*
Soit $G = (V, E)$ un graphe connexe orienté[3]*.*

Un parcours en profondeur depuis r est un parcours récursif des voisins d'un sommet r de G. Les sommets atteints sont successivement ajoutés au parcours. Un sommet est ignoré s'il a déjà été atteint précédemment.

Un parcours en profondeur d'un graphe G connexe décrit une arborescence particulière A, telle que :
- l'arborescence A est un sous-graphe de G, avec $V(A) = V(G)$,
- deux sommets adjacents dans G sont liés par une relation de descendance dans A,
- quels que soient a et b deux sommets sans relation de descendance dans A, appelons $S_A[a, b]$ l'ensemble des maillons internes de la chaîne de A reliant a et b. Alors dans le sous-graphe de G induit par $V(G) - S_A[a, b]$, les sommets a et b sont dans deux composantes connexes différentes.

1.3 Complexité algorithmique

La théorie de la complexité s'intéresse à l'étude de la difficulté de la résolution de problèmes pour lesquels on sait qu'il existe des solutions. L'objectif est de déterminer si ces solutions peuvent être trouvées efficacement ou pas, en se basant sur une estimation (théorique) des temps de calcul et des besoins en mémoire informatique. Nous présentons dans les sous-sections suivantes les principales notions de la théorie de la complexité.

[3]La définition s'applique aux graphes non orientés. En pareil cas, le graphe non orienté est ramené à un graphe orienté symétrique

1.3.1 Machines déterministes

Une ***machine déterministe*** est le modèle formel d'une machine de calcul (tel nos ordinateurs). Les deux principaux modèles de machines sont :

1. la machine de Turing,
2. la machine RAM (Random Access Machine)

Ces modèles supposent un ordinateur idéal avec une mémoire infinie, dans lequel une cellule mémoire est accessible en un temps constant, et les instructions sont exécutées séquentiellement.

Pour l'ensemble des algorithmes étudiés, les calculs de complexité en temps d'exécution supposent l'utilisation d'une machine de Turing. Une large part de la théorie de la complexité utilise ce modèle de machine, énoncé par Alan Turing en 1936 dans [Tur36] et largement repris dans la littérature. Le lecteur est renvoyé à [TG95, Tur36] pour une description détaillée de ce modèle de machines. Pour cette architecture, le codage d'une donnée est supposé ***raisonnable***. Par exemple :

1. Le codage d'un nombre n utilise $\log n$ bits.
2. Le codage d'un graphe[4] $G = (V, E)$ est polynomial en $|V|$.

La description formelle des algorithmes s'inspire du modèle RAM. Ce modèle présenté dans [AHU74] suppose un accès direct à la mémoire (il modélise la machine de Von Neumann[5] telle que ce dernier la décrivait en 1945). Nous ne ferons apparaître que les principales instructions d'un algorithme, afin de faire apparaître plus clairement sa complexité. Notons que les machines RAM sont équivalentes aux machines de Turing : tout algorithme qui peut s'exécuter sur l'un de ces modèles peut également s'exécuter sur l'autre, avec des temps d'exécution polynomialement liés.

1.3.2 Problèmes algorithmiques

Deux grandes familles de problèmes sont considérées en informatique :

1. les problèmes de décision
2. les problèmes d'optimisation

Définition 21 (un problème de décision) *:*
En algorithmique, un ***problème de décision*** *est une question mathématiquement définie portant sur des paramètres donnés, et dont la réponse est oui ou non.*

Nous illustrons cette définition par le problème de décision bien connu SET-COVER (problème 1.3.1), que nous réutilisons dans certains des travaux de ce mémoire.

Donnée	**:**	Une collection C de sous-ensembles d'un ensemble fini S, un entier k.
Question	**:**	Existe-t-il un sous-ensemble $C' \subseteq C$ de cardinalité k ou moins, tel que tout élément de S appartient à au moins un membre de C' ?

Problème de décision 1.3.1: Le problème SET-COVER

[4] le codage d'un graphe utilise de l'ordre de $|V| + |E|$ cases mémoires, une case mémoire est un emplacement pouvant stocker un entier.

[5] Von Neumann est considéré comme le père des ordinateurs modernes

Définition 22 (un problème d'optimisation) *:*
*Un **problème d'optimisation** consiste à trouver la meilleure solution dans un ensemble des solutions réalisables. Cet ensemble peut être fini mais compter un très grand nombre d'éléments. Il est décrit de manière implicite, par une liste de contraintes que doivent satisfaire les solutions réalisables.*

*La **qualité** d'une solution s'évalue par une **fonction objectif**; la meilleure solution est celle dont la qualité correspond le plus à un objectif recherché (généralement, maximiser ou minimiser une quantité).*

Nous présentons MIN-SET-COVER (1.3.1), la version optimisation du problème SET-COVER sur la cardinalité de l'ensemble solution.

Donnée	:	Une collection C de sous-ensembles d'un ensemble fini S.
Résultat	:	Un sous-ensemble $C' \subseteq C$, tel que tout élément de S appartient au moins à un membre de C'.
Objectif	:	Minimiser $\|C\|$

Problème d'optimisation 1.3.1: Le problème MIN-SET-COVER

À chaque problème d'optimisation correspond un problème de décision. La résolution d'un problème d'optimisation peut s'effectuer en résolvant plusieurs fois le problème de décision associé, en faisant varier de manière dichotomique le paramètre de qualité d'une solution[6]. Dans notre définition de SET-COVER, ce paramètre est k. Une pratique commune consiste à extraire ce paramètre de la donnée, et à définir un problème de décision pour chaque valeur de son domaine. Ainsi dans notre exemple nous définissons les problèmes de décision 1-SET-COVER, 2-SET-COVER, 3-SET-COVER, ... et les désignons comme la famille des problèmes de décision k-SET-COVER.

La donnée d'un problème de décision ou d'optimisation est appelée ***une instance*** du problème. Nous utilisons ce terme par la suite.

1.3.3 Classes de complexité

Nous présentons quelques classes de complexité dans lesquelles les problèmes de décision et d'optimisation se situent. Un problème de décision est dans $\boldsymbol{P}$ s'il peut être résolu par un algorithme déterministe en un temps polynomial par rapport à la taille de l'instance. Le problème est dit ***polynomial***. La classe $\boldsymbol{NP}$ est la classe des problèmes de décision pour lesquels il existe un algorithme polynomial déterministe $\mathcal{A}$ et, pour chaque instance positive[7] I un ***certificat*** $c(I)$, tels que l'algorithme $\mathcal{A}$ sait reconnaître à l'aide de $c(I)$ que la réponse est oui pour I. Intuitivement, les problèmes dans NP sont tous les problèmes qui peuvent être résolus en énumérant l'ensemble des solutions possibles et en les testant avec un algorithme polynomial. Le nombre de solutions peut néanmoins être exponentiellement grand.

Par définition $P \subseteq NP$. On ne sait pas si cette inclusion est stricte, ou si $P = NP$. Une grande part de la théorie de la complexité s'est construite sous l'hypothèse que $P \neq NP$.

[6] Ceci n'est possible qu'en cas de monotonie du problème de décision. Nous assurons cette monotonie en recherchant un paramètre de qualité inférieur ou égal à une constante donnée, et non simplement égal à cette constante.

[7] et seulement pour les instances positives.

La classe ***NP-complet*** est une sous-classe des problèmes NP. Un problème de décision est NP-Complet lorsque tous les problèmes de décision appartenant à NP lui sont réductibles par un algorithme polynomial.

Un problème de décision est ***NP-Difficile*** si tous les problèmes d'optimisation NP-difficiles lui sont réductibles par un algorithme polynomial.

Si $P \neq NP$, alors tout problème de décision NP-complet a une version optimisation qui est NP-difficile, et il n'existe pas d'algorithme polynomial pour le résoudre.

Pour les problèmes d'optimisation, il est parfois possible de calculer en temps polynomial des solutions avec des ***garanties de performances***, c'est à dire des solutions dont le coût est comparable avec le coût optimum. Ces solutions peuvent être caractérisées par un ***rapport à l'optimum.***

Définition 23 (rapport à l'optimum) *:*
Soient une instance I d'un problème d'optimisation, une solution S pour I, et une fonction objectif val. Alors le rapport à l'optimum de S pour I est égal à :

$$\max\left(\frac{val(S)}{val(S^*)}; \frac{val(S^*)}{val(S)}\right)$$

où S^* est la solution optimale pour I.

Pour un problème d'optimisation donné, nous dirons qu'un algorithme polynomial $\mathcal{A}$ est $l(n)$-approché, si pour toute instance I du problème, $\mathcal{A}$ renvoie une solution S dont le rapport à l'optimum est borné supérieurement par $l(n)$. Lorsqu'il existe un algorithme polynomial k-approché avec k constant, le problème d'optimisation appartient à une classe particulière, la classe ***APX***.

1.3.4 Réduction de problèmes

La réduction de problème en théorie de la complexité consiste à réduire un problème connu vers le problème que l'on souhaite classer. Une réduction d'un problème P_1 vers un problème P_2 est une transformation de toute instance de P_1 en une instance de P_2, de sorte que toute solution à P_1 puisse induire une solution à P_2. Cette transformation doit être polynomiale en temps et mémoire selon la taille de P_1. Ainsi, il y a une relation de difficulté de résolution entre ces deux problèmes.

Si P_1 est polynomial, alors P_2 le sera aussi, et inversement, si P_1 est NP-complet, alors P_2 le sera de même. Le théorème de Cook a classé en premier le problème SAT comme NP-complet. Il affirme qu'il est possible de réduire tous les problèmes NP-complets au problème SAT, prouvant ainsi leur NP-complétude.

Première partie

Diffusion dans un réseau radio multi-sauts

Introduction

Le travail présenté dans cette première partie cible le problème de la diffusion dans les réseaux sans fil. Diffuser un message depuis un nœud source vers l'ensemble des participants du réseau est une opération essentielle dans de nombreux protocoles de routage. La diffusion s'opère efficacement dans les réseaux filaires grâce à des techniques d'inondation bien connues [DM78]. L'adaptation de ces méthodes dans les réseaux sans fil où les nœuds partagent la même fréquence se révèle désastreuse : dans un environnement où deux émissions géographiquement proches s'interfèrent mutuellement, le trafic généré par inondation conduit à une saturation du média radio, et à un effondrement des performances du réseau.

Nous proposons dans cette partie l'étude du problème de diffusion dans un réseau sans fil multi-sauts, sujet aux contraintes de communications propres à l'utilisation du média radio, et en supposant que les nœuds ne disposent pas de couche MAC idéale[8]. Le problème se décline dans un modèle de communication synchrone - dans lequel les émissions des nœuds s'effectuent par étapes de durée égale - et asynchrone. En supposant que la topologie du réseau est connue (par un superviseur ou intrinsèque à chaque nœud), nous nous proposons de définir des stratégies de diffusion et d'étudier leurs performances en terme de temps de diffusion et nombre d'émissions.

Organisation de cette partie

Le chapitre 2 introduit le problème de diffusion dans les réseaux sans fil. Nous y proposons un formalisme permettant une classification des contextes dans lesquels ce problème se décline. Forts de ce formalisme nous dressons un état de l'art sur les méthodes, bornes et algorithmes existants, et dessinons les limites de notre étude. Nous proposons enfin une nouvelle stratégie de diffusion originale, dite ***stratégie en arborescence***.

Le chapitre 3 présente les premiers résultats de complexité sur les stratégies introduites. Nous y établissons notamment les conditions d'existence d'une solution pour une stratégie donnée, et nous intéressons aux coûts minimums en temps et en émissions générées par chacune d'entre elles. Nous montrons que minimiser l'un ou l'autre de ces coûts est un problème NP-difficile, et regardons l'impact de la topologie sur la complexité de ces problèmes.

Le chapitre 4 est dédié à la résolution du problème de diffusion. Nous y présentons un nouvel outil, la mv-décomposition, qui va servir de support à l'élaboration d'heuristiques avec garantie de performance lorsque les émissions sont synchrones. Nous étudions également le problème de diffu-

[8] Une couche MAC idéale suppose que deux nœuds en contention n'accèdent jamais simultanément au canal

sion dans la grille et le tore dans un modèle asynchrone.

Le chapitre 5 conclut notre étude par des expérimentations destinées à comparer les coûts optimaux entre différentes stratégies de diffusion, dont notre stratégie en arborescence. Les résultats obtenus nous permettent d'établir la pertinence de notre stratégie dans des instances générées aléatoirement.

Chapitre 2

Le problème de la diffusion

La première section de ce chapitre présente le problème de la diffusion, et explique la difficulté à diffuser efficacement un message dans les réseaux radios. La seconde section définit le formalisme employé, ainsi que différentes stratégies que nous considérons dans cette partie.

2.1 Introduction au problème de diffusion

De la diffusion dans les réseaux filaires...

Une ***diffusion*** se définit comme l'« *Action de propager une idée, des connaissances, des techniques ou de distribuer un bien dans un large public, et le résultat de cette action* » (définition du Trésor de la Langue Française informatisé). Pouvoir diffuser une information émise depuis une source vers l'ensemble des entités d'un réseau informatique est une opération essentielle, utilisée intrinsèquement dans de nombreux protocoles. Par exemple, les protocoles de routage à état de lien utilisent la diffusion lorsqu'un routeur doit notifier un changement d'état de liens aux autres routeurs.

La méthode de diffusion la plus élémentaire est ***l'inondation aveugle***[1] [DM78] (en anglais *blind flooding*). Son fonctionnement est le suivant :

- Le nœud initiant l'inondation envoie le paquet à tous ses voisins directs.
- Si un nœud quelconque du réseau reçoit le paquet, il le rediffuse à tous ses voisins.

L'instauration d'un TTL permet d'éviter des situations de bouclage, limite la duplication des messages, et garantit l'arrêt de la diffusion. Les nœuds peuvent être amenés à appliquer durant l'inondation certains traitements de contrôle, afin de minimiser le nombre de messages émis (estampillage, ...). Bien qu'un peu coûteux dans le cas où le réseau est très dense, les mécanismes d'inondation aveugle se déroulent relativement bien dans les réseaux filaires.

Il est important de souligner que les procédés d'inondations classiques génèrent beaucoup de messages redondants. Chaque nœud peut potentiellement recevoir le message autant de fois qu'il y a de liens. Cependant cette observation ne pose guère de problème dans les réseaux filaires (qui utilisent pour la plupart le protocole bas niveau ethernet) : un routeur possède un canal dédié par voisin ce qui permet la réception simultanée de messages provenant de deux routeurs adjacents. Seuls les nœuds partageant le même canal filaire (réseaux en architecture de bus, utilisation de hubs,

[1] également nommée « diffusion pure » dans certains ouvrages

...) sont susceptibles de générer des collisions s'ils émettent un message simultanément. En pareille situation, l'utilisation d'un protocole d'accès multiples avec surveillance de porteuse et détection de collision (CSMA/CD) permet de corriger les transmissions infructueuses : ce protocole s'appuie sur la capacité d'un nœud à pouvoir écouter le canal alors qu'il procède à une transmission. Par l'utilisation de CSMA/CD, tout nœud est en mesure de savoir si son émission a été correctement réceptionnée.

... à la difficulté de diffuser dans les réseaux radio

Caractéristiques du média radio :

Dans un réseau radio, les nœuds communiquent entre eux en utilisant une ***onde radio-électrique*** comme support de transmission du message. La transmission d'un message s'effectue sur une fréquence spécifique et ne peut être captée que dans une zone suffisamment proche de la source d'émission (à cause des phénomènes d'amortissement d'ondes) dite **zone d'émission**. Deux transmissions de fréquences trop voisines et de sources géographiquement proches peuvent créer une ***interférence***. La région où les deux transmissions se superposent est appelée ***zone d'interférence***. Dans les réseaux où les nœuds utilisent tous la même fréquence, l'utilisation du média radio dessine un contexte plus hostile qu'ethernet : un nœud émetteur communiquant avec un récepteur est physiquement incapable de détecter si l'information qu'il a transmise a été correctement reçue. Sa vision locale ne lui permet pas de connaître l'existence de transmissions en concurrence avec la sienne. Ce problème est celui du ***nœud caché*** pour lequel des protocoles de prévention de collision comme CSMA/CA[2] se révèlent inefficaces. L'utilisation de mécanismes d'esquives de collision basés sur un principe d'accusé de réception réciproque entre l'émetteur et le récepteur (CSMA/CA avec RTS/CTS[3]) est nécessaire pour garantir l'acheminement d'une transmission. Mais l'utilisation de ces accusés est susceptible de saturer le média radio et de détériorer des performances du réseau en terme de bande passante.

Deux grandes familles de réseaux radio :

On distingue deux grandes familles de réseaux radio :

1. ***les réseaux à fréquences multiples*** (MFN : Multi-Frequency Network). Ces réseaux sont généralement déployés en mode infrastructure sur le fonctionnement suivant :
   - Un réseau couvre une surface subdivisée en zones, chacune couverte par une station de base. Les stations de base sont elle-mêmes organisées en réseau. Les nœuds du réseau ne communiquent pas directement, mais dialoguent avec la station de base de la zone où ils se situent.
   - Lorsqu'un nœud souhaite établir une communication vers la station de base, cette dernière lui attribue, le temps de la transmission, deux fréquences propres : une dédiée à l'émission vers la station de base (fréquence montante), et une pour la réception des messages de la station de base (fréquence descendante). Aucun autre nœud voisin n'utilise alors l'une de ces fréquences à ce moment-là.
   - À la fin d'une communication, les fréquences utilisées par un nœud sont libérées.

[2]CSMA/CA se présente comme la version de CSMA/CD adaptée aux réseaux sans fil. Lorsque deux transmetteurs concurrents sont à portée l'un de l'autre, CSMA/CA permet d'éviter les transmissions simultanées.

[3]Ready To Send / Clear To Send

Les réseaux de téléphonie mobile GSM fonctionnent sur ce principe. Le fait que chaque nœud possède sa propre fréquence d'émission/réception évite les interférences et offre des performances en bande passante intéressantes. Cependant, le nombre de participants par zone est clairement limité par le nombre de fréquences que peut attribuer une station de base. De plus, l'attribution et l'utilisation des fréquences est soumise à une règlementation stricte propre à chaque pays.

2. ***les réseaux à fréquence unique*** (SFN : single-frequency network) : Les nœuds émettent et reçoivent sur une fréquence commune, ce qui ne limite plus le nombre de participants. Mais l'apparition de phénomènes d'interférences dans des zones à forte densité risque d'altérer les performances du réseau. Les communications entre nœuds peuvent s'effectuer :
   - en mode infrastructure : les nœuds s'appuient sur une station de base fixe qui agit comme un concentrateur.
   - en mode ad-hoc : les nœuds s'auto-organisent sans architecture fixe afin de constituer un réseau point à point. Si deux des stations du réseau sont hors de portée l'une de l'autre, elles ne pourront pas communiquer, même si elles sont à portée d'une troisième station. En effet dans sa version actuelle le mode ad-hoc de 802.11 ne propose pas de système de routage permettant d'acheminer les trames d'une station à une autre, en passant par une station intermédiaire.

Dans la suite de ce mémoire, nous ne considérons que des réseaux à fréquence unique, qui regroupent notamment les réseaux utilisant la norme 802.11.

Diffusion dans les réseaux à fréquence unique

Les procédés d'inondation créent dans les réseaux sans fil un problème de redondance, de contention et de collision, connu sous le problème de la tempête d'inondation [TNCS02]. L'apparition de ce problème est inhérent à l'utilisation du médium radio comme support de communication. Le fait qu'un nœud reçoive un message diffusé autant de fois qu'il a de voisins pose ici un sérieux problème d'accès au média et de saturation du canal.

Le problème de la diffusion a été très étudié sous l'hypothèse d'une couche MAC idéale : les collisions sont inexistantes ou corrigées par un protocole bas niveau, type 802.11 (utilisé par exemple dans les réseaux ad-hoc ou de capteurs). Les nœuds de ces réseaux peuvent être des systèmes autonomes ayant une source d'énergie embarquée et limitée. Une attention particulière est alors apportée à la gestion de l'énergie. La diffusion se révèle ici efficace si elle minimise le nombre d'émissions nécessaires à la propagation du message vers les autres participants. Le temps nécessaire à la diffusion, ou le nombre de nœuds mobilisés sont quelques-uns des critères pouvant justifier la pertinence d'une solution de diffusion.

Dans cette partie, nous ciblons notre étude sur les réseaux sans fil pour lesquels les nœuds ne disposent pas de couche MAC idéale et ont un modèle d'antennes omnidirectionnelles avec puissance d'émission fixe. Les réseaux que nous considérons supposent les caractéristiques suivantes (proposées notamment dans [HHL86, FL94, Che04]) :

- ***Δ-port émission*** : pour une transmission donnée, tous les nœuds voisins de la source sont susceptibles de recevoir le message.

- 1-***port réception*** : un nœud ne peut recevoir qu'un seul message à la fois. La réception simultanée de deux ou plusieurs messages entraîne des interférences et une altération des messages envoyés.
- ***Half-duplex*** : un nœud du réseau ne peut à la fois émettre un message et en recevoir un autre. Cependant il peut choisir d'ignorer une réception pour procéder à une émission.

Dans ce modèle, le fait que le réseau soit en 1-port réception rend les techniques d'inondation peu efficaces.

2.2 Modélisation et formalisme

Cette section décrit le modèle que nous utilisons et introduit un formalisme permettant de classer les différentes approches que nous envisageons.

2.2.1 Quelques définitions et propriétés utiles

Un réseau radio multi-sauts peut se modéliser par un graphe orienté $G = (V, E)$, où V représente l'ensemble des nœuds du réseau, et E l'ensemble des liaisons radios entre ces nœuds. Sauf précision, nous considérons que si un nœud x peut communiquer vers un nœud y, alors y peut communiquer avec x. Nous en déduisons que le graphe orienté est symétrique et nous l'assimilons à un graphe non orienté. Par la suite, l'utilisation d'un graphe $G = (V, E)$ sans précision implique que ce dernier peut être orienté ou non. Enfin, nous supposons que chacun des réseaux considérés est ***connexe***, condition trivialement nécessaire à l'accomplissement d'une diffusion, et possède au moins deux nœuds.

Le long de cette partie, nous restreignons notre étude à la diffusion d'un seul message émis d'une source unique. Une instance du problème de la diffusion dans sa forme générale se décrit par un couple (G, s), où G représente le graphe du réseau, et s le nœud émetteur du message, avec $s \in V(G)$. Toute stratégie dont l'exécution aboutit à la diffusion d'un message depuis s vers l'ensemble des autres nœuds du graphe G, est appelée ***stratégie de diffusion valide*** sur l'instance (G, s).

Nous introduisons le vocabulaire suivant :

Définition 24 (Transmission correcte) *:*
*Soient x et y deux nœuds d'un réseau sans fil. Le nœud x reçoit une **transmission correcte** (ou **réception correcte**) de y si et seulement si elle n'a pas été altérée par des interférences : aucun autre nœud voisin de x n'a émis en même temps que y.*

Définition 25 (Durée de transmission) *:*
*Soit une transmission directe[4] d'un nœud x vers un nœud y. Nous définissons la **durée de transmission** comme l'intervalle de temps séparant le début de l'émission de x et la fin de la réception par y.*

Définition 26 (transmetteur, récepteur) *:*
Soient un graphe $G = (V, E)$, un nœud source s, et S une stratégie de diffusion sur G :

Nous utilisons le vocabulaire suivant :

[4] ou transmission à un saut.

- *Tout nœud de G désigné dans S pour émettre un message est un **transmetteur**. On note T l'ensemble des transmetteurs de G.*
- *Tout nœud de G qui reçoit dans S un message sans avoir à le relayer est un **récepteur**. On note R l'ensemble des récepteurs de G.*

Propriété 1 *:*
Soient un graphe $G = (V, E)$, un nœud source s, et S une stratégie de diffusion sur G.

1. *Les ensembles T et R partitionnent V.*
2. *La source du message s appartient à T.*
3. *L'ensemble T forme un ensemble dominant connexe dans G.*

Les points 1 et 2 de la propriété 1 se déduisent de la définition 26. Tout nœud transmetteur t doit pouvoir recevoir le message depuis s, d'où l'existence d'une chaîne de transmetteurs entre t et s. Le point 3 est alors immédiat.

La portée des définitions suivantes se limite dans ce mémoire à la partie consacrée à la diffusion. Notamment la notion d'***assignation de dates*** sera redéfinie autrement dans la seconde partie présentant le problème de satisfaction de requêtes.

Définition 27 (ordonnancement d'émissions, assignation de dates) *:*
Soient un graphe $G = (V, E)$, et un message m.

*Un ordonnancement de dates d'émissions des sommets de G, ou **ordonnancement d'émissions**, est une application f qui à tout couple (x, i) où $x \in V(G)$ et $i \in \mathbb{N}^*$ associe une valeur dans $\{0, 1\}$, que l'on interprète comme suit :*
- *$f(x, i) = 1$ si le nœud x du réseau émet à l'étape i le message m,*
- *$f(x, i) = 0$ autrement.*

Par la suite et selon certaines hypothèses, l'ordonnancement d'émissions est utilisé pour décrire des stratégies de diffusion valides.

*Une assignation de dates d'émissions des sommets de G, appelée **assignation de dates**, est un ordonnancement d'émissions dans lequel chaque nœud x émet au plus 1 fois le message m. Lorsqu'une stratégie peut se définir par un ordonnancement d'émission, l'assignation de dates remplace ce dernier si le modèle considéré impose qu'un nœud ne peut émettre au plus qu'une fois le message.*

*La **durée d'un ordonnancement** f (resp. **durée d'une assignation**), notée $duree(f)$, est la plus grande date à laquelle un sommet émet. Formellement :*

$$duree(f) = \max\{i | f(x, i) = 1, \forall x \in V(G)\}$$

*Le **nombre d'émissions d'un ordonnancement** f (resp. **nombre d'émissions d'une assignation**), noté $emissions(f)$, est égal à :*

$$emissions(f) = \sum_{x \in V(G)} \sum_{i \in \mathbb{N}^*} f(x, i)$$

Remarque 1 *:*
Une assignation de dates peut se définir comme une application associant à tout $x \in G$ une date d'émission dans $i \in \mathbb{N}$. La date 0 désigne alors un sommet récepteur. Cette seconde définition présente un avantage considérable en terme de mémoire, puisqu'avec ce codage $O(n)$ bits sont utilisés pour coder l'ensemble des dates d'émissions d'une stratégie. En utilisant la formulation de la définition 27, il faut $O(n \times D)$, D étant la plus grande date affectée.

Nous avons cependant privilégié une définition selon une application de $V(G) \times \mathbb{N}^*$ dans $\{0,1\}$ par souci de clarté, et pour assurer une continuité avec la définition d'ordonnancement. Nous pouvons ainsi établir des relations plus directes entre stratégies. En pratique, il convient de représenter un résultat selon le format le plus compact.

2.2.2 Modèle de communication

Nous envisageons plusieurs approches au problème de diffusion. Pour classer et référencer ces approches, nous proposons une notation à cinq champs inspirée de la classification de Graham pour les problèmes d´ordonnancement [Gra66], et définie comme suit :

Définition 28 (modèle de communication) *:*
Soit un problème de communication sur un réseau donné.

*Un **modèle de communication** offre une notation simplifiée reprenant les principaux éléments et caractéristiques du problème. Nous définissons un tel modèle sous la forme d'une notation à cinq champs **[SC,TT,CÉ,CR,To]** renseignant les éléments suivants :*
- ***S**chéma de **C**ommunication*
- ***T**ype de **T**ransmissions*
- ***C**ontexte d'**É**mission*
- ***C**ontexte de **R**éception*
- ***To**pologie*

L'objet et le domaine de chacun de ces champs est expliqué ci-après. Un champ vide prend la valeur définie par défaut. Ce modèle ne se limite pas au problème de la diffusion, qui constitue cependant l'unique objet de cette première partie.

Schéma de communication :

Le schéma de communication désigne le problème de communication que nous considérons. Parmi les principaux schémas étudiés dans la littérature citons :
- **[Diff]** ***la Diffusion*** : un message émis depuis un nœud source s doit être acheminé vers l'ensemble des nœuds du réseau.
- **[CommGrp]** ***la Communication de groupes***, ou diffusion Partielle (en anglais *Multicast*) : un message émis depuis un nœud source s doit être acheminé vers un sous-ensemble des nœuds du réseau.
- **[Rassmblt]** ***le Rassemblement*** (en anglais *Gathering*) : chaque nœud du réseau doit envoyer un message vers un nœud collecte t.
- **[Requêtes]** ***la Satisfaction de Requêtes*** : un ensemble de requêtes doit être satisfait, une requête $r_i = (s_i, t_i)$ étant satisfaite si un message émis depuis s_i a transité vers sa destination t_i.

Type de transmissions :

Deux Types de transmissions sont considérés :

1. **[Sync]** ***transmissions synchrones*** : le temps est divisé en étapes de longueurs égales (en anglais *slots*). Ces étapes sont numérotées dans $\mathbb{N}^*$. Une communication de x vers y entraîne une émission du sommet x au début de l'étape t, reçue et traitée par y avant la fin de l'étape. Un nœud ne peut pas émettre plus d'une fois durant une étape et les messages ne sont pas multiplexables : une émission contient exactement un message.
2. **[Async]** ***transmissions asynchrones*** : dans ce type de transmissions, le délai de traitement consécutif à la réception d'un message est supposé non borné. Ce délai résulte de l'algorithmique sous-jacente au traitement du message (passage à travers des différentes couches du modèle réseau, ...). L'absence de borne précise sur ce délai empêche tout découpage temporel en étapes. Une hypothèse supplémentaire doit être faite afin de pouvoir exploiter efficacement ce modèle. Nous supposons ici que les temps de transmission sont tous identiques. Cette hypothèse se justifie du fait que les nœuds émettent des messages de taille et contenu similaires : le champs de données est le même pour chaque émission ; seul le champs d'en-tête, de faible taille, est propre à une transmission.

Contexte d'émission :

Le contexte d'émission définit les caractéristiques des nœuds transmetteurs du réseau en terme d'émission. Lorsqu'un transmetteur reçoit un message sans interférence et doit le relayer, nous appliquons l'une des politiques suivantes :

- **[]** ***Immédiate*** : le transmetteur n'a aucune capacité. Il procède à l'émission du message immédiatement après l'avoir réceptionné (par exemple : à l'étape suivante dans un modèle de communication synchrone).
- **[Tmp]** ***Temporisation*** : le transmetteur peut temporiser l'unique émission du message durant un nombre indéterminé d'étapes (dans un modèle de communication synchrone) ou au bout d'une durée déterminée (dans un modèle de communication asynchrone).
- **[Tmp-EM]** ***Temporisation et émissions Multiples*** : le transmetteur peut effectuer plusieurs transmissions, en instaurant éventuellement un temps d'attente entre deux émissions (désigné par un nombre d'étapes dans un modèle synchrone, une durée autrement).

Dans le quintuplet décrivant le modèle de communications, nous supposons les émissions ***immédiates*** si aucune valeur ne renseigne le contexte d'émission.

Contexte de réception :

Le contexte de réception définit les différentes politiques que peut adopter un nœud lorsqu'il reçoit une transmission sans interférence. La propagation du message dans le réseau peut être contrôlée en définissant pour chaque nœud un sous-ensemble de ses voisins, nommé ***ensemble de pères***. Cette politique s'applique exclusivement aux nœuds qui sont désignés comme transmetteurs dans une solution S :

1. Si j est un transmetteur dans S : une réception sans interférence est ***validée*** si et seulement si son émetteur i est inclus dans l'ensemble des pères de j. Elle est ignorée dans le cas contraire. Cette politique permet, en association avec le contexte d'émission défini ci-avant, de lier causalement des émissions afin de prévenir les interférences.

2. Si j est un récepteur dans S : la première transmission correcte reçue est validée.

Il peut paraître absurde d'ignorer un message bien reçu et de vouloir le recevoir de quelqu'un d'autre. Cette approche se justifie par le fait que la décision d'émission peut être une conséquence immédiate d'une réception : par exemple dans des modèles de communication où les émissions doivent être immédiatement consécutives à des réceptions valides. Valider une réception selon son émetteur permet alors un contrôle sur la propagation du message.

Nous considérons les contextes de réception suivants :

- **[]** ***Réception neutre*** : l'ensemble des pères de chaque transmetteur est égal à son voisinage. Une réception correcte est validée quelle qu'en soit sa provenance.
- **[Arbo]** ***Réception en arborescence*** : l'ensemble des pères de chaque transmetteur i est réduit à un élément désigné comme le ***père du nœud***, et noté $pere(i)$. La structure engendrée par les relations de parenté est une arborescence[5] enracinée en la source du message s.

Un contexte de réception non renseigné suppose une réception neutre.

Topologie :

Le dernier champ d'un modèle de communications indique les particularités topologiques du réseau. De nombreuses topologies peuvent être envisagées : ***arbre***, ***grille***, ***graphes triangulés***, ***disques unitaires***, ... En l'absence de valeur, le graphe du réseau est supposé connexe quelconque.

Tableau récapitulatif :

CHAMP :	DOMAINE :			
Schéma de **C**ommunication	**[Diff]**	**[CommGrp]**	**[Rassmblt]**	**[Requêtes]**
Type de **T**ransmissions	**[Sync]**	**[Async]**		
Contexte d'**E**mission	[]	**[Tmp]**	**[Tmp-EM]**	
Contexte de **R**éception	[]	**[Arbre]**		
Topologie	[]	**[arbre]**	**[grille]**	**[triangulé]**, ...

TAB. 2.1 – Définition d'un modèle de communication : tableau récapitulatif

Notre notation permet une meilleure classification des travaux existants. Par exemple [BP05, BGK+06b, BGK+06a] réfèrent au modèle [Rassmblt, Sync, Tmp-EM„], et plus précisément aux modèles [Rassmblt, Sync, Tmp-EM„ grille] pour [BP05] et [Rassmblt, Sync, Tmp-EM„ chaîne] pour [BGK+06a]. De nombreux auteurs ont également étudié le modèle [Diff,Sync,Tmp-EM„], pour lequel une bibliographie plus exhaustive est présentée en sous-section 2.3.1. Enfin, la seconde partie de ce mémoire se focalise sur le modèle [Requêtes, Sync, Tmp„].

Nous présentons dans la section suivante quelques modèles pour lesquels le schéma de communication est la diffusion.

[5] Nous verrons qu'en fonction du contexte d'émission et du type de transmissions, cette arborescence présente des caractéristiques spécifiques.

2.3 Le problème de la diffusion : modèles, objectifs et bibliographie

Nous présentons dans cette section le problème de diffusion dans un réseau radio $G = (V, E)$ d'un message m émis par une source unique s dans différents modèles de communications, et suivant deux objectifs distincts. Nous étudions quatre modèles de communication, et définissons pour chacun deux problèmes différenciés uniquement par leur fonction objectif :

- minimiser les coûts en temps d'une part,
- minimiser les coûts en nombre d'émissions d'autre part.

Dans les deux premières sous-sections, nous présentons deux modèles de communication très étudiés dans la littérature : [Diff,Sync,Tmp-EM„] et [Diff,Sync,Tmp„]. Résoudre le problème de diffusion dans ces approches consiste essentiellement à établir des ordonnancement d'émissions des sommets de G.

Dans les sous-sections 2.3.3 et 2.3.4, nous introduisons de nouveaux modèles [Diff,Sync„Arbo,] et [Diff,Async„Arbo,]. À notre connaissance, ces modèles n'ont encore jamais été étudiés. Nous les étudions afin de savoir s'ils peuvent constituer une alternative intéressante aux stratégies d'ordonnancement classiques.

2.3.1 Le modèle [Diff,Sync,Tmp-EM„]

Description et problèmes considérés

Considérons le modèle [Diff,Sync,Tmp-EM„] : le problème de diffusion est abordé dans un environnement où les émissions sont synchrones, et les nœuds peuvent émettre plusieurs fois à différentes étapes. Un nœud considère un message comme acquis dès la première réception correcte, quel qu'en soit l'émetteur. Une stratégie de diffusion valide d'un message m dans ce modèle se représente par un ***ordonnancement cohérent pour [Diff,Sync,Tmp-EM„]*** défini comme suit :

Définition 29 (ordonnancement cohérent pour [Diff,Sync,Tmp-EM„]) *:*
Soient un graphe $G = (V, E)$, une source unique $s \in V$ possédant un message m, et un ordonnancement d'émissions f dans G.

*Cet ordonnancement est **cohérent pour [Diff,Sync,Tmp-EM„]** si et seulement si la stratégie de diffusion qu'il décrit est valide, c'est-à-dire que :*

- *la source s émet m à l'étape 1,*
- *un transmetteur relaie m à une étape j s'il a pu le recevoir correctement à une étape précédente $i < j$,*
- *tout sommet de G doit recevoir le message.*

Formellement, f est cohérent pour [Diff,Sync,Tmp-EM„] si et seulement si :

1. $f(s, 1) = 1$,
2. $\forall x \in V - \{s\}, f(x, 1) = 0$
3. $\forall x \in V - \{s\}, f(x, j) = 1 \rightarrow \exists i < j \mid \sum_{y \in N(x)} f(y, i) = 1$,
4. $\forall x \in V, \exists i > 0 | \sum_{y \in N(x)} f(y, i) = 1$.

Donnée	:	un graphe $G = (V, E)$, un sommet $s \in V$, un entier k
Question	:	Existe-t-il un ordonnancement f cohérent pour [Diff,Sync,Tmp-EM$_\infty$] et tel que $duree(f) \leq k$?

Problème de décision 2.3.1: Le problème [Diff,Sync,Tmp-EM$_\infty$]-Temps

Nous introduisons le problème de décision 2.3.1 ***[Diff,Sync,Tmp-EM$_\infty$]-Temps*** :

Nous appelons [Diff,Sync,Tmp-EM$_\infty$]-minTemps la version optimisation de [Diff,Sync,Tmp-EM$_\infty$]-Temps qui consiste à trouver un ordonnancement f minimisant $duree(f)$.

Remarque 2 :
Nous ne nous intéressons pas au problème de minimisation du nombre d'émissions dans le modèle [Diff,Sync,Tmp-EM$_\infty$] : si l'on cherche à minimiser uniquement le nombre d'émissions, le nombre d'étapes utilisées n'est plus une contrainte critique, et peut être de l'ordre du nombre d'émissions. Soit S une stratégie de diffusion valide pour [Diff,Sync,Tmp-EM$_\infty$] utilisant k émissions. On peut en déduire une nouvelle stratégie dans laquelle une seule émission a lieu à chaque étape. Il devient alors inutile qu'un nœud émette à plusieurs étapes, puisque l'ensemble de ses voisins aura reçu le message dès la première transmission. Il en résulte une stratégie S' de coût $k' \leq k$ valide pour le modèle [Diff,Sync,Tmp$_\infty$] (présenté en section 2.3.2). Il est donc plus pertinent d'étudier la minimisation du nombre d'émissions dans ce modèle. Nous verrons par la suite que ce problème se réduit alors à un problème très étudié de la théorie des graphes : la recherche d'un ensemble dominant connexe de cardinalité minimum.

État de l'art sur [Diff,Sync,Tmp-EM$_\infty$]-Temps

Le problème [Diff,Sync,Tmp-EM$_\infty$]-Temps a fait l'objet de nombreux travaux de recherche. Dans les références suivantes, les auteurs utilisent pour représenter la topologie du réseau tantôt un graphe orienté, tantôt un graphe non orienté et considèrent alors le réseau comme symétrique. Si une confusion peut naître de la disparité de ces approches, il est important de souligner qu'en général, les algorithmes produits se déclinent efficacement d'une modélisation vers une autre sans perte de performance.

L'approche retenue dans les travaux suivants implique que la topologie du réseau est connue de tous les nœuds du réseau ou par un superviseur dont le rôle est d'ordonnancer les émissions. Chlamtac et Kutten ont introduit le problème [Diff,Sync,Tmp-EM$_\infty$]-Temps en 1985 dans [CK85], en proposant une preuve de NP-complétude basée sur le problème MINIMUM-SET-COVER. Des algorithmes polynomiaux avec garantie de performance ont alors été proposés pour réaliser la diffusion dans un graphe G. L'objectif de ces algorithmes est de produire un ordonnancement f cohérent pour [Diff,Sync,Tmp-EM$_\infty$] pour lequel $duree(f)$ est fonction de la taille du réseau. Ainsi :

- Un premier algorithme polynomial construisant un ordonnancement en $O(D.\log^2 n)$ étapes est proposé dans [CW91], où D est le diamètre du graphe et n le nombre de nœuds.
- Cette borne supérieure fut améliorée peu après par Bar-Yehuda et al.[BYGI92] qui démontrèrent l'existence d'un algorithme permettant une diffusion en $O(D.\log n + \log^2 n)$ étapes. Cette borne fut reprise dans [KP04] par une preuve présentant des caractéristiques algorithmiques plus intéressantes que la précédente.

- En 1995, Gaber et Mansour [GM95] présentent alors un algorithme polynomial construisant une solution en $O(D + \log^5 n)$ étapes.
- Cet algorithme est repris par Elkin et Kortsarz [KE05], qui proposent alors une borne supérieure en $O(D+\log^4 n)$ dans les graphes quelconques, et annoncent même un nombre d'étapes borné par $O(D + \log^3 n)$ lorsque le graphe est planaire.
- Concernant la recherche de bornes inférieure, les auteurs prouvent dans [ABNLP91] l'existence d'une famille de graphes de diamètre 2 pour lesquels une diffusion requiert nécessairement $\Omega(\log^2 n)$ étapes.

Ce problème a également été abordé sous la contrainte supplémentaire que les nœuds n'ont a priori aucune connaissance du réseau, pas même de leurs voisins proches[6]. Les nœuds sont seulement distingués par un identifiant unique. Quelques résultats relatifs à une telle approche sont les suivants : dans [BYGI87], les auteurs proposent une borne inférieure en $\Omega(n)$. Cette borne est améliorée dans [CGG$^+$00], où les auteurs prouvent l'existence d'une famille de réseaux non symétriques ayant besoin d'au moins $\Omega(n \log n)$ étapes pour orchestrer une diffusion. Une construction similaire dans [BP97] généralise cette borne pour les graphes symétriques. Une succession d'algorithmes déterministes ou randomisés ont également été proposés :

- Un premier algorithme dans [CGG$^+$00] construit une diffusion en $O(n^{11/6})$ étapes.
- Cette borne supérieure est ramenée à $O(n^{5/3} \log^{1/3} n)$ dans [MP01].
- Indépendamment les auteurs de [CGOR00] annoncent une solution en $O(n^{3/2})$ étapes.
- Une solution en $O(n^{3/2}\sqrt{(}\log n))$ est proposée dans [Pel00] en utilisant une méthode différente.
- Finalement, un algorithme produisant un ordonnancement en $O(n \log^2 n)$ étapes a été détaillé dans [CGR02].

Notons enfin que le problème a été évoqué dans [CF94, BCB99, CMS03] lorsque les nœuds n'ont aucune connaissance de la topologie du réseau, mais que le degré maximum de ce dernier est borné par Δ :

- Ces travaux sont initié en [CF94] avec un algorithme proposant une borne supérieure de $O(D\frac{\Delta^2}{\log^2 \Delta} \log^2 n)$.
- Cette borne sera par la suite ramenée à $O(D\Delta \log^{\log \Delta} n)$ dans [BCB99].
- Enfin la borne sera enfin descendue à $O(D\Delta \log n \log(n/\Delta))$ dans [CMS03].

2.3.2 Le modèle [Diff,Sync,Tmp„]

Description et problèmes considérés

Considérons le modèle [Diff,Sync,Tmp„] : le problème de diffusion est abordé dans un environnement où les émissions sont synchrones, mais les nœuds peuvent émettre au plus une fois. Un nœud considère un message comme acquis dès la première réception correcte, quel qu'en soit l'émetteur. Une stratégie de diffusion valide d'un message m dans ce modèle se représente par une ***assignation cohérente pour [Diff,Sync,Tmp„]*** définie comme suit :

Définition 30 (assignation cohérente pour [Diff,Sync,Tmp„]) *:*
Soient un graphe $G = (V, E)$, une source unique $s \in V$ possédant un message m, et une assignation

[6] Dans une telle configuration, une diffusion peut être achevée sans qu'aucun nœud ne le sache. La définition précise d'une diffusion en t étapes doit donc être la suivante : une diffusion s'opère en t étapes si tous les nœuds connaissent le message après t étapes et qu'aucune émission n'a lieu après cette étape.

de dates f dans G.

Cette assignation est ***cohérente pour [Diff,Sync,Tmp„]*** *si et seulement si :*

1. *Elle est cohérente pour [Diff,Sync,Tmp-EM„],*
2. $\forall x,\ \sum_{i \in I\!N^*} f(x,i) \leq 1$.

Clairement une assignation de date cohérente pour [Diff,Sync,Tmp„] définit une stratégie de diffusion valide, puisque le point 2 de la définition 29 implique que tout sommet émet au plus une fois.

Nous présentons le problème de décision ***[Diff,Sync,Tmp„]-Temps*** (2.3.2) :

Donnée	**:**	un graphe $G = (V, E)$, un sommet $s \in V$, un entier k
Question	**:**	Existe-t-il une assignation g cohérente pour [Diff,Sync,Tmp„] et telle que $duree(g) \leq k$?

Problème de décision 2.3.2: Le problème [Diff,Sync,Tmp„]-Temps

Nous appelons [Diff,Sync,Tmp„]-minTemps la version optimisation de [Diff,Sync,Tmp„]-Temps, qui consiste à trouver une assignation f minimisant $duree(f)$.

Nous ne présentons pas le problème consistant à minimiser le nombre d'émissions dans ce modèle : si l'on suppose que le nombre d'étapes n'est plus une contrainte critique, il peut être de l'ordre du nombre d'émissions, avec une seule émission par étape. Minimiser le nombre d'émissions consiste alors à trouver un ensemble dominant connexe (comprenant la source du message) de cardinalité minimum dans le graphe du réseau G (en anglais : MINIMUM CONNECTED DOMINATING SET (MCDS)). Ce problème est connu pour être NP-complet [GJ79, GK96] et a fait l'objet des travaux suivants :

Il est polynomialement équivalent à la recherche d'un arbre couvrant G ayant un nombre maximum de feuilles [CWY00] (en anglais : MAXIMUM LEAF SPANNING TREE (MLST)).

Il a été prouvé NP-complet même pour les topologies comme les graphes de disques unitaires [CCJ90]. Ces résultats ont poussé à rechercher des algorithmes d'approximation avec garantie de performances dans ce type de graphes [CHLD, MBH+95, WAF02, FKMS06]. Breu et al. ont présenté dans [MBH+95] un algorithme centralisé calculant un ensemble dominant connexe de cardinalité d'au plus 10 fois l'optimal. D'autres travaux ont par la suite proposé des algorithmes distribués pour la construction d'ensembles dominants connexes toujours dans les graphes de disques unitaires . Wan et al. ont montré l'existence d'un algorithme ayant une garantie de performance de 8 [WAF02], par la suite ramenée à 6.91[FKMS06].

Certains résultats à l'origine présentés pour [Diff,Sync,Tmp-EM„]-Temps proposent des ordonnancements d'émissions dans lesquels chaque nœud n'émet au plus qu'une fois. Ces résultats s'étendent donc naturellement à [Diff,Sync,Tmp„]-Temps. Ainsi dans [CK87] les auteurs proposent une stratégie de diffusion sans collision basée sur un arbre. Cette stratégie, d'abord présentée de manière centralisée, est ensuite adaptée pour être calculée dans un contexte distribué.

2.3.3 Le modèle [Diff,Sync„Arbo,]

Considérons le modèle [Diff,Sync„Arbo,] : le problème de diffusion est abordé dans un environnement où les émissions sont synchrones, et les transmetteurs ne peuvent émettre plus d'une fois le message. De plus un transmetteur ne peut temporiser son émission : sitôt une réception sans interférence acceptée, il la réémet à l'étape suivante. Cependant tout transmetteur n'accepte que les transmissions émises par le nœud qui lui a été désigné comme son père, et ignore les autres. Les nœud définis comme récepteurs - ne retransmettent pas le message - peuvent accepter la première réception valide. Une stratégie de diffusion valide d'un message m dans ce modèle se représente par une arborescence particulière A, dite ***cohérente pour [Diff,Sync„Arbo,]*** et définie comme suit :

Définition 31 (Arborescence cohérente pour [Diff,Sync„Arbo,]) *:*
Soient un graphe $G = (V, E)$, une source unique $s \in V$ possédant un message m, et une arborescence $A = (V', E')$ avec $V' \subseteq V$ et $E' \subseteq E$.

*Cette arborescence est **cohérente pour [Diff,Sync„Arbo,]** si et seulement si la stratégie de diffusion définie comme suit est valide :*

- *Les nœuds de $V(A)$ représentent l'ensemble des transmetteurs.*
- *Un arc (x, y) de $E(A)$ désigne le nœud x comme père de y dans la stratégie de diffusion.*

Un transmetteur x avec $profondeur_A(x) = t$ va donc émettre à l'étape t. Cette stratégie est valide si, pour un transmetteur donné, aucune transmission voisine ne doit interfèrer avec l'émission de son père. De plus tout récepteur doit être en mesure de recevoir le message, à une étape durant laquelle seul un transmetteur voisin émet.

Formellement, toute arborescence A cohérente pour [Diff,Sync„Arbo,] définit une stratégie de diffusion valide si et seulement si :

1. *A est enracinée en s,*
2. $\forall \{x, y\} \in E \cap \mathcal{P}_2(V' - \{s\}) | pere_A(x) \neq y \rightarrow profondeur_A(pere_A(x)) \neq profondeur_A(y)$,
3. $\forall x \in V - V', \quad \exists y \in V' \cap N_G(x), \forall z \in V' \cap N_G(x) | profondeur_A(y) \neq profondeur_A(z)$.

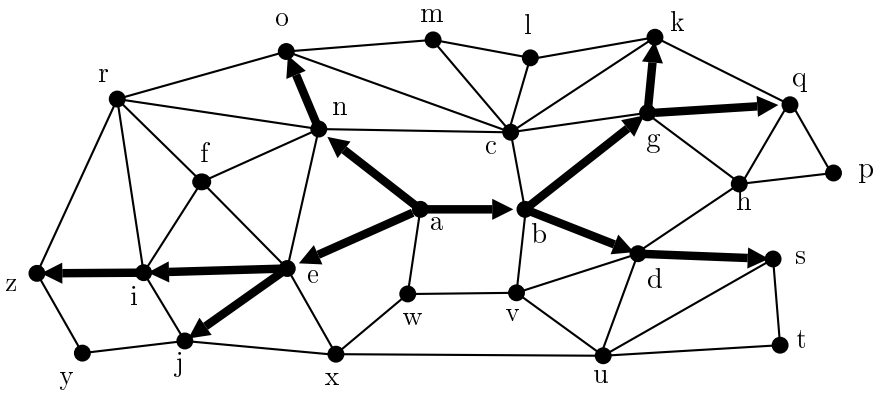
Le nombre d'étapes de la stratégie de diffusion est clairement égal à $hauteur(A)+1$. Son nombre d'émissions est égal à $|V'|$. La figure 2.1(a) présente un exemple d'arborescence cohérente pour [Diff,Sync„Arbo,] sur un réseau de 26 nœuds. La figure 2.1(b) détaille les dates des émissions des sommets de l'arborescence, et les dates de réception des nœuds récepteurs (c'est-à-dire les nœuds qui ne sont pas dans l'arborescence).

Nous introduisons les deux problèmes de décision :

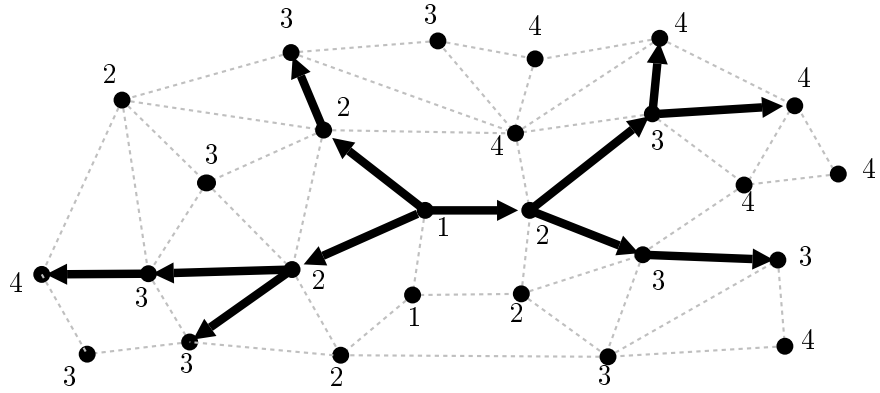
1. ***[Diff,Sync„Arbo,]-Temps*** (2.3.3),
2. ***[Diff,Sync„Arbo,]-émissions*** (2.3.4)

Donnée	**:**	un graphe $G = (V, E)$, un sommet $s \in V$, un entier k.
Question	**:**	Existe-t-il une arborescence A cohérente pour [Diff,Sync„Arbo,] et telle que $hauteur(A) \leq k - 1$?

Problème de décision 2.3.3: Le problème [Diff,Sync„Arbo,]-Temps



(a) Un graphe, et une arborescence cohérente pour [Diff,Sync,,Arbo,].



(b) Les dates d'émissions des nœuds de l'arborescences. Les dates associées aux autres nœuds indique au terme de quelle étape chacun reçoit correctement l'information pour la première fois.

FIG. 2.1 – Une arborescence de diffusion dans un modèle synchrone

Donnée	:	un graphe $G = (V, E)$, un sommet $s \in V$, un entier k.
Question	:	Existe-t-il une arborescence A cohérente pour [Diff,Sync„Arbo,] et telle que $\|V(A)\| \leq k$?

Problème de décision 2.3.4: Le problème [Diff,Sync„Arbo,]-émissions

Nous appelons [Diff,Sync„Arbo,]-minTemps et [Diff,Sync„Arbo,]-minémissions les versions optimisation de [Diff,Sync„Arbo,]-Temps et [Diff,Sync„Arbo,]-émissions.

2.3.4 Le modèle [Diff,Async„Arbo,]

Le modèle [Diff,Async„Arbo,] est le seul modèle asynchrone que nous étudions. Le problème de diffusion est abordé dans un environnement où chaque transmetteur n'émet qu'une fois, et n'accepte le message que du nœud père qui lui a été assigné. Un transmetteur ne peut pas temporiser son émission : sitôt la réception de son père acceptée, il la réémet au terme d'un délai de traitement non borné et propre au nœud. Nous introduisons les définitions suivantes :

Définition 32 (élément dominant) *:*
Soient A une arborescence et un ensemble $S \subseteq V(A)$.

*L'ensemble S possède un **élément dominant** dans A si et seulement s'il existe un élément x de S tel que x est ancêtre de y dans A, pour tout $y \in S - \{x\}$.*

La figure 2.2 représente une partie d'un graphe G, et une arborescence A sur ce graphe enracinée en s (désignée par les arcs épais). Soit $S = N_G(m)$. Alors a est l'élément dominant de S.

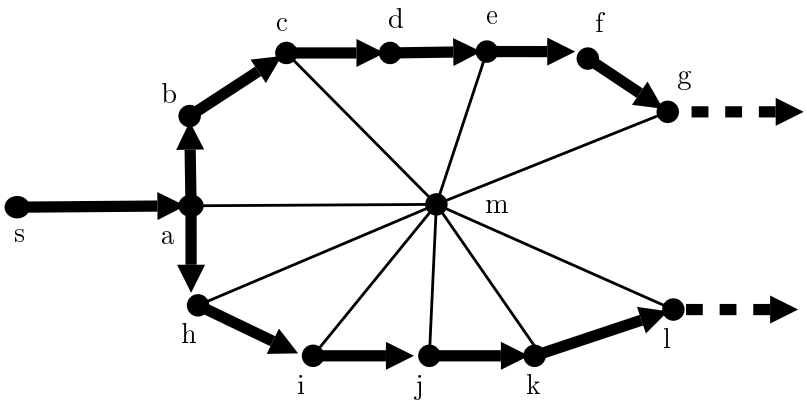


FIG. 2.2 – Représentation partielle d'un graphe G, d'une arborescence A

Définition 33 (séquence d'émissions, robustesse, coefficient de robustesse) *:*
Soient A une arborescence et un ensemble $S \subseteq V(A)$.

Un sous-ensemble $S' \subseteq S$ avec $|S' = k|$ décrit une ***séquence d'émissions*** *dans A si et seulement s'il existe un ordre $O = (s_1, s_2, \ldots, s_k)$ sur les éléments de S' tel que s_i est ancêtre de s_j dans A, quels que soient i et j tels que $1 \leq i < j \leq k$.*

La ***robustesse*** *d'une séquence d'émissions dans A décrite par S, notée $rse_A(S)$, est la cardinalité du plus grand sous-ensemble $S' \subseteq S$ tel que $pere_A(x) \neq y$ pour tout $x, y \in S'$.*

Le ***coefficient de robustesse*** *$cr_A(S)$ d'un ensemble S dans A est défini par :*

$$cr_A(S) = \max_{S' \subseteq S} \left(rse_A(S') - \left| S - S' \right| \right)$$

pour toute partie S' de S telle que S' décrit une séquence d'émissions dans A.

Calculons sur la figure 2.2 le coefficient de robustesse du sommet récepteur m. Soit S l'ensemble des voisins transmetteurs de m. On a $S = N_G(m) \cap V(A)$. Alors $S_1 = \{a, c, e, g\} \subseteq S$ et $S_2 = \{a, h, i, j, k, l\} \subseteq S$ sont les deux séquences d'émissions dans A maximales au sens de l'inclusion. On a $rse_A(S_1) = |\{a, c, e, g\}| = 4$ et $rse_A(S_2) = |\{a, i, k\}| = 3$. Le coefficient de robustesse $cr_A(S)$ est maximum pour S_2, et de valeur 0 (Pour S_1, $rse_A(S_1) - |S - S_1| = -1$).

Une hypothèse du modèle asynchrone est que les transmissions sont de durées identiques. Soient trois nœuds a, b, c situés dans le voisinage de d, tels que les émissions de b et c sont liées par une relation de précédence. Le nœud a peut émettre indépendamment de l'émission de b et/ou de c . Alors a peut interférer avec b et c si b et c sont immédiatement consécutives. Dans le cas contraire, a ne peut interférer qu'avec au plus une seule de ces deux émissions.

Propriété 2 *Soit x un sommet donné n'appartenant pas à A et dont le coefficient de robustesse de son voisinage $N_G(x)$ est supérieur ou égal à 1. Alors il existe un nombre suffisant d'émissions consécutives dans $N_G(x) \cap A$, de sorte que x puisse recevoir au moins l'une de ces émissions sans interférence.*

Une stratégie de diffusion valide dans le modèle [Diff,Async,,Arbo,] se représente alors par une arborescence particulière A dite ***cohérente pour [Diff,Async,,Arbo,]***, définie comme suit :

Définition 34 (Arborescence cohérente pour [Diff,Async,,Arbo,]) *:*
Soient un graphe $G = (V, E)$, une source unique $s \in V$ possédant un message m, et une arborescence $A = (V', E')$ telle que $V' \subseteq V, E' \subseteq E$.

Cette arborescence est ***cohérente pour [Diff,Async,,Arbo,]*** *si et seulement si la stratégie de diffusion définie comme suit est valide :*

- *Les nœuds de $V(A)$ représentent l'ensemble des transmetteurs.*
- *Un arc (x, y) de $E(A)$ désigne le nœud x comme père de y dans la stratégie de diffusion.*

Cette stratégie est effectivement valide si et seulement si pour toute paire de transmetteurs $\{x, y\}$ adjacents dans G, les émissions de x et de $pere(y)$ ne sont pas simultanées, sans quoi y pourrait ne pas recevoir la transmission de $pere(y)$. De même les émissions de y et $pere(x)$ doivent être successives. Il en résulte que les émissions de x et y doivent être successives, ou que $pere(x) = pere(y)$. Enfin, soit x un récepteur. S'il existe un élément dominant dans $N_G(x) \cap V(A)$, alors x doit recevoir correctement l'émission de ce dernier, ou alors il doit exister dans $N_G(x) \cap V(A)$ une séquence d'émissions dans A suffisamment robuste pour assurer la réception de x.

L'ensemble de ces conditions s'exprime formellement comme suit :

1. *A est enracinée en s.*
2. *$\forall \{x,y\} \in \mathcal{P}_2(V') \cap E$, l'une des trois conditions suivantes est satisfaite :*
   (a) *x est ancêtre de y dans A,*
   (b) *y est ancêtre de x dans A,*
   (c) *$pere_A(x) = pere_A(y)$.*
3. *$\forall x \in V - V'$, $N_G(x) \cap V'$ possède un élément dominant, ou bien $cr_A(N_G(x) \cap V') \geq 1$*

La figure 2.1(a) ne représente pas une arborescence cohérente pour [Diff,Async„Arbo,] : par exemple le sommet récepteur c ne possède ni élément dominant, ni un coefficient de robustesse suffisamment grand. En revanche la figure 2.3 illustre une telle configuration.

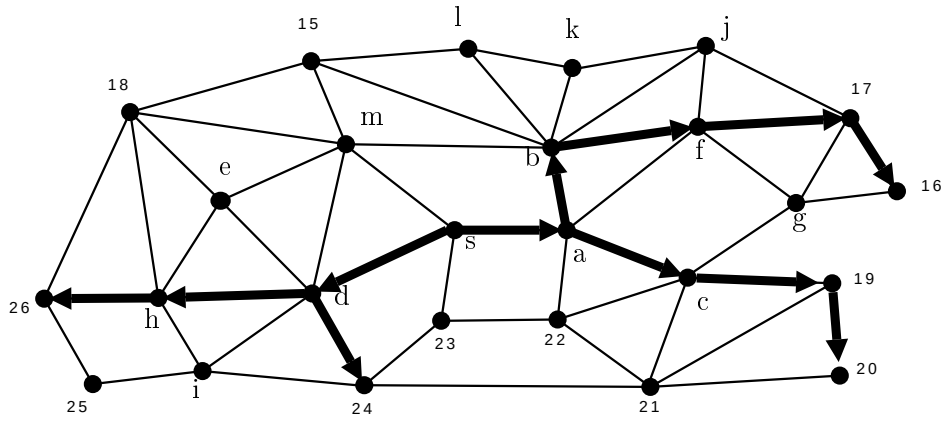


FIG. 2.3 – Une arborescence cohérente pour [Diff,Async„Arbo,].

Le temps de diffusion d'une telle stratégie ne peut être évalué avec précision, puisque les délais de traitement sont non bornés. Cependant l'indicateur le plus fiable en moyenne est la hauteur de l'arborescence, $hauteur(A)$. Nous supposons le temps de diffusion proportionnel à $hauteur(A)$. Le nombre d'émissions est ici égal à la cardinalité de l'ensemble $V(A)$. Nous introduisons deux problèmes de décision ***[Diff,Async„Arbo,]-Temps*** (problème 2.3.5), et***[Diff,Async„Arbo,]-émissions*** (problème 2.3.6), dont les coûts sont liés à chacun de ces deux paramètres.

Donnée	:	un graphe $G = (V, E)$, un sommet $s \in V$, un entier k
Question	:	Existe-t-il une arborescence A cohérente pour [Diff,Async„Arbo,] et telle que $hauteur(A) \leq k - 1$?

Problème de décision 2.3.5: Le problème [Diff,Async„Arbo,]-Temps

Donnée	:	un graphe $G = (V, E)$, un sommet $s \in V$, un entier k
Question	:	Existe-t-il une arborescence A cohérente pour [Diff,Async„Arbo,] et telle que $V(A) \leq k$?

Problème de décision 2.3.6: Le problème [Diff,Async„Arbo,]-émissions

Nous appelons [Diff,Async„Arbo,]-minTemps et [Diff,Async„Arbo,]-minÉmissions les versions optimisation de [Diff,Async„Arbo,]-Temps et [Diff,Async„Arbo,]-émissions.

Chapitre 3

Étude de stratégies de diffusion

Nous menons une étude sur les stratégies introduites dans le chapitre précédent, et précisément sur la stratégie de diffusion en arborescence. Nous définissons dans une première section les conditions d'existence d'une solution en fonction de la stratégie retenue. Le seconde section s'oriente sur la complexité des problèmes de minimisation du temps de diffusion et du nombre d'émissions pour une stratégie donnée. Nous étudions notamment dans la troisième section l'impact de la topologie sur de tels problèmes.

3.1 Satisfiabilité et comparaison de modèles

3.1.1 Le modèle [Diff,Sync„,], un oubli du chapitre précédent ?

Dans le chapitre précédent, nous n'avons pas considéré le modèle [Diff,Sync„,]. Dans une telle configuration, le problème de la diffusion est envisagé dans un réseau où les émissions sont synchrones mais les nœuds ne peuvent émettre qu'une fois au plus, et sans pouvoir temporiser leur émission. Les possibilités d'un transmetteur sont réduites à relayer le message au plus une fois, et à l'étape immédiatement successive à une première réception sans interférence. Une stratégie de diffusion valide se représente de manière réduite par une bipartition des nœuds du réseau en transmetteurs / récepteurs. Pour une bipartition donnée, il est possible de vérifier en temps polynomial si une diffusion couvre bien l'ensemble des nœuds du réseau, simplement en simulant une diffusion et en vérifiant quels nœuds ont été atteints une fois que le messsage ne se propage plus.

Nous avons volontairement choisi d'ignorer ce modèle car pour certains graphes il n'existe pas de solution, comme en atteste la propriété 4. Sa preuve repose sur la propriété 3

Propriété 3 *:*
Soient G=(V,E) un graphe, et x un point d'articulation de G.

Alors dans toute stratégie de diffusion, x est un transmetteur.

Preuve :
Immédiat par le point 3 de la propriété 1

□

Propriété 4 *:*
Il existe des réseaux pour lesquels aucune stratégie ne peut satisfaire [Diff,Sync$_{\parallel}$,].

Preuve :
Soit G le graphe X_{84} présenté dans la figure 3.1.

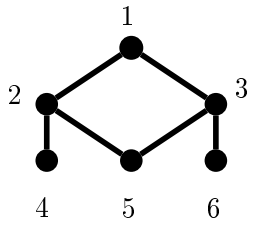


FIG. 3.1 – Le graphe X_{84}

Soit s le sommet 1 de G. Supposons qu'il existe une stratégie de diffusion S cohérente pour [Diff,Sync$_{\parallel}$,X_{84}]. Alors les sommets 2 et 3 sont transmetteurs, en accord avec la propriété 3. L'émission de 1 à l'étape 1 est correctement perçue par 2 et 3 qui émettent à l'étape suivante, empêchant le sommet 5 de recevoir une émission.

□

Remarquons que X_{84} est 1-connexe, imposant de ce fait que certains nœuds obligatoirement des transmetteurs dans la diffusion. Aussi peut-on se demander si l'infaisabilité de [Diff,Sync$_{\parallel}$,] n'est pas simplement liée à a présence de ces transmetteurs imposés. La propriété suivante généralise la propriété 4 sur des instances k-connexes :

Propriété 5 *:*
Il existe des réseaux pour lesquels aucune stratégie ne satisfait [Diff,Sync$_{\parallel}$,k-connexe].

Preuve :
Soit $G = (V, E)$ le graphe tel que :
- $V = \{s, x_0, x'_0, x_1, x'_1, \ldots, x_k, x'_k\}, k \in I\!N$
- $E = \big\{\{s, x_i\}, \forall i \leq k\big\} \bigcup \big\{\{x_i, x'_j\} | i \neq j, \forall i, j \leq k\big\}$

Clairement G est k-connexe. La figure 3.1.1 présente un tel graphe pour $k = 3$.

Supposons qu'il existe une stratégie S satisfaisant [Diff,Sync$_{\parallel}$,] sur l'instance décrite par le graphe G, où s désigne la source du message. Appelons S_i l'ensemble des sommets situés à distance i de la source. Clairement S_i partitionne S, avec $S_0 = \{s\}$, $S_1 = \big\{x_i\big\}_{0 \leq i \leq k}$, et $S_2 = \big\{x'_i\big\}_{0 \leq i \leq k}$.

L'émission de s à l'étape 1 touche correctement l'ensemble des sommets de S_1. Puisque tout nœud x_i possède au plus k voisins dans S_2, lui-même de cardinalité $k + 1$, au moins deux nœuds a et b de S_1 doivent être transmetteurs. Par les caractéristiques de [Diff,Sync$_{\parallel}$,], a et b doivent émettre à l'étape 2. Dans le graphe G, pour toute paire de nœuds $\{a, b\}$ avec $a, b \in S_1$, l'ensemble $N_G(a) \cap N_G(b) \cap S_2$ contient au moins un élément c. Cet élément ne peut recevoir le message si a et b émettent simultanément. Ceci conclut notre preuve.

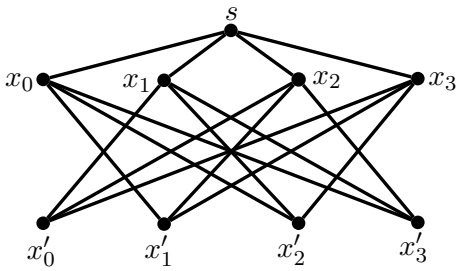


FIG. 3.2 – Un graphe 3-connexe

□

Ces observations sont le point de départ d'une réflexion consistant à déterminer les conditions d'existence des stratégies dans différents modèles. Dans la sous-section suivante, nous établissons quelques relations entre stratégies.

3.1.2 Hiérarchisation des contextes

Nous établissons brièvement une hiérarchisation des modèles étudiés. Sur la figure 3.3, un lien ascendant entre deux modèles A et B signifie qu'une stratégie de diffusion satisfaisant A satisfait également B. Ces relations sont décrites dans la propriété 6 :

Propriété 6 *:*
Soient $G = (V, E)$ un graphe, et une source $s \in V$. Alors :

1. *toute stratégie satisfaisant [Diff,Sync,Tmp„] satisfait également [Diff,Sync,Tmp-EM„],*
2. *toute stratégie satisfaisant [Diff,Sync„Arbo,] satisfait également [Diff,Sync,Tmp„],*
3. *toute stratégie satisfaisant [Diff,Sync„,] satisfait également [Diff,Sync„Arbo,],*
4. *toute stratégie satisfaisant [Diff,Async„Arbo,] satisfait également [Diff,Sync„Arbo,].*

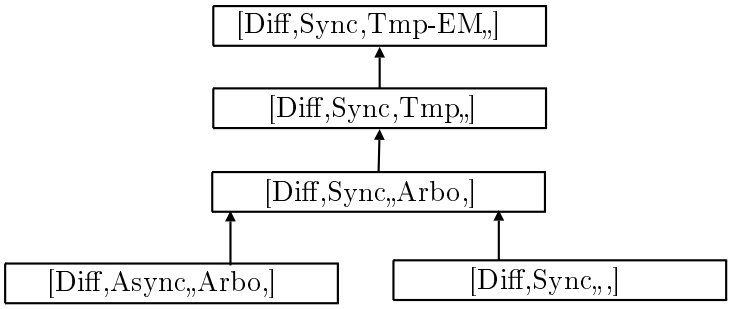


FIG. 3.3 – Relations de satisfiabilité entre différents modèles de communication

Preuve :

- Preuve du point 1 : immédiate par les définitions 27 et 30.
- Preuve du point 2 : soit A une arborescence cohérente pour [Diff,Sync„Arbo,]. Un transmetteur ignore les émissions provenant des nœuds autres que son père. Une fois l'émission de son père reçue à l'étape t, il émet à l'étape $t+1$. Ce mode de fonctionnement est plus restrictive que pouvoir accepter la première émission correcte, et réémettre le message à une étape postérieure de son choix. On en déduit l'inclusion de l'espace des solutions.
- Preuve du point 3 : dans une stratégie S satisfaisant [Diff,Sync„,], tout transmetteur émet à l'étape immédiatement successive à la première transmission correctement reçue. Nous pouvons définir une arborescence A enracinée en s, dont les nœuds sont les transmetteurs de S. Un arc (i,j) indique que l'émission de i est la première qui a été correctement reçue par j. Clairement A est cohérent pour [Diff,Sync„Arbo,] sur la même instance.
- Preuve du point 4 : intuitivement, une stratégie satisfaisant [Diff,Async„Arbo,] doit satisfaire son homologue synchrone. Une étude approfondie des arborescentes cohérentes pour les contextes [Diff,Sync„Arbo,] (définition 31) et [Diff,Async„Arbo,] (définition 34) permet de valider l'inclusion des solutions d'un contexte dans l'autre.

□

3.1.3 Satisfiabilité des modèles

Dans la sous-section 3.1.1 nous avons présenté un modèle pour lequel il n'existe pas toujours de stratégie satisfaisant une instance donnée. La sous-section 3.1.2 a établi des relations de satisfiabilité entre modèles. Notre étude de satisfiabilité se conclut ici par la démonstration qu'il existe toujours des stratégies de diffusion satisfaisant respectivement les modèles [Diff,Sync,Tmp„], [Diff,Sync,Tmp-EM„],[Diff,Sync„Arbo,] et [Diff,Async„Arbo,].

Intuitivement, une stratégie de diffusion semble toujours pouvoir émerger dans les modèles [Diff,Sync,Tmp„] et [Diff,Sync,Tmp-EM„]. Les possibilités de temporisation d'émissions sont un outil puissant permettant de prévenir d'éventuels brouillages. La réponse est moins immédiate pour les modèles [Diff,Sync„Arbo,] et [Diff,Async„Arbo,].

Reprenons l'instance présentée dans la preuve de la propriété 4. Si l'instance n'est pas satisfiable dans le modèle [Diff,Sync„,], elle l'est en revanche dans le modèle [Diff,Sync„Arbo,] : l'arborescence A définie par $V(A) = \{1,3,5\}$ et $E(A) = \{\{1,3\},\{3,5\},\{5,2\}\}$ est cohérente pour [Diff,Sync„Arbo,]. La figure 3.4 présente une telle arborescence.

Remarque 3 *:*
L'arborescence A a une hauteur de 3, et constitue ici la stratégie la plus efficace en terme de temps de diffusion, avec 4 étapes. Une stratégie dans un modèle [Diff,Sync,Tmp„] optimale en temps n'aurait nécessité que 3 étapes. Bien que A induise une assignation de dates cohérente dans [Diff,Sync,Tmp„], la qualité de cette dernière n'est plus aussi bonne dans un modèle hiérarchiquement supérieur en terme de faisabilité.

Terminons cette section par l'énoncé du théorème 1 :

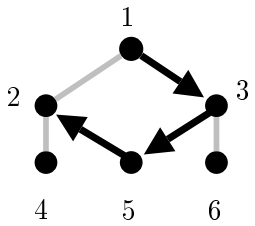


FIG. 3.4 – Une arborescence cohérente pour [Diff,Sync„Arbo,] sur X_{84}. La source s est le sommet 1.

Théorème 1 *:*
Soient $G = (V, E)$ un graphe connexe, un sommet source $s \in V$.

Alors il existe toujours une stratégie satisfaisant [Diff,Async„Arbo,].

Preuve :
Soit A' une arborescence résultant d'un parcours en profondeur de G depuis le sommet s. Nous appelons A l'arborescence construite par copie de A' et en y supprimant les nœuds feuilles et leur arête incidente. Clairement A est cohérente pour le modèle [Diff,Async„Arbo,] :
- La construction de A implique que deux nœuds de $V(A)$ adjacents dans G sont forcément liés par une relation de descendance.
- Pour tout nœud x de $V(G) - V(A)$, l'ensemble de ses voisins est constitué de transmetteurs, tous situés sur la même branche de l'arborescence.

Ces propriétés se déduisent de la construction d'une arborescence décrivant un parcours en profondeur. L'existence d'un tel parcours est établie pour tout graphe connexe, ce qui termine la preuve. □

Corollaire 1 *:*
Soient $G = (V, E)$ un graphe connexe, un sommet source $s \in V$.

Alors il existe toujours des stratégies de diffusion cohérentes pour les modèles [Diff,Async„Arbo,], [Diff,Sync„Arbo,], [Diff,Async,Tmp„] et [Diff,Async,Tmp-EM„].

Preuve :
Immédiat d'après le théorème 1 et les relations hiérarchiques établies dans la sous-section 3.1.2. □

Ces résultats permettent d'affirmer que l'ensemble des problèmes de minimisation définis dans la section 2.3 ont tous une solution réalisable.

3.2 Minimiser le temps ou l'énergie d'une diffusion

Nous étudions dans cette section deux familles de problèmes consistant à construire une diffusion minimisant respectivement le temps nécessaire et l'énergie requise, selon différents modèles de com-

munication. L'ensemble des problèmes considérés a été introduit et formalisé dans les sous-sections 2.3.1 à 2.3.4.

La donnée de chacun des problèmes définis comporte un graphe $G = (V, E)$ connexe, et un nœud source $s \in V$. Les problèmes de décision portant sur l'existence d'une solution avec nombre d'étapes borné possèdent une donnée complémentaire, notée k, correspondant à la la plus grande date allouable.

Les trois problèmes [Diff,Sync,Tmp„]-Temps, [Diff,Sync,Tmp„]-émissions, [Diff,Sync,Tmp-EM„]-Temps et [Diff,Sync,Tmp-EM„]-émissions ont été montrés NP-complets dans de nombreux travaux. Nous ciblons notre étude de complexité sur les problèmes dont le contexte de réception est en arborescence.

3.2.1 Minimiser le temps de diffusion

Dans cette première sous-section, nous montrons la NP-complétude du problème de décision [Diff,Sync„Arbo,]-Temps. La preuve présentée s'étend naturellement pour prouver la NP-complétude de [Diff,Async„Arbo,]-Temps.

Théorème 2 *:*
Le problème de décision [Diff,Sync„Arbo,]-Temps est NP-complet.

Preuve :
Nous montrons que le problème est NP-complet par une réduction depuis le problème NP-complet EXACT SET COVER, désigné dans la suite du document par l'acronyme ESC. Ce problème est une version particulière du problème MINIMUM SET COVER dans laquelle les ensembles retenus dans une solution doivent être disjoints.

Tout d'abord, le problème [Diff,Sync„Arbo,]-Temps est dans NP : soient $I = (G, s, k)$ une instance de [Diff,Sync„Arbo,]-Temps, et T une arborescence T donnée. Il est possible de vérifier en un temps polynomial que T est bien cohérente pour [Diff,Sync„Arbo,] sur I, et de hauteur inférieure ou égale à $k - 1$.

Soit $I_{ESC} = (S, C)$ une instance du problème ESC, où $C = (S_1, S_2, \ldots, S_k)$ est une collection de k sous-ensembles d'un ensemble S de l éléments $S = \{e_1, e_2, \ldots, e_l\}$. Par une transformation polynomiale en temps et en mémoire, nous construisons le graphe $G = (V, E)$ tel que :
- $V = \{x_1, x_2, \ldots, x_k\} \bigcup \{y_1, y_2, \ldots, y_l\} \bigcup \{s\}$.
- $E = \big\{\{s, x_i\}, \forall i \leq k\big\} \bigcup \big\{\{x_i, y_j\}, \forall i \leq k, j \leq l | e_j \in S_i\big\}$

Considérons le problème [Diff,Sync„Arbo,]-Temps sur l'instance I définie par le graphe $G = (V, E)$ précédemment construit, le sommet source $s \in V$, et un entier $k = 2$. Résoudre I revient à trouver s'il existe une stratégie de diffusion en 2 étapes au plus. Nous montrons qu'il existe solution au problème [Diff,Sync„Arbo,]-Temps sur I si et seulement s'il existe une solution à I_{ESC}.

Soit A une arborescence cohérente pour [Diff,Sync„Arbo,] et de hauteur 1. Clairement A décrit une solution de [Diff,Sync„Arbo,]-Temps sur I. Dans cette solution, le sommet s émet à l'étape 1

et tous les sommets de $\{x_i\}_{1\leq i\leq k}$ reçoivent correctement le message. À l'étape suivante un sous-ensemble U de ces sommets émet. Cet ensemble est désigné par l'ensemble des sommets fils de s dans A. Puisque A décrit une solution, tous les sommets $\{y_j\}_{1\leq j\leq l}$ reçoivent correctement le message. On en déduit une couverture de tous les sommets $\{y_j\}_{1\leq j\leq l}$ par les sommets de U. En sélectionnant les ensembles S_i tels que $x_i \in U$, nous obtenons une solution au problème ESC.

La réciproque est immédiate.

Rappelons enfin que [Diff,Sync„Arbo,]-Temps est dans NP. Puisque ESC est un problème NP-complet, alors nous en concluons que [Diff,Sync„Arbo,]-Temps également.

□

Corollaire 2 *:*
Le problème [Diff,Async„Arbo,]-Temps est NP-complet.

Preuve :
L'arborescence A construite dans la preuve précédente est de hauteur 1 et est également cohérente pour le modèle [Diff,Async„Arbo,]. La preuve se généralise dans le cas asynchrone.

□

Signalons enfin la propriété suivante :

Propriété 7 *:*
Dans un contexte synchrone, une borne inférieure au nombre d'étapes est définie par l'excentricité du sommet source. Une arborescence cohérente pour le modèle [Diff,Sync„Arbo,] (respectivement [Diff,Async„Arbo,]) possède une hauteur au moins égale à l'excentricité du sommet source moins un.

3.2.2 Minimiser le nombre d'émissions

Dans la sous-section suivante, nous établissons la NP-complétude du problème de décision [Diff,Sync„Arbo,]-émissions. Ce résultat s'étend dans le modèle asynchrone.

Théorème 3 *:*
- *Le problème de décision [Diff,Async„Arbo,]-émissions est NP-complet.*
- *Le problème de minimisation associé [Diff,Async„Arbo,]-minémissions est NP-difficile et non approximable à un facteur constant près.*

Preuve :
Nous montrons que le problème de décision [Diff,Async„Arbo,]-émissions est NP-complet par une réduction depuis le problème NP-complet SET COVER (SC), qui est le problème de décision associé au problème d'optimisation MINIMUM SET COVER (MSC). Nous montrons que notre réduction conserve le rapport d'approximation de la solution. Puisque MSC est un problème NP-difficile et non approximable à une constante près, nous en déduirons que [Diff,Async„Arbo,]-minémissions également.

Le problème [Diff,Async„Arbo,]-émissions est dans NP : il est possible de vérifier en un temps polynomial qu'une arborescence A donnée est bien cohérente pour [Diff,Async„Arbo,] sur une instance $I = (G, s, k)$ donnée, et que son nombre de nœuds est inférieur ou égal à $k - 1$.

Soit $I_{SC} = (C, S, k)$ une instance de SC, où $C = \{S_1, S_2, \ldots, S_m\}$ est une collection de sous-ensembles d'un ensemble S de l éléments $S = \{e_1, e_2, \ldots, e_l\}$, et k un entier naturel. Par une transformation polynomiale en temps et en mémoire, nous construisons depuis I_{SC} le graphe $G = (V, E)$ tel que :

- $V = \{x_1, x_2, \ldots, x_m\} \bigcup \{y_1, y_2, \ldots, y_l\} \bigcup \{s\}$.
- $E = \big\{\{s, x_i\}, \forall i \leq m\big\} \bigcup \big\{\{x_i, y_j\}, \forall i \leq m, j \leq l | e_j \in S_i\big\} \bigcup \big\{\{x_i, x_j\}, \forall i, j \leq m\big\}$

Considérons le problème [Diff,Async„Arbo,]-émissions sur l'instance $I = (G, s, k+1)$. Résoudre I revient à trouver une arborescence A cohérente dans [Diff,Sync„Arbo,] telle que $emissions(A) = k+1$.

Soit A une arborescence cohérente pour [Diff,Async„Arbo,]-émissions où $emissions(A) = k + 1$. À l'étape 1 seul s émet et l'ensemble $\{x_i\}_{1 \leq i \leq k}$ peut recevoir (et potentiellement accepter) le message. Aucun sommet y_i n'est transmetteur, puisque l'ensemble de ses sommets voisins est inclus dans $\{x_i\}_{1 \leq i \leq k}$. Exactement k sommets parmi $\{x_i\}_{1 \leq i \leq k}$ sont transmetteurs. L'ensemble T des transmetteurs forme une clique dans G. L'arborescence A est donc une chaîne, en accord avec la définition 34. Les relations de parenté sont secondaires, et peuvent se définir par un ordre quelconque sur les sommets de T. Il se déduit que tout sommet de $\{y_j\}_{0 \leq j \leq l}$ est voisin d'au moins un sommet de $\{x_i\}_{1 \leq i \leq k}$. En sélectionnant les ensembles S_i tels que $x_i \in V(A)$, nous obtenons une solution sur l'instance I_{SC}.

La réciproque est immédiate.

Cette réduction conserve le rapport entre solutions, et donc l'inaproximabilité de la solution. Rappelons enfin que [Diff,Async„Arbo,]-émissions est dans NP et que MSC est un problème NP-difficile et non approximable à un facteur constant. Nous concluons que [Diff,Async„Arbo,]-émissions est NP-complet et que [Diff,Async„Arbo,]-minémissions est un problème NP-difficile et non approximable à un facteur constant.

□

Corollaire 3 *:*

- *Le problème de décision [Diff,Async„Arbo,]-émissions est NP-complet.*
- *Le problème de minimisation [Diff,Async„Arbo,]-minémissions est NP-difficile et non approximable à un facteur constant.*

Preuve :

La preuve présentée précédemment se décline dans le cas synchrone. L'arborescence A n'est plus forcément une chaîne. Cependant dans le graphe G construit dans la preuve précédente, s'il existe une arborescence A cohérente pour [Diff,Sync„Arbo,] avec $emissions(A) = k + 1$, nous pouvons en déduire une arborescence B cohérente pour [Diff,Async„Arbo,] avec $emissions(B) = k + 1$, où B est une chaîne avec $V(B) = V(A)$ et $s \in ext(B)$. Les résultats de la preuve s'étendent alors naturellement au modèle synchrone.

□

Terminons avec la propriété suivante :

Propriété 8 *:*
Soit $G = (V, E)$ un graphe connexe.

Alors dans toute stratégie de diffusion, le nombre de nœuds transmetteurs $|T|$ vérifie l'équation :

$$|T| \geq \frac{|V| - 2}{\Delta_G - 1} \tag{3.1}$$

où Δ_G est le degré maximum du réseau.

Preuve :
En accord avec le point 3 de la propriété 1, $|T| \geq |D|$, où $|D|$ est la cardinalité du plus petit ensemble dominant connexe.

Soit D un ensemble dominant connexe de cardinalité minimum dans G, et G' le graphe induit de G par les sommets de D. Appelons ***potentiel couvrant*** d'un sommet $x \in D$ le nombre de nœuds de $V(G) - V(G')$ adjacents à x. Clairement ce potentiel est égal à $d_G(x) - d'_G(x)$. La somme des potentiels couvrant des $x \in D$ doit être supérieure à la cardinalité de l'ensemble des sommets de $V(G) - V(G')$, c'est-à-dire $|V| - |D|$. En remarquant que G' doit être connexe, (donc contient nécessairement $|D| - 1$ arêtes) et que le degré de G est borné par Δ, nous obtenons :

$$\begin{aligned}
\sum_{x \in D} (d_G(x) - d_{G'}(x)) &\geq n - |D| \\
|D| + \sum_{x \in D} d_G(x) &\geq n + \sum_{x \in D} d_{G'}(x)) \\
|D| + \Delta \times |D| &\geq n + 2 \times (|D| - 1) \\
|D| &\geq \frac{n - 2}{\Delta - 1}
\end{aligned}$$

□

3.3 Impact de la topologie sur le nombre d'émissions

Le travail présenté dans la section suivante naît de l'observation suivante :

Observation 1 *:*
Il existe des graphes pour lesquels une stratégie cohérente pour [Diff,Sync„Arbo,] requiert un nombre de transmetteurs strictement supérieur à la cardinalité de l'ensemble dominant connexe minimum contenant la source.

Cette constatation s'applique pour le modèle [Diff,Async„Arbo,].

Preuve :
Soient G le graphe X_{84} présenté en figure 3.1, S une stratégie cohérente pour [Diff,Sync„Arbo,] permettant une diffusion depuis 1, et D un ensemble dominant connexe minimum sur G.

Une arborescence A induisant une stratégie valide pour ce modèle est présentée en figure 3.4. D'après le point 3 de la propriété 1, $|V(A)| \geq |D|$. La cardinalité de l'ensemble dominant connexe minimum D est 3, par exemple $D = \{1, 2, 3\}$. Clairement le nombre de nœuds de cette arborescence est 4, et ne peut être minimisé.

Même démonstration pour le modèle [Diff,Async„Arbo,].

□

3.3.1 Nombre d'émissions et ensemble dominant connexe

Nous répondons dans cette section à la question suivante : considérons la donnée d'un graphe G et d'un sommet source s. Soient A une arborescence cohérente pour [Diff,Async„Arbo,], et D un ensemble dominant connexe pour G, incluant s et de cardinalité minimum[1]. Par le point 3 de la propriété 1, $|V(A)| \geq |D|$. De plus l'obervation 1 montre qu'il existe des graphes (ex., X_{84}) pour lesquels $|V(A)| > |D|$. Existe-t-il alors des familles de graphes pour lesquelles l'existence de A avec $|V(A)| = |D|$ est garantie ?

Nous allons montrer que les graphes triangulés (définition 15) constituent une de ces familles. Les résultats présentés s'étendent naturellement au modèle [Diff,Sync„Arbo,].

L'arbre des cliques d'un graphe triangulé :

Les graphes triangulés ont une structure particulière induisant de nombreuses propriétés. Plusieurs outils de traitement et notions ont pu être élaborés sur ces graphes, parmi lesquels l'arbre de cliques [GHP95] que nous introduisons ici :

Définition 35 (Arbre de cliques) *:*
Soit $G_\Delta = (V, E)$ un graphe triangulé.

*Un **arbre de cliques** d'un graphe triangulé G_Δ est une représentation de G par un arbre T_{G_Δ} dans lequel :*
- *$V(T_{G_\Delta})$ est l'ensemble $\{C_i\}_{1 \leq i \leq k}$ des cliques maximales de G_Δ,*
- *$E(T_{G_\Delta})$ contient toute arête $\{C_i, C_j\}$ si l'intersection $C_i \cap C_j$ constitue un séparateur pour G_Δ.*

Un arbre de cliques satisfait la propriété d'intersection des cliques : soient C_i et C_j deux cliques maximales, l'ensemble $C_i \cap C_j$ est contenu dans toutes les cliques sur la chaîne reliant C_i et C_j dans T_{G_Δ}. En 1974, Gavril annonce le théorème suivant :

[1] Nous ajoutons un sommet s' et une arête $\{s, s'\}$ à G pour assurer que la cardinalité d'un ensemble connexe minimal incluant s est aussi minimum.

Théorème 4 ([Gav74]) *:*
Un graphe est triangulé si et seulement s'il admet un arbre de cliques.

Pour un graphe triangulé, nous pouvons généralement associer plusieurs arbres de cliques non isomorphes. Le théorème suivant est un résultat énoncé par Blair et Peyton dans [BP94] :

Théorème 5 *:*
Soit G_Δ un graphe triangulé.

Il est possible de calculer un arbre de cliques de diamètre minimum en temps linéaire.

Vers un algorithme de diffusion dans les graphes triangulés

Dans la suite, nous enracinons tout arbre de cliques sur un nœud contenant la source s. Nous parlons alors d'une arborescence de cliques, et sous-entendons que son nœud racine est une clique maximale de G_Δ contenant s. Nous proposons l'algorithme 1.

Algorithme 1 : ArborescenceTriangulé

Entrées : Un graphe triangulé G_Δ, $s \in V(G)$, un ensemble dominant connexe D dans G_Δ contenant s.
Sorties : Une arborescence A.

```
1  début
2  |  Calculer T_{G_Δ} une arborescence de cliques de G_Δ
3  |  Calculer O un ordre sur V(T_{G_Δ}) résultant d'un parcours en largeur de T_{G_Δ}
4  |  V(A) = s
5  |  E(A) = ∅
6  |  pour i allant de 1 à |V(T_{G_Δ})| faire
7  |  |  C ⟵ i-ème élément de O
8  |  |  tant que (C ∩ D) − V(A) ≠ ∅ faire
9  |  |  |  choisir x le sommet dans V(A) ∩ C de profondeur maximale dans A
10 |  |  |  choisir y dans (C ∩ D) − V(A)
11 |  |  |  V(A) = V(A) ∪ {x}
12 |  |  |  E(A) = E(A) ∪ (x, y)
13 |  retourner A
14 fin
```

Théorème 6 *:*
L'algorithme 1 renvoie une arborescence A cohérente pour [Diff,Async„Arbo,triangulé], et telle que les nœuds de A sont exactement les éléments de D.

Trouver un ensemble connexe dominant de taille minimum dans un graphe triangulé permet alors de résoudre le problème [Diff,Async„Arbo,triangulé]-minémissions sur ce dernier.

Preuve :
L'algorithme s'exécute selon deux invariants :

1. Au terme de chaque itération de premier niveau (qui s'étend des lignes 6 à 12) , l'ensemble $C \cap D$ est contenu dans $V(A)$.
2. Au début de chaque itération de premier niveau, la clique considérée C contient au moins un élément qui est dans $V(A)$:
   - Si cette clique est la première, $V(A) \cap C = \{s\}$.
   - Sinon, appelons $p(C)$ la clique parente de C dans A. En accord avec l'ordre O, $p(C)$ a été déjà traitée, et $p(C) \cap D \subseteq V(A)$. Puisque $C \cap p(C)$ est un séparateur, $C \cap p(C) \cap D \neq \emptyset$.

Au départ $V(A)$ contient uniquement s. Les éléments de A sont successivement ajoutés depuis l'ensemble $(C \cap D) - V(A)$. Donc $V(A) \subseteq D$. L'algorithme parcourt bien l'ensemble des cliques maximales de G_Δ. Donc $V(A) = D$.

L'arborescence retournée est bien cohérente pour [Diff,Async„Arbo,triangulé] :
- Soit x et y deux transmetteurs de $V(A)$ adjacents dans G_Δ. Le premier élément rencontré est forcément ancêtre du second.
- Soit x un récepteur de $V(G_\Delta) - V(A)$. Alors l'ensemble de ses voisins transmetteurs est sur la même branche de A.

□

Corollaire 4 *:*
Optimisé, l'algorithme 1 propose une solution au problème [Diff,Async„Arbo,]-minTemps avec un facteur d'approximation de $\max(l, \Delta_{G_\Delta})$ *où* l *est la taille de la clique maximale de* G_Δ*.*

Lorsque le degré est borné par k, le rapport d'approximation est k.

Preuve :
Supposons que l'algorithme 1 calcule une arborescence de hauteur minimale, notée $T^*_{G_\Delta}$. La hauteur de $T^*_{G_\Delta}$ est égale à l'excentricité de la source. Si D est un ensemble dominant connexe minimal, alors aucune arête n'est ajoutée à l'arborescence A lors de l'exploration d'une feuilles de $T^*_{G_\Delta}$, puisque les éléments de D qu'elle contient auront été ajoutés précédemment.
La hauteur de A est au plus de $l \times (hauteur(A_{G_\Delta}) - 1)$, où l est la taille de la clique maximale de G_Δ.
La propriété 7, et le fait que l est bornée par Δ_{G_Δ} permettent de conclure.

□

3.3.2 Graphes triangulés et ensemble dominant connexe minimum

D'après le théorème 6, résoudre [Diff,Async„Arbo,triangulé]-minTemps revient à trouver un ensemble dominant connexe de cardinalité minimum dans le graphe triangulé donné. Nous terminons ce chapitre par l'étude de la complexité de ce problème dans les graphes triangulés et ses sous-classes.

Trois classes de graphes sont considérées :

1. les graphes triangulés,
2. les path-graphes,
3. les graphes d'intervalles.

Un graphe est triangulé si et seulement s'il est le graphe d'intersection de sous-arbres d'un arbre. Les path-graphes sont une sous-famille des graphes triangulés, décrite par l'ensemble des graphes d'intersection de sous-chaines d'un arbre. Enfin les graphes d'intervalles, très étudiés dans la littératures sont les graphes d'intersection de sous-chaines d'une chaîne. Les relations d'inclusion entre ces classes sont immédiates. Nous énonçons le théorème suivant :

Théorème 7 *:*
Résoudre les problèmes [Diff,Async„Arbo,]-minTemps et [Diff,Sync„Arbo,]-minTemps est :
- *polynomial sur les graphes d'intervalles,*
- *NP-difficile sur les path-graphes,*
- *NP-difficile et non approximable à une constante sur les graphes triangulés.*

Preuve :
La complexité est identique à la recherche d'un ensemble connexe dominant de cardinalité minimum :
- Ce problème est polynomial dans les graphes d'intervalles : un algorithme polynomial en $O(n+m)$ a été proposé dans [RR88].
- Dans les path-graphes, la recherche d'un tel ensemble est un problème NP-difficile[2].
- Enfin, il est NP-difficile et non approximable à une constante près dans les graphes triangulés. La preuve du théorème 3 annonçait déjà ce résultat : le graphe construit dans cette dernière était triangulé et le résultat aboutissait à la non approximabilité du nombre minimum d'émissions.

□

[2]Communication personnelle de Dieter Kratsch : la preuve est une modification d'une démonstration de [LPHH84], prouvant que trouver un ensemble dominant total de cardinalité minimum dans les path-graphes est NP-difficile.

Chapitre 4

Mise en place d'heuristiques de diffusion

Les résultats du chapitre précédent sont sans appel : minimiser le temps de diffusion ou le nombre d'émissions est un problème difficile, quelle que soit la stratégie adoptée. Ces résultats conduisent à la nécessité de définir des heuristiques de diffusion. La première section de ce chapitre s'intéresse à la diffusion d'un message aux nœuds situés à distance 2 de la source. Le principal résultat de cette section est la mise en œuvre d'un algorithme proposant une stratégie en $O(\log^2 n)$ étapes sur cette classe de graphes. La seconde section est dédiée à la définition d'une heuristique sur la grille. Nous montrons que cette heuristique ne peut s'appliquer au tore, et proposons alors une nouvelle heuristique dans une troisième section.

4.1 Diffusion à distance 2

Les résultats présentés dans cette section n'auraient pu être établis sans l'aide de Genèvieve Simonet qui s'est particulièrement impliquée dans ces travaux. Qu'elle en soit ici remerciée.

Dans cette section nous restreignons notre étude au problème de la diffusion à une distance 2 dans un type de communications synchrone. Plus précisément, nous nous situons exclusivement dans le modèle [Diff,Sync,Tmp-EM„]. Le problème se résume ainsi : au terme de l'étape initiale, le message est parvenu à l'ensemble des nœuds situés à distance 1 de la source. Comment informer alors l'ensemble des nœuds à distance 2 ? Nous supposons que les sommets à distance 2 sont de simples récepteurs et ne rediffusent pas le message, donc que seuls les sommets de distance 1 peuvent émettre.

La donnée de ce problème peut se représenter par un graphe biparti $G = (X, Y, E)$. L'ensemble X contient les sommets qui ont connaissance du message, et l'ensemble Y représente les nœuds à distance 2. Les arêtes de E représentent les relations de voisinage. Nous ne représentons pas dans G la source s, ni les liens reliant s à l'ensemble des éléments de X.

Les problèmes [Diff,Sync,TMP-EM„]-minTemps et [Diff,Sync,TMP-EM„]-minémissions ont été montrés NP-difficiles, même lorsque l'excentricité de la source est d'au plus 2. Les principaux résultats de la section sont :

- l'introduction d'un nouvel outil : la mv-décomposition,
- la mise en place d'un algorithme proposant une stratégie de diffusion en $O(\log^2 n)$ étapes cohérente pour [Diff,Sync,TMP-EM„] ,
- l'étude du problème dans une famille de graphes particulière, les graphes convexes circulaires.

Chacun de ces points fait l'objet d'une sous-section.

4.1.1 Un outil pertinent : la mv-décomposition

Nous introduisons un nouvel outil appelé la ***mv-décomposition***. Les définitions suivantes nous permettent de définir cet outil :

Définition 36 (mono-voisinage, réceptivité, saturation, coût d'une saturation) *:*
Soit $G = (X, Y, E)$ un graphe biparti tel que X couvre Y.

*Pour $X' \subseteq X$, on appelle **mono-voisinage** de X' l'ensemble des voisins de X' qui ne sont voisins que d'un seul sommet de X'. On le note $mv_G(X')$.*

La réceptivité de G, notée $\rho(G)$, est alors définie par :

$$\rho(G) = \max_{X' \subseteq X} \left| mv_G(X') \right|$$

*On dit qu'un ensemble $\{X_i\}_{i \in I}$ de parties de X **sature** Y dans G lorsque :*

$$Y = \bigcup_{i \in I} mv_G(X_i)$$

*Le **coût de saturation** de G, noté $\sigma(G)$, est le cardinal minimal d'un ensemble de parties de X qui sature Y dans G*

Sur la figure 4.1 nous représentons un graphe biparti $G = (X, Y, E)$, si $X_1 = \{A, B, D\}$, alors $mv_G(X_1) = \{1, 3, 5, 6\}$. La réceptivité $\rho(G) = 5$ est obtenue pour $X_2 = \{A, D\}$, ce qui donne $mv_G(X_2) = \{1, 2, 4, 5, 6\}$. L'ensemble de parties $\{X_1, X_2\}$ sature Y dans G, et porte le coût de saturation $\sigma(G)$ à 2.

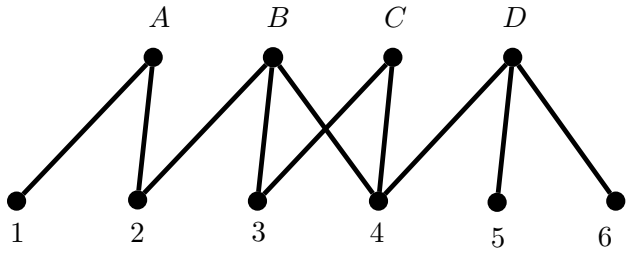


FIG. 4.1 – un graphe biparti.

Appliquons ces définitions au problème de diffusion à distance 2. Les nœuds d'une partie X_i émettent à une même étape, durant laquelle seuls les nœuds de $mv_G(X_i)$ reçoivent un message sans interférence. La démarche entreprise le long de cette section consiste à définir un ensemble $\{X_i\}_{i \in I}$ de parties de X saturant Y. Cet ensemble se construit ici au moyen d'une mv-décomposition. Enfin le coût de saturation est le nombre d'étapes minimum requis par une diffusion.

La couverture minimale, clé de voute de la mv-décomposition

La construction d'une mv-décomposition s'opère par le calcul d'une ***couverture minimale***, que nous définissons comme suit :

Définition 37 (couverture, couverture minimale) *:*
Soit $G = (X, Y, E)$ un graphe biparti.

*Une **couverture** dans G d'un sous-ensemble $Y' \subseteq Y$ est un sous-ensemble $X' \subseteq X$ tel que $Y' \subseteq N_G(X')$, où $N_G(X')$ désigne l'ensemble des voisins des sommets de X' dans G.*

*On dit que X' est une **couverture minimale** (au sens de l'inclusion) de Y' dans G lorsque X' est une couverture de Y mais qu'aucun de ses sous-ensembles stricts n'en est une.*

Sur l'exemple de la figure 4.1, $\{A, B, C, D\}$ est une couverture de Y qui n'est pas minimale, à l'inverse de $\{A, C, D\}$. Nous proposons le lemme suivant :

Lemme 1 *:*
Soient $G = (X, Y, E)$ un graphe biparti, et $X' \subseteq X$ une couverture minimale de $Y' \subseteq Y$.
Alors chaque sommet de X' possède dans Y' un voisin qui n'est voisin d'aucun autre sommet de X'.
Autrement dit, $mvg(X') \geq |X'|$.

Preuve :
Soient $X' \subseteq X$ une couverture dans G de $Y' \subseteq Y$ et x un sommet de X'. Si chaque voisin de x dans Y' est également voisin d'un autre sommet de X', alors $X' - \{x\}$ couvre encore Y'.

□

La mv-décomposition : Définitions et propriétés

Nous présentons maintenant un nouvel outil, la mv-décomposition. Un des résultats de la mv-décomposition d'un graphe biparti $G = (V, E)$ est un ensemble de parties de X saturant Y, qui va servir de support à une diffusion. La mv-décomposition se définit formellement comme suit :

Définition 38 (mv-décomposition) *:*
Soit $G = (X, Y, E)$ un graphe biparti tel que X couvre Y.

Une mv-décomposition de G consiste en la donnée d'un entier K et de trois suites $(X_i)_{0 \leq i \leq K}$, $(Y_i)_{0 \leq i \leq K}$ et $(Z_i)_{0 \leq i \leq K}$ satisfaisant les conditions suivantes :

1. *Pour $i = 0$:*
   - *$X_0 = X$*
   - *$Y_0 = Y$*
2. *Pour tout i tel que X_i soit défini et non vide :*
   - *$X_{i+1} \subseteq X_i$ est une couverture minimale de Y_i,*
   - *Z_i est défini de sorte que le biparti H_i, sous-graphe de G induit par $X_i \cup Z_i$, décrit un couplage parfait*[1]*,*

[1] tous ses sommets sont de degré égal à 1

- $Y_{i+1} = Y_i - Z_{i+1}$.

3. *K est l'entier pour lequel $Y_K = \emptyset$.*

L'entier K est la ***profondeur*** *de la mv-décomposition.*

Nous énonçons les propriétés suivantes :

Propriété 9 *:*
Soit $G = (X, Y, E)$ un graphe biparti tel que X couvre Y.

Alors G possède au moins une mv-décomposition, et pour toute mv-décomposition :

1. *$\{X_i\}_{0 \leq i \leq K}$ et $\{Y_i\}_{0 \leq i \leq K}$ sont deux suites emboitées pour l'inclusion, avec $X_K \neq \emptyset$ et $Y_K = \emptyset$. Par ailleurs X_i couvre Y_i pour $0 \leq i \leq K$.*
2. *$\{Z_j\}_{i \leq j \leq K}$ est une partition de Y_{i-1}. En particulier $\{Z_i\}_{1 \leq i \leq K}$ est une partition de Y.*
3. *pour tout i tel que $1 \leq i \leq K$, on a :*
   - *$|Z_i| = |X_i| \neq \emptyset$.*
   - *$Z_i \subseteq mv_G(X_i)$.*
4. *Pour tout i tel que $1 \leq i \leq K$, tout sommet x de X_i possède, pour tout j tel que $1 \leq j \leq i$, exactement un voisin dans Z_j qui n'est voisin d'aucun autre sommet de X_i.*

Preuve :
Soient $X_i \subseteq X$ et $Y_i \subseteq Y$, Y_i non vide, tels que X_i couvre Y_i (vrai pour $i = 0$). Alors Y_i possède au moins une couverture minimale $X_{i+1} \subseteq X_i$. Le lemme 1 permet d'affirmer que pour tout X_{i+1} non vide, Z_{i+1} est bien défini et non vide, et donc que Y_{i+1} est strictement inclus dans Y_i. Ceci assure également que X_{i+1} couvre Y_{i+1} L'existence d'une mv-décomposition pour tout G est donc assurée, de même que les points (1), (2) et (3).

Pour tout i et j tels que $1 \leq j \leq i \leq K$, $X_i \subseteq X_j$. Alors tout sommet x de X_i est aussi un sommet de X_j. Comme le sous-graphe induit de G par les sommets $X_j \cup Z_j$ définit un couplage parfait, il existe donc un sommet $z_j \in Z_j$ voisin de x qui n'est voisin d'aucun autre sommet de X_j. □

La figure 4.2 présente une mv-décomposition d'un graphe biparti. Les arêtes en noir désignent pour chaque nœud y la provenance d'un message sans interférence. Dans une diffusion, l'émission simultanée des sommets d'un ensemble X_i de cardinalité l entraîne une réception correcte pour les l sommets de Z_i, ainsi que pour l sommets de chaque ensemble Z_j avec $1 \leq j \leq i$. Les sommets de Y_i reçoivent obligatoirement tous une réception avec interférences.

Propriété 10 *:*
Soit $G = (X, Y, E)$ un graphe biparti tel que X couvre Y. Alors pour toute mv-décomposition de G de profondeur K, on a :

$$K \leq \Delta_G(X) \tag{4.1}$$

en notant $\Delta_G(X)$ le degré maximum dans G d'un sommet de X.

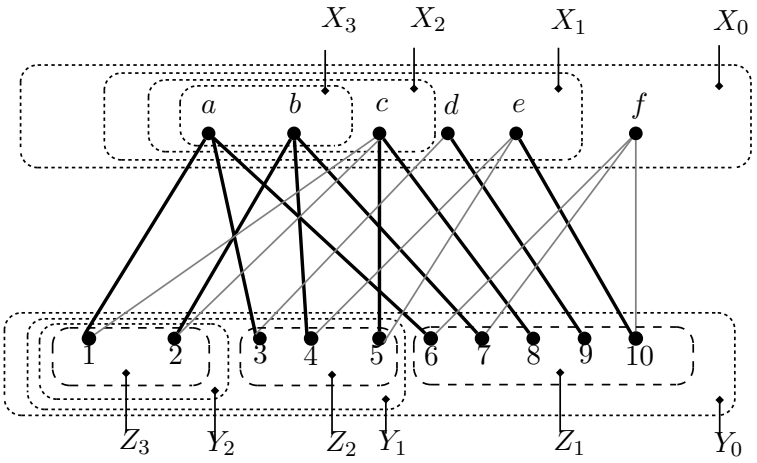


FIG. 4.2 – Construction d'une mv-décomposition

Preuve :

En reprenant les notations de la définition 38, et d'après les points 2 et 4 de la propriété 9, on a $d_G(x) \geq K$. Ce qui permet de conclure.

□

Nous terminons cette sous-section par la remarque suivante :

Remarque 4 *:*

Intuitivement dans le problème de diffusion, on peut se convaincre du fait que l'émission simultanée des nœuds d'un ensemble X_i quelconque va assurer une diffusion sans interférence vers les nœuds de Z_i.

Dans la définition formelle d'une mv-décomposition $(X_i)_{0\leq i\leq K}$, $(Y_i)_{0\leq i\leq K}$ et $(Z_i)_{0\leq i\leq K}$, l'équation suivante est toujours satisfaite :

$$|X_i| = |Z_i| \tag{4.2}$$

On peut définir une mv-décomposition que nous qualifions d'alternative, en remplaçant l'équation 4.2 par l'inéquation suivante, sans altérer pour autant la diffusion des X_i vers les Z_i :

$$|X_i| \leq |Z_i| \tag{4.3}$$

Cette modification permettrait de calculer des mv-décompositions plus pertinentes : par exemple dans des topologies comme l'étoile à n branches, la profondeur minimale d'une mv-décomposition alternative serait de 1, alors que celle d'une mv-décomposition classique serait de $n-1$.

Dans l'étude théorique menée en sous-section 4.1.4, il est nécessaire d'avoir une égalité entre les cardinaux de X_i et Z_i pour établier le rapport d'approximation de l'algorithme Saturation. Intuitivement, remplacer l'égalité 4.2 par l'inégalité 4.3 doit conduire à une approximation au moins aussi bonne, mais cette intuition n'a pas été prouvée par le calcul.

4.1.2 Calcul d'une mv-décomposition

Nous présentons l'algorithme *mv-decomposition* qui calcule la mv-décomposition d'un graphe biparti $G = (X, Y, E)$ donné.

Algorithme 2 : mv-décomposition

Données : Un graphe biparti $G = (X, Y, E)$
Résultat : Une suite $(X_i)_{1 \leq i \leq K}$ de parties de X qui sature Y dans G
// Déclaration des variables :
1 Pile $P[x]$: pile des sommets de Y dont le seul voisin est x, $\forall x \in X$.
2 int L : nombre d'éléments de Y qui ont été traités.
3 int i : indice de profondeur actuelle
// Initialisation des variables :
4 $L = 0$; $X[0] = X$; $i = 1$
5 Initialiser $P[x]$, $\forall x \in X$.
6 **tant que** $L < |Y|$ **faire**
7 $X[i] = \emptyset$
// Calcul d'une couverture minimale X[i] :
8 **pour tous les** $x \in X[i-1]$ **faire**
9 **si** $|P[x]| = 0$ **alors**
// Suppression de x du voisinage de ses voisins :
10 **pour tous les** $y \in N(x)$ **faire**
11 $N(y) = N(y) - \{x\}$
12 **si** $|N(y)| = 1$ **alors**
// Si y n'a plus qu'un voisin z, il est ajouté à $P[z]$
13 $P[N(y)] = P[N(y)] \cup \{y\}$
14 **fin**
15 **fin**
16 **fin**
17 **sinon**
// x est sélectionné dans la couverture.
18 $X[i] = X[i] \cup \{x\}$
// un sommet y devient récepteur de l'émission de x :
19 Soit $y \in P[x]$. $P[x] = P[x] - \{y\}$.
20 $L++$
21 **fin**
22 **fin**
23 **fin**

Théorème 8 *L'algorithme mv-decomposition a une complexité de* $O(m)$.

Preuve :

La phase d'initialisation (ligne 5) se déroule en $O(m)$: un parcours des arêtes puis des sommets de Y permettent d'initialiser les piles $P[x], \forall x \in X$.

Par la suite, et pour un sommet $x \in X$ donné :

- on exécute au plus une fois la portion de code située entre les lignes 9 et 16, et qui consiste à supprimer le sommet x (il ne sera alors plus considéré par la suite).
- on exécute au plus $d_G(x)$ fois la portion de code située entre les lignes 17 et 21.

La complexité de la portion de code située entre les lignes 9 et 15 est en $O(d_G(x))$, à la condition que l'accès au voisinage d'un sommet (ligne 10) ainsi que que la suppression du sommet x du

voisinage de y (ligne 11) soient en $O(1)$. Une telle contrainte peut s'obtenir en programmation par l'utilisation d'une matrice à trous[2] pour représenter les relations de voisinage.

La complexité de la portion de code située entre les lignes 17 et 21 est en $O(1)$.

On en conclut que la complexité globale de l'algorithme est de l'ordre de :

$$O(\sum_{x \in X} d_G(x)) = O(m)$$

□

Le calcul d'une mv-décomposition par cet algorithme permet l'évaluation du rapport d'approximation de l'algorithme Saturation de la sous-section 4.1.4, en accord avec l'équation 4.2 de la remarque 4,

Nous proposons une seconde version de cet algorithme qui renvoie une mv-decomposition alternative, évoquée dans la remarque 4. Cette version offre de meilleurs résultats pratiques, puisque le nombre de sommets de Y éliminés par étape peut être plus important. La complexité de cet algorithme demeure en $O(m)$.

Un des éléments de cet algorithme est le tableau de booléens M, qui indique pour chaque sommet $x \in X$ s'il existe un élément y dont il est le seul voisin, auquel cas x doit être ajouté à l'ensemble X_i en cours.

[2]La mémoire réservée peut être de l'ordre de $O(n^2)$, mais le fait de ne pas initialiser cette mémoire et de n'utiliser que certaines des cases réduit la complexité en fonction du nombre de cases initialisées.

Algorithme 3 : L'algorithme mv-decomposition alternative

Données : Un graphe biparti $G = (X, Y, E)$
Résultat : Une suite $(X_i)_{1 \leq i \leq K}$ de parties de X qui sature Y dans G
`// Déclaration des variables :`
M : tableau de booléens indexé par les sommets de X
L : liste des sommets de Y de degré 1
`// Initialisation des variables :`
Initialiser M, L $\forall x \in X, y \in Y$.
$Y' = Y$, $X[0] = X$, $i = 1$
tant que $Y' \neq \emptyset$ **faire**
  $X[i] = \emptyset$
  pour tous les $x \in X[i-1]$ **faire**
    si $M[x]$ **alors**
      $X[i] = X[i] \cup \{x\}$
      $M[x] = 0$
    fin
    sinon
      pour tous les $y \in N_G(x) \cap Y'$ **faire**
        $N_G(y) = N_G(y) - \{x\}$
        si $|N_G(y)| = 1$ **alors**
          $L = L \cup \{y\}$
          $M[N_G(y)] = 1$
      fin
    fin
  fin
  $Y' = Y' - L$
  $L = \emptyset$
  $i = i + 1$
fin

4.1.3 mv-décomposition et minimisation du nombre d'émissions

Cette sous-section se consacre au problème de minimisation du nombre d'émissions lors d'une diffusion à distance 2 dans le modèle [Diff,Sync,TMp-EM$_{\mathbb{N}}$]. Le théorème suivant établit le lien entre la couverture des sommets Y d'un graphe biparti $= (X, Y, E)$, et nombre d'émetteurs requis pour le problème de la diffusion à distance 2 sur le même graphe.

Théorème 9 *:*
Soit $G = (X, Y, E)$ un graphe biparti tel que X couvre Y. Alors :

1. *Si C est une couverture minimale de Y, il en résulte une diffusion de X à Y avec un nombre minimal d'émetteurs de même cardinalité que C.*
2. *Trouver un nombre minimum d'émetteurs revient à trouver une couverture minimum.*

3. *S'il existe un algorithme ρ-approché pour le problème de la couverture minimum, alors il existe un algorithme ρ-approché pour le problème de la diffusion de X à Y en minimisant le nombre d'émissions.*

Preuve :
Soit $C \subseteq X$ une couverture minimale de Y de cardinalité k, avec $C = \{c_i\}_{1 \leq i \leq k}$. Nous faisons émettre séquentiellement chacun des sommets de C. Au terme de k étapes tous les sommets de C ont émis, sans générer d'interférences. Les sommets de Y ont donc bien tous reçu l'information. Nous en déduisons les points 1 et 2. La cardinalité du nombre d'émetteurs est clairement égale à celle de la couverture C. Toute transformation conserve donc le rapport d'approximation d'une solution.

□

Soit $G = (X, Y, E)$ un graphe biparti, et un ensemble $\left\{\{X_i\}_{1 \leq i \leq K}\right\}$ de parties de X résultant de la mv-décomposition de G. De toute mv-décomposition de G nous pouvons déduire une stratégie de diffusion cohérente pour [Diff,Sync,Tmp-EM„]. Par exemple nous définissons une stratégie S_1 comme suit :

– À l'étape i, tous les sommets de X_{K+1-i} émettent le message.

Cette stratégie aboutit bien à une diffusion de X à Y : puisque l'ensemble $X_{i1 \leq i \leq K}$ sature Y, le message sera bien reçu par les sommets de Y au terme de l'étape K. Analysons les coûts en nombre d'émissions de S_1. Son nombre d'émissions est :

$$\sum_{i=1}^{K} |X_i|$$

D'après le théorème 9, le coût en nombre d'émissions de la stratégie S_1 n'est pas minimal, puisqu'à lui seul X_1 est une couverture minimale de Y.

Nous définissons alors la stratégie suivante à partir de toute mv-décomposition de G :

– A l'étape 1, tous les sommets de X_K émettent le message.

– A l'étape i tel que $2 \leq i \leq K$, tous les sommets de $X_{K+1-i} - X_{K+2-i}$ émettent le message.

La stratégie de diffusion S_2 diffère de S_1 du fait que les sommets qui ont émis à l'étape i n'émettent plus à l'étape $i+1$. Nous affirmons la propriété suivante :

Propriété 11 *:*
La stratégie de diffusion S_2 aboutit à une diffusion de X à Y.

Preuve :
Comparons la stratégie S_2 à la stratégie S_1. À l'étape 1, les sommets de X_K émettent. À l'étape 2 ces derniers n'émettent plus dans $S2$ alors qu'ils émettent dans $S1$.

Or l'émission de ces derniers est superflue : rappelons que $X_{i+1} \subseteq X_i$ pour tout i tel que $1 \leq i \leq k-1$. Donc :

– si les émissions des X_K à l'étape 2 atteignaient correctement un sommet y dans la stratégie S_1, y avait déjà reçu correctement le message à l'étape 1.

– si les émissions des X_K provoquaient des interférences sur un sommet y à l'étape 1, alors dans S_1, l'émission des mêmes sommets à l'étape 2 provoquerait également des interférences sur y.

Nous étendons ce raisonnement à chaque étape. Il en découle que S_2 est une stratégie valide.

□

Analysons les coûts de S_2 :
Le nombre d'étapes requis par la stratégie S2 est K. Son coût en nombre d'émissions est :

$$|X_K| + \sum_{i=2}^{K} |X_{K+1-i} - X_{K+2-i}| = |X_1|$$

L'ensemble X_1 constitue une couverture minimale de Y. Nous obtenons donc une stratégie de diffusion valide, dont le coût en nombre d'émissions est minimal par rapport à la décomposition, en accord avec le théorème 9.

Le nombre d'étapes requis ici est inférieur ou égal à $\Delta_G(X)$ d'après la propriété 10. Nous allons voir qu'il est possible d'obtenir une stratégie présentant de meilleurs coûts en nombre d'étapes.

4.1.4 mv-décomposition et minimisation du temps de diffusion

Cette sous-section s'intéresse au problème de minimisation du nombre d'étapes d'une diffusion de X à Y. Nous utilisons la mv-décomposition pour produire un algorithme retournant une solution dont le nombre d'étapes est en $O(\log^2 n)$.

Propriétés et théorèmes

Propriété 12 *:*
Soit $G = (X, Y, E)$ un graphe biparti tel que X couvre Y. Alors, la réceptivité $\rho(G)$ vérifie l'inéquation suivante :

$$\max_{X' \in X} |mv_G(X')| = \rho(G) \geq \max(\Delta_G(X), \frac{|Y|}{\Delta_G(X)})$$

Preuve :
Soit x un sommet de G de degré $\Delta_G(X)$. Alors $|mv_G(\{x\})| = \Delta_G(X)$. L'inégalité $\rho(G) \geq \Delta_G(X)$ se déduit alors de la définition de $\rho(G)$.

Soit $X' \subseteq X$ une couverture minimale de Y dans G. Comme $Y = \bigcup_{x \in X'} N_G(x)$, alors :

$$|Y| \leq \sum_{x \in X'} |N_G(x)| = \sum_{x \in X'} d_G(x) \leq \left|X'\right| \times \Delta_G(X) \leq mv_G(X') \times \Delta_G(X)$$

La seconde inégalité $\rho(G) \geq \frac{|Y|}{\Delta_G(X)}$ se déduit encore de la définition de $\rho(G)$.

□

En corollaire immédiat à la propriété 12, on a donc :

$$\rho(G) \geq \sqrt{|Y|}$$

En fait nous allons affiner cette borne et montrer que :

$$\rho(G) \geq \frac{|Y|}{1 + \ln |Y|}$$

Propriété 13 *:*
Soit $G = (X, Y, E)$ un graphe biparti tel que X couvre Y. Alors, pour toute mv-décomposition de G, on a, en reprenant les notations de la définition 38 :

$$\forall i | 1 \leq i \leq K, |mv_G(X_i)| \geq i \times |X_i| \tag{4.4}$$

Preuve :
D'après le point 4 de la propriété 9, tout sommet x de X_i possède dans chaque Z_j, pour $1 \leq j \leq i$, un voisin qui n'est voisin d'aucun autre sommet de X_i. D'après le point 2 de cette même propriété, ces i voisins sont tous différents.

□

Théorème 10 *:*
Soit $G = (X, Y, E)$ un graphe biparti tel que X couvre Y. Alors pour toute mv-décomposition de G, on a :

$$\rho(G) \geq \max_{1 \leq i \leq K} |mv_G(X_i)| \geq \frac{|Y|}{H_K} \tag{4.5}$$

$$\sigma(G) \leq K \tag{4.6}$$

où H_n est le nombre harmonique $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$.

Preuve :
La premère inégalité de la première inéquation découle de la définition de $\rho(G)$. La seconde se déduit du calcul suivant :

$$\begin{aligned}
|Y| &= \sum_{i=1}^{K} |Z_i| \\
&= \sum_{i=1}^{K} |X_i| \\
&\leq \sum_{i=1}^{K} \frac{|mv_G(X_i)|}{i} \\
&\leq \sum_{i=1}^{K} \frac{\max_{1 \leq i \leq K} |mv_G(X_i)|}{i} = \max_{1 \leq i \leq K} |mv_G(X_i)| \times H_K
\end{aligned}$$

Montrons l'inéquation 4.6. Soit y un sommet quelconque de Y. Par le point 2 de la propriété 9, $y \in Z_i$ pour un i tel que $1 \leq i \leq K$. Par le point 3 de cette même propriété, $y \in mv_G(X_i)$. Donc $Y = \bigcup_{i=1}^{K} mv_G(X_i)$, ce qui permet de conclure.

□

Théorème 11 *:*
Soit $G = (X, Y, E)$ un graphe biparti tel que X couvre Y. Alors :

$$\rho(G) \geq \frac{|Y|}{1 + ln\Delta_G(X)} \tag{4.7}$$

$$\sigma(G) \leq \Delta_G(X) \tag{4.8}$$

Preuve :
Se déduit du théorème 10 et de la propriété 10, compte tenu du fait que le nombre harmonique H_n est une fonction croissante de n qui satisfait $H_n \leq 1 + \ln n$.

□

l'algorithme Saturation

Nous proposons l'alorithme suivant :

Algorithme 4 : L'algorithme Saturation

Données : Un graphe biparti $G = (X, Y, E)$ tel que X couvre Y
Résultat : Un ensemble $\{W_t\}_{1 \leq t \leq L}$ qui sature Y dans G

```
1  R = Y;
2  t = 0;
3  tant que R ≠ ∅ faire
4  |  t = t + 1;
5  |  Calculer une mv-décomposition de G[X,R];
6  |  Soit K_t sa profondeur, et soit (X_i^t)_{0≤i≤K_t} la suite décomposant X;
7  |   choisir i dans {1, ..., K_t} de sorte que mv_{G_[X,R]}(X_i^t) soit de cardinal maximum;
8  |  R = R − mv_{G_[X,R]};
9  |  W_t = X_i^t;
10 fin
11 L = t;
12 Retourner {W_t}_{1≤t≤L} ;
```

Clairement $\{W_t\}_{1 \leq t \leq L}$ est un ensemble de sous-ensembles de X saturant Y. Une stratégie de diffusion valide se déduit alors logiquement de ce dernier en faisant émettre les sommets de W_i à l'étape i. Le nombre d'étapes de cette stratégie est la cardinalité de l'ensemble résultat, c'est-à-dire le nombre d'itérations L de l'algorithme.

Théorème 12 *:*
L'algorithme Saturation s'exécute en $O(\ln^2 |Y|)$ itérations, c'est-à-dire que toute stratégie de diffusion construite à partir de l'ensemble $\{W_t\}_{1 \leq t \leq L}$ s'exécute en $O(\ln^2 |Y|)$ étapes.

Preuve :
En accord avec la propriété 10, $K_t \leq \Delta_{G_{[X,R]}} \leq |R|$. Au cours d'une itération, on a, compte tenu

du théorème 10 :

$$\left|mv_{G_{[X,R]}}(X_i^t)\right| \geq \frac{|R|}{H_k} \geq \frac{|R|}{1+\ln|R|}$$

Notons u_n le cardinal de R après la n^{eme} itération. On a alors :

$$\begin{aligned} u_0 &= |Y| \\ u_{n+1} &\leq u_n\left(1-\frac{1}{1+\ln u_n}\right), 0 \leq n \leq L \\ u_L &= 0 \end{aligned}$$

Soit $(v_n)_{n\in N}$ la suite géométrique définie par :

$$v_n = |Y|\left(1-\frac{1}{1+\ln|Y|}\right)^n$$

On a :

$$v_{\theta(\ln^2|Y|)} = 0$$

En effet :

$$\begin{aligned} v_n < 1 &\Leftrightarrow \ln v_n < 0 \\ &\Leftrightarrow \ln|Y| + n\ln\left(1-\frac{1}{1+\ln|Y|}\right) < 0 \end{aligned}$$

Comme $\ln(1+x) \leq x$, pour que $v_n < 1$, il suffit que $\ln|Y| - \frac{n}{1+\ln|Y|} < 0$, soit $n > \ln|Y| \times (1+\ln|Y|)$.

Clairement on a $v_n \geq u_n, \forall n$ et donc $L \leq \ln|Y| \times (1+\ln|Y|)$.

□

Dans des travaux précédents, [CW91] proposait déjà un algorithme renvoyant une stratégie de diffusion en $O(\ln^2 n)$ étapes. Ce résultat n'est pas nouveau. Cependant, l'algorithme proposé dans ce papier s'exécute en un temps d'exécution en $O(n.m\ln^2 n)$. Le théorème suivant atteste que notre algorithme possède une meilleure complexité :

Théorème 13 *:*
La complexité en temps de l'algorithme Saturation est de l'ordre de $O(m \times \ln^2|Y|)$.

Preuve :

À chaque itération, l'algorithme calcule une mv-décomposition. Une décomposition se calcule en $O(m)$ d'après le théorème 8. Enfin rappelons que le nombre d'itérations de l'algorithme est en $O(\ln^2|Y|)$, en accord avec le théorème 12.

□

4.1.5 diffusion à distance 2 dans les graphes convexes circulaires

Nous étudions la diffusion à distance 2 dans le modèle [Diff,Sync,Tmp-EM$_\infty$] sur un graphe biparti convexe circulaire $G = (X, Y, E)$. Un graphe biparti $G = (X, Y, E)$ est dit ***convexe circulaire*** en Y s'il existe un ordre circulaire sur Y tel que pour tout sommet $x \in X$, les sommets de Y adjacents à x apparaissent consécutivement dans cet ordre.

La figure 4.1.5 présente un graphe convexe circulaire.

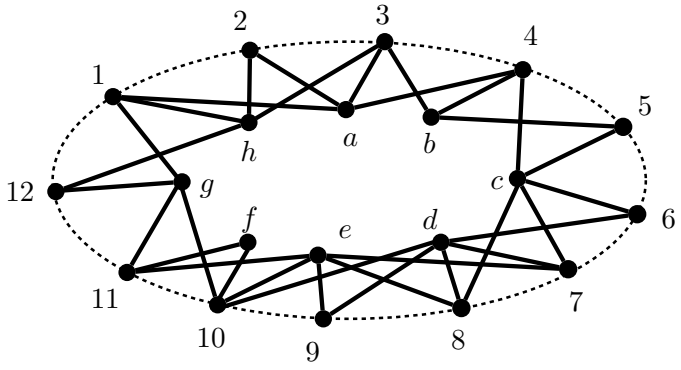


FIG. 4.3 – Un graphe convexe circulaire

Remarque 5 *:*
Une représentation naturelle consiste à définir les sommets de Y comme les points entiers d'un cercle de périmètre $|Y|$. Nous posons ainsi $Y = \{1, 2, \ldots, |Y|\}$. Les sommets de X sont alors des intervalles, tels que x contient y si et seulement si l'arête $\{x, y\}$ existe dans E.

Nous proposons dans cette sous-section une stratégie de diffusion dont la construction repose sur le calcul d'une couverture minimum. Nous énonçons à ce propos le théorème suivant :

Théorème 14 *:*
Le calcul d'une couverture minimum des sommets de Y par $X' \subseteq X$ sur un graphe convexe circulaire $G = (X, Y, E)$ d'effectue en temps polynomial.

Preuve :
Nous utilisons la représentation introduite dans la remarque 5. Le problème de la couverture minimum équivaut à sélectionner un ensemble d'intervalles de cardinalité minimum et dont l'union doit contenir l'ensemble des entiers de 1 à $|Y|$.

L'entier 1 est inclus dans X_1 intervalles dont au moins un appartient à une couverture minimum. Pour chaque élément x_i de X_1, nous construisons une couverture minimale C_{x_i} comme suit :

1. Nous posons $C_{x_i} = \{x_i\}$.
2. Soit y le plus petit entier de $[1, |Y|]$ inclus dans aucun des intervalles de C_{x_i}. Alors parmi les intervalles de X couvrant y, nous ajoutons à C_{x_i} celui qui contient le plus grand entier.
3. Nous répétons la procédure précédente jusqu'à ce que tous les éléments de Y soient couverts par des intervalles de C_{x_i}.

Parmi l'ensemble des couvertures minimales $C_{x_i}|x_i \in X_1$, se trouve une couverture minimum. □

Lemme 2 *:*
Soit C une couverture minimale de Y de cardinalité k. Alors il existe un ordre $\{x_1, x_2, \ldots, x_k\}$ sur les éléments de C tel que $N(x_i) \cap N(x_j) \neq 0 \Rightarrow |j-i| \ mod \ (k-1) \leq 1$.

Preuve :
Nous détaillons la procédure permettant de définir un tel ordre, en utilisant la représentation en intervalles introduite dans la remarque 5.

Nous posons $x_0 \in C$ le premier sommet de l'ordre, avec $x_0 = [a_0, b_0]$[3]. Il existe un unique élément $x_1 = [a_1, b_1]$ de C tel que $b_0 + 1 \in N(x_1)$, que nous ajoutons immédiatement après x_0 dans l'ordre. Nous continuons cette procédure, en ajoutant successivement dans l'ordre les éléments $x_i = [a_i, b_i]$ tels que $b_{i-1} + 1 \in N(x_i)$, et ce jusqu'à parcourir tous les éléments de X.

Remarquons que dans le calcul d'une couverture minimum présenté dans la preuve du théorème 14, l'ordre dans lequel sont ajoutés les sommets vérifie bien la propriété du lemme. □

Par la suite, pour une couverture minimale C de cardinalité k donnée, nous appelons $O_C = (x_1, x_2, \ldots, x_k)$ l'ordre sur les éléments de C renvoyé par la construction de la preuve du lemme 2.

Propriété 14 *:*
Soient $G = (X, Y, E)$ un graphe convexe circulaire, et C une couverture minimum de Y de cardinalité k. Alors pour tout sommet $y \in Y$:
- *y est adjacent à au moins un sommet $x_i \in C$,*
- *y est adjacent à au plus deux sommets qui sont consécutifs dans O_C (ou alors au premier et au dernier élément de O_C),*
- *y ne peut être adjacent à trois sommets ou plus de C, autrement la couverture n'est pas minimale.*

Considérons C une couverture minimum de Y de cardinalité k. La propriété précédente nous permet de partitionner l'ensemble Y comme suit :
- Y_i désigne l'ensemble des sommets de Y adjacents au sommet x_i uniquement.
- $Y_{\{i,j\}}$ désigne l'ensemble des sommets de Y tels que $Y_{\{i,j\}} = N(x_i) \cap N(x_j)$.

En accord avec l'ordre O_C défini, $Y_{\{i,j\}} = \emptyset$ si $|j-i| \ mod \ (k-1) > 1$.

Considérons l'émission de sommets de C à une étape donnée :
- Si x_i émet seulement, alors Y_i, $Y_{\{i,i-1\}}$, et $Y_{\{i,i+1\}}$ reçoivent correctement l'information
- Si x_i et x_{i+1} émettent seulement, alors Y_i, $Y_{\{i,i-1\}}$, Y_{i+1} et $Y_{\{i+1,i+2\}}$ reçoivent correctement l'information, mais pas $Y_{\{i,i+1\}}$

Théorème 15 *:*
Il est possible d'informer au moins $2|Y|/3$ sommets de Y en une étape.

[3]Si x_i contient $|Y|$ et 1, alors on supposera que b_0 est plus petit que a_0. Autrement $a_0 \leq b_0$.

Preuve :
Soit C une couverture minimum de Y de cardinalité k et d'ordre associé $O_C = (x_1, x_2, \ldots, x_k)$.

Définissons trois ensembles T_1, T_2, et T_3 en fonction de la parité de k. Soit Y_{T_i} l'ensemble des sommets de Y qui reçoivent le message lorsque seulement tous les éléments de T_i émettent simultanément.

Si k est pair, nous définissons les sous-ensembles suivants :
- $T_1 = \{x_1, x_3, x_5, x_7, \ldots, x_{k-1}\}$.
- $T_2 = \{x_2, x_4, x_6, x_8, \ldots, x_k\}$.
- $T_3 = \{x_1, x_2, x_3, x_4, x_5, \ldots, x_k\}$.

Les ensembles Y_{T_i} correspondants sont :
- $Y_{T_1} = \big\{\{Y_{2i-1}, \forall i \in [1, k/2]\} \cup \{Y_{\{i,i+1\}}, \forall i \in [1, k-1]\} \cup Y_{\{1,k\}}\big\}$.
- $Y_{T_2} = \big\{\{Y_{2i}, \forall i \in [1, k/2]\} \cup \{Y_{\{i,i+1\}}, \forall i \in [1, k-1]\} \cup Y_{\{1,k\}}\big\}$.
- $Y_{T_3} = \big\{\{Y_i, \forall i \in [1, k]\}\big\}$.

Si k est impair, nous définissons les sous-ensembles suivants :
- $T_1 = \{x_1, x_3, x_5, x_7, x_9, \ldots, x_k\}$.
- $T_2 = \{x_1, x_2, x_4, x_6, x_8, \ldots, x_{k-1}\}$.
- $T_3 = \{x_2, x_3, x_4, x_5, x_6, \ldots, x_k\}$.

Les ensembles Y_{T_i} correspondants sont :
- $Y_{T_1} = \big\{\{Y_{2i-1}, \forall i \in [1, k/2]\} \cup \{Y_{\{i,i+1\}}, \forall i \in [1, k-1]\}\big\}$.
- $Y_{T_2} = \big\{\{Y_{2i}, \forall i \in [1, k/2]\} \cup \{Y_{\{i,i+1\}}, \forall i \in [2, k-1]\} \cup Y_{\{1,k\}}\big\}$.
- $Y_{T_3} = \big\{\{Y_i, \forall i \in [1, k]\} \cup Y_{\{1,k\}}\big\}$.

Quelle que soit la parité de k, chaque sous-ensemble de la forme Y_i ou $Y_{\{i,i+1\}}$ est inclus dans deux des trois ensembles Y_{T_1}, Y_{T_2} et Y_{T_3}. L'union de ces sous-ensembles forme l'ensemble Y. Si on fait émettre les sommets de T_1, puis de T_2, puis de T_3, chaque sommet y reçoit deux fois l'information. Donc dans un cas au moins $\frac{2}{3}$ des sommets de Y reçoivent l'information.

□

Proposition 1 *:*
Il est possible d'informer la totalité des sommets de Y en 2 étapes au plus.

Preuve :
Reprenons la preuve du théorème 15. Tout sommet $y \in Y$ est inclus dans deux des trois sous-ensembles Y_{T_1}, Y_{T_2} et Y_{T_3}, donc dans au moins un ensemble parmi Y_{T_1} et Y_{T_2}. En faisant émettre les sommets de l'ensemble T_1 à l'étape 1, et T_2 à l'étape 2, nous assurons la réception par tous les sommets de y.

□

4.2 Une heuristique décentralisée dans la grille

Nous présentons dans la section suivante une heuristique qui retourne une stratégie de diffusion valide pour le modèle [Diff,Sync,Tmp-EM, grille].

Une grille $G_{M\times N}$ représente une topologie très structurée. Elle constitue un intérêt particulier dans le déploiement de réseaux sans fil à l'échelle d'une ville, d'un bâtiment. Nous présentons d'abord une heuristique centralisée, puis évaluons ses performances en temps et diffusion. Nous montrons par la suite comment décentraliser cette heuristique.

4.2.1 Mise en place d'une heuristique centralisée

Nous présentons les grandes lignes d'une heuristique retournant une arborescence A cohérente pour [Diff,Sync,Tmp-EM,,grille], sur une grille de dimensions $P \times Q$. Cette heuristique se découpe en trois phases :

1. Découpage de la grille en blocs de colonnes, et calcul d'une solution S_1.
2. Découpage de la grille en blocs de lignes, et calcul d'une solution S_2.
3. Évaluation de la meilleure solution parmi S_1 et S_2 en fonction de la valeur à minimiser (temps ou nombre d'émissions).

Les deux premières phases sont identiques, à la rotation de la grille près. Aussi dans cette sous-section nous ne détaillons que la première phase relative au découpage en colonnes.

Découpage en blocs :

Soient $G_{P\times Q}$ une grille de dimensions $P \times Q$, et un nœud source de coordonnées $[x_s, y_s]$. Nous notons (x, y) le sommet de coordonnées $[x, y]$. Les colonnes sont indicées de 1 à Q.

La grille $G_{P\times Q}$ est découpée en blocs de deux ou trois colonnes d'indices consécutifs, chaque colonne appartenant nécessairement à un et un seul bloc. Ce découpage dépend du nombre de colonnes Q :

- Si $Q = 3k$, la grille est découpée en k blocs de trois colonnes chacun. Le bloc d'indice i contient les colonnes d'indices $3i - 2$, $3i - 1$ et $3i$.
- Si $Q = 3k + 1$, la grille est découpée en $(k - 1)$ blocs de trois colonnes, et en 2 blocs de deux colonnes organisés comme suit : le bloc d'indice 1 contient les colonnes 1 et 2. Les blocs 2 à k englobent trois colonnes, le bloc i contenant les colonnes $3i-3$, $3i-2$ et $3i-1$ pour $i \in [2, k]$. Le bloc $k + 1$ contient les colonnes $Q - 1$ et Q.
- Si $Q = 3k + 2$, la grille est découpée en k blocs de trois colonnes, le bloc i contenant les colonnes $3i - 2$, $3i - 1$ et $3i$. Un bloc d'indice $k+1$ contenant deux colonnes $Q-1$ et Q vient compléter le découpage.

Définition des relations de parenté :

L'idée générale de l'heuristique se présente comme suit :
Dans chaque bloc de trois colonnes, les nœuds de la colonne centrale sont des transmetteurs. Les sommets de la colonne 1 sont également des transmetteurs, à condition qu'elle appartienne à un

bloc de deux colonnes. Même chose pour les sommets de la colonne Q. L'ajout de transmetteurs supplémentaires va permettre de connecter ces colonnes de transmetteurs au sommet source.

Nous appelons r l'indice du bloc contenant le sommet source s. Suivant la position de s dans le bloc r, l'heuristique construit deux arbres de diffusion légèrement différents.

Détaillons la construction d'une arborescence A_1 lorsque le sommet source ne se situe pas sur la colonne centrale du bloc r. Nous définissons les relations de parenté suivantes[4] (la compréhension de ces relations est facilitée par la figure 4.4) :

- Pour tout bloc composé de trois colonnes $\{a-1, a, a+1\}$:
  - Le père de $(a, i-1)$ est $(a, i) \forall i \in [1, y_s - 1]$.
  - Le père de $(a, i+1)$ est $(a, i) \forall i \in [y_s + 1, q - 1]$.
- Pour tout bloc d'indice $l < r$ composé de trois colonnes $\{a-1, a, a+1\}$:
  - Le père de $(a, y_s + 1)$ est $(a-1, y_s + 1)$
  - Le père de $(a, y_s - 1)$ est $(a-1, y_s - 1)$
  - Le père de $(a-1, y_s + 1)$ est $(a-1, y_s)$
  - Le père de $(a-1, y_s - 1)$ est $(a-1, y_s)$
  - Le père de $(a-1, y_s)$ est (a, y_s)
  - Le père de (a, y_s) est $(a+1, y_s)$
  - Le père de $(a+1, y_s - 1)$ est $(a+2, y_s)$
- Pour tout bloc d'indice $l > r$ composé de trois colonnes $\{a-1, a, a+1\}$:
  - Le père de $(a, y_s + 1)$ est $(a+1, y_s + 1)$
  - Le père de $(a, y_s - 1)$ est $(a+1, y_s - 1)$
  - Le père de $(a+1, y_s + 1)$ est $(a+1, y_s)$
  - Le père de $(a+1, y_s - 1)$ est $(a+1, y_s)$
  - Le père de $(a+1, y_s)$ est (a, y_s)
  - Le père de (a, y_s) est $(a-1, y_s)$
  - Le père de $(a+1, y_s - 1)$ est $(a-2, y_s)$

Traitons le cas particulier du bloc r, composé de trois colonnes $\{a-1, a, a+1\}$ et dans lequel le nœud source s n'est pas sur la colonne centrale a :

- Si s se situe sur la colonne $a-1$ alors :
  - Le père de $(a, y_s + 1)$ est $(a+1, y_s + 1)$
  - Le père de $(a, y_s - 1)$ est $(a+1, y_s - 1)$
  - Le père de $(a+1, y_s + 1)$ est $(a+1, y_s)$
  - Le père de $(a+1, y_s - 1)$ est $(a+1, y_s)$
  - Le père de $(a+1, y_s)$ est (a, y_s)
  - Le père de (a, y_s) est s
- Si s se situe sur la colonne $a+1$ alors :
  - Le père de $(a, y_s + 1)$ est $(a-1, y_s + 1)$
  - Le père de $(a, y_s - 1)$ est $(a-1, y_s - 1)$
  - Le père de $(a-1, y_s + 1)$ est $(a-1, y_s)$
  - Le père de $(a-1, y_s - 1)$ est $(a-1, y_s)$
  - Le père de $(a-1, y_s)$ est (a, y_s)
  - Le père de (a, y_s) est s

[4] une relation entre i et j implique naturellement que ces deux sommets sont des transmetteurs.

Enfin définissons les relations pour les blocs de deux colonnes. S'il existe un bloc d'indice 1 de deux colonnes :

- Le père de $(1, i-1)$ est $(1, i) \forall i \in [1, y_s]$.
- Le père de $(1, i+1)$ est $(1, i) \forall i \in [y_s, q-1]$.
- Le père de $(1, y_s)$ est $(2, y_s)$
- Le père de $(2, y_s)$ est $(3, y_s)$

Symétriquement, s'il existe un bloc d'indice $K+1$ de deux colonnes :

- Le père de $(Q, i-1)$ est $(1, i) \forall i \in [1, y_s]$.
- Le père de $(Q, i+1)$ est $(1, i) \forall i \in [y_s, q-1]$.
- Le père de $(1, y_s)$ est $(2, y_s)$
- Le père de $(2, y_s)$ est $(3, y_s)$

L'ensemble de ces relations de parenté induit une arborescence solution. La figure 4.4 présente cette arborescence A_1 dans une grille dont le nombre de colonnes est $Q = 3k + 1$. Le découpage en blocs est indiqué par des intervalles au dessus de la grille. Notons que le sommet source (représenté en gris clair) ne se situe pas sur la colonne centrale du bloc auquel il appartient. Les colonnes 1 et Q sont composées de transmetteurs. Intuitivement, lorsqu'on « ajoute » une colonne de récepteurs de part et d'autre de la grille, le schéma de diffusion reste inchangé. Ceci donne une idée d'une solution lorsque $Q = 3k$ ou $Q = 3k + 2$, si s ne se situe pas sur une colonne centrale de bloc r.

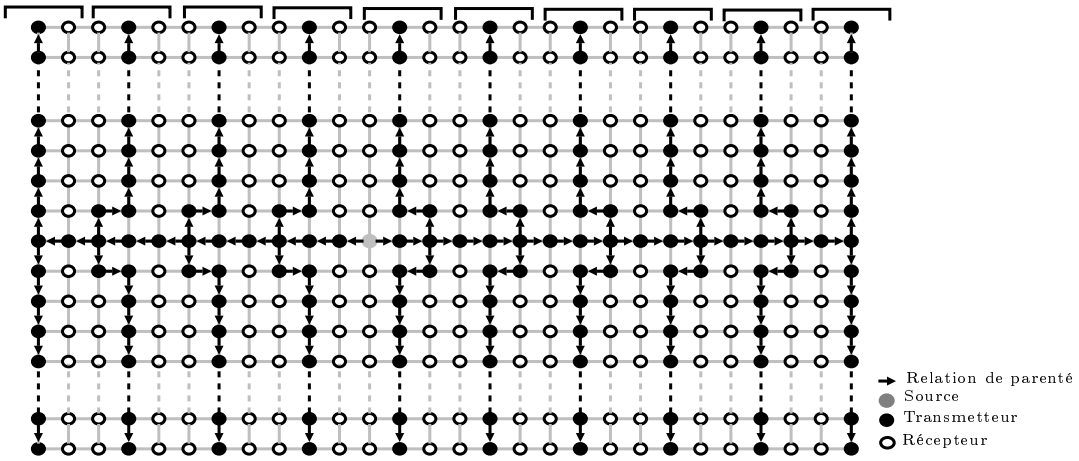


FIG. 4.4 – Une grille $G_{P \times Q}$, dans laquelle s ne se situe pas sur la colonne centrale d'un bloc

En gardant la même arborescence, et si $s = (x_s, y_s)$ est situé sur une colonne centrale, alors un problème survient au niveau des nœuds $(x_s - 1, y_s - 1)$ et $(x_s - 1, y_s + 1)$ (comme sur la figure 4.4) ou bien des nœuds $(x_s + 1, y_s - 1)$ et $(x_s + 1, y_s + 1)$. En effet les émissions des deux transmetteurs voisins peuvent survenir indépendamment - donc simultanément dans le pire des cas - empêchant alors ces nœuds de recevoir correctement le message. Pour corriger ce problème local, nous présentons une seconde arborescence A_2 pour le cas où s est situé sur un colonne centrale. Cette arborescence se déduit de la figure 4.5 [5].

[5] Nous nous abstenons d'énumérer les différentes relations de parenté lorsque s se situe sur une colonne centrale.

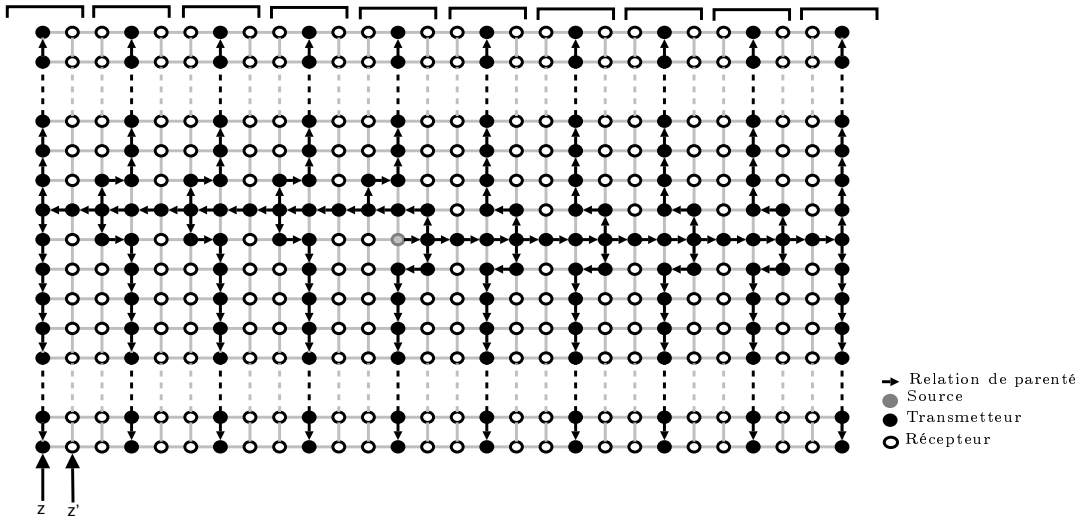


FIG. 4.5 – Une grille $G_{P\times Q}$, dans laquelle s se situe sur la colonne centrale d'un bloc

Remarque 6 *:*
La seconde arborescence possède des performances un peu moins bonnes que la précédente : en comparant les figures 4.4 et 4.5, la branche reliant s au sommet $(1,1)$ (le sommet z sur la figure) est un peu plus longue sur A_2. Il est parfois nécessaire de dégrader la qualité globale d'une solution simplement pour assurer la réception du message par une poignée de nœuds supplémentaires. L'expérience nous montre que cette remarque est souvent observable.

4.2.2 Évaluation de performances

Nous évaluons les performances de notre heuristique en temps et en nombre d'émissions par rapport aux bornes inférieures connues.

Théorème 16 *:*
L'heuristique précédemment présentée renvoie une stratégie optimale en temps à une constante additive de 5 près.

En terme d'émissions, cette même stratégie est ρ-approchée du nombre d'émissions minimum, avec :

$$\rho < 1 + \frac{6}{\max(P,Q) - 2}$$

Ces relations se déduisent logiquement de la figure 4.5 depuis la présentation du cas où s ne se situe pas sur une colonne centrale.

Preuve :

Soit A une arborescence décrivant une solution au problème [Diff,Sync,Tmp-EM, grille] sur une grille $G_{P\times Q}$. Une borne inférieure à la hauteur de A est l'excentricité du sommé source s moins un. Le sommet le plus éloigné de s est l'un des quatre sommets-coin de la grille. La hauteur optimale h^* est borné inférieurement par $\max(|x_s - x| + |y_s - y|) - 1$, pour $(x, y) \in \{(1, 1), (P, 1), (1, Q), (P, Q)\}$. Dans notre heuristique, la hauteur de l'arborescence est plus coûteuse lorsque s se situe sur une colonne centrale du bloc. Dans pareil cas, la hauteur de T est $\max(|x_s - x| + |y - s_y|) + 4$, pour $(x, y) \in \{(1, 1), (P, 1), (1, Q), (P, Q)\}$. Nous obtenons une hauteur optimale à un facteur additif de 5 près.

Une borne inférieure au nombre d'émissions est la cardinalité du plus petit ensemble connexe dominant de $G_{P\times Q}$, lui même borné inférieurement par

$$\frac{1}{3}(P \times Q - 2)$$

en accord avec la propriété 8. Le cas critique en terme d'émissions intervient pour $Q = 3k + 1$, et lorsque s se situe sur une colonne centrale de bloc. En pareille situation, le nombre de transmetteurs est de :

$$\frac{1}{3}(P \times Q + 4Q + 2P - 7)$$

Le calcul du rapport entre la solution trouvée et la borne inférieure permet de conclure.

□

Remarque 7 *:*
Le facteur d'approximation ρ tend très vite vers 1 pour des instances de grilles suffisamment grandes. Par ailleurs, la borne minorante utilisée n'est pas atteignable pour la majorité des instances. Ceci laisse suggérer que le ratio de performance est un peu plus faible que celui proposé par le calcul, renforçant ainsi la pertinence de notre heuristique.

4.2.3 Décentralisation de l'heuristique

Si l'on s'en réfère à l'heuristique précédente et pour un nœud donné i, le statut de transmetteur ou récepteur et la relation de parenté sont déterminés à partir :

- des coordonnées (x_i, y_i) de i dans la grille,
- des coordonnées (x_s, y_s) du sommet source s,
- du dernier nœud ayant relayé le message,
- du positionnement de s sur une colonne centrale de bloc ou non.

La dernière information dépend des dimensions $P \times Q$ de la grille $G_{P\times Q}$, et de la position de s dans cette dernière[6]. Ces deux informations permettent de retrouver immédiatement un découpage en blocs adéquat et le positionnement de s.

Supposons que chaque nœud connaît les dimensions de la grille ainsi que ses propres coordonnées. Considérons un message reçu par un nœud i sans interférence. Si le nœud source s et le nœud ayant effectué la transmission ont précisé dans l'entête du message leurs coordonnées, alors i détient

[6] de manière anecdotique, elle dépend également de la stratégie de découpages de blocs en lignes ou colonnes

l'ensemble des éléments nécessaires à l'heuristique précédente. En calculant localement l'heuristique décrite, un nœud peut donc déterminer par lui même et pour chaque réception s'il doit ou non relayer le message.

Remarque 8 :
Le choix du découpage de $G_{P\times Q}$ en lignes ou colonnes doit être identique pour tous les nœuds. Ceci implique un procédé de choix déterministe, par exemple :

1. *Valoriser le découpage pour lequel s ne se situe pas sur la colonne centrale.*
2. *En cas d'égalité, privilégier la dimension de plus grande taille.*
3. *En cas d'égalité, toujours découper par colonnes.*

Remarque 9 :
Les dimensions de la grille peuvent également être passées dans l'entête du message, de manière absolue ou relatives par rapport aux coordonnées de la source. Ceci permet de gérer l'ajout et le retrait de colonnes dans la grille, ou la diffusion aux nœuds présents dans une sous-grille G' de $G_{P\times Q}$.

4.3 De la grille vers le tore

4.3.1 Une migration possible ?

Nous reprenons dans cette section l'heuristique présentée sur la grille. Peut-on adapter cette heuristique dans un tore $T_{P\times Q}$, afin de fournir une stratégie cohérente pour [Diff,Sync,Tmp-EM„tore] ?

Le tore est une grille dont les nœuds qui sont situés sur les bords sont connectés aux extrémités correspondantes. Il peut se voir comme une grille $G_{P\times Q}$ repliée sur elle même, d'abord dans une dimension puis une autre. Ceci engendre de nouveaux liens, entre les sommets $(1, i)$ et $(P, i), \forall i \in Q$ d'une part, et entre les sommets $(i, 1)$ et $(i, Q), \forall i \in P$ d'autre part.

L'idée naturelle pour adapter l'heuristique précédente de la grille vers le tore consiste à utiliser une représentation en deux dimensions du tore, en positionnant par exemple le sommet source s au centre de la représentation. On définit alors les relations de parenté en se référant aux coordonnées inhérentes à la représentation $2D$. Cependant l'apparition de ces nouveaux liens entraîne de nouveaux conflits en termes d'émissions. Il en résulte que tous les sommets aux abords de la représentation $2D$ peuvent ne pas recevoir correctement le message. Naturellement, ces nœuds ne représentent qu'une petite proportion des sommets du tore, qui diminue lorsque les dimensions augmentent. Mais puisque nous imposons une réception correcte par tous les nœuds, la définition d'une heuristique dédiée au tore est nécessaire.

Dans le reste de la section, nous associons un tore avec sa représentation $2D$. Nous parlons, un peu abusivement, de lignes ou de colonnes pour désigner respectivement les sommets de mêmes abscisses ou ordonnées sur la représentation $2D$. Notons que la source s peut se situer à n'importe quel endroit de la représentation $2D$.

4.3.2 Une heuristique dédiée au tore

Tout comme son homologue sur la grille, l'heuristique sur le tore effectue un découpage de la représentation $2D$ du tore en blocs de colonnes, et calcule une solution S_1. Dans une seconde phase elle effectue un découpage en blocs de lignes, et calcule une solution S_2. Elle renvoie enfin la meilleure solution en fonction de la valeur à minimiser (temps ou nombre d'émissions). Les deux premières phases sont identiques, à la rotation de la représentation près. Aussi nous ne détaillons que la première phase relative au découpage en colonnes.

Découpage en blocs :

Soient $T_{P\times Q}$ un tore de dimensions $P \times Q$, et un nœud source de coordonnées $[x_s, y_s]$. Soit (x, y) le sommet de coordonnées $[x, y]$. Les lignes sont indicées de 1 à P, les colonnes de 1 à Q.

Nous découpons $T_{P\times Q}$ en blocs de trois, quatre, et cinq colonnes d'indices consécutifs. Chaque colonne appartient nécessairement à un et un seul bloc. Ce découpage dépend du nombre de colonnes Q. Les caractéristiques du tore nous laisse plus de liberté dans ce découpage. Nous pouvons donc effectuer un découpage, et placer la source s par la suite :

- Si $Q = 3k$, le tore est découpée en k blocs de trois colonnes chacun, de sorte que le sommet source s soit situé sur la colonne centrale d'un bloc.
- Si $Q = 3k + 1$, le tore est découpé en $(k - 1)$ blocs de trois colonnes, et en 1 bloc de quatre colonnes, de sorte que s soit situé sur l'une des deux colonnes centrales du bloc de 4 colonnes.
- Si $Q = 3k + 2$, le tore est découpée en $k - 1$ blocs de trois colonnes, et en 1 bloc de cinq colonnes. Le sommet source s se situe sur l'une des deux colonnes adjacentes à la colonne centrale du bloc de cinq colonnes.

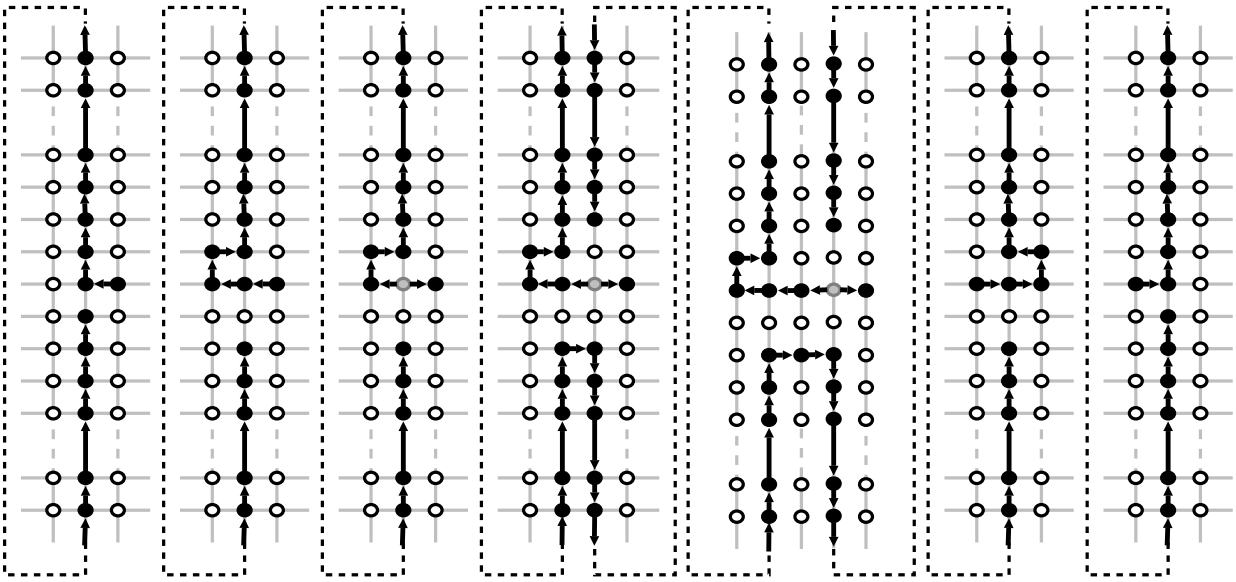
Les blocs sont indicés de 1 à $k = \lfloor Q/3 \rfloor$ de sorte que :
- le bloc contenant la source s possède l'indice $R = \lfloor k/2 \rfloor$,
- deux blocs adjacents possédent des numéros consécutifs,
- les blocs 1 et k sont adjacents.

Définition des relations de parenté :

Nous définissons les relations de parenté entre sommets d'un même bloc. Ces relations sont plus explicites lorsqu'elles apparaissent sur un schéma. Nous proposons 7 schémas possibles présentés en figure 4.6, et nous abstenons de les énumérer de manière formelle.

Nous appliquons les relations de parenté sur les blocs en fonction de leur indice et de leur taille, suivant la règle suivante :
Pour un bloc d'indice $i \neq R$:
- Si $i = 1$, nous appliquons les relations décrites par le schéma 4.6(a).
- Si $1 < i < R$ nous appliquons les relations décrites par le schéma 4.6(b).
- Si $R < i < k$: voir figure 4.6(f)
- Si $i = k$: voir figure 4.6(g)

Sur le bloc R :
- Si R contient 3 colonnes : voir figure 4.6(c).
- Si R contient 4 colonnes : voir figure 4.6(d).
- Si R contient 5 colonnes : voir figure 4.6(e).



(a) schéma 1 (b) schéma 2 (c) schéma 3 (d) schéma 4 (e) schéma 5 (f) schéma 6 (g) schéma 7

FIG. 4.6 – Huit schémas représentant les relations de parenté pour un bloc

Enfin nous définissons les relations entre deux transmetteurs a et b de 2 blocs adjacents d'indices i et j, avec $i < j$:

- Si $R < i$, alors a est le père de b.
- Si $j < R$, alors b est le père de a.

La figure 4.7 présente une représentation $2D$ d'un tore $T_{\times}$. Sur cette dernière apparaissent le découpage en blocs (au dessus) et les relations de parenté définies par notre heuristique.

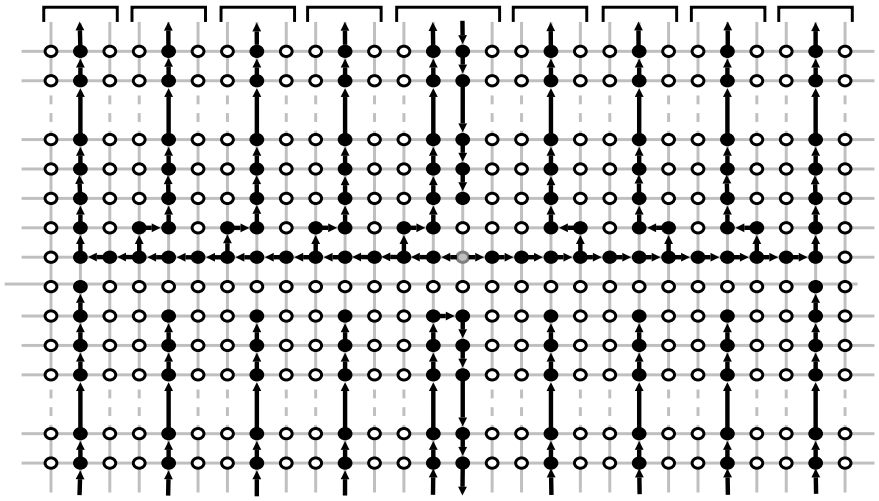


FIG. 4.7 – Schéma de diffusion en arborescence sur la représentation $2D$ d'un tore

4.3.3 Évaluation de performances

Nous évaluons dans cette sous-section les performances de l'heuristique sur le tore.

Remarque 10 *:*
Intuitivement, les performances de cette heuristique en temps sont moins bonnes que celle présentée dans la grille :

- *Les conflits apparus en repliant la grille horizontalement ont été résolu dans le bloc source en définissant une longue chaîne d'émissions. Cette chaîne est présente pour $Q = 3k$ et $Q = 3k+1$ est apparaît clairement dans les figures 4.6(d) et 4.6(e) par une chaîne faisant un aller-retour. La longueur de cette chaîne va affecter la hauteur de l'arborescence résultat A. Aussi définir cette chaîne dans le bloc R permet de minimiser son impact sur le coût en temps de la solution.*
- *Les conflits apparus en repliant la grille verticalement sont résolus dans chaque bloc i : les transmissions de la colonne centrale forment alors une grande chaîne d'émissions, affectant également le coût en temps de la solution.*

Le théorème suivant annonce les coûts de notre heuristique :

Théorème 17 *:*
L'heuristique sur le tore renvoie une solution au problème [Diff,Sync,Tmp-EM„tore]-minTemps avec un rapport d'approximation d'au plus 2*.*

Elle renvoie une solution ρ-approchée au problème [Diff,Sync,Tmp-EM„tore]-minémissions avec :

$$\rho < 1 + \frac{4}{\min(P,Q) - 2}$$

Preuve :
En accord avec la propriété 7, la borne inférieure à la hauteur d'une arborescence cohérente pour [Diff,Sync,Tmp-EM„tore] est l'excentricité de la source, c'est-à-dire :

$$\lfloor \frac{P}{2} \rfloor + \lfloor \frac{Q}{2} \rfloor$$

Le cas critique de l'heuristique intervient lorsque la dimension dans laquelle le découpage en blocs a été défini est de la forme $3k+2$. En supposant que $P = 3k+2$ et $Q = 3k'+2$, nous obtenons une solution dont le coût est :

$$\min\left(\max(\lceil \frac{P}{2} \rceil + Q, 2Q), \max(\lceil \frac{Q}{2} \rceil + P, 2P)\right)$$

D'après la propriété 8, une borne inférieure du nombre d'émissions est :

$$\frac{1}{3}(P \times Q - 2)$$

Le cas critique en terme d'émissions intervient pour une dimension de tore $Q = 3k+1$: le bloc R contient alors autant de transmetteurs que de récepteurs. En pareille situation, le nombre total de transmetteurs est de :

$$\frac{1}{3}(P \times Q + 2Q + 2P - 14)$$

□

Remarque 11 *:*
Le résultat en nombre d'émissions est légèrement meilleur que sur la grille. Était-ce prévisible ? A priori oui, car tous les sommets ont ici un degré 4*. En comparaison, les sommets en bord de grille n'ont que* 3 *voire* 2 *voisins ce qui est plus contraignant.*

Remarque 12 *:*
Le rapport d'approximation ρ *au problème [Diff,Sync,Tmp-EM„tore]-minémissions tend très vite vers* 1 *pour des tores de dimensions suffisamment grandes. Par ailleurs, la borne minorante utilisée n'est pas atteignable pour la majorité des instances. Ceci laisse suggérer que le ratio de performance est un peu plus faible que celui proposé par le calcul, renforçant ainsi la pertinence de notre heuristique.*

Notons enfin que la décentralisation de l'heuristique reste possible, par affectation de coordonnées aux nœuds selon une représentation $2D$. Le principe est alors identique à celui employé dans la grille.

Chapitre 5

Outils de résolution exacte

L'étude théorique menée dans le chapitre 3 permet d'affirmer ceci : toute stratégie cohérente pour [Diff,Sync,Arbo$_{\infty}$] possède des coûts en nombre d'étapes ou d'émissions supérieurs ou égaux à une stratégie optimale cohérente pour [Diff,Sync,Tmp-EM$_{\infty}$]. En d'autres termes dans un modèle synchrone, on ne peut pas espérer une meilleure solution en préférant notre stratégie en arborescence plutôt que les stratégies d'ordonnancement classiques. Mais la différence de coûts est-elle significative en moyenne ? Nous proposons dans ce chapitre des expérimentations destinées à répondre à cette question. La première section décrit une formulation linéaire en nombre entiers destinée à produire des solutions selon différentes stratégies. La seconde section détaille le protocole expérimental que nous avons mis en œuvre et les résultats que nous avons obtenus.

5.1 Formulation linéaire en nombres entiers

Nous définissons dans cette section une stratégie optimale pour résoudre le problème de la diffusion dans un modèle de communication synchrone. Nous proposons des formulations linéaires en nombres entier pour les problèmes [Diff,Sync,Tmp-EM$_{\infty}$]-minTemps, [Diff,Sync,Tmp$_{\infty}$]-minTemps et [Diff,Sync,Arbo$_{\infty}$]-minTemps. L'intérêt de ces formulations réside dans le fait qu'elles permettent de minimiser lexicographiquement le temps de diffusion, puis le nombre d'émissions. Les solutions renvoyées sont donc optimales en temps, et utilisent un nombre minimal d'émissions.

La donnée du programme linéaire en nombres entiers (PLNE) est composée d'un graphe $G = (V, E)$ supposé connexe et d'un nœud source s.

Le nombre d'étapes et d'émissions sont minimisés comme suit :

- Une variable T représente le nombre d'étapes maximum autorisé. Le PLNE est défini selon cette variable.
- La variable T est initialisé avec l'excentricité de s dans G lors de la première exécution du PLNE. Tant que ce dernier renvoie que l'instance donnée n'a pas de solution, nous incrémentons T et relançons l'exécution.
- Dès qu'une solution réalisable est trouvée sur l'instance, on en déduit que T a atteint la valeur minimum du nombre d'étapes. La fonction objectif consiste alors à minimiser le nombre d'émissions utilisées. Au terme de son exécution, le programme renvoie une solution minimum en nombre d'étapes, et minimale en nombre d'émissions.

5.1.1 Stratégie cohérente pour [Diff,Sync,Tmp-EM„]

Nous proposons une formulation linéaire mixte qui renvoie, lorsqu'elle existe, une stratégie cohérente pour le contexte [Diff,Sync,Tmp-EM„] sous forme d'un ordonnancement d'émissions. Les critères de temps et de transmissions sont minimisés comme expliqué ci-avant.

Soient $G = (V, E)$, $s \in V$, et $T \in \mathbb{N}^*$ la donnée de notre programme linéaire.

Nous utilisons deux types de variables :

1. $emet[i,t] \in \{0,1\}, \forall i \in V, t \in [1,T]$
2. $recoitBien[i,t] \in \{0,1\}, \forall i \in V, t \in [1,T]$

Les variables entières $emet[i,t]$ prennent la valeur 1 si et seulement si le nœud i émet à l'étape t, et 0 autrement. De même chaque variable $recoitBien[i,t]$ est affectée à 1 si et seulement si le nœud i reçoit une transmission sans interférence à l'étape t, et 0 autrement. La formulation de nos contraintes nous autorise à relaxer le problème, c'est-à-dire définir les variables $recoitBien[i,t]$ de manière continue sur $[0,1]$, et non plus binaires.

La minimisation du temps de diffusion est assurée en faisant varier le paramètre d'entrée T. Aussi l'objectif de la fonction de minimisation du programme est la minimisation du nombre d'émissions :

$$\min \sum_{i \in V, t \in [1,T]} emet[i,t]$$

Cet objectif est sujet aux contraintes suivantes :

1. L'inéquation 5.1 impose que chaque nœud (hormis la source) doit recevoir au moins une fois le message.

$$\sum_{t \in [1,T]} recoitBien[i,t] \geq 1 \quad \forall i \in V - \{s\} \tag{5.1}$$

2. Les deux inéquations suivantes lient la réception d'un nœud à l'émission de ses voisins. Un nœud ne peut rien recevoir à l'étape t si aucun de ses voisins n'émet (inéquation 5.2), ou si deux de ses voisins (ou plus) émettent simultanément (inéquation 5.3).

$$recoitBien[i,t] \leq \sum_{j \in N_G(i)} emet[j,t] \quad \forall i \in V, t \in [1,T] \tag{5.2}$$

$$recoitBien[i,t] \leq 2 - emet[j,t] - emet[k,t] \quad \forall i \in V, t \in [1,T], j,k \in N(i) \tag{5.3}$$

3. Un nœud autre que la source ne peut émettre qu'à la condition qu'il ait reçu le message à une étape précédente, comme l'atteste l'inéquation 5.4 :

$$emet[i,t] \leq \sum_{u \in [1,t-1]} recoitBien[i,u] \quad \forall i \in (V) - \{s\}, t \in [1,T] \tag{5.4}$$

4. Enfin le nœud source est le seul à émettre à l'étape 1 :

$$emet[s,1] = 1 \tag{5.5}$$

Des opérations de filtrage sont réalisées pour optimiser le temps d'exécution de ce programme. Ainsi pour un nœud i situé à une distance d de la source, nous pouvons déjà attribuer la valeur 0 aux variables $emet[i,t]$, et $recoitBien[i,t-1]$, $\forall t \leq d$. De même seuls les nœuds adjacents à la source reçoivent bien le message à l'étape 1. Enfin un nœud qui est un point d'articulation émet obligatoirement au moins une fois dans toute solution, ce qui nous conduit à définir une nouvelle inéquation satisfaisant cette condition.

5.1.2 Stratégie cohérente pour [Diff,Sync,Tmp„]

La formulation présentée précédemment se modifie très simplement si l'on souhaite définir des stratégies cohérentes dans d'autres contextes. Si l'on souhaite que l'ordonnancement résultat décrive une stratégie cohérente pour [Diff,Sync,Tmp„], il nous suffit d'ajouter l'inéquation 5.6. Cette inéquation contraint chaque nœud à n'émettre qu'au plus une fois dans la solution obtenue.

$$\sum_{t\in[1,T]} emet[i,t] \leq 1 \quad \forall i \in V \tag{5.6}$$

5.1.3 Stratégie cohérente pour [Diff,Sync„Arbo,]

Une stratégie cohérente pour [Diff,Sync,Tmp„] l'est également pour [Diff,Sync„Arbo,] si et seulement si toute émission d'un nœud i quelconque à l'étape t est précédée d'une réception correcte à l'étape $t-1$. Nous formulons cela en reprenant la formulation conduisant à l'obtention d'une stratégie valide pour [Diff,Sync,Tmp„], et en remplaçant l'inéquation 5.4 par l'inéquation suivante :

$$emet[i,t] \leq recoitBien[i,t-1] \quad \forall i \in (V) - \{s\}, t \in [2,T] \tag{5.7}$$

5.2 Simulations et résultats

5.2.1 Mode opératoire et génération de graphes aléatoires

Nous avons choisi d'effectuer nos simulations sur des graphes de disques unitaires. Cette classe de graphes a été retenue car elle modélise efficacement les réseaux radio dans lesquels les émissions sont omnidirectionnelles, ce qui est le cas des réseaux ad-hoc ou des réseaux de capteurs.

Nous générons ces graphes à l'aide de deux paramètres que sont :

1. n le nombre de sommets du graphe,
2. r le rayon d'émission.

Nous tirons pour chacun des n sommets deux coordonnées correspondant à la position du sommet dans un plan euclidien de dimensions $M \times M$. Deux sommets sont voisins si leur distance est inférieure ou égale à r. Si le graphe résultat n'est pas connexe, nous augmentons progressivement r jusqu'à ce qu'il le devienne sans remettre en cause les coordonnées précédemment tirées. En pareil cas les graphes obtenus ont une densité généralement faible, et ont une plus grande probabilité de contenir le graphe X_{84} comme sous-graphe induit. La présence d'un tel motif n'est généralement pas à l'avantage de notre stratégie en arborescence.

Le sommet source s est choisi au hasard dans le graphe résultat.

Pour chaque graphe, nous calculons le coût optimal en nombre d'étapes puis d'émissions de solutions cohérentes pour [Diff,Sync,Tmp-EM„], [Diff,Sync,Tmp„] et [Diff,Sync„Arbo,], et reportons

à titre indicatif le temps nécessaire au calcul de cette solution. Nous générons un millier de graphes connexes entre 5 et 70 sommets, et de densités diverses.

5.2.2 Résultats et analyse

Les simulations ont été réalisées sur une station de travail équipée d'un processeur Intel Core 2 Duo 2.0 Ghz. Nous avons utilisé la librairie GLPK[1] pour implémenter nos programmes. Le tableau 5.1 présente une partie significative des résultats obtenus. Pour chaque instance et chaque stratégie, nous indiquons le nombre d'étapes minimum et le nombre minimal d'émissions (colonnes ***étapes*** et ***émissions***), ainsi que le temps en secondes pour trouver la solution (colonne ***temps***).

Excentricité et nombre d'étapes

La remarque suivante découle d'une analyse théorique :

Remarque 13 *:*
Il existe des instances de graphes de disques unitaires pour lesquelles le coût optimal d'une solution à [Diff,Sync„Arbo,]-minTemps peut être éloigné à un facteur multiplicatif près du coût optimal de [Diff,Sync,Tmp„]-minTemps.

Par exemple, si l'on considère l'instance de la figure 5.1 :

- *toute stratégie de diffusion cohérente pour [Diff,Sync„Arbo,] requiert au moins* 16 *étapes : en effet, chacun des sommets formant des cycles de taille* 4 *doit être transmetteur, et aucun ne peut émettre à la même étape (voir pour rappel la remarque 3).*
- *En revanche, on peut trouver une stratégie de diffusion cohérente pour [Diff,Sync,Tmp„] en* 12 *étapes, puisque dans chaque cycle de taille* 4 *il n'est pas nécessaire de procéder à l'émission des* 4 *sommets (on temporise pour éviter un conflit).*

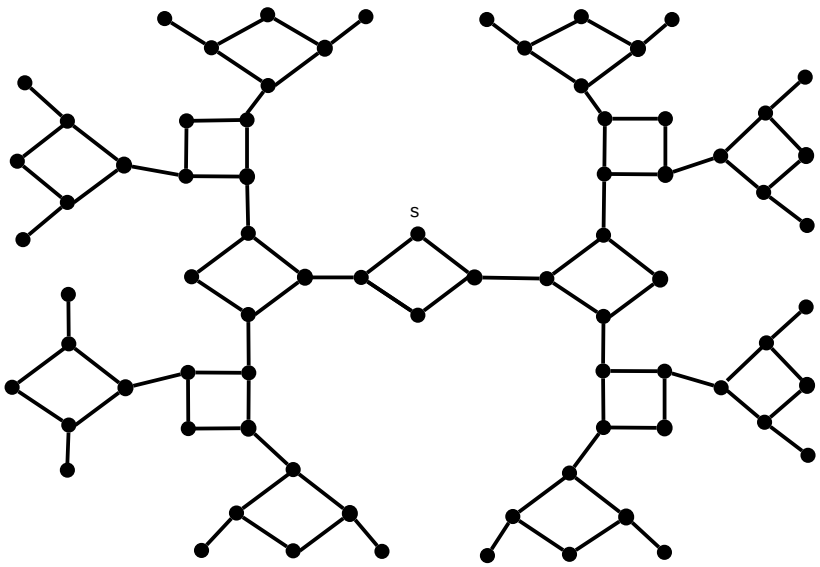


FIG. 5.1 – Un graphe de disques unitaires, et une source s.

[1] Gnu Linear Programming Tool

Graphe			*[Diff,Sync,Tmp-EM,,]*			*[Diff,Sync,Tmp,,]*			*[Diff,Sync,,Arbo,]*		
|V|	|E |	excentricité(s)	étapes	émissions	temps	étapes	émissions	temps	étapes	émissions	temps
5	7	2	2	2	0	2	2	0	2	2	0
5	5	3	3	3	0	3	3	0	3	3	0
10	12	4	4	6	0	4	6	0	4	6	1
10	25	3	3	3	0	3	3	0	3	3	0
15	51	3	4	4	0	4	4	0	4	4	0
15	30	6	6	7	0	6	7	0	6	7	0
20	34	6	6	7	0	6	7	0	6	7	0
20	85	3	4	4	1	4	4	0	4	4	0
25	62	7	7	10	0	7	10	0	7	10	0
25	48	11	11	12	1	11	12	0	11	12	0
30	74	9	9	11	0	9	11	1	9	11	0
30	106	4	4	5	0	4	5	0	4	5	0
35	90	6	6	12	1	6	12	0	6	12	1
35	147	5	5	8	0	5	8	1	5	8	0
40	136	7	7	12	14	7	12	16	7	12	12
40	375	2	3	4	2	3	4	0	3	4	1
45	134	10	10	16	2	10	16	3	10	16	3
45	175	7	7	12	1	7	12	2	7	12	1
50	195	6	6	12	1	6	12	1	7	8	18
50	86	17	17	25	29	17	25	27	17	25	10
55	381	4	5	8	15	5	8	9	5	8	4
55	151	11	11	17	5	11	17	6	11	18	101
60	261	7	7	12	261	7	12	102	7	12	242
60	252	8	8	12	6	8	12	6	8	12	3
65	348	6	6	12	774	6	12	>1200	6	12	545
65	312	6	6	12	24	6	12	29	6	12	32
70	395	6	6	13	146	6	13	196	6	13	301
70	312	6	6	16	213	6	16	149	6	16	108
75	352	6	7	16	>1200	7	16	>1200	7	16	722
75	245	7	7	21	19	7	21	29	7	21	27
80	229	8	8	27	49	8	27	21	8	27	18
80	213	12	12	29	>1200	12	29	>1200	12	30	1026

TAB. 5.1 – Résultats expérimentaux de trois stratégies synchrones

En généralisant la construction dont l'idée découle de la figure 5.1, nous pouvons proposer des instance composées de graphes de disques unitaires dans lesquelles l'optimum de [Diff,Sync,,Arbo,]-minTemps est au moins égal à 4/3 fois l'optimum de [Diff,Sync,Tmp,,]-minTemps.

Or les résultats des simulations en terme de nombre d'étapes, synthétisés dans le tableau 5.2, montrent qu'au moins 93% des solutions trouvées utilisent autant d'étapes que l'excentricité de la source, quelle que soit la stratégie employée. Les solutions restantes utilisent seulement une étape de plus que l'excentricité.

	Nombre d'étapes utilisées		
Modèle	excentricité	excentricité + 1	$\geq$ excentricité + 2
[Diff,Sync,Tmp-EM,,]	93.1%	6.9%	0.0%
[Diff,Sync,Tmp,,]	93.1%	6.9%	0.0%
[Diff,Sync,,Arbo,]	93.1%	7.0%	0.0%

TAB. 5.2 – Relation entre le nombre d'étapes et l'excentricité de la source, selon les modèles

Sur le millier d'instances générées, les optimums en nombre d'étapes selon les stratégies diffèrent sur seulement 2 instances : sur chacune d'entre elles, le nombre d'étapes des stratégie cohérentes pour [Diff,Sync,Tmp-EM,,] et [Diff,Sync,Tmp,,] est égal à l'excentricité, et à l'excentricité plus un pour [Diff,Sync,,Arbo,].

- En moyenne et sur les graphes de disques unitaires aléatoires, des stratégies cohérentes pour [Diff,Sync,Tmp-EM,,], [Diff,Sync,Tmp,,] ou [Diff,Sync,,Arbo,] ont des optimaux en nombre d'étapes globalement identiques.
- Ces optimaux sont égaux à l'excentricité de la source pour plus de 90% des cas.

En examinant les instances où le nombre d'étapes requis est strictement supérieur à l'excentricité, on observe qu'en général la perte d'une étape est souvent liée à la présence d'une configuration difficile à franchir (par exemple le sous-graphe X_{84}) aux abords du nœud source, ou vers les sommets les plus éloignés de la source.

Nombre d'émissions selon les stratégies

Sur l'ensemble des simulations, aucune stratégie cohérente pour [Diff,Sync,Tmp-EM,,] n'a utilisé moins d'émissions qu'une stratégie cohérente pour [Diff,Sync,Tmp,,] sur la même instance. Cette constatation nous emmène à penser qu'en pratique, la capacité d'un nœud à pouvoir émettre plusieurs fois un même message n'est pas pertinente, et qu'une solution à coût équivalent peut être trouvée avec au plus une seule émission par nœud.

De plus, le nombre d'émissions optimal d'une stratégie cohérente pour [Diff,Sync,,Arbo,] est égal à l'optimal d'une stratégie cohérente pour [Diff,Sync,Tmp-EM,,] sur 97.1% des instances, et à l'optimal plus un pour les autres. Nous avons seulement identifié une instance sur laquelle le nombre d'émissions optimal pour [Diff,Sync,,Arbo,] était exactement celui de [Diff,Sync,Tmp-EM,,] plus deux. Ceci nous conforte dans l'idée que toute stratégie cohérente pour [Diff,Sync,,Arbo,] offre des performances équiavalentes aux autres stratégies, bien que théoriquement moins bonnes.

Par ailleurs, sur les deux instances pour lesquelles la stratégie cohérence pour [Diff,Sync„Arbo,] requiert une étape de plus que celle cohérente pour [Diff,Sync,Tmp„], le nombre d'émissions optimal est inférieur de deux. L'augmentation du nombre d'étapes permet ici de diminuer le nombre d'émissions nécessaires.

Temps de calcul

L'objectif de ces simulations consistait essentiellement à montrer que les stratégies basées sur le modèle [Diff,Sync„Arbo,] offrent des performances globalement équivalentes aux autres stratégies synchrones. Nous ne cherchons pas à résoudre ici le plus rapidement possible des instances de grande taille. Néanmoins, nous terminons ce chapitre en soulevant une interrogation liée aux temps de calcul, et qui relève plutôt du domaine de la recherche opérationnelle.

Il est clair que l'implémentation de nos programmes permet de résoudre en quelques secondes des instances de l'ordre de 50 sommets ou moins. Lorsque les instances deviennent de plus en plus grandes, le temps nécessaire à la résolution d'une instance devient de plus en plus aléatoire, et peut varier de quelques secondes à quelques heures pour deux instances pourtant très proches. Aucun des facteurs que nous avons étudiés (nombre de sommets, densité, nombre de variables et de contraintes) ne sont à l'origine de telles variations. Nous citons à titre d'exemple l'expérimentation suivante :

1. À partir de mêmes coordonnées de sommets, nous avons créé plusieurs graphes en faisant varier seulement le rayon d'émission lors du processus de génération.
2. Nous avons sélectionné un modèle, et noté le temps d'exécution du solveur pour calculer la stratégie optimale en temps puis en émissions sur chacun des graphes générés.
3. Nous avons dressé la courbe du temps de calcul en fonction du rayon d'émission.

Il apparaît que cette courbe n'est pas régulière, et qu'une augmentation même légère du rayon d'émission a une variation imprévisible sur le temps de calcul de la stratégie optimale. Nous pensons que la variation du temps de calcul est certainement liée à la topologie du graphe, et au comportement interne du solveur qui, dans son processus de résolution, arrive à effectuer souvent de bonnes affectations de variables mais pas systématiquement.

Notons enfin qu'en général et pour une instance donnée, nous trouvons plus rapidement la stratégie optimale cohérente pour [Diff,Sync„Arbo,] que les autres. Ceci est probablement dû au fait que l'espace des solutions est plus restreint, et engendre plus de coupes dans le processus de recherche exhaustive des solutions. Le phénomène est très présent en recherche opérationnelle : une solution de coût optimal peut être déterminée rapidement (par rapport à la durée totale d'exécution du solveur), mais prouver l'optimalité de cette dernière nécessite un parcours exhaustif de toutes les solutions très coûteux en temps. Curieusement, cette remarque ne s'applique pas pour [Diff,Sync,Tmp„]. Le temps de recherche de la stratégie cohérente pour [Diff,Sync,Tmp„] est globalement plus long que pour [Diff,Sync,Tmp-EM„].

Conclusion

Nous avons étudié le long de cette partie le problème de la diffusion depuis une source unique s dans un réseau radio multi-sauts, sous diverses approches et contraintes. Les caractéristiques les plus représentatives des réseaux que nous avons considéré sont :

- des émissions omnidirectionnelles,
- une même fréquence d'émission pour chaque nœud,
- l'absence de couche MAC idéale, ce qui prévient la réception simultanée de deux transmissions voisines, et l'incapacité à émettre et recevoir simultanément.

Le chapitre 2 a permis d'établir une classification des différentes approches et travaux réalisés. Aux approches traditionnelles proposant un ordonnancement d'émissions, nous avons opposés une nouvelle stratégie de diffusion, dite en arborescence, dont un des intérêt est l'application dans un modèle asynchrone.

L'étude de menée dans le chapitre 3 a montré l'existence systématique d'une stratégie de diffusion en arborescence. Malheureusement, les problèmes qui consistent à trouver une arborescence minimisant les coûts en temps de diffusion ou en nombre d'émissions ont été montré NP-difficiles dans les cas synchrones ou asynchrones. L'étude de l'impact de la topologie du réseau sur ces problèmes nous a permis d'établir que lorsque le graphe du réseau est triangulé, le nombre d'émissions minimum d'une stratégie de diffusion en arborescence est exactement la taille d'un ensemble dominant connexe. Ce résultat aboutit à un algorithme polynomial minimisant le nombre d'émissions dans les graphes d'intervalles. Nous avons par ailleurs proposé un algorithme approximant le temps de diffusion par Δ_G lorsque le graphe est triangulé.

La première section du chapitre 4 reprend les stratégies d'ordonnancement propres aux modèles synchrones sur les graphes dont l'excentricité de la source est 2. L'introduction d'un nouvel outil, la mv-décomposition, permet de définir une stratégie de diffusion en $O(\ln^2 n)$. Le résultat était déja annoncé dans [CW91], mais l'algorithme que nous proposons a une meilleure complexité de l'ordre de $O(m \times O(\ln^2 n))$. Nous annonçons également une stratégie minimum en nombre d'étapes et d'émissions sur les graphes convexes circulaires. Les seconde et troisième sections du chapitre sont dédiées à la définition d'heuristiques sur la grille et le tore. Le principal résultat est la mise en place d'une stratégie de diffusion en arborescence dont les performances sont indiquées dans le tableau 5.3 .

Enfin, les méthodes de résolutions exactes présentées dans le chapitre 5 nous ont permis d'établir des observations intéressantes entre les trois stratégies de diffusion synchrones en ordonnancement, en ordonnancement et émission unique par nœud, et en arborescence : expérimentalement, les trois stratégies renvoient des solutions optimales de coûts équivalents en nombre d'étapes et nombre

Topologie	***Temps de diffusion***	***Nombre d'émissions***
Grille $[P \times Q]$	OPT + 5	OPT+ $\frac{6}{\max(P,Q)-2}$
Tore $[P \times Q]$	$2 \times OPT$	OPT+ $\frac{4}{\max(P,Q)-2}$

TAB. 5.3 – Performance des heuristiques proposées sur la grille et le tore

d'émissions sur les graphes de disques unitaires. Ceci justifie l'intérêt porté aux stratégies de diffusion en arborescence, que l'on pouvait penser moins pertinentes au terme de l'étude théorique.

Le principal apport dont peut bénéficier ce travail est la mise en place d'heuristiques de diffusion pertinentes pour les stratégies en arborescences sur des topologies générales.

Par ailleurs les résultats expérimentaux observés nous incitent à nous demander quel est l'écart entre les performances théoriques d'une stratégie de diffusion et ses performances réelles. Nous avons par exemple proposé une stratégie de diffusion en $O(m.\log N)$ étapes sur des graphes pour lesquels l'excentricité de la source est 2. Nous connaissons l'existence de graphes sur lesquels cette borne est atteinte. Mais qu'en est-il en général sur des instances quelconques ? Obtient-t-on des résultats proches de cette borne ou au contraire bien meilleurs ? Et en généralisant cette approches sur des graphes de diamètre D ? L'étude de ces questions sur les algorithmes de diffusion existants permettrait de vérifier si l'analyse théorique est un bon indicateur de performances, ou si la comparaison de deux algorithmes doit obligatoirement passer par l'expérimentation.

Deuxième partie

Satisfaction de requêtes de communication dans un réseau radio multi-sauts synchrone

Introduction

Nous étudions deux familles de problèmes algorithmiques inspirés des contraintes de routage rencontrées dans des réseaux sans fil multi-sauts synchrones. Ces problèmes consistent à satisfaire des requêtes de communication dans un environnement où deux nœuds ne sont pas nécessairement à portée directe d'émission et où les émissions trop proches génèrent des zones de brouillage (pas de couche MAC idéale).

Pour satisfaire une requête il est possible de lui affecter une route que devra suivre le message de la requête dans le réseau. Pour éviter les brouillages nous pouvons envisager de temporiser l'émission de certains nœuds. Notre objectif est de minimiser le temps nécessaire pour satisfaire toutes les requêtes. Les deux familles de problèmes diffèrent en ce que les routes sont imposées dans l'une, et à déterminer dans l'autre. Ces problèmes de routage constituent une suite naturelle au problème de diffusion abordé dans la première partie de cette thèse.

Nous proposons dans cette partie une étude de complexité de ces problèmes, et nous mettons en évidence l'impact de certains paramètres sur la difficulté de ces problèmes.

Organisation de cette partie

Cette partie est organisée de la manière suivante :

Le chapitre 6 introduit formellement les deux familles de problèmes, respectivement nommées DAWN-paths et DAWN-requests. Nous y présentons les sous-problèmes constituant chaque famille.

Nous débutons dans le chapitre 7 une étude de complexité de l'ensemble des problèmes définis au chapitre 6. La complexité des problèmes de DAWN-paths et DAWN-requests est envisagée en général, selon la distance maximum séparant chaque source d'une requête de sa destination, dans des topologies restreintes et selon la valeur de la date maximum que l'on peut allouer. Nous dressons pour ce dernier paramètre la frontière entre la polynomialité et la NP-complétude de DAWN-paths et DAWN-requests.

Le chapitre 8 conclut par l'existence d'algorithmes de résolution exacte en temps polynomial lorsque le nombre de requêtes d'une instance est borné.

Chapitre 6

Le problème de satisfaction de requêtes de communication

La première section de ce chapitre définit la notion de requêtes de communication, et introduit les conditions d'exécution dans lesquelles nous considérons le problème de satisfaction de requêtes de communication. Le formalisme proposé dans la seconde section permet la définition des problèmes DAWN-paths et DAWN-requests dans la section suivante. Nous concluons dans une quatrième section par la définition de ces problèmes dans un modèle dit « online », qui revient à considérer ces problèmes sur des réseaux dynamiques.

6.1 Introduction au problème de satisfaction de requêtes

Nous avons introduit dans la première partie de ce mémoire les principales caractéristiques liées à l'utilisation d'un média radio, et avons étudié le problème de la diffusion dans différents modèles. Nous proposons l'étude de deux nouveaux problèmes algorithmiques inspirés du fonctionnement des réseaux radio, et liés à la ***satisfaction de requêtes de communication***.

Une ***requête*** entre deux nœuds est la demande de ressources (slot de connexion, bande passante, donnée, ...) d'un nœud à l'autre. Dans le cadre d'une requête de communication d'un nœud a vers un nœub b, il convient de délivrer une information depuis le nœud a à destination du nœud b. On qualifie la requête de ***satisfaite*** si le message a été acheminé correctement de a vers b. De nombreux problèmes sont inhérents à la satisfaction d'un ensemble de requêtes de communications, par exemple :

- Des problèmes de gestion de trafic : l'émission d'une requête consomme les ressources d'un réseau, principalement de la bande passante et du temps de calcul des nœuds routeurs. Le nombre de requêtes pouvant être traitées simultanément est donc limité par les ressources de l'infrastructure. Afin d'éviter des phénomènes d'engorgement, qui peuvent dégrader les performances du réseau, il est souvent nécessaire de retarder l'émission de requêtes, ou d'en contrôler au moins le débit. Un problème parfois évoqué consiste à trouver des stratégies permettant de satisfaire un nombre maximum de requêtes simultanément, et a été notamment étudié dans les réseaux WDM (Wavelength Division Multiplexing) [CR02].
- Des problèmes de routage d'information : un nœud ne sait pas nécessairement par quelle route atteindre le destinataire d'une requête.

Nous étudions le problème de satisfaction de requêtes exclusivement dans le modèle [Requêtes, Sync, Tmp„] dont nous rappelons les caractéristiques principales :

- Les nœuds ne disposent pas de couche MAC idéale.
- Les nœuds ont un modèle d'antennes omnidirectionnelles sans limitation de puissance (le rayon d'émission ne peut varier au cours du temps).

Les communications sont sujettes aux contraintes suivantes :

- Δ-port émission : les émissions sont omnidirectionnelles. Pour une transmission donnée, tous les nœuds voisins de l'émetteur recoivent le message.
- 1-port réception : un nœud ne peut recevoir qu'un seul message à la fois. La réception simultanée de deux ou plusieurs messages peut entraîner des interférences et une altération des messages envoyés.
- Half-duplex : un nœud du réseau ne peut pas simultanément émettre un message et en recevoir un autre. Cependant il peut choisir d'ignorer une réception pour procéder à une émission.

Dès lors, toute émission a lieu à une étape donnée, sans multiplexage possible. Prévenir les conflits d'émissions s'opère en temporisant les émissions. Nous supposons qu'un superviseur connaît la topologie du réseau.

C'est dans ce contexte qu'apparaissent nos deux problèmes algorithmiques : satisfaire le plus rapidement possible une collection de requêtes de communication en indiquant aux nœuds du réseau les dates auxquelles ils doivent faire suivre les paquets qui transitent par eux.

6.2 Modélisation et formalisme

Nous modélisons un réseau radio multi-sauts par un graphe orienté $G = (V, E)$. L'ensemble V représente l'ensemble des nœuds du réseau, et E représente l'ensemble des liaisons radios entre ces nœuds. Nous considérons généralement que le réseau est symétrique : si un nœud x peut communiquer avec un nœud y, alors y peut communiquer avec x. Nous préférons dans ce cas une modélisation avec un graphe non orienté, faisant ainsi abstraction de l'orientation des liaisons. Une requête de communication r est définie comme un couple (s, t) dans lequel s désigne l'émetteur de la requête et t son destinataire. Une route de communication dans le réseau est modélisée par un parcours.

Définition 39 (fonction de routage) *:*
Soient un graphe $G = (V, E)$ et une collection de requêtes R.

*Une **fonction de routage** est une fonction P qui à toute requête $r = (s, t)$ de R associe un parcours $P(r)$ dans G commençant par s et terminant par t. Pour un parcours donné, nous supposons les sommets deux à deux différents.*

Définition 40 (affectation de dates, correcte, sans conflit) *:*
Soient un graphe $G = (V, E)$, une collection de requêtes R, une fonction de routage P.

*Une **affectation de dates** est une fonction qui, à tout couple (r, x) où $r = (s, t) \in R$ et $x \in P(r)$ avec $x \neq t$[1] associe un entier naturel positif : la date à laquelle le sommet x doit relayer le message de la requête r.*

*Une affectation de dates d est **correcte** si et seulement si les sommets x_i de chaque parcours $P(r)$ émettent dans un ordre cohérent. Formellement, pour toute requête r avec $P(r) = (x_0, x_1, \ldots, x_k)$ on a $d(r, x_0) < d(r, x_1) < ... < d(r, x_k)$.*

*Nous dirons qu'une affectation de dates d correcte est de plus **sans conflit** si et seulement si pour toute paire de requêtes r et r' de R avec $P(r) = (\ldots, x_i, x_{i+1}, \ldots)$ et $P(r') = (\ldots, y_i, y_{i+1}, \ldots)$ les émissions respectent le modèle de communication et ne s'interfèrent pas au niveau d'un destinataire. D'une manière formelle, lorsque $d(r, x_i) = d(r', y_j)$ les conditions suivantes doivent être satisfaites :*

1. *$x_i \neq y_j$: prévient tout multiplexage, un même nœud ne peut émettre à la même étape les messages de deux requêtes différentes.*
2. *$x_{i+1} \neq y_j$ et $y_{j+1} \neq x_i$: implique qu'aucun nœud ne peut à la fois être récepteur et émetteur à la même étape (half-duplex).*
3. *$\{x_i, y_{j+1}\} \notin E(G)$ et $\{y_j, x_{i+1}\} \notin E(G)$: assure qu'un nœud ne peut recevoir un message que s'il n'est pas brouillé par une transmission voisine. Il implique également que l'émission d'un nœud affecte l'ensemble de ses voisins (Δ-port-émission et 1-port-réception).*

Le coût d'une affectation est la valeur de la plus grande date affectée, c'est à dire $\max(d(r, x) | r \in R, x \in P(r))$. *Nous le notons duree($d$).*

Définition 41 (affectation inarrêtable) :
Soient un graphe $G = (V, E)$, une collection de requêtes R, une fonction de routage P, et une affectation de dates d correcte et sans conflit des requêtes de R sur le graphe G.

*L'affectation est dite **inarrêtable** si et seulement si pour toute requête $r = (s, t) \in R$, dès que s émet alors la progression du message de r n'est plus temporisée et se propage sur $P(r)$ jusqu'à sa destination t. Formellement, pour $P(r) = (x_0, x_1, \ldots, x_i, \ldots, x_k)$, alors :*

$$d(r, x_i) = d(r, x_0) + i, \quad \forall 0 \leq i \leq k$$

Seuls les parcours et la date d'émission de la source de chaque requête nécessitent d'être connus dans la solution.

6.3 Les problèmes DAWN-paths et DAWN-requests

Compte tenu d'une collection de requêtes de communication à satisfaire dans un réseau radio synchrone, le problème DAWN (***Date Assignment in Wireless Network***) consiste à trouver une affectation de dates correcte et sans conflit le long de routes de communication. Selon si les routes de communication à suivre sont imposées (données par des tables de routage) ou pas, nous distinguons respectivement deux types de problèmes : DAWN-paths et DAWN-requests. Chacun de ces problèmes fait l'objet d'une des deux sections suivantes.

[1] le dernier nœud du parcours, destinataire du message, n'a pas à le relayer.

6.3.1 DAWN-paths

Le problème de décision DAWN-paths, présenté dans cette sous-section, consiste à se demander s'il est possible de satisfaire une collection de requêtes donnée en un nombre imposé d'étapes, lorsque les routes de communication sont données.

Donnée	:	Un graphe $G = (V, E)$, une collection de requêtes $R = \big(r_i = (s_i, t_i)\big)_{1 \leq i \leq K}$, une fonction de routage P qui associe à chaque requête r_i un parcours $P(r_i)$ reliant les sommets de r_i, un entier naturel D.
Question	:	Existe-t-il une affectation de dates correcte et sans conflit, telle que la plus grande date affectéenée soit inférieure ou égale à D ?

Problème de décision 6.3.1: Le problème DAWN-paths

(a) (b)

FIG. 6.1 – (a) Une instance $I = (G, R, P, 6)$ de DAWN-paths contenant 2 requêtes $r_1 = (A, F)$, $r_2 = (G, K)$, $P(r_1) = (A, B, C, D, E, F)$ and $P(r_2) = (G, H, I, J, K)$. (b) Une affectation de dates correcte et sans conflit pour I.

Une instance I de DAWN-paths est représentée par un 4-uplet (G, R, P, D) désignant respectivement un graphe, une collection de requêtes, une fonction de routage et une date d'émission maximum. La figure 6.1 illustre une instance de ce problème. Toute affectation d correcte et sans conflit des requêtes de R sur P et de coût inférieur à D représente donc une solution réalisable pour I, dont le coût est égal à $duree(d)$. Par souci de clarté, nous dirons que l'affectation de dates est ***réalisable*** sur I.

À partir de DAWN-paths nous définissons les problèmes suivants :

- Nous appelons ***min-DAWN-paths*** la version optimisation du problème DAWN-paths consistant à trouver une affectation de dates correcte et sans conflit ayant un nombre minimum d'étapes.
- Nous appelons ***DAWN-K-paths*** la version du problème DAWN-paths dans laquelle le nombre de requêtes est borné par une constante k.
- Nous appelons ***D-DAWN-paths*** pour $D \in I\!N^*$ la version du problème DAWN-paths dans laquelle le nombre maximum d'étapes autorisé est extrait de la donnée et se situe dans le nom du problème.

Les instances de min-DAWN-paths et D-DAWN-paths se définissent par un triplet (G, R, P).

6.3.2 DAWN-requests

Cette sous-section décrit un problème approché de DAWN-paths, nommé DAWN-requests. L'étude du problème DAWN-requests naît de l'observation suivante :

Observation 2 *:*
Recalculer les routes de communication en fonction des requêtes peut permettre de diminuer le nombre d'étapes minimum requis. Autrement dit, il n'existe pas de routage fixe permettant d'atteindre une solution optimale pour n'importe quelle collection de requêtes donnée.

Preuve :
Soit G le graphe présenté dans la figure 6.2.

Définissons $I_1 = (G, R_1, P)$ et $I_2 = (G, R_2, P)$ deux instances du problème min-DAWN-paths avec $R_1 = (r_1^1 = (A, H), r_2^1 = (A, G))$ et $R_2 = (r_1^2 = (A, H), r_2^2 = (A, D))$. On peut montrer par une étude de cas que l'instance I_1 peut être satisfaite en 4 étapes si et seulement si $P(r_1^1) = (A, B, E, H)$ et $P(r_2^1) = (A, C, F, G)$. Tout autre routage entraîne une augmentation du nombre d'étapes minimum. L'instance I_1 avec le routage précédemment énoncé est présentée en figure 6.2 (a). La figure (b) présente une solution de cette instance.

Or dans ce cas, la solution optimale du problème I_2 est au mieux de 6 étapes, alors qu'avec le routage P' tel que $P'(r_1^2) = (A, C, F, H)$ $P'(r_2^2) = (A, B, E, D)$ on obtient une solution en 4 étapes. L'instance I_2 avec le dernier routage énoncé est présentée en figure 6.2 (c). La figure (d) présente une solution de cette instance.

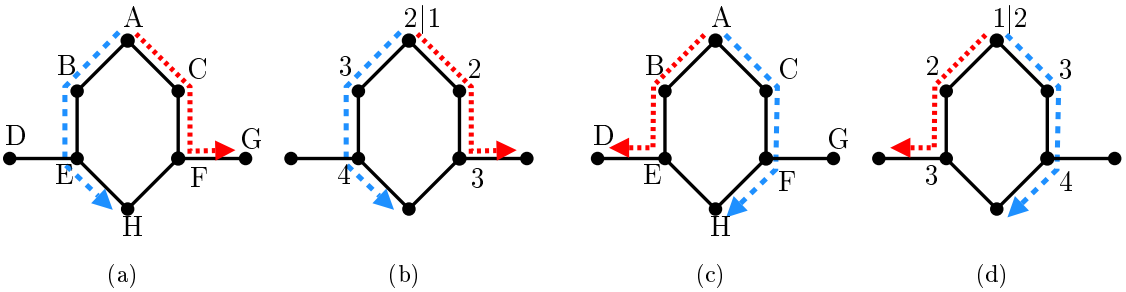


FIG. 6.2 – Il n'y a pas de routage optimal fixe pour le problème DAWN-paths

Clairement pour deux instances données, une même requête (A, H) doit tantôt suivre le parcours (A, B, E, H), tantôt (A, C, F, H) si l'on souhaite satisfaire les instances en un nombre minimum d'étapes. □

L'observation 2 nous incite à définir un second problème de décision, nommé DAWN-requests et présenté en 6.3.2. Ce problème consiste à se demander s'il est possible de satisfaire une collection de requêtes données en un nombre imposé d'étapes, avec la difficulté supplémentaire que les routes de communication sont libres. Une solution réalisable à une instance ce problème consiste à définir les routes de communications empruntées par chaque requête ainsi qu'à produire une affectation de dates d correcte et sans conflit pour ces dernières.

Donnée	:	Un graphe $G = (V, E)$, une collection de requêtes $R = \left(r_i = (s_i, t_i)\right)_{1 \leq i \leq K}$, un entier naturel D,
Question	:	Existe-t-il une fonction de routage P sur R, et une affectation de dates correcte et sans conflit, telle que la plus grande date affectée soit inférieure ou égale à D ?

Problème de décision 6.3.2: Le problème DAWN-requests

Nous notons une instance I de DAWN-requests par un triplet (G, R, D) désignant respectivement un graphe, une collection de requêtes, et une date d'émission maximum. Une solution à cette instance se décrit par le couple (P, d) où P est la fonction de routage décrivant les routes de communications et d une affectation de dates correcte et sans conflit de R sur P. Le coût de cette solution est alors égal à $duree(d)$. Par souci de clarté, nous dirons qu'une affectation de dates est ***réalisable*** pour l'instance (G, R, D) de DAWN-requests si et seulement s'il existe un routage P sur R, tel que l'affectation est réalisable pour l'instance (G, R, P, D) de DAWN-paths.

À partir de DAWN-requests nous définissons les problèmes suivants :

- Nous appelons ***min-DAWN-requests*** la version optimisation du problème DAWN-requests consistant à définir la fonction de routage et à trouver une affectation de dates correcte et sans conflit ayant un nombre minimum d'étapes.
- Nous appelons ***DAWN-K-requests*** la version du problème DAWN-requests dans laquelle le nombre de requêtes est borné par une constante k.
- Nous appelons ***D-DAWN-requests*** pour $D \in I\!N^*$ la version du problème DAWN-requests dans laquelle le nombre maximum d'étapes autorisé est extrait de la donnée et se situe dans le nom du problème.

Une instance de min-DAWN-requests ou D-DAWN-requests est représentée par un couple (G, R).

Remarque 14 *:*

Dans la définition d'un parcours donné par une fonction de routage, nous supposons que les sommets d'un même parcours sont tous distincts. Cette condition n'est pas essentielle, mais permet d'éviter qu'un message boucle dans le réseau.

Supprimer cette supposition n'affecte en rien le coût d'une solution optimale au problème DAWN-requests. En effet, si dans une solution réalisable un message parcours une boucle dans le réseau, il suffit de temporiser l'émission responsable de l'entrée dans la boucle d'autant d'étapes que nécessite la traversée de la boucle. Nous revenons alors à un état identique à celui affiché en sortie de boucle si le message avait transité par elle.

D'un point de vue formel, supposons l'existence d'une solution (P, d) de coût k sur une instance $I = (G, R, D)$ de DAWN-requests, telle que $\exists r = (s, t) \in R | P(r) = (s, \ldots, x, \ldots, x \ldots, t)$. Alors on peut définir une solution de coût $k' \leq k$ en supprimant simplement dans $P(r)$ les sommets contenus entre les deux occurrences du même sommet x, et la première occurrence de x. L'affectation de dates reste inchangée sur les éléments restants de $P(r)$.

Remarque 15 *:*

L'ajout de la particule « unstoppable » devant un problème désigne sa version dans laquelle toute solution réalisable est représentée par une affectation de date correcte, sans conflit, et inarrêtable.

En accord avec cette dernière remarque, nous définissons les problèmes suivants :
- ***unstoppable-min-DAWN-paths***,
- ***unstoppable-DAWN-K-paths***,
- ***unstoppable-D-DAWN-paths***,
- ***unstoppable-min-DAWN-requests***,
- ***unstoppable-DAWN-K-requests***
- ***unstoppable-D-DAWN-requests*** .

Remarque 16 *:*
Reprenons la condition citée en remarque 14 et stipulant que les sommets d'un parcours sont deux à deux distincts. Nous annonçons dans cette même remarque que cette condition est facultative pour les instances de DAWN-requests. La condition devient essentielle dès lors que l'on s'intéresse au problème unstoppable-DAWN-requests, car le coût de la solution optimale est affecté en conséquence.

Nous illustrons cette remarque par l'instance $I = (G, R)$ de unstoppable-min-DAWN-requests où G est le graphe de la figure 6.3(a), et $R = \{r_1 = (a, n), r_2 = (l, a)\}$. Si l'on n'autorise pas de boucle, les seuls parcours possibles sont $P(r_1) = (a, b, c, d, e, f, k, l, m)$ et $P(r_2) = (l, k, f, j, e, d, c, b)$ (figure 6.3(b)). Les requêtes ne pouvant être temporisées une fois parties, alors r_2 ne peut pas débuter avant que r_1 n'ait atteint le sommet m, à l'étape 8. Le coût de la solution optimale est alors de 15.

Si l'on autorise un message à boucler dans le réseau, alors une solution optimale de coût 12 utilisant les parcours $P(r_1) = (a, b, c, d, e, f, k, l, m)$ et $P(r_2) = (l, k, f, j, i, h, g, f, e, d, c, b)$ peut être obtenue, en faisant débuter chaque requête à la première étape (figure 6.3(c)). Le passage de r_1 dans la boucle permet aux deux requêtes de se « croiser ».

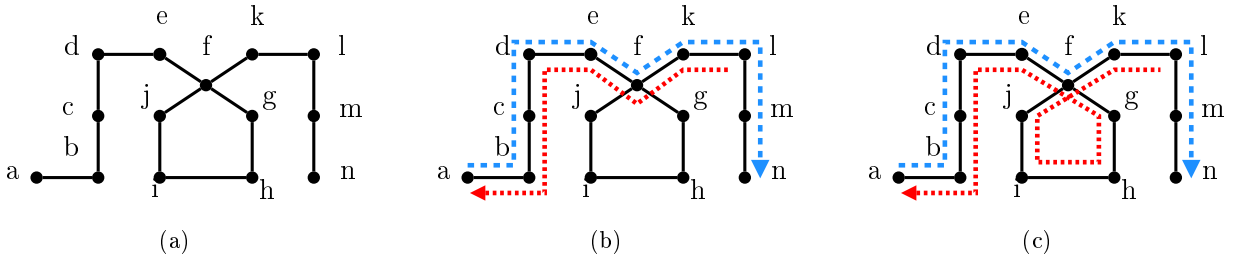


FIG. 6.3 – Une instance de unstoppable-DAWN-requests

La mise en place d'un routage incitant un message à boucler dans le réseau n'est pas à notre sens une bonne option. Toute possibilité de boucle reste donc exclue par la suite.

6.4 Vers un modèle online

L'ensemble des problèmes jusqu'alors énoncés a été établi sur un réseau radio synchrone dans lequel aucune émission n'a été planifiée ou n'est en cours. Supposons le scénario suivant :

1. Une collection R_1 de requêtes doit être routée sur le réseau.
2. Un superviseur calcule une solution en D_1 étapes, communique l'affectation de dates (éventuellement les politiques de routage) aux nœuds du réseau qui démarrent les émissions.

3. Alors que le processus de satisfaction des requêtes de R_1 est en pleine exécution, une seconde collection de requêtes R_2 à satisfaire apparaît.

Parmi les différentes politiques qui peuvent être envisagées, nous distinguons deux approches :

1. Attendre que toutes les requêtes R_1 soient satisfaites, calculer ensuite une affectation de dates des requêtes R_2 sur le graphe et exécuter cette affectation.
2. Calculer une affectation de dates des requêtes R_2 et l'exécuter sans attendre, en tenant compte des indisponibilités des sommets déjà mobilisés pour satisfaire les requêtes de R_1.

Nous qualifions la première approche d'***offline***, en opposition avec la seconde que nous dirons ***online***. Intuitivement l'approche online offre de meilleures performances en terme de temps que son équivalent offline. Pourquoi attendre qu'une première collection de requêtes soit satisfaite pour faire partir la seconde ? Ce modèle online présente en outre des aspects plus réalistes : si une région très localisée du réseau est mobilisée pour satisfaire R_1, la politique offline est pénalisante, car aucune requête de R_2 ne peut débuter même si l'ensemble des nœuds des parcours des requêtes de R_2 ne participe à l'émission des requêtes de R_1.

Nous formalisons dans cette section l'approche online du problème DAWN.

6.4.1 Formalisme des versions online de DAWN

Dans sa version online, le problème DAWN consiste à trouver une affectation de dates correcte et sans conflit pour un ensemble de requêtes donné, dans un réseau radio où des émissions ont déjà été planifiées. Ces émissions résultent d'une ou plusieurs affectations de date, précédemment calculées à partir d'un ou plusieurs ensembles de requêtes. L'ensemble de ces émissions planifiées peut se décrire par une affectation de dates d_1 correcte et sans conflit. Durant toute l'exécution de d_1, certaines arêtes deviennent indisponibles à certaines étapes, tandis que d'autres redeviennent utilisables. Le problème online-DAWN-paths consiste donc à satisfaire un ensemble de requêtes R dans un réseau où la topologie varie au cours du temps de manière prédictible.

Modéliser un réseau dynamique

Nous introduisons le vocabulaire suivant :

Définition 42 (affectations compatibles) *:*
Soient un graphe G, deux collections de requêtes R_1 et R_2 et deux affectations de dates d_1 et d_2 valides et sans conflit satisfaisant respectivement les requêtes de R_1 et R_2.

*Les affectations d_1 et d_2 sont **compatibles** si l'affectation d des requêtes $R_1 \cup R_2$ telle que :*
- $d(r,x) = d_1(r,x), \forall r \in R_1, x \in P(r)$
- $d(r,x) = d_2(r,x), \forall r \in R_2, x \in P(r)$

est correcte et sans conflit.

*Nous dirons que l'affectation d_2 est **compatible pour** d_1 si et seulement si d_1 a été défini avant d_2 sans tenir compte des requêtes satisfaites par d_2, et que les deux affectations sont compatibles.*

La donnée de la version online du problème DAWN peut se décrire par un graphe et une affectation correcte et sans conflit d_1 définissant des émissions invalidant des liens du réseau le temps

d'une étape, et par un ensemble de requêtes R à satisfaire. Une solution consiste donc à trouver une affectation d_2 correcte de R sur un graphe G, et qui soit compatible avec d_1. Concrètement, les émissions de l'affectation d_1 invalident des liens du réseau le temps d'une étape. La recherche d'une affectation d_2 compatible avec d_1 sur G peut se ramener à la recherche d'une affectation sur un réseau dynamique. Récemment, les graphes évolutifs [Fer02] ont été proposés comme une abstraction formelle pour les réseaux dynamiques. Nous reprenons la définition proposée dans [BXFJ03] :

Définition 43 (Graphe évolutif) *:*
Soient un graphe $G = (V, E)$ un graphe, et une séquence ordonnée $S_G = (G_1, G_2, \ldots, G_\tau)$ de graphes partiels de $\tilde{G}$ où $\tilde{G} = (V, \tilde{E})$ est un graphe orienté symétrique tel que $\{x, y\} \in E \Leftrightarrow (x, y) \in \tilde{E}$ et $(y, x) \in \tilde{E}$.

*Alors le couple (G, S_G) est appelé un **graphe évolutif**.*

Dans la suite, nous notons $S_G[i]$ le i-ème graphe de la séquence S_G. Concrètement, chaque graphe partiel correspond à la connectivité du réseau durant une période de temps désignée par son index.

La définition suivante permet de construire un graphe évolutif à partir d'un graphe et d'une affectation de dates correcte et sans conflit.

Définition 44 (Graphe évolutif représentant une affectation) *:*
Soient une instance (G, R, P, D) de DAWN-paths, et d une affectation réalisable pour I.

*Un **graphe évolutif représentant une affectation** est un couple (G, S_G) qui décrit la disponibilité des liens du réseau durant l'exécution de d. Le graphe $S_G[i]$ de la séquence S_G représente l'état du réseau à l'étape i, pour tout $i \in [1, duree(d)]$. Le graphe G représente l'état du réseau au repos. Cet état est considéré pour toute étape $i \geq duree(d) + 1$.*

Ce graphe évolutif se construit par l'algorithme 5 :

L'algorithme 5 intègre toutes les contraintes relatives à notre modèle. Soit (G, S_G) le graphe évolutif construit à partir d'une affectation d_1. Supposons une transmission de x_k à x_{k+1} à l'étape i dans d_1, alors :

- ligne 9 : le sommet x est isolé dans $S_G[i]$. Il ne peut émettre ou recevoir tout autre message. Les émissions de $y \in N_G(x_k)$ sont tolérées, mais ne peuvent être perçues par x_k.
- ligne 10 : aucun voisin de x_k dans G ne peut recevoir de message à l'étape i.
- ligne 11 : le sommet x_{k+1} est isolé dans $S_G[i]$. Il ne peut émettre ou recevoir tout autre message.
- ligne 12 : aucun nœud $y \in N_G(x_{k+1})$ ne peut émettre à l'étape i, pour ne pas interférer l'émission de x_k vers x_{k+1}. Néanmoins y peut recevoir une transmission voisine.

L'utilisation des graphes évolutifs permet de définir le problème de satisfaction de requêtes de manière concise. Une requête dans un graphe évolutif (G, S_G) est définie par un couple (s, t), avec $s, t \in V(G)$. La définition d'une affectation de dates dans sa version originale ne tient pas compte du caractère dynamique de la topologie et doit être revue en conséquence. Nous proposons les définitions suivantes :

Algorithme 5 : Construction d'un graphe évolutif représentant une affectation

Entrées : une instance $I = (G, R, P, D)$ de DAWN-paths, et d une affectation réalisable pour I.

Sorties : Un graphe évolutif (G, S_G) représentant l'affectation d.

```
1  début
2  |  G̃ = (V, Ẽ)|{x, y} ∈ E ⇔ (x, y) ∈ Ẽ et (y, x) ∈ Ẽ.
3  |  Fixer le nombre d'éléments τ de la séquence S_G à duree(d)
4  |  pour i allant de 1 à τ faire
5  |  |  S_G[i] = G̃
6  |  pour tous les r = (x_1, x_l) ∈ R avec P(r) = (x_1, x_2, ..., x_l) faire
7  |  |  pour tous les k ∈ [1, l] faire
8  |  |  |  si d(r, x_k) = i alors
9  |  |  |  |  E(S_G[i]) = E(S_G[i]) − {(x_k, a), (a, x_k)}, ∀a ∈ V(G)
10 |  |  |  |  E(S_G[i]) = E(S_G[i]) − {(a, z)}, ∀z ∈ N_G(x_k), ∀a ∈ V(G)
11 |  |  |  |  E(S_G[i]) = E(S_G[i]) − {(x_{k+1}, a), (a, x_{k+1})}, ∀a ∈ V(G)
12 |  |  |  |  E(S_G[i]) = E(S_G[i]) − {(z, a)}, ∀z ∈ N_G(x_{k+1}), ∀a ∈ V(G)
13 fin
```

Définition 45 (affectation online de dates, correcte, sans conflit) *:*
Soient un graphe évolutif (G, S_G) avec $|S_G| = \tau$, une collection de requêtes R, et une fonction de routage P dans G.

*Une **affectation online de dates** est une fonction qui, à tout couple (r, x) où $r = (s, t) \in R$ et $x \in P(r)$ avec $x \neq t$, associe un entier naturel positif : la date à laquelle le sommet x doit relayer le message.*

*Une affectation online de dates d est **correcte** si et seulement si les sommets x_i de chaque parcours $P(r)$ émettent dans un ordre cohérent, et sont possibles à la date fixée. Formellement, pour toute requête r avec $P(r) = (x_0, x_1, x_2, \ldots, x_k)$ on a :*

1. $d(r, x_0) < d(r, x_1) < d(r, x_2) < \cdots < d(r, x_k)$
2. *si* $d(r, x_i) = p$ *avec* $p \leq \tau$, *alors* $(x_i, x_{i+1}) \in E(S_G[p]), \forall i \in [0, k-1]$

*Une affectation online de dates d est **sans conflit** si et seulement si lorsque $d(r, x_i) = d(r', y_j) = p$ les conditions suivantes sont satisfaites :*

1. $x_i \neq y_j$
2. $x_{i+1} \neq y_j$ *et* $y_{j+1} \neq x_i$
3. *si* $p \leq \tau$, $(x_i, y_{j+1}) \notin E(S_G[p])$ *et* $(y_j, x_{i+1}) \notin E(S_G[p])$
4. *si* $p > \tau$, $(x_i, y_{j+1}) \notin E(G)$ *et* $(y_j, x_{i+1}) \notin E(G)$

Remarque 17 *:*
Reprenons les notations de la définition 45. Pour chaque requête r, un parcours $P(r)$ est défini dans G. Ce parcours est valide dans (G, S_G) car pour toute date $t > \tau$ la topologie du réseau est

représentée par G.

Déclinaison de DAWN dans sa version online

Nous déclinons les problèmes présentés dans cette seconde partie dans leur version online. Dans chacun des problèmes, la topologie est désormais représentée par un graphe évolutif (G, S_G), et toute affectation de date correcte et sans conflit répondant au problème doit être online. Nous ne considérons pas seulement les graphes évolutifs représentant une affectation, mais l'ensemble des graphes évolutifs. Cette approche permet d'inclure à notre modèle des évènements ponctuels comme par exemple une intervention sur un lien.

Nous ajoutons la particule « online- » devant le nom d'un problème pour désigner sa version online.

À titre d'exemple, le problème de décision online-DAWN-paths, présenté dans cette sous-section, consiste à se demander s'il est possible de satisfaire dans un réseau dynamique (modélisé par un graphe évolutif) une collection de requêtes données en un nombre imposé d'étapes, lorsque les routes de communication sont données.

Donnée	**:**	Un graphe évolutif (G, S), une collection de requêtes $\big(r_i = (s_i, t_i)\big)_{1 \leq i \leq K}$, une fonction de routage P qui associe à chaque requête r_i un parcours $P(r_i)$ reliant les sommets de r_i, un entier naturel D.
Question	**:**	Existe-t-il une affectation de dates online correcte et sans conflit, telle que la plus grande date affectée soit inférieure ou égale à D ?

Problème de décision 6.4.1: Le problème online-DAWN-paths

Chapitre 7

Étude de complexité

Ce chapitre est dédié à l'étude de complexité des problèmes DAWN-paths et DAWN-requests, et des sous-problèmes qui leurs sont associés. Nous montrons dans la première section que tous les problèmes introduits dans le chapitre précédent sont NP-complets en général. Nous montrons que ceci reste vrai, même lorsque la topologie du réseau est restreinte, ou lorsque le nombre maximum d'étapes autorisées est très petit (de l'ordre de 3 ou plus). Nous définissons dans la seconde section la limite exacte entre la polynomialité et la NP-complétude de DAWN-paths et DAWN-requests, selon la valeur de ce nombre maximum d'étapes.

7.1 Des problèmes difficiles

Nous montrons dans cette section que les problèmes DAWN-paths et DAWN-requests sont en général NP-difficiles et non approximables à un facteur constant près. Ces résultats concernant la NP-difficulté et la non approximabilité reposent sur des résultats connus pour le problème de coloration de graphes. Pour tout entier naturel D le problème D-COLORING cherche à déterminer s'il est possible d'associer un entier de l'intervalle $[1, D]$ à chaque sommet d'un graphe non orienté donné de sorte que deux sommets adjacents n'aient pas le même entier associé. On qualifie alors la coloration obtenue de ***propre***. Pour $D \geq 3$, le problème D-COLORING est NP-complet. Le problème de minimisation associé est min-COLORING, connu pour être NP-difficile et non-approximable à un facteur constant près.

Nous montrons que pour tout entier naturel $D \geq 3$ le problème D-DAWN-paths est NP-complet (ce qui implique naturellement la NP-complétude du problème DAWN-paths et la NP-difficulté de min-DAWN-paths). Nous montrons par ailleurs qu'il n'existe pas, sauf si $P = NP$, d'algorithme polynomial d'approximation à un facteur constant pour le problème min-DAWN-paths.

7.1.1 Des problèmes difficiles dans le cas général

Nous démontrons dans cette sous-section le théorème suivant :

Théorème 18 *:*
Les problèmes de décision D-DAWN-paths et D-DAWN-requests sont NP-complets pour tout $D \geq 3$.

Les problèmes d'optimisation min-DAWN-paths et min-DAWN-Requests sont NP-difficiles et non approximables à un facteur constant près.

Preuve :
Nous montrons d'abord que D-DAWN-requests est NP-complet depuis une réduction au problème D-COLORING.

Le problème D-DAWN-requests est dans NP : on peut vérifier en temps polynomial si une fonction de routage P permet de router efficacement les requêtes, et si une affectation de dates f est correcte, sans conflit et requiert moins de D étapes.

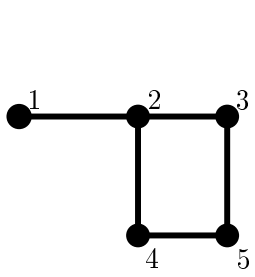
Soit $I_C = (G_C)$ une instance du problème D-COLORING. Nous construisons à partir de I_C une instance I=(G,R) de D-DAWN-requests, où G est un graphe avec :

$$V(G) = \{s_x, t_x | x \in V(G_C)\}^1$$

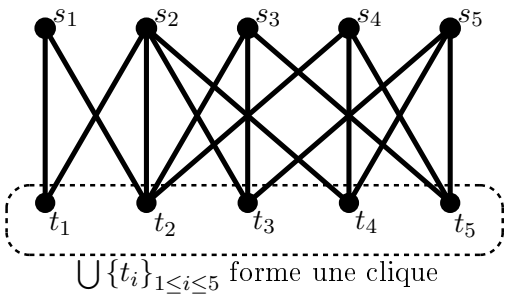
et

$$E(G) = \big\{\{s_x, t_x\} | x \in V(G_C)\big\} \cup \big\{\{s_x, t_y\} | \{x, y\} \in E(G_C)\big\} \cup \big\{\{t_x, t_y\} | x, y \in V(G_C)\big\}$$

Nous définissons sur G une collection de requêtes $R = \big(r_x = (s_x, t_x) | x \in V(G_C)\big)$. La construction de G et de l'ensemble R est clairement polynomiale. Les figures 7.1(a) et 7.1(b) présentent un graphe G_C et le graphe résultat G construit depuis G_C.



(a) Un graphe G_C



(b) Un graphe G construit depuis G_C

FIG. 7.1 – Exemple de construction de G depuis G_c

Nous montrons que s'il existe une affectation de dates valide pour I en k étapes, alors il existe une solution au problème D-COLORING pour l'instance I_C avec k couleurs ($k \leq D$), et réciproquement.

Notons $n = |V(G_C)|$. Soient $S = (P, d)$ un couple désignant une fonction de routage P et une affectation de dates d réalisable pour (G, R, D) et de coût $k \leq D$. Supposons qu'il existe une requête $r_i = (s_i, t_i)$ de sorte que le message n'est pas directement émis de s_i à t_i, mais nécessite au moins un nœud relais $t_j | j \neq i$. Si t_j émet le message de s_i à l'étape t, alors aucun autre nœud s_l ne peut transmettre à cette étape, puisque $\bigcup \{t_i\}_{1 \leq i \leq n}$ forme une clique. Nous pouvons alors construire une solution S' à partir de S avec un coût $k' \leq k$ dans laquelle s_i transmet directement à t_i à l'étape t. La fonction de routage est alors immédiate.

Soit $S' = (P, d)$ une telle solution, où d est une affectation valide pour l'instance I, et $P(r_i) = (s_i, t_i) \forall r_i \in R$. Soit c la fonction qui associe à chaque sommet $x \in V(G_C)$ la couleur $d(r_x, s_x)$.

[1] Formellement, la notation $s_x, t_x | x \in V(G_C)$ suppose que les éléments de $V(G_C)$ sont tous nommés. Nous supposons par la suite que cette condition est toujours vérifiée lorsque nous utilisons cette notation.

Notons que $duree(d) = \max(c)$. La coloration résultante est propre puisque si x et y sont adjacents dans G_C, alors par construction les arêtes $\{r_x, t_y\}$ et $\{r_y, t_x\}$ existent dans G. Ceci implique que $d(r_x, s_x) \neq d(r_y, s_y)$.

Réciproquement, soit c une coloration propre des sommets de G. Soit d l'affectation de dates qui à tout couple (r_x, s_x) associe la date $c(x)$. Alors d est une affectation de dates correcte (évident) et sans conflit (puisque si $\{s_x, t_y\}$ est une arête de G' alors $\{x, y\}$ est une arête de G et donc $c(x) \neq c(y)$ et par conséquent $d(r_x, s_x) \neq d(r_y, s_y)$). L'affectation est donc valide pour I. Notons que $\max(c) = duree(d)$.

Pour conclure, nous rappelons qu'à toute coloration c de G_C correspond une solution (P, d) à l'instance I telle que $P(r_i) = (s_i, t_i) \forall r_i \in R$ et $\max(c) = duree(d)$. La réciproque est également vraie et donc : puisque D-COLORING est NP-complet pour tout $D \geq 3$, et que D-DAWN-requests est un problème dans NP alors D-DAWN-requests est NP-complet.

Par ailleurs, la réduction proposée conserve le coût d'une solution. Puisque le problème de minimisation min-COLORING est connu pour être NP-difficile et non approximable à une constante près, il s'en déduit la NP-difficulté de min-DAWN-requests.

Cette preuve s'étend aux problèmes D-DAWN-paths et min-DAWN-paths, en créant une instance (G, R, P) de D-DAWN-paths depuis l'instance I de D-DAWN-requests, où P est une fonction de routage qui associe le parcours $p_{r_x} = (s_x, t_x)$ à chaque requête $r_x = (s_x, t_x)$.

□

Nous énonçons le corollaire suivant :

Corollaire 5 *:*
Les problèmes min-DAWN-paths et min-DAWN-requests restent NP-difficiles et non-approximables à une constante près même si pour chaque requête $r_i = (s_i, t_i)$ les nœuds s_i et t_i sont adjacents.

Preuve :
Immédiat puisque dans la preuve du théorème 18, chaque source d'une requête est adjacente à sa destination. □

Ce dernier corollaire insiste sur la difficulté de min-DAWN-paths et min-DAWN-requests, puisqu'il montre que la distance maximum entre source et destination n'a aucune influence sur la complexité de ces problèmes.

Remarque 18 *:*
Intuitivement, on ne peut s'empêcher de penser qu'il est préférable de faire débuter le plus tôt possible les requêtes dont la source est très éloignée de la destination qui lui est associée. Cette stratégie est certainement efficace dans des instances aléatoires, mais ne peut aboutir à une solution approchant l'optimum à une constante multiplicative près, au vu du théorème 18.

Notons enfin que le théorème 18 et le corollaire 5 s'étendent aux problèmes de décision unstoppable-min-DAWN-paths et unstoppable-min-DAWN-requests et à leur version décision : sur l'instance

construite dans la preuve de ce dernier, chaque requête doit être satisfaite en 1 seule émission (les routes de communication sont de longueur 1). Les problèmes unstoppable-min-DAWN-paths et unstoppable-min-DAWN-requests sont donc aussi difficiles que min-DAWN-paths et min-DAWN-requests. De même, les problèmes unstoppable-D-DAWN-paths et unstoppable-D-DAWN-requests sont NP-complets pour $D \geq 3$.

Signalons enfin que lorsqu'un problème est montré NP-complet, alors sa version online l'est également (il suffit de considérer un graphe évolutif (G, S_G) avec $S_G[i] = G$ pour tout $1 \leq i \leq \tau$). Nous en déduisons la NP-complétude de online-D-DAWN-paths, online-D-DAWN-requests , online-unstoppable-D-DAWN-paths et online-unstoppable-D-DAWN-requests, pour tout $D \geq 3$. Les versions optimisation de ces problèmes sont NP-difficiles et non approximables à une constante près.

7.1.2 Une difficulté établie même dans des topologies restreintes

Les résultats de la sous-section précédente montrent que la majorité des problèmes étudiés sont difficiles à résoudre en général. Nous montrons dans cette sous-section que l'ajout de conditions fortes sur la topologie du réseau n'a qu'un impact modéré sur la complexité des problèmes. Précisément, le principal résultat démontré au terme de cette sous-section est la NP-complétude des problèmes DAWN-paths et DAWN-requests sur les arbres binaires. La preuve de ce résultat requiert les quatre prochains lemmes.

Lemme 3 *:*
Soit une instance (G, R, P, D) de DAWN-paths, telle qu'il existe une requête $r \in R$ avec $P(r) = (x_1, x_2, x_3, \ldots x_{D+1})$.

Alors pour toute affectation d valide dans (G, R, P, D), on a $d(r, x_i) = i, \forall 1 \leq i \leq D$.

Preuve :
Immédiat puisque la source et la destination de la requête r sont distantes de D sauts, qui est le nombre d'émissions maximum autorisé. □

Le lemme suivant annonce que l'on peut, par construction, empêcher que dans toute solution d'une instance I de DAWN-Paths un sommet x émette le message d'une requête r vers un sommet y à l'étape i, et ce de deux façons :
- en imposant l'existence d'une transmission inéluctable à cette étape dans le voisinage de y,
- ou en imposant l'existence d'une transmission inéluctable à cette étape à destination d'un sommet adjacent à x.

Lemme 4 *:*

Soit $I = (G, R, P, D)$ une instance de DAWN-Paths, et soient x un sommet de $V(G)$ et i un entier. Alors il existe une instance $I' = (G', R', P', D)$ du même problème avec $V(G) \subseteq V(G')$, $R \subseteq R'$ et telle que :
- *toute affectation d correcte et sans conflit pour I' comporte exactement D étapes, et est également correcte et sans conflit pour I,*

- *pour toute affectation d correcte et sans conflit pour I' et toute requête $r \in R'$, on a $d(r,x) \neq i$,*
- *l'instance I' est constructible en temps polynomial.*

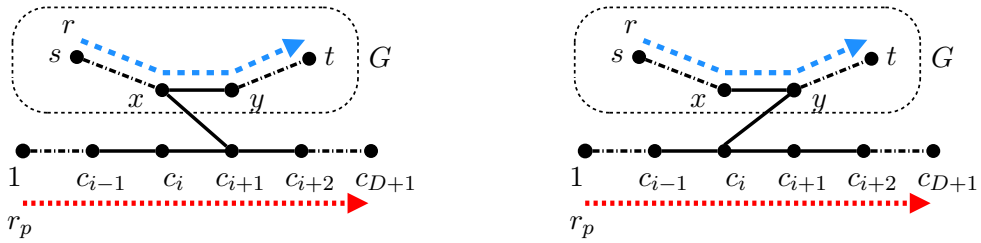
Preuve :

Soient les éléments suivants :
- une instance $I = (G, R, P, D)$ de DAWN-paths,
- une requête $r \in R$ avec $P(r) = (s, \ldots, x, y, \ldots, t)$
- une chaîne $C = \{c_1, c_2, c_3, \ldots, c_{D+1}\}$, avec $V(G) \cap V(C) = \emptyset$,
- une requête $r' = (c_1, c_{D+1})$,
- un entier $i \in [1, D]$.

Nous notons H_1 le graphe $(V(G) \cup V(C), E(G) \cup E(C) \cup \{\{y, c_i\}\})$, et H_2 le graphe $(V(G) \cup V(C), E(G) \cup E(C) \cup \{\{x, c_{i+1}\}\})$.

Soit $H = H_1$ et posons $P(r') = (c_1, c_2, \ldots c_{D+1})$. Nous affirmons que :
- L'instance $(H, R \cup \{r'\}, P, D)$ peut être construire depuis (G, R, P, D) en temps polynomial,
- soit d une affectation de date correcte et sans conflit pour $(H, R \cup \{r'\}, P, D)$, alors on a $d(r', c_i) = i$, $d(r, x) \neq i$, et d est réalisable pour (G, R, P, D).

Ces affirmations restent vraies si $H = H_2$. Une instance $(H, R \cup \{r'\}, P, D)$ est représentée sur la figure 7.2 lorsque $H = H_1$ (figure 7.2(a)) et $H = H_2$ (figure 7.2(b)).



(a) $H = H_1$ (b) $H = H_2$

FIG. 7.2 – Interdire à un sommet x d'émettre à l'étape i. Deux approches.

La source et la destination de la requête r_p sont distantes de D sauts. Dans toute affectation valide d pour (G, R, P, D), on a obligatoirement $d(r_p, c_i) = i$. On a donc $d(r, x) \neq i$ en accord avec la contraposée du point 3 de la définition 40. □

Remarque 19 *:*
Dans la construction proposée dans la preuve du lemme 4, si G est un arbre, alors les graphes H_1 et H_2 sont des arbres.

Pour une instance $I = (G, R, P, Q)$, une requête $r \in R$, un sommet $x \in P(r)$ et un entier i donnés, la construction présentée dans la preuve du lemme 4 implique que pour toute affectation d valide pour I, on a $d(r, x) \neq i$. En appliquant plusieurs fois cette construction, il est possible pour un même sommet x d'interdire plusieurs dates. L'inconvénient de cette approche est la nécessité d'ajouter une chaîne de $D+1$ sommets ainsi qu'une requête pour chaque date que l'on veut interdire

pour $d(r,x)$. Le lemme 5 généralise le lemme 4 pour un ensemble de dates donné, et propose dans sa preuve une construction plus pertinente.

Lemme 5 *:*

Soit $I = (G, R, P, D)$ une instance de DAWN-Paths, et soient x un sommet de $V(G)$ et T un ensemble d'entier. Alors il existe une instance $I' = (G', R', P', D)$ du même problème avec $V(G) \subseteq V(G')$, $R \subseteq R'$ et telle que :

- *toute affectation d correcte et sans conflit pour I' comporte exactement D étapes, et est également correcte et sans conflit pour I,*
- *pour toute affectation d correcte et sans conflit pour I', toute requête $r \in R'$ et tout entier $i \in T$, on a $d(r,x) \neq i$,*
- *l'instance I' est constructible en temps polynomial,*
- *on a, en notant $d_G(x)$ le degré de x dans le graphe G :*
  - $0 \leq d_{G'}(x) - d_G(x) \leq 3$,
  - $0 \leq d_{G'}(y) - d_G(y) \leq 3$,
  - $d_{G'}(x) - d_G(x) + d_{G'}(y) - d_G(y) = 3$,

Preuve :

Nous considérons les éléments suivants :

- une instance $I = (G, R, P, D)$ de DAWN-paths,
- une requête $r \in R$ de parcours associé $P(r) = (s, \ldots, x, y, \ldots, t)$
- une chaîne $C^0 = \{c_1^0, c_2^0, c_3^0, \ldots, c_{2D}^0\}$ avec $V(G) \cap V(C^0) = \emptyset$,
- une chaîne $C^1 = \{c_1^1, c_2^1, c_3^1, \ldots, c_{2D}^1\}$ avec $V(G) \cap V(C^1) = \emptyset$,
- une chaîne $C^2 = \{c_1^2, c_2^2, c_3^2, \ldots, c_{2D}^2\}$ avec $V(G) \cap V(C^2) = \emptyset$,
- un ensemble T de k entiers naturels $\{a_1, a_2, \ldots, a_k\}$ non nuls inférieurs ou égaux à D.
- un ensemble de requêtes R' de cardinalité k, créé à partir des éléments de T. La requête $r_a \in R'$ est associée à l'entier $a \in T$ et se définit comme suit :
  - Si $a \equiv 0[3]$, $r_a = (c_{D+1-a}^0, c_{2D+1-a}^0)$
  - Si $a \equiv 1[3]$, $r_a = (c_{D+1-a}^1, c_{2D+1-a}^1)$
  - Si $a \equiv 2[3]$, $r_a = (c_{D+1-a}^2, c_{2D+1-a}^2)$

Notons H le graphe tel que $V(H) = V(G) \cup V(C^0) \cup V(C^1) \cup V(C^2)$ et $E(H) = E(G) \cup E(C_0) \cup E(C^1) \cup E(C^2) \cup \{e_0, e_1, e_2\}$, où e_0, e_1, e_2 sont trois arêtes telles que :

- $e_0 \in \{\{y, c_D^0\}, \{x, c_{D+1}^0\}\}$
- $e_1 \in \{\{y, c_D^1\}, \{x, c_{D+1}^1\}\}$
- $e_2 \in \{\{y, c_D^2\}, \{x, c_{D+1}^2\}\}$

Alors :

1. L'instance $(H, R \cup R', P, D)$ peut être construire depuis (G, R, P, D) en temps polynomial.
2. Soient d une affectation de date valide pour $(H, R \cup R', P, D)$. Alors :
   (a) L'affectation d est valide pour (G, R, P, D).
   (b) Soit un entier $a \in T$. Posons $i \in [0, 2]$ tel que $a \equiv i[3]$. Alors $\forall j \in [D + 1 - a, 2D - a]$ on a $d(r_a, c_j^i) = j - D + a$.
   (c) Pour tout $a \in T$, on a $d(r, x) \neq a$.

Soit une affectation de dates d valide pour (G, R, P, D).

Les requêtes de R' sont partitionnées sur trois chaînes distinctes en fonction de la valeur du modulo de a par 3, où a est l'entier associé à une requête r_a. Cette partition implique que pour tout ensemble K quelconque, et tout élément $i \in K$ avec $i \leq D$, la requête r_i peut être définie sur une chaîne C_i, avec $i \in [0, 2]$. Réciproquement et pour tout $i \in [0, 2]$, une chaîne C^i ne contient que des requêtes r_a avec $a \equiv i[3]$.

Les requêtes situées sur la même chaîne sont orientées dans le même sens. Si deux requêtes r_a et r_b associées aux entiers a et b sont sur la même chaîne C^i, alors :
- $b - a \equiv 0[3]$,
- leur source sont situées sur C_i à une distance de $3|b - a1|$.

La source et la destination de toute requête $r \in R'$ sont distantes de D sauts. Chaque requête démarre donc à la première étape sans pouvoir être stoppée, en accord avec le lemme 3. Les sources des requêtes de R' d'une même chaîne sont suffisamment espacées pour ne pas se gêner.

Pour tout entier $a \in T$, et tout entier $j \in [D + 1 - a, 2D - a]$, et en posant $i \in [0, 2]$ tel que $a \equiv i[3]$, on a donc $r_a = (c^i_{D+1-a}, c^i_{2D+1-a})$, et donc $d(r_a, c^i_j) = j - D + a$.

Ainsi pour une date $a \in T$, alors le sommet c^i_D émet à l'étape a, avec $a - i \equiv 0[3]$. Notre construction implique que si c^0_D, c^1_D, ou c^2_D émet à une étape a, alors $d(r, x) \neq a$. □

Remarque 20 *:*
Reprenons le contenu du lemme 5. Alors :

1. *Si G est un arbre, alors le graphe H est un arbre.*
2. *$d_G(x) + d_G(y) + 3 = d_H(x) + d_H(y)$.*
3. *Si $e_0 = \{x, c^0_{D+1}\}$, $e_1 = \{y, c^1_D\}$ et $e_2 = \{y, c^2_D\}$, alors $d_H(x) = d_G(x)+1$ et $d_H(y) = d_G(y)+2$.*

Clairement, la combinaison du lemme 5 et du point 3 de la remarque 20 annonce que :
- on peut par construction interdire à un sommet x d'émettre vers un sommet y à un ensemble d'étapes donné,
- une des constructions possibles augmentera le degré de x de seulement 1, et le degré de y de seulement 2, quel que soit le nombre de dates interdites.

La figure 7.3 présente un exemple de cette construction depuis une instance $I = (G, R, P, D)$ où le graphe G est schématisé, et D est fixé à 7. Dans le graphe H résultant, le degré de x est augmenté de seulement 1, et le degré de y de seulement 2. Soit une affectation valide dans $(H, R \cup R', P', 7)$. Les requêtes définies sur la chaîne C_0 empêchent x d'émettre aux étapes 3 et 6, celles sur chaîne C_1 aux étapes 1 et 7, et celles de la chaîne C_2 aux étapes 2 et 5. Seule la date 4 reste possible et donc $d(r, x) = 4$.

La construction du lemme 5 est utilisée dans la preuve du théorème 19, pour empêcher certains nœuds d'émettre à toutes les étapes exceptées certaines préalablement choisies. Nous introduisons la définition suivante :

Définition 46 (une instance (u,D)-tendue) *:*
Soient u et D deux entier naturels, avec $D \geq 7$. Nous définissons une ***instance*** *(u, D)-**tendue***

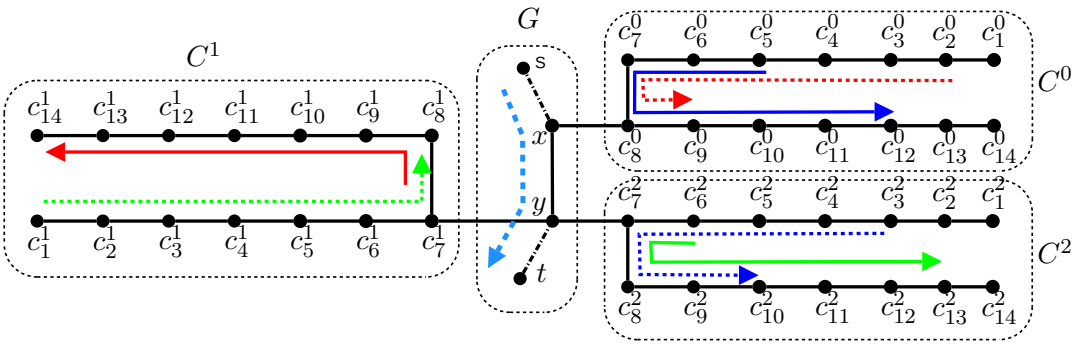


FIG. 7.3 – Interdire à un sommet x d'émettre à l'étape i. Seconde approche

comme une instance de DAWN-paths $(C_{[1,D+7u]}, R, P, D)$ *où* $C_{[1,D+7u]}$ *est une chaîne de* $D+7u$ *sommets* $\{1, 2, \ldots, D+7u\}$ *comportant les arêtes* $\{i, i+1\}|\forall 1 \leq i \leq D+7u-1$. R *est un ensemble de* $2u$ *requêtes* $\{r_1, \bar{r}_1, r_2, \bar{r}_2, \ldots, r_u, \bar{r}_u\}$ *avec* $r_i = \{(7i-5, 7i+D-9)\}$ *et* $\bar{r}_i = \{(7i-6, 7i+D-8)\}|\forall i \in [1, u]$.

Dans la figure 7.4, l'ensemble des requêtes $r_{x_1}, r_{\bar{x_1}}, r_{x_2}, r_{\bar{x_2}}, r_{x_3}, r_{\bar{x_3}}$ sur la chaîne $C_{[1,47]}$ représente une instance $(3, 26)$-tendue. Pour satisfaire ces requêtes en 26 étapes seulement deux possibilités se présentent pour chaque couple de requêtes r_i et $r_{\bar{i}}$: la requête r_i part à l'étape 1 et $r_{\bar{i}}$ à l'étape 3, ou bien r_i part à l'étape 5 et $r_{\bar{i}}$ à l'étape 1. Ensuite, aucune temporisation n'est possible pour n'importe quelle requête. Le lemme 6 formalise cette propriété des instances tendues :

Lemme 6 *:*
Considérons une instance (u, D)*-tendue,* u *et* D *étant deux entiers définis. Soit* d *une affectation valide et sans conflit pour cette instance. Alors pour tout* $\in [1, u]$*, on a :*
- $(d(r_i, 7i-5), d(\bar{r}_i, 7i-6)) \in \{(5, 1), (1, 3)\}$,
- $d(r_i, j+1) = d(r_i, j) + 1, \quad \forall j \in P(r_i) - \{7i + D - 9\}$,
- $d(\bar{r}_i, j+1) = d(\bar{r}_i, j) + 1, \quad \forall j \in P(\bar{r}_i) - \{7i + D - 8\}$.

Pour tour entier $i \in [1, u]$ *et* $j \in P(r_i)$ *:*
- *si* $d(r_i, 7i-5) = 1$ *alors* $d(r_i, j) = j - 7i + 6$ *et* $d(\bar{r}_i, j) = j - 7i + 9$
- *si* $d(r_i, 7i-5) = 5$ *alors* $d(r_i, j) = j - 7i + 10$ *et* $d(\bar{r}_i, j) = j - 7i + 7$

NP-complétude dans les arbres

Nous énonçons le théorème suivant :

Théorème 19 *:*
Le problème DAWN-paths est NP-complet lorsque le graphe du réseau est un arbre de degré 3.

Preuve :
La preuve est basée sur une réduction polynomiale de n'importe quelle instance du problème NP-complet 3-SAT , vers une instance (G, R, P, D) de DAWN-paths dans laquelle G est un arbre.

Soit $I_{SAT} = (U, W)$ une instance de 3-SAT sous forme normale conjonctive, où U est un ensemble de variables $\{x_1, x_2, \ldots, x_n\}$ et W un ensemble de clauses disjonctives $\{c_1, c_2, \ldots, c_m\}$ de

trois littéraux chacune. Notons $n = |U|$, $m = |W|$, et posons $D = m + 7n + 3$.

Considérons une instance (n, D)-tendue (G_1, R_1, P, D). L'idée générale de la preuve consiste à faire correspondre deux requêtes r_i et $\bar{r}_i$ à chaque variable $x_i \in U$. Par souci de clarté nous notons r_{x_i} la requête r_i et $r_{\bar{x}_i}$ la requête $\bar{r}_i$.

Soit G_2 l'arbre tel que $V(G_2) = V(G_1) \cup \{c_i^s, c_i^t | i \in [1, m]\}$ et $E(G_2) = E(G_1) \cup \{\{c_i^s, c_i^t\}, \{7n + i, c_i^s\} | i \in [1, m]\}$. Soit R_2 l'ensemble de requêtes $\{r_{c_i} = (c_i^s, c_i^t) | i \in [1, m]\}$. Considérons maintenant l'instance $(G_2, R_1 \cup R_2, P, D)$ (voir la figure 7.4 pour un exemple) et notons que $d_{G_2}(c_i^s) = 2$, et $d_{G_2}(c_i^t) = 1$. .

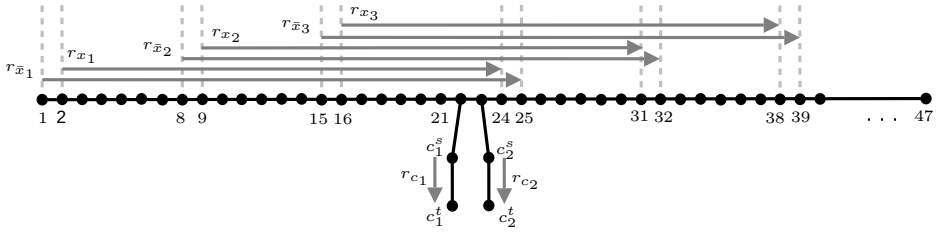


FIG. 7.4 – Un graphe G_2 et un ensemble de requêtes $R_1 \cup R_2$, construits depuis une instance $I_{SAT} = (U, W)$ où $U = \{x_1, x_2, x_3\}$ et $W = \{c_1, c_2\}$. Ici $D = m + 7n + 3 = 26$.

En appliquant la construction proposée dans le lemme 5 plusieurs fois, nous créons une instance $I = (G, R, P, D)$ comme suit : pour chaque requête $r_{c_i} = (c_i^s, c_i^t)$ nous empêchons c_i^s d'émettre vers c_i^t le message de la requête r_{c_i} à toutes les dates $t \leq D$, excepté pour 3 étapes définies à partir des littéraux contenus dans la clause c_i : si c_i contient le littéral positif (resp. négatif) associé à la variable $x_j | j \leq n$, alors c_i^s est autorisé à transmettre à l'étape $7n + i - 7j + 6$ *(resp.* $7n + i - 7j + 8$*)*. Nous choisissons la construction qui augmente le degré de c_i^s de 1 et le degré de c_i^t de 2, en accord avec le lemme 5 et de point 3 de la remarque 20. Cette construction était déjà introduite dans la figure 7.3.

Clairement puisque G_2 est un arbre, le graphe résultat G est également un arbre. Par ailleurs G est de degré maximum 3.

La taille de I est polynomiale dans la taille de I_{SAT}, et I peut être construite en temps polynomial. Nous vérifions que s'il existe une solution en D étapes pour I, alors il existe une solution à I_{SAT} et réciproquement.

Soit d une affectation de dates valide et sans conflit sur I. Toute requête $r_{c_i} | i \in [1, m]$ est clairement satisfaite dans d. D'après le lemme 6 et pour chaque entier $i \in [1, n]$, une requête parmi $\{r_{x_i}, r_{\bar{x}_i}\}$ doit démarrer à l'étape 1, et l'autre dès que possible. Une date a été affectée au couple (r_{c_i}, c_i^s), parmi les trois dates définies depuis les trois littéraux de la clause c_i. Notre construction implique qu'il existe un entier j tel que $d(r_{c_i}, c_i^s)$ est de la forme $7n+i-7j+6$ ou $7n+i-7j+8$, selon que x_j est un littéral positif ou négatif. Sj x_j est positif, la source de la requête r_{x_j} est située sur le sommet $7j - 5$ (c'est à dire à une distance de $7n + i - 7j + 5$ du sommet $7n + i$ lui même adjacent à c_i^s). La requête r_{x_j} débute obligatoirement à l'étape 1 - et on a $d(r_{x_j}, 7n+i) = d(r_{c_i}, c_i^s)$ - autrement

les émissions des requêtes se seraient interférées. Nous adoptons un raisonnement similaire si x_j est un littéral négatif.

Pour chaque clause c_i il existe donc au moins un littéral l (positif ou négatif) contenu dans la clause c_i, tel que la requête r_l a débuté avant $r_{\bar{l}}$. En effet, en affectant la valeur « Vrai » à toutes les variables x_i si r_{x_i} débute à l'étape 1 (c'est-à-dire avant $r_{\bar{x}_i}$) et « Faux » sinon, nous obtenons une solution à l'instance I_{SAT}.

Réciproquement, s'il existe une solution à I_{SAT}, alors nous pouvons déduire une solution pour l'instance I : pour chaque variable x_i nous débutons la requête r_{x_i} avant $r_{\bar{x}_i}$ si et seulement si x_i a la valeur « Vrai ». Pour chaque clause c_i, r_{c_i} démarre à la première étape autorisée et disponible (cette étape existe puisque la clause c_i est satisfaite par au moins un littéral).

Pour conclure nous rappelons que 3-SAT est NP-complet et que DAWN-paths appartient à NP. DAWN-paths est donc NP-complet sur les arbres binaires. □

Corollaire 6 *:*
Le problème de décision unstoppable-DAWN-paths est NP-complet sur les arbres binaires.

Preuve :
Dans la preuve du théorème 19, toute requête qui démarre à une étape donnée se propage alors jusqu'à sa destination sans être temporisée. Les résultats s'étendent donc logiquement à unstoppable-DAWN-paths. □

Remarque 21 *:*
Dans le théorème 19, le graphe est supposé être un arbre. Une seule chaîne connecte toute paire de sommets. Toute instance de DAWN-paths est clairement une instance de DAWN-requests et réciproquement. Le problème DAWN-requests a donc une complexité identique à DAWN-paths dans les arbres.

Corollaire 7 *:*
Les problèmes de décision DAWN-paths et DAWN-requests sont NP-complets sur les graphes planaires.

Preuve :
Immédiat puisqu'un arbre est un graphe planaire. □

Nous mentionnons que la preuve du théorème 19 peut être adaptée afin de montrer la NP-complétude de DAWN-paths et DAWN-requests dans les graphes de disques unitaires. Ces graphes sont souvent utilisés pour modéliser efficacement les réseaux de capteurs ou les réseaux ad-hoc, ce qui fait de cette remarque un résultat intéressant dans le cadre de notre étude. L'idée de la preuve consiste à définir un arbre qui est aussi un graphe de disques unitaires : on reprend la construction de l'arbre, mais en ajoutant des sommets intermédiaires entre les sommets $7n + i | i \in [1, m]$ afin de les éloigner suffisamment les uns des autres. Il est alors possible de plonger les de l'arbre dans un plan euclidien de sorte qu'une arête existe si et seulement si les sommets incidents sont distants

d'au plus une distance r fixée.

Signalons enfin qu'ici encore, les résultats de NP-complétude des problèmes s'étendent sur leur version online.

7.2 Limite entre polynomialité et NP-complétude

Comme nous venons de le voir, les problèmes DAWN-paths et DAWN-requests restent NP-complets même sur des classes de graphes très restreintes. L'impact de la topologie ne semble pas déterminant dans la complexité du problème. Dans la section suivante, nous étudions l'impact de la plus grande date allouable D.

Un premier résultat a été proposé dans le théorème 18, et annonce que pour tout $D \geq 3$, les problèmes de décision D-DAWN-paths et D-DAWN-requests sont NP-complets.

Le théorème suivant annonce la complexité de 1-DAWN-paths et 1-DAWN-requests :

Théorème 20 *:*
Les problèmes de décision 1-DAWN-paths et 1-DAWN-requests sont polynomiaux.

Preuve :
Soit (G,R,P) une instance de 1-DAWN-paths.

Les émissions ne peuvent avoir lieu qu'à la première étape. Appelons S_s (respectivement S_t) l'ensemble des sommets s (respectivement t) tels que $\exists r \in R, r = (s,t)$.

L'instance I possède une solution réalisable si et seulement si :

1. $\forall (r_1 = (s_1,t_1), r_2 = (s_2,t_2)) \in R^2,\ ((s_1 = s_2) \vee (t_1 = t_2)) \mapsto r_1 = r_2$,
2. $\forall r = (s,t) \in R, \{S_s \cap N_G(t)\} = \{s\}$.

Auquel cas la solution est immédiate. La preuve est identique pour 1-DAWN-requests. □

La frontière entre la polynomialité et la NP-complétude se situe donc entre $D = 2$ et $D = 3$ pour chacun des problèmes.

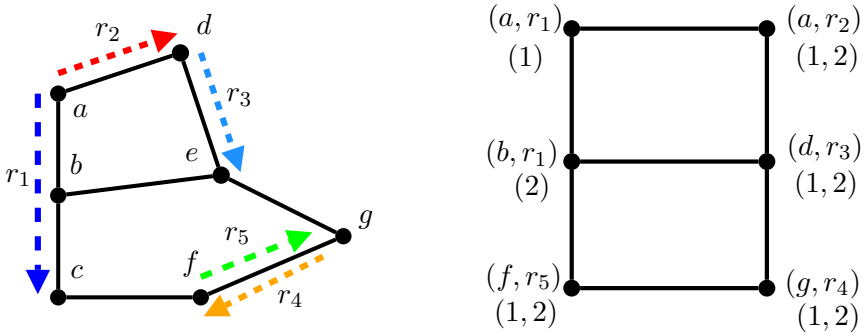
Le théorème 21 précise cette frontière sur D-DAWN-paths. La preuve de ce théorème utilise un résultat connu du problème LIST-COLORING (LC). Ce problème est défini par la donnée d'un graphe $G = (V,E)$ non orienté, d'un ensemble de couleurs C (par exemple un intervalle d'entiers) et d'une fonction L de V dans $P(C)$ (l'ensemble des parties de C). Le but est de trouver une fonction c de V dans C de telle sorte que pour chaque sommet v du graphe, $c(v)$ appartienne à $L(v)$ et soit différent de chaque couleur choisie pour ses voisins. Lorsqu'on se restreint aux instances dont les listes sont de longueur au plus 2 ce problème est connu pour être polynomial.

Théorème 21 *:*
Le problème de décision 2-DAWN-paths est polynomial.

Preuve :
Soit $I = (G, R, P)$ une instance du problème 2-DAWN-paths. Nous pouvons supposer que pour chaque requête r le parcours $P(r)$ est de la forme (s, t) ou (s, l, t), où l est adjacent à s et t dans G, puisque s'il existe un parcours plus long, l'instance est clairement sans solution.

Nous construisons depuis I une instance de LC dans laquelle les listes contiennent au plus deux éléments parmi les couleurs $\{1, 2\}$. Soit G' le graphe non orienté avec $V(G') = \{(r, x) | r \in R, x \in P(r)\}$ avec $x \neq t$. Il existe dans G' une arête entre deux sommets (u, x) et (v, y) si et seulement si pour toute affectation de dates d correcte et sans conflit pour l'instance I, on a $d(u, x) \neq d(v, y)$. Dit autrement $E(G')$ contient l'ensemble des arêtes $\{(u, x), (v, y)\}$, telles que les sommets x et y ne peuvent émettre respectivement les requêtes u et v à la même étape sur l'instance I. Soit L une affectation de listes de couleurs aux sommets de G'. Pour toute requête r telle que $P(r)$ contient 3 sommets (s, l, t) nous assignons dans G' la liste (1) au sommet (r, s) et la liste (2) au sommet (r, l). Nous assignons la liste de couleurs $(1, 2)$ pour les autres sommets du graphe G'. Clairement $I_{LC} = (G', \{1, 2\}, L)$ est une instance de LC construite à partir de I en temps polynomial.

La figure 7.5(a) présente une instance I_{LC} construite à partir de l'instance I de la figure 7.5(b).



(a) une instance de 2-DAWN-paths (b) une instance de LC

FIG. 7.5 – Construction d'une instance de LC depuis une instance de DAWN-paths

S'il existe une solution c au problème de coloration sur listes pour l'instance I_{LC}, alors il existe une affectation de dates d correcte, sans conflit et en 2 étapes au plus pour l'instance I : il suffit de poser $d(r, x) = c(r, x)$. La réciproque est également vraie.

Rappelons pour conclure que la construction de l'instance I_{LC} est polynomiale en temps et espace, et que le problème de coloration sur listes est polynomial lorsque les listes sont de longueur au plus 2.

□

Le théorème suivant complète notre étude de D-DAWN-requests.

Théorème 22 *:*
Le problème de décision 2-DAWN-requests est NP-complet.

Preuve :
Soit I une instance du problème 3-SAT construite sur un ensemble de variables W.

Pour chaque variable x de X soit $H_x = (X_x, Y_x, E_x)$ le biparti complet dont les sommets des deux partitions sont $X_x = \{(1,x),(1,\neg x)\}$ et $Y_x = \{(2,x),(2,\neg x)\}$.

Pour chaque clause $C = \{l_1, l_2, l_3\}$ de I, soit $F_C = (X_c, Y_c, E_c)$ le biparti complet dont les deux partitions sont $X_c = \{(1,C),(2,C)\}$ et $Y_c = \{(C,l_1),(C,l_2),(C,l_3)\}$.

Soit $I' = (G, R, 2)$ l'instance de DAWN-requests où G contient tous les sommets et les arêtes des graphes H_x et F_C pour toute variable x de W et toute clause C de I. De plus G contient les arêtes $\{(C,l),(2,l)\}$ pour tout littéral l et toute clause C tels que $l \in C$. La figure 7.6 présente un exemple de graphe G construit à partir d'une instance de 3-SAT. La collection R contient exactement toutes les requêtes de la forme $((1,C),(2,C))$ où C est une clause de I et toutes les requêtes de la forme $((1,l),(2,l))$ où l est un littéral d'une clause de I. Clairement notre construction s'effectue en temps et en espace polynomiaux.

Supposons que I' admette une solution. Un littéral l de I étant donné, affectons-lui la valeur « Vrai » si et seulement si $(1,l)$ émet à la date 1. Une clause C étant donnée, exactement un sommet du type (C,l) émet à l'étape 2. Ce sommet est relié à un sommet $(2,l)$ qui n'a pu recevoir le message qu'à l'étape 1. Donc l vaut « Vrai » et C est satisfait.

Réciproquement, supposons que I admette une solution. Pour tout littéral l de I qui a pour valeur « Vrai », affectonsgnons la date 1 au sommet $(1,l)$ et la date 2 au sommet $(2,\neg l)$. Soit C une clause de I. La date 1 est affectée au sommet $(1,C)$. La date 2 doit être affectée à un des sommets voisins (et un seul). Nous pouvons choisir un des couples dont le littéral l vaut « Vrai ». Cette date est possible puisque (C,l) est seulement relié à $(2,C)$ (destinataire) et $(2,l)$ qui a déjà reçu le message à la date 1.

Enfin, puisque 2-DAWN-requests est dans NP, il s'agit bien d'un problème NP-complet.

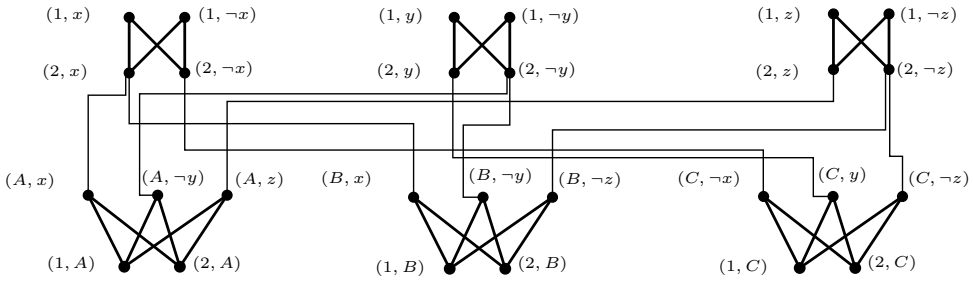


FIG. 7.6 – graphe construit à partir de l'instance $\{\{x,\neg y,z\},\{x,\neg y,\neg z\},\{\neg x,y,\neg z\}\}$

□

Le résultat suivant est un corollaire des théorèmes 20,21, et 22 :

Corollaire 8 *:*
Les problèmes de décision :
- *unstoppable-1-DAWN-paths ,*
- *unstoppable-1-DAWN-requests,*
- *et unstoppable-2-DAWN-paths*

sont polynomiaux.

Les problèmes de décision :
- *unstoppable-2-DAWN-requests,*
- *unstoppable-D-DAWN-requests avec* $D \geq 3$*,*
- *et unstoppable-D-DAWN-paths avec* $D \geq 3$

sont NP-complets.

Preuve :
Les résultats proposés sur D-DAWN-paths et D-DAWN-requests s'étendent à leur version inarrêtable, puisque dans chacune des preuves présentées, aucune requête n'est temporisée une fois démarrée. □

Chapitre 8

Algorithmes de résolution d'instances

Il découle du chapitre précédent que nos deux familles de problèmes restent difficiles même sur des topologies restreintes. Lorsque l'on étudie leur complexité en fonction de la date maximale allouable D, ces problèmes deviennent vite NP-complets même pour des petites valeurs de D. Nous proposons l'étude d'un dernier paramètre de l'instance, le nombre de requêtes. Nous montrons dans la première section, que lorsque ce paramètre est borné par une constante, disons K, alors les deux problèmes deviennent polynomiaux. Nous montrons dans la seconde section que ces résultats s'étendent lorsque la topologie du réseau est dynamique.

8.1 Polynomialité et nombre de requêtes

Pour tout entier naturel K, le problème min-DAWN-K-paths consiste à minimiser le nombre d'étapes nécessaires pour satisfaire K requêtes. Nous proposons le théorème suivant :

Théorème 23 *:*
Pour tout entier naturel K, le problème DAWN-K-paths est polynomial.

Preuve :
Pour un entier K donné, considérons (G, R, P) une instance du problème min-DAWN-K-paths, respectivement un graphe, une collection de requêtes $(r_1 \dots r_K)$ et une fonction de routage P. Nous construisons un graphe $G' = (V', E')$ comme suit :

Soit V' l'ensemble des K-uplets $(e_1 \dots e_K)$ où e_i est un sommet du parcours associé par P à la requête r_i. Chacun de ces K-uplets représente un état possible du réseau : la composante e_i indiquant la position du message de la requête r_i sur son parcours.

Deux états $(e_1, \dots, e_k)$ et $(f_1, \dots, f_k)$ sont compatibles s'il est possible de passer du premier au deuxième en une étape, en faisant émettre simultanément l'ensemble des nœuds $\{e_i | e_i \neq f_i, 1 \leq i \leq K\}$. Auquel cas et pour tout $e_i \neq f_i$ les conditions suivantes doivent être respectées :
- les sommets e_i et f_i doivent être immédiatement consécutifs dans le parcours $P(r_i)$,
- il n'existe pas d'indice $j \neq i$, tel que $e_j \neq f_j$ et $\{e_j, f_i\} \in E(G)$,
- pour tout $j \neq i$ tel que $e_j \neq f_j$, alors $|\{e_i, f_i, e_j, f_j\}| = 4$.

Le K-uplet $E_s = (s_1, \ldots, s_i, \ldots, s_k)$ représente l'état initial du réseau, et le K-uplet $E_t = (t_1, \ldots, t_i, \ldots, t_k)$ l'état final du réseau.

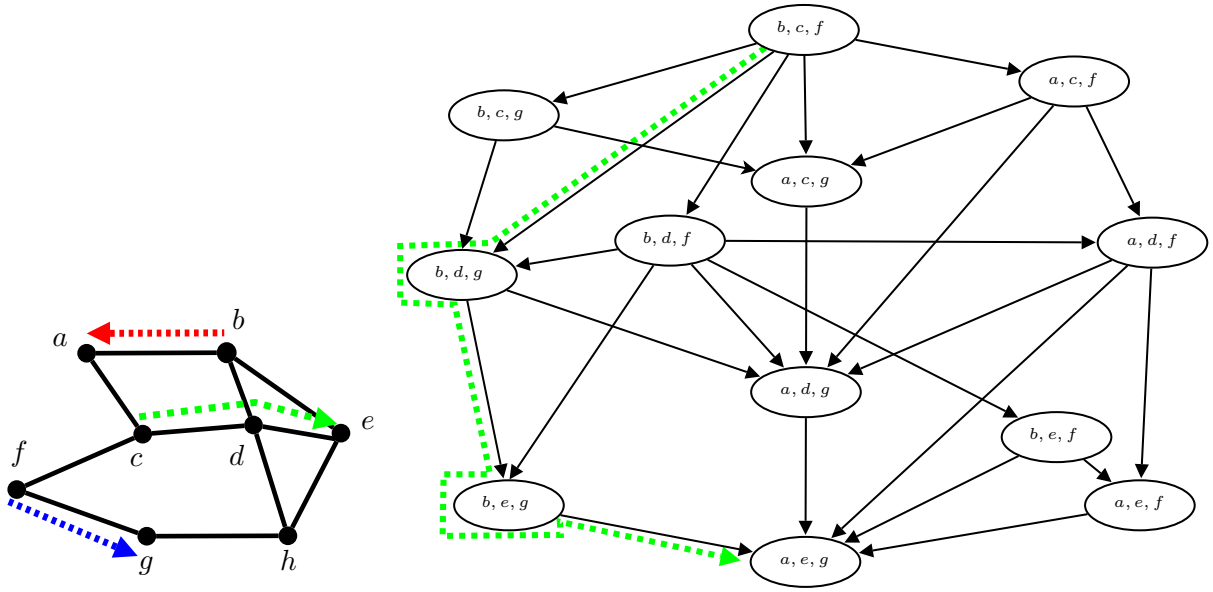
Soit E' l'ensemble de tous les couples (x, y) où x et y sont des états compatibles de V'. Clairement (V', E') est un graphe orienté constructible en temps polynomial. Nous pouvons associer à tout chemin $Q = (E_s, \ldots, A, \ldots, E - t)$ entre l'état $(s_1, \ldots, s_i, \ldots, s_k)$ et l'état $(t_1, \ldots, t_i, \ldots, t_k)$ une affectation de dates d, correcte et sans conflit, telle que la plus grande date affectée est égale au nombre d'arcs du chemin, et réciproquement.

Précisément, soient A et B deux sommets consécutifs dans Q. Si le i-ème sommet x de A est différent du i-ème sommet de B alors $d(r_i, x)$ est égal à l'indice de A dans le chemin Q.

Puisque la recherche d'un plus court chemin dans un graphe est un problème connu pour être polynomial nous pouvons directement conclure.

□

Nous illustrons une telle preuve par la figure 8.1(b), construite à partir de l'instance de la figure 8.1(a).



(a) une instance de DAWN-paths

(b) le graphe des états construit depuis 8.1(a)

FIG. 8.1 – Construction du graphe des états

Dans l'exemple de la figure 8.1(b) les états (b, c, f) et (a, e, g) sont respectivement les états initiaux et finaux du graphe. Le plus court chemin entre ces états est de 3, par exemple le chemin $((b, c, f), (b, d, g), (b, e, g), (a, e, g))$. Il donne une solution en 3 étapes :

- les sommets c et f émettent à l'étape 1,
- le sommet d est le seul à émettre à l'étape 2,
- enfin le sommet b émet à l'étape 3.

Corollaire 9 *:*

Pour tout entier K donné, le problème DAWN-K-requests est polynomial

Preuve :
Reprenons la preuve de la polynomialité de min-DAWN-K-paths. Les problèmes min-DAWN-K-paths et min-DAWN-K-requests sont très similaires, à l'unique différence que l'on ne sait pas dans le second par quels nœuds chaque requête transite.

Dans l'algorithme proposé dans la preuve du théorème 23, il suffit de ne plus générer seulement l'ensemble des K-uplets $(e_1 \dots e_K), \forall e_i \in P(r_i)$, mais l'ensemble des k-uplets $(e_1 \dots e_K), \forall e_i \in V(G)$. Le nombre d'états augmente mais reste polynomial en fonction de la donnée (de l'ordre de n^{2K} avec K constant). Deux états sont compatibles si l'on peut passer de l'un à l'autre en une étape. La condition relatant la consécutivité de e_i et f_i dans le parcours $P(r_i)$ est clairement hors de propos ici. La suite de la preuve reste inchangée.

Nous soulignons que si G est non orienté, alors le graphe des états est orienté symétrique.
□

Dans la preuve précédente, la solution retournée S peut nécessiter un post-traitement : pour une requête r, un même sommet x peut apparaître plusieurs fois dans $P(r)$. À partir de S nous générons une solution S' en supprimant tous les sommets entre la première et la dernière occurrence de x dans P, comme annoncé dans la remarque 14.

En appliquant successivement ce post-traitement nous générons une solution S^* dans laquelle chaque sommet apparaît au plus une fois dans un parcours donné.

Corollaire 10 *:*
Le problème unstoppable-DAWN-K-paths est polynomial.

Preuve :
Il suffit de modifier la preuve du théorème 23 :
Deux états $(e_1, \dots, e_k)$ et $(f_1, \dots, f_k)$ étaient compatibles s'il était possible de passer du premier au deuxième en une étape, en faisant émettre simultanément l'ensemble des nœuds $\{e_i | e_i \neq f_i, 1 \leq i \leq K\}$. Nous ajoutons aux conditions de compatibilité entre états une contrainte qui rend deux états incompatibles si la propagation d'un message est stoppée par un sommet autre que la source. Formellement, pour toute requête $r_i = (s_i, t_i)$, si $e_i = f_i$ alors $e_i = s_i$.

La suite de la preuve reste inchangée. □

Remarque 22 *:*
Dans la preuve de la polynomialité de unstoppable-min-DAWN-K-paths, le fait que la fonction de routage est connue est un point important : Pour toute solution S déduite du graphe des états, nous sommes certains que chaque sommet apparaît au plus une fois dans un parcours donné.

Or si nous essayons d'étendre cette preuve à unstoppable-min-DAWN-K-requests (sur le même principe que l'extension de la preuve de min-DAWN-K-paths à min-DAWN-K-requests), une solution

S peut être retournée dans laquelle un parcours boucle sur un sommet, ce qui simule une temporisation. Le coût de S peut alors être strictement inférieur au coût optimal d'une solution lorsqu'aucun sommet n'apparaît plus de deux fois dans un parcours, comme annoncé dans la remarque 16.

Il n'est donc pas possible de conclure à la polynomialité de unstoppable-min-DAWN-K-requests avec l'utilisation d'un graphe des états tel que nous l'avons défini.

8.2 Extension dans les topologies dynamiques

Nous avons précédemment introduit ce que pourraient être DAWN-paths et DAWN-requests dans un cadre plus pratique, dans lequel de nouvelles requêtes sont ajoutées spontanément au court du temps. Ceci nous a conduit à définir les problèmes online-DAWN-paths et online-DAWN-requests, dans lesquels la topologie du réseau est modélisée par un graphe évolutif. Naturellement, ces problèmes sont NP-complets dès lors que leurs équivalents offline le sont. Les caractéristiques des réseaux dynamiques laissent penser qu'il existe des familles d'instances de DAWN-paths/DAWN-requests que l'on sait résoudre polynomialement, mais qui deviennent difficiles dans leurs versions online. Cependant, nous montrons dans cette section que les problèmes online-DAWN-paths et online-DAWN-requests sont polynomiaux lorsque le nombre de requêtes est borné par K. Ce résultat permet d'envisager pour la suite des heuristiques destinées à résoudre les instances sans borne sur le nombre de requêtes.

La preuve du théorème suivant reprend l'idée de graphe des états introduit dans la preuve du théorème 23, et l'applique aux graphes évolutifs.

Théorème 24 *:*
Pour tout entier naturel K, le problème online-DAWN-K-paths est polynomial.

Preuve :
Pour un entier K donné, considérons $((G, S_G), R, P)$ une instance du problème online-min-DAWN-K-paths, où S est une séquences de sous-graphes du graphe orienté symétrique $G = (V, E)$, avec $|S| = \tau$, R une collection de K requêtes $(r_1 \dots r_K)$ et P une fonction de routage. Nous définissons le graphe $G' = (V', E')$ comme suit :

Soit V' l'ensemble des (K+1)-uplets $(e_1 \dots e_K, d)$ où e_i est un sommet du parcours associé par P à la requête r_i, et d un entier tel que $d \in [1, \tau + 1]$. Chacun de ces (K+1)-uplets représente un état possible du réseau : la composante e_i indique la position du message de la requête r_i sur son parcours, et l'entier d l'état du réseau, à savoir le graphe $S_G[d]$ pour $d \leq \tau$ et le graphe G lorsque $d = \tau + 1$.

Soit E' l'ensemble de tous les couples (X, Y) où X et Y sont des états compatibles de V' : deux états $(e_1, \dots, e_k, d_1)$ et $(f_1, \dots, f_k, d_2)$ sont compatibles s'il est possible de passer du premier au deuxième en une étape, en faisant émettre simultanément l'ensemble des nœuds $\{e_i | e_i \neq f_i, 1 \leq i \leq K\}$, lorsque le graphe du réseau est $S_G[d_1]$ pour $d_i \leq \tau$, ou G si $d = \tau + 1$. Compte tenu de l'évolution de la topologie, si $d_1 \neq \tau+1$, alors nécessairement $d_2 = d_1+1$, autrement $d_1 = d_2 = \tau+1$. Pour tout $e_i \neq f_i$, les sommets e_i et f_i doivent être immédiatement consécutifs dans le parcours $P(r_i)$.

Clairement $G' = (V', E')$ est un graphe orienté constructible en temps polynomial, puisque le nombre d'états est d'au plus $(|V(G)|^K \times \tau)$. Le (K+1)-uplet $(s_1, \dots s_i, \dots s_k, 1)$ représente l'état initial du réseau. Chacun des (K+1)-uplets $(t_1, \dots t_i, \dots t_k, d)|d \in [1, \tau + 1]$ désigne un état final, dans lequel les requêtes de R ont été satisfaites.

Nous pouvons associer à tout chemin Q entre l'état initial $(s_1, \dots s_i, \dots s_k)$ et tout état final $(t_1, \dots t_i, \dots t_k, d)$ une affectation de dates d, correcte et sans conflit, telle que la plus grande date affectée est égale au nombre d'arcs du chemin, et réciproquement.

Précisément, soient A et B deux sommets de G' consécutifs dans Q. Si le i-ème sommet x de A est différent du i-ème sommet de B alors $d(r_i, x)$ est égal au l'indice de A dans le chemin Q.

En recherchant le plus court chemin entre l'état initial et un état final quelconque, nous en déduisons une affectation de dates online correcte et sans conflit sur l'instance $((G, S_G), R, P)$. Puisque la recherche d'un plus court chemin dans un graphe est un problème connu pour être polynomial nous pouvons directement conclure.

□

Corollaire 11 *:*
Pour tout entier naturel K, le problème online-DAWN-K-requests est polynomial.

Preuve :
Sur le même principe que la preuve du corollaire 9. □

Vers une heuristique pour DAWN-paths ?

La polynomialité de online-DAWN-K-requests ouvre de nouvelles perspectives pour résoudre efficacement les instances de DAWN-paths. Supposons un ensemble non borné R de requêtes à satisfaire sur un graphe G. Notre étude a montré que trouver le nombre minimum d'étapes requis est NP-difficile et non approximable à une facteur constant près. Aussi doit-on envisager l'utilisation d'heuristiques afin de pouvoir formuler une solution approchée dans des temps raisonnables.

Une idée d'heuristique consécutive aux travaux sur online-DAWN-K-paths consisterait à découper l'ensemble de requêtes en paquets $R_1, R_2, \dots$ de K requêtes chacun :

- On définit une affectation optimale d_1 du premier paquet R_1 sur G, en un temps polynomial.
- Le système (G, d_1) permet de définir un graphe évolutif (G, S^1_G), représentant l'état du réseau lors de l'exécution des requêtes de R_1.
- On définit alors une affectation optimale d_2 du second paquet R_2 sur le graphe évolutif (G, S^1_G), en un temps polynomial.
- Le système $((G, S^1_G), d_2)$ permet à son tour de définir un graphe évolutif (G, S^2_G), représentant l'état du réseau lors de l'exécution des requêtes de R_1 et R_2.
- Nous continuons cette exécution jusqu'à traiter l'ensemble des paquets de requêtes.

Cette heuristique, à l'heure non implémentée, permettrait de définir en temps polynomial des solutions dont le coût moyen semble intuitivement bon par rapport à la solution optimale. Naturellement, ce coût est d'autant meilleur que K est grand, mais induit un temps d'exécution beaucoup plus long. Aussi la mise en place d'une telle heuristique, la comparaison des résultats obtenus avec

la solution optimale, et l'impact du paramètre K sur le coût des solutions constitueraient des suites intéressantes à donner à ce travail.

Conclusion

Dans cette seconde partie, nous avons étudié le problème de satisfaction de requêtes de communication dans un réseau radio multi-sauts synchrone. Ce problème a été envisagé selon l'hypothèse que les routes de communications sont connues ou non, ce qui nous a conduit à définir au chapitre 6 les deux familles de problèmes DAWN-paths et DAWN-requests, et les problèmes qu'elles englobent.

Les résultats présentés dans le chapitre 7 établissent la complexité de ces problèmes. Ils insistent sur le fait que ces problèmes restent difficiles même lorsque des conditions supplémentaires sont envisagées sur les instances étudiées. Considérer seulement les instances sur des topologies très spécifiques comme l'arbre binaire, ou dans lesquelles la source de chaque requête est adjacente à sa destination, ne change en rien la complexité de ces problèmes.

Le chapitre 8 apporte cependant un résultat encourageant, puisque les résultats qui y sont présentés annoncent que les problèmes sont polynomiaux dès lors que le nombre de requêtes est borné par une constante K. Cette polynomialité est garantie même sur des réseaux dynamiques, qui modélisent efficacement des réseaux sur lesquels un ensemble de requêtes est en cours de satisfaction. Ceci permet d'envisager un processus de satisfaction de requêtes dans des conditions d'exécution plus réalistes et ouvre une réflexion sur la mise en œuvre d'heuristiques.

Le tableau 8.1 résume l'ensemble des résultats de complexité présentés dans les chapitres 7 et 8.
Nous ne pouvons affirmer que unstoppable-min-DAWN-requests peut se résoudre en temps polynomial, bien que l'ensemble des résultats énoncés nous incitent à penser que oui.

Soulignons dans les résultats présentés que D-DAWN-requests est NP-complet pour $D \geq 2$, tandis que D-DAWN-paths reste polynomial pour $D = 2$. On constate sans surprise que la perte de l'information de routage rend le problème plus difficile. En revanche il est plus étonnant de constater que la version inarrêtable d'un problème semble aussi difficile que le problème lui-même : le fait d'imposer qu'une requête ne soit plus temporisée une fois démarrée ne semble pas jouer un rôle important dans la complexité des problèmes. Pourtant, et de manière intuitive, il est raisonnable de penser qu'il existe une classe d'instances sur laquelle DAWN-paths (respectivement DAWN-requests) serait NP-complet, mais pas unstoppable-DAWN-paths (respectivement unstoppable-DAWN-requests). Un apport intéressant à ce travail consisterait à mettre en évidence une telle classe d'instances.

Au terme de cette partie, le principal problème ouvert consiste à établir la complexité de DAWN-paths lorsque le graphe est une chaîne, et que les requêtes sont orientées dans la même direction. Nous savons le problème approximable à une constante près (facilement majorable par 3), et qu'il est même polynomial lorsque pour chaque requête source et destination ne sont distantes que d'au

Problèmes	**Complexite**	**Remarques**
DAWN-paths DAWN-requests unstoppable-DAWN-paths unstoppable-DAWN-requests	NP-complet	Reste NP-complet sur les arbres binaires, ou si chaque source est adjacente à sa destination.
D-DAWN-paths, $D \leq 2$	Polynomial	Réduction vers 2-LIST-COLORING
D-DAWN-paths, $D \geq 3$	NP-complet	Réduction depuis D-COLORING
D-DAWN-requests, $D = 2$	NP-complet	Réduction depuis 3-SAT
D-DAWN-requests, $D \geq 3$	NP-complet	Réduction depuis D-COLORING
min-DAWN-paths min-DAWN-requests unstoppable-min-DAWN-paths unstoppable-min-DAWN-requests	NP-difficile	Non approximable à une constante près. Reste NP-difficile sur les arbres binaires, les graphes planaires, les graphes de disques unitaires ou si chaque source est adjacente à sa destination.
min-DAWN-K-paths min-DAWN-K-requests unstoppable-min-DAWN-K-paths	Polynomial	Algorithme en $O(n^{2K})$.
online-DAWN-K-paths online-DAWN-K-requests	Polynomial	Algorithme en $O(n^{2K})$.

TAB. 8.1 – Tableau récapitulatif des résultats de complexité

plus 3. Quelques algorithmes polynomiaux ont été proposés et évalués de manière expérimentale. Les résultats des simulations laissent penser qu'en améliorant ces algorithmes, il serait peut être possible d'obtenir un algorithme polynomial pour résoudre le problème. Pourtant, et en dépit de nombreux efforts, il ne nous a pas été possible d'aboutir à un résultat concluant à la polynomialité ou la NP-complétude de DAWN-paths dans les chaînes, ni même de unstoppable-DAWN-paths.

Pareillement, nous n'avons pas encore établi la complexité de DAWN-paths lorsque la topologie est une arborescence et que les requêtes sont orientées de la racine vers les feuilles. Nous supposons la complexité de ce problème identique à celle de DAWN-paths sur les chaînes.

Troisième partie

Conception de réseaux radio multi-sauts robustes

Introduction

Nous consacrons cette troisième et dernière partie à l'étude d'un problème de ***conception*** de réseaux radio multi-sauts. Ce problème nous a été initialement proposé par une entreprise française de télécommunications. L'objectif de cette entreprise est d'assurer la distribution d'un flux internet haut-débit dans des zones rurales non couvertes en technologie ADSL. Ce flux internet est récupéré à partir de villes sources et doit être acheminé à destination de villages clients par un jeu de relais. Un village client peut également servir de relais. La technologie radio est utilisée pour établir un lien entre deux nœuds du réseau (villes, villages ou relais). De nombreuses contraintes viennent cadrer le déploiement d'un tel réseau. La difficulté du problème consiste à définir quels liens établir pour assurer une distribution efficace, et ce en respectant les contraintes posées.

Le modèle de communication considéré se démarque de celui retenu dans les deux premières parties de ce mémoire : nous n'utilisons pas d'antennes omnidirectionnelles pour établir les communications entre nœuds. Un lien est rendu possible par la mise en place d'une antenne directionnelle sur chacune de ses extrémités. La communication est possible si et seulement si la qualité du signal est suffisamment élevée. L'utilisation de cette technologie offre une meilleure robustesse aux interférences. Nous supposerons ces dernières inexistantes.

Organisation de cette partie

Cette partie se décompose comme suit :

Le chapitre 9 précise les conditions dans lesquelles sont abordées le problème de conception. Nous y présentons notamment les différentes contraintes que doit satisfaire le réseau à concevoir, ainsi qu'une modélisation formelle du problème.

Nous proposons dans le chapitre 10 une étude de complexité du problème de conception. Nous distinguons trois paramètres régissant trois principales contraintes de déploiement. Notre étude établit la difficulté du problème selon si certains de ces trois paramètres sont considérés ou non. Nous montrons notamment dans quelles mesures le problème de conception peut se ramener à des problèmes algorithmiques biens connus, et quels résultats existent alors sur ces derniers.

Nous proposons dans le chapitre 11 trois outils de résolution, dont deux méthodes optimales et une heuristique, afin de résoudre rapidement le problème de conception sur des instances de taille raisonnable. Nous comparons les résultats obtenus par ces méthodes en terme de temps d'exécution.

Chapitre 9

Conception de réseaux radio multi-sauts robustes : objectifs et contraintes

Ce chapitre introductif définit dans une première section le problème de conception d'un réseau radio multi-sauts, dont la résolution doit permettre l'acheminement d'un flux réseau vers un ensemble de destinations. Nous y présentons les contraintes de déploiement que le réseau déployé doit satisfaire. La seconde section précise la modélisation du problème que nous adoptons le long de cette partie.

9.1 Le problème de conception

L'objectif des travaux de cette partie consiste à concevoir un réseau radio assurant la distribution de bande passante depuis des nœuds sources vers des nœuds clients géographiquement proches. Cette distribution est soumise à plusieurs contraintes de divers ordres.

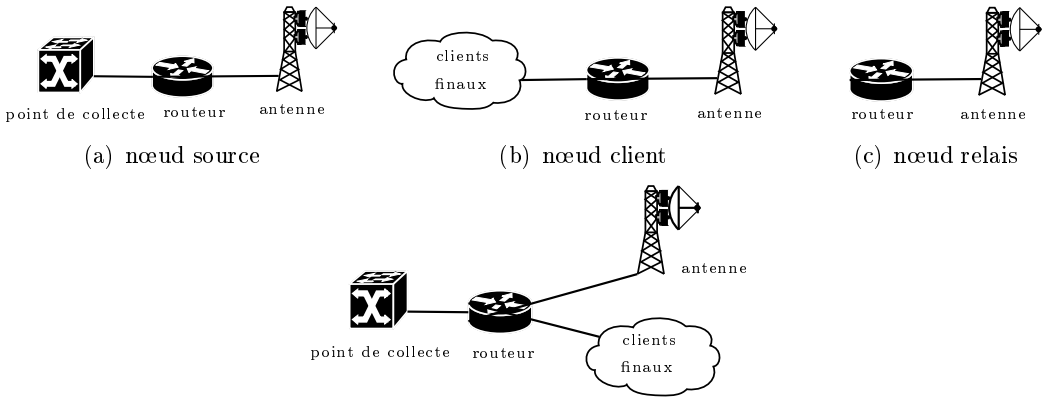
D'un point de vue fonctionnel, les nœuds du réseau représentent :
- des nœuds sources,
- des nœuds clients,
- des relais additionnels qu'il est possible de mettre en place pour assurer la distribution.

La dimension des réseaux considérés est à l'échelle pluri-urbaine, et peut s'étendre à un département ou une région :
- Les nœuds sources représentent des villes disposant d'un accès haut-débit à Internet et qu'il est possible d'exploiter. Un nœud source est donc l'association d'un routeur et d'un ***point de collecte*** fournissant un flux haut-débit.
- Les nœuds clients, ou ***points de desserte***, sont des villages pour lesquels une demande de couverture haut-débit (une bande passante) a été définie, de taille par exemple proportionnelle à la taille du village. D'un point de vue technique, ces nœuds désignent un routeur auquel est connecté un sous-réseau. Ce sous-réseau représente l'ensemble des utilisateurs qui vont bénéficier de la bande passante demandée par ce routeur. Remarquons qu'un nœud client peut également être un nœud source. Ce statut implique la connexion d'un point de collecte au routeur du nœud.

– Les nœuds ***relais*** sont des sites présentant un intérêt certain pour être envisagés comme relais potentiels dans le réseau à concevoir. Ils peuvent être des villes dont la demande de couverture est nulle, ou des points stratégiques (ex. structures ou sommets élevés, ...) permettant d'atteindre des destinations plus éloignées. L'équipement déployé sur ces sites se limite à un routeur, mais nécessite une source électrique à proximité.

Nous utilisons le média radio comme support de communication. Une communication directe entre deux nœuds du réseau nécessite sur chacun d'entre eux la mise en place d'une antenne directionnelle. Ces antennes doivent être en vision directe l'une de l'autre, et sont par conséquent placées sur des sites situés en hauteur (clochers, châteaux d'eau, ...).

La figure 9.1 présente les différents types de nœuds rencontrés. Nous limiterons notre étude au déploiement d'un réseau radio assurant la connexion inter-routeurs (Nous n'étudierons pas la distribution du flux depuis un routeur aux utilisateurs finaux qui lui sont rattachés).



(a) nœud source (b) nœud client (c) nœud relais

(d) nœud source et client

FIG. 9.1 – Types de nœuds envisagés.

Établir d'une communication directe

L'utilisation d'antennes directionnelles s'opère par le déploiement suivant :

1. Les deux antennes d'un doivent être orientées en direction l'une de l'autre, et en ligne de vue.
2. Une antenne est reliée à un routeur par une carte réseau qui sert d'interface entre ces deux entités.
3. Le fonctionnement d'une antenne est régi par un protocole issu de la norme 802.11. Une communication directe entre deux nœuds requiert que les deux antennes impliquées utilisent le même protocole.

Remarque 23 *:*
Le long de notre étude, nous limitons par souci de clarté le nombre de protocoles disponibles, et ne considérons que les normes 802.11a et 802.11b. Néanmoins l'ajout de technologies supplémentaires n'affecte en rien la définition et la résolution du problème de conception.

L'existence d'un lien radio entre deux nœuds est déterminée par la qualité du signal émis depuis l'un des nœuds vers l'autre, qui dépend notamment :

- de la puissance d'émission de la transmission,
- de facteurs environnementaux et géographiques.

Établir la qualité d'un lien entre deux sites n'est pas évident. Une méthode exacte consiste à positionner un émetteur sur un site, un récepteur sur l'autre site, et mesurer la qualité du lien. Cette mesure doit également être opérée dans l'autre sens, la mesure la plus faible est alors retenue. Cette démarche est coûteuse en temps et en moyens déployés. Une solution à coût réduit consiste à utiliser un simulateur de qualité de liens, qui prend en entrée :

- les données topologiques et géographiques d'une zone,
- les coordonnées GPS des sites sélectionnés.

Il en ressort deux matrices :

1. une matrice de qualité des liens par paire de nœuds,
2. une matrice des distances par paire de nœuds.

Un premier filtrage des liens est opéré en supprimant les liens dont la qualité de signal se situe au dessous d'un seuil critique. Lorsqu'un lien radio possède un signal suffisamment fort, il convient d'en définir alors sa capacité, c'est-à-dire le débit maximal qu'il peut faire transiter. Les principaux facteurs qui peuvent influer sur la capacité d'un lien sont :

1. le type d'antennes (matériel),
2. la distance entre les antennes,
3. le protocole de communication utilisé.

Les caractéristiques techniques d'une antenne directionnelle indiquent les débits atteignables en fonction de la distance et du protocole utilisé. Nous associons à chaque lien le protocole qui permet la plus grosse capacité, comme sur la figure 9.2.

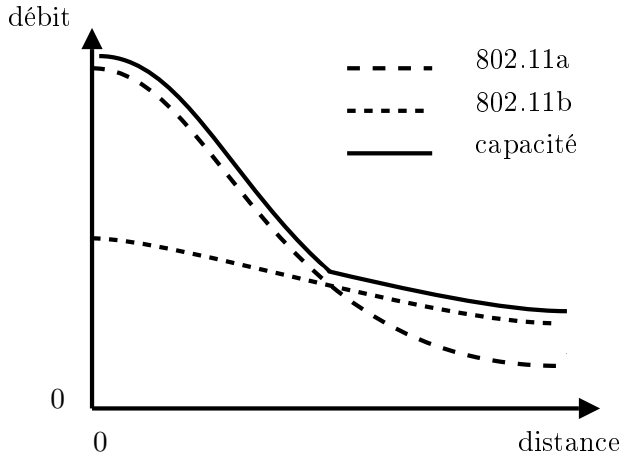


FIG. 9.2 – Capacité d'un lien en fonction de la distance et du protocole.

Remarque 24 *:*
Nous supposons ici que le coût d'un lien n'est pas lié au choix d'un protocole. Dans le cas contraire,

on peut considérer pour chaque paire de nœuds autant de liens que de protocoles disponibles, et associer à chacun de ces liens un coût en fonction de son protocole.

Pannes envisageables

Trois types de pannes peuvent intervenir :

1. une panne sur un lien radio : la liaison entre deux nœuds est interrompue. Ce cas intervient généralement lors de pannes matérielles sur une carte réseau ou une antenne, mais peut aussi avoir des raisons logicielles. Parmi les causes liées à ces pannes citons la défaillance d'une carte réseau suite à une surchauffe, ou un coup de foudre sur l'antenne correspondante. Le routeur peut cependant continuer à distribuer un flux sur le sous-réseau qu'il connecte, ou acheminer les paquets aux autres routeurs.
2. une panne sur un routeur : si une panne intervient sur le routeur d'un nœud, tous les liens assurant les communications vers les routeurs voisins deviennent hors d'usage. Aucun flux ne peut être acheminé vers le sous-réseau qu'il connecte. Ce type de panne intervient généralement si un impact de foudre sur une antenne s'est propagé à tout le routeur, ou en cas de coupure d'électricité.
3. une panne sur un point d'accès : en cas de panne sur un point d'accès, le routeur reste connecté au réseau et à son sous-réseau, mais perd sa capacité à délivrer un flux. Il se comporte comme un simple relai. Lorsque le nœud concerné est également un nœud client, le sous-réseau correspondant ne reçoit plus le flux associé au routeur du nœud.

Contraintes de déploiement

Le déploiement du réseau est soumis à plusieurs contraintes de divers ordres :

- ***Contraintes de demande*** : chaque nœud client exige un débit spécifique en bande passante, qui doit lui être assuré dans des conditions normales de fonctionnement.
- ***Contraintes de capacité des liens*** : le flux circulant sur un lien est borné par la capacité de ce dernier.
- ***Contraintes de degré*** : le degré d'un nœud du réseau est égal au nombre d'antennes directionnelles présentes sur ce dernier. La charge de travail d'un routeur est proportionnelle au nombre d'antennes qu'il commande. Ainsi un nombre élevé d'antennes peut entraîner des difficultés de fonctionnement (temps de traitement de paquets élevés, surchauffe du routeur, ...). De plus, le nombre de slots disponibles pour les cartes réseau est borné sur chaque routeur. Le nombre d'antennes associées au même routeur ne doit donc pas excéder une certaine valeur.
- ***Contraintes de nombre de sauts*** : la distribution d'un nœud client est assurée par un ou plusieurs nœuds sources. Pour des raisons de temps d'accès et de réponse, le nombre de sauts entre entre un nœud client et la (les) source(s) dont il dépend ne peut excéder une certaine valeur.
- ***Contraintes de robustesse*** : Dans la mesure du possible, le réseau doit être résistant aux pannes, c'est-à-dire assurer une bande passante non nulle à tous les nœuds clients si une anomalie survient, sur un nœud, un lien. Quelle que soit la localisation de la panne (routeur, lien, ou point d'accès), les nœuds du réseau [1] doivent restés connectés à au moins un nœud

[1] à l'exception du nœud sur lequel intervient une panne

source, tout en respectant les contraintes de nombre de sauts énoncées précédemment. Nous nous autorisons en cas de panne à dégrader le débit reçu par certains nœuds clients, afin d'assurer une connexion avec l'extérieur à tous les nœuds.

Le déploiement d'un tel réseau possède un coût, que nous associons principalement à l'achat et l'installation d'une antenne directionnelle. Aussi est-il pertinent dans un tel déploiement de minimiser le nombre d'antennes utilisées.

Nous appelons ***NetworkDesign*** le problème consistant à définir un réseau entre les routeurs des nœuds qui répond à toutes les contraintes énoncées ci-avant. La définition formelle de ce problème est donnée en fin de section 9.2.

9.2 Modélisation

La section suivante propose la formalisation du problème NetworkDesign. La première sous-section présente les éléments d'une instance de NetworkDesign. La seconde sous-section introduit le vocabulaire inhérent au problème, et se conclut par la définition formelle de NetworkDesign.

9.2.1 Modélisation de la donnée

Nous modélisons l'ensemble des informations nécessaires à la conception du réseau comme suit :
- Nous modélisons la donnée par un graphe non orienté $G = (V, E)$, où V désigne l'ensemble des nœuds du réseau, et E l'ensemble des liens déployables de signal suffisament grand.
- Une fonction *capa* associe à chaque arête $e \in E$ un réel dans R^{*+} correspondant à sa capacité.
- Nous nommons $S \subseteq V$ et $T \subseteq V$ les ensembles de sommets correspondant respectivement aux nœuds sources et aux nœuds clients.
- Une fonction *dem* associe pour chaque élément $t \in T$ un réel dans R^{*+} désignant la demande de bande passante de t.

Répondre au problème NetworkDesign consiste à trouver un graphe H, sous-graphe partiel de G, et répondant aux contraintes présentées précédemment. Nous proposons dans la sous-section suivante une définition formelle de ce problème. Parmi les contraintes de conception énoncées, le degré des nœuds et le nombre de sauts entre une source et un client doivent être bornés. De plus le réseau doit supporter un certain nombre de pannes simultanées. Nous notons respectivement *deg*, *hop* et r le degré maximal, le nombre de sauts maximum, et le nombre de pannes simultanées que peut supporter le réseau.

9.2.2 formulation du problème

Le vocabulaire suivant précède la définition du problème :

Définition 47 (r-robustesse) *:*
Soient un graphe $G = (V, E)$, un sous-ensemble $S \subseteq V$, un entier naturel dist ≥ 1, et un sommet $x \in V$.

*Un sommet $x \notin S$ est r-**robuste** dans (G, S, d), si et seulement s'il existe $r + 1$ chaînes $\{C_1, C_2, \ldots C_{r+1}\}$, $C_i \subseteq G$, telles que :*
- *les extrémités de C_i sont x et s_i, avec $s_i \in S$,*

- $V(C_i) \cap V(C_j) = \{x\},\ \forall 1 \leq i < j \leq r+1$.

Réciproquement, nous disons que l'ensemble de chaînes $\{C_1, C_2, \ldots C_{r+1}\}$, $C_i \subseteq G$, ***assure la r-robustesse*** *de* $x \notin S$ *dans* (G, S).

Si x *est inclus dans* S, x *est* r*-robuste dans* (G, S) *si et seulement s'il existe* r *chaînes avec les conditions énoncées précédemment, puisque* x *est déjà connecté à un point de collecte.*

Nous disons qu'un ensemble de sommets T *est* r*-robuste dans* (G, S) *si chacun de ses sommets* $x \in T$ *est* r*-robuste dans* (G, S). *Réciproquement, nous disons qu'un ensemble de chaînes assure la* r*-robustesse de* T *dans* (G, S) *si et seulement s'il assure la* r*-robustesse de chacun des éléments de* T.

Remarque 25 *:*
La notion de résistance aux pannes est ici portée sur les nœuds du réseau, et se veut logiquement plus forte que la résistance aux pannes sur les liens. Le modèle de robustesse proposé et l'ensemble de nos travaux peuvent se décliner aisément pour ne considérer que des pannes intervenant sur les liens.

Dans le problème de conception, l'ensemble des nœuds clients T doit donc être r-robuste dans le réseau déployé.

Le terme de flot est couramment employé dans la terminologie des réseaux pour décrire la quantité de matière transitant sur chacun des liens du réseau. Les problèmes de flots cherchent généralement à satisfaire des demande de trafic entre paires de nœuds selon des contraintes liées aux capacité des liens. Nous proposons ici une définition d'un flot chaînes-contraint :

Définition 48 (flot chaînes-contraint) *:*
Soient un graphe non orienté $G = (V, E)$, *une valuation des arêtes* $capa : E \rightarrow R^{*+}$, *un ensemble* $S \subseteq V$ *de nœuds sources, un ensemble* $T \subseteq V$ *de nœuds clients, et une fonction* $dem : T \rightarrow R^{*+}$.

Considérons alors un ensemble de chaînes C *tel que* $\forall c \in C, ext(c) = \{s_c, t_c\} | s_c \in S, t_c \in T$. *Appelons* $P_C(e)$ *l'ensemble des chaînes de* C *contenant l'arête* e.

Un ***flot chaînes-contraint*** *valide est une fonction* $f : C \rightarrow Z+$ *qui associe à chaque chaîne* c *de* C *un réel positif ou nul tel que, en notant* $h(e) = \sum_{c \in P_C(e)} f(c)$ *avec* $e \in E$ *:*
- $h(e) \leq capa(e), \forall e \in E$
- $\sum_{c \in C | t \in ext(c)} f(c) \geq dem(t), \forall t \in T$

Nous formalisons NetworkDesign par le problème de décision 9.2.1.

Clairement, le problème NetworkDesign recherche à partir d'un réseau donné un sous-réseau dans lequel :
- les nœuds sont de degré borné,
- en fonctionnement normal, les demandes de nœuds clients sont satisfaites par des nœuds sources situés à un nombre de sauts borné,
- le sous-réseau peut supporter un nombre donné de pannes.

Donnée :
- un graphe non orienté $G = (V, E)$,
- une fonction $capa : E \rightarrow R^{*+}$,
- deux sous-ensembles S et T de V,
- une fonction $dem : T \rightarrow R^{*+}$,
- les entiers naturels deg, hop, r, k.

Question : existe-t-il un ensemble de chaînes C de G avec $ext(c) = \{s_c, t_c\} | s_c \in S, t_c \in T, \forall c \in C$, et un flot chaînes-contraint f sur C tel que, en notant $H = (V_H, E_H)$ le graphe pour lequel lequel $V_H = \bigcup_{c \in C} V(c)$ et $E_H = \bigcup_{c \in C} E(c)$, on a :

1. $|N_H(x)| \leq deg, \quad \forall x \in V(H)$,
2. $\forall c \in C, |E(C)| \leq hop$,
3. f est valide pour l'instance $(H, capa, S, T, dem, C)$,
4. l'ensemble des chaînes C assure la r-robustesse de T dans (H, S),
5. $|E_H| \leq k$.

Problème de décision 9.2.1: le problème NetworkDesign

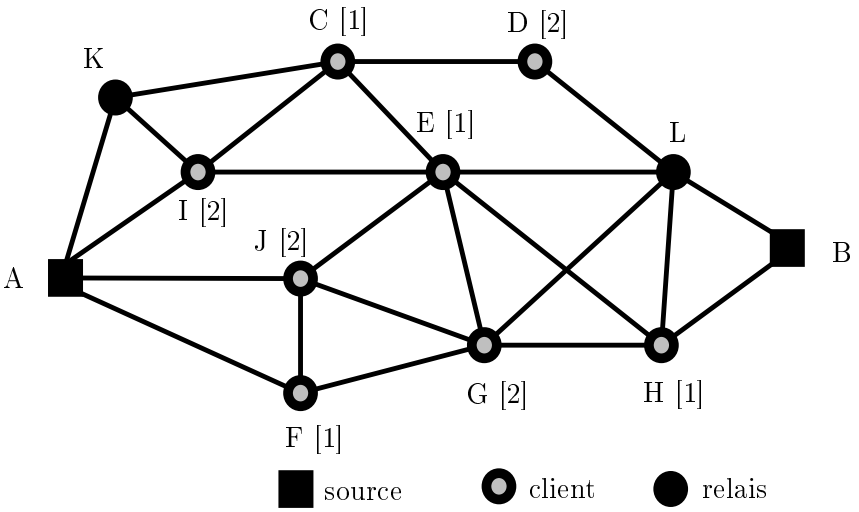
Une instance de NetworkDesign est a priori décrite par un 9-uplet (G, $capa$, S,T,dem, k, deg, hop, r). Nous cherchons une topologie dont le nombre de liens est borné par une constante k. La version optimisation de ce problème, appelée min-NetworkDesign, consiste à trouver le sous-réseau ayant un nombre minimum de liens. La donnée k est alors supprimée de la donnée.

Remarquons que la donnée des nœuds sources, des nœuds clients, des demandes qui leur sont associées, des liens et de leur capacité sont des données de terrain. À l'opposé, les contraintes de robustesse, de distance et de degré relèvent plutôt de choix techniques et de politiques de déploiement. Par souci de clarté, nous utiliserons parfois une notation consistant à extraire les paramètres techniques deg, hop et r de la donnée du problème. Ainsi le problème [deg,hop,r]-NetworkDesign réfère à une instance de NetworkDesign dans laquelle le degré maximum est borné par deg, le nombre de sauts entre une source et un client est d'au plus hop, et le réseau déployé doit être r-robuste. La donnée du problème est réduite à un 6-uplet (G, $capa$, S,T,dem, k).

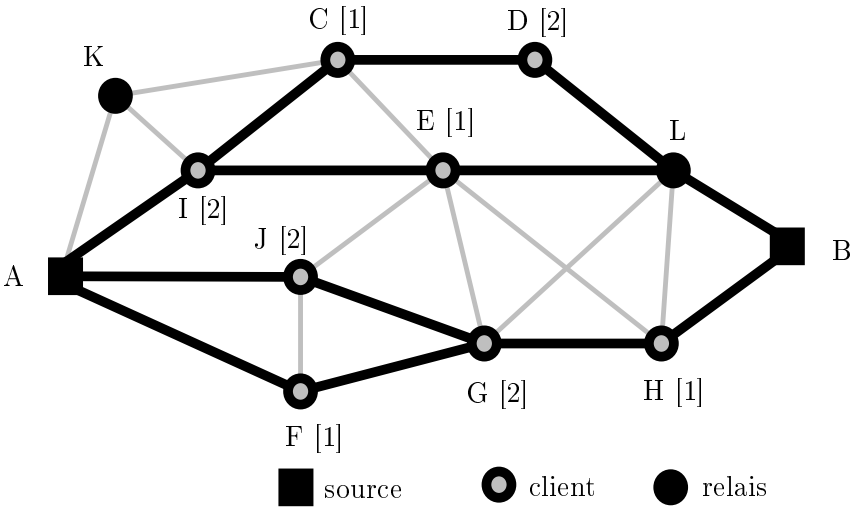
La figure 9.3(a) présente une instance de [3,3,1]-min-NetworkDesign. Les sommets A et F sont des sommets sources. Les sommets B à F sont des sommets clients. La demande de chacun des sommets est indiquée entre crochets. La capacité de chaque arête est ici fixée à 3. La figure 9.3(b) présente une solution de coût 13. Le tableau 9.1 détaille la solution, en énumérant pour chaque client la ou les chaînes assurant le transport du flot, et les chaînes de secours utilisées en cas de panne.

Nous terminons cette section par l'énoncé de la remarque suivante :

Remarque 26 *:*
Dans le problème tel que nous avons énoncé, les sommets du graphe résultat ont les mêmes contraintes de degré, de robustesse, et de distance. En adaptant notre formulation, nous pourrions définir en résultat un graphe dans lequel les paramètres critiques sont propres à chaque nœud. En effet :



(a) Une instance ayant 2 sources et 8 clients



(b) Une solution de coût 13

FIG. 9.3 – Une instance de [3,3,1]-min-NetworkDesign et sa solution

client	*chaînes [flot]*	*secours*	*client*	*chaînes [flot]*	*secours*
C	A-I-C [1]	B-L-D-C	G	B-H-G [2]	A-K-G
D	B-L-D [2]	A-I-C-D	H	B-H [1]	A-F-G
E	B-L-E [1]	A-I-E	I	A-I [2]	B-L-E-I
F	A-F [1]	B-H-G-F	J	A-J [2]	B-H-G-K

TAB. 9.1 – Listes des chaînes assurant la distribution du flot et la robustesse par sommet.

- *Au niveau du degré, certains sites peuvent accueillir plus d'antennes que d'autres.*
- *Au niveau de la robustesse, on peut souhaiter que certains clients soient plus robustes que d'autres si la proportion d'utilisateurs finaux est supérieure.*
- *Enfin, augmenter la distance maximum entre une source et un client précis peut amener de nouvelles solutions moins coûteuses en nombre de liens.*

Nous mentionnons l'existence de ces variantes qui peuvent se formaliser aisément par le vocabulaire que nous avons introduit. Cependant, nous restreindrons notre étude dans ce mémoire au problème présentant des contraintes identiques pour tous les nœuds.

Chapitre 10

Étude de complexité

Nous proposons une étude complexité du problème min-NetworkDesign en fonction des valeurs des paramètres de degré, nombre de sauts et robustesse, respectivement désignées par les valeurs deg, hop et r. Ce chapitre regroupe à la fois un état de l'art sur des problèmes similaires, et instaure de nouvelles preuves de complexité. Par la suite, lorsqu'une contrainte est ignorée, nous lui attribuons une valeur correspondant à cet état : les affectations $deg = \infty$ et $hop = \infty$ impliquent respectivement que le degré maximum autorisé et le nombre maximum de sauts peuvent être arbitrairement grands (de l'ordre de n), et l'affectation $r = 0$ que le réseau ne tolère aucune panne.

Organiser les nombreux problèmes résultant des variations de ces paramètres n'est pas chose facile. Aussi nous découperons notre étude en trois sections. La première section décrit quelques résultats sur le problème min-NetworkDesign lorsqu'aucun des trois types de contraintes (degré, nombre de sauts, robustesse) n'est considéré. Les sections 2, et 3 s'intéressent aux problèmes pour lesquels respectivement 1 et 2 types de contraintes sont envisagés.

10.1 NetworkDesign sans contrainte

Le problème [∞,∞,0]-NetworkDesign décrit exactement le problème du EDGE-COST FLOW (ECF), connu pour être NP-complet [GJ79]. Il reste NP-complet même sous l'hypothèse que les capacités des arêtes sont infinies. On rencontre parfois ce dernier problème sous le nom INFINITE CAPACITY EDGE-COST FLOW (ICECF) [EKS05].

Les deux propositions suivantes illustrent l'impact de la cardinalité des ensembles sources et clients sur la complexité de ces deux familles de problèmes.

Proposition 2 *:*

1. *Les deux précédents problèmes restent NP-complets, lorsque le nombre de sources est de* 1.
2. *Lorsque le nombre de clients est de* 1*, ECF reste NP-complet.*
3. *En revanche, lorsque le nombre de clients est de* 1*, ICECF devient polynomial.*

Preuve :

Soit une instance I de ECF contenant (entre autres) un ensemble de sommets sources S dans un graphe $G = (V, E)$. À partir de I, on peut construire polynomialement une instance I' du même

problème dans laquelle le nombre de sources est de 1, en copiant I dans I' et en ***fusionnant***[1] les sommets de S dans G. Une solution à I de coût k induit alors une solution a I de coût équivalent, et réciproquement.

Lorsque les arêtes sont de capacité infinies, et qu'il n'y a qu'une seule source, ce problème n'est autre que celui du STEINER TREE (StT), connu pour être NP-complet. Ceci conclut la preuve du point 1 de la proposition 2.

Soit une instance I de ECF contenant (entre autres) un ensemble de clients $C = \{c_1, c_2, \ldots, c_m\}$ dans un graphe $G = (V, E)$. On peut construire polynomialement une instance I' du même problème comme suit :

- I' est initialisé par copie de I.
- Nous ajoutons à G un sommet c, et les arêtes $\{c_i, c\}|c_i \in C$.
- La capacité de $\{c_i, c\}$ est égale à $dem(c_i)$.
- Le sommet c est désigné comme seul client, avec une demande $dem(c) = \sum_{c_i \in C} dem(c_i)$.

Par la suite, nous nommerons cette construction l'***ajout d'un super client***.

S'il existe une solution à I' de coût k, alors elle mobilise chacune des arêtes $\{c_i, c\}|c_i \in C$ à pleine capacité. On peut en déduire une solution à I de coût $k - |C|$. Réciproquement, toute solution de I de coût k implique une solution dans I' de coût $k + |C|$. La complexité du problème reste inchangée, comme l'indique le point 2 de la proposition 2.

Nous terminons par la justification du point 3 : lorsque les capacités des arêtes sont infinies et qu'il n'y a qu'un seul client c, il suffit de déterminer le plus court chemin entre s et une source quelconque pour obtenir un ensemble d'arêtes permettant le transit du flux. Cette opération est connue pour s'exécuter avec une complexité en $O(n^2)$. □

La preuve de la proposition 3 utilise une réduction vers le problème BIPARTITION. Ce problème consiste à se demander s'il existe une bipartition $\{N_A, N_B\}$ d'un ensemble S d'entiers, telle que la somme des éléments de N_1 égale celle des éléments de N_2. Ce problème est connu pour être NP-complet.

Proposition 3 *:*
Lorsque tous les nœuds du réseau sont des clients :

1. *ECF reste NP-complet.*
2. *ICECF est polynomial.*

Preuve :

Nous prouvons le point 1 de la remarque par une réduction au problème BIPARTITION.

Le problème ECF est dans NP : pour toute solution potentielle de coût k décrite par un ensemble d'arêtes E_S, on peut vérifier en un temps polynomial que le flot est réalisable sur le graphe partiel induit par les arêtes de E_S. Nous construisons à partir de toute instance $N = \{n_1, n_2, n_3, \ldots n_m\}$ du problème BIPARTITION une instance $I_{ECF} = (G, capa, S, T, dem, k)$ de ECF comme suit : Le

[1] L'opération de fusion d'un ensemble de sommets $S = \{s_1, s_2 \ldots s_k\}$ dans un graphe $G = (V, E)$ consiste à supprimer dans V les sommets de S, et à ajouter un nouveau sommet s. Chaque arête du type $\{a, s_i\}|a \in V - S, s_i \in S$ est supprimée de E, et est remplacée par l'arête $\{a, s\}$. Les arêtes $\{s_i, s_j\}|s_i, s_j \in S$ sont simplement supprimées.

graphe $G = (V, E)$ est défini par les ensembles :

$$V = \{s\} \cup N \cup \{N_1, N_2\}$$

et

$$E = \big\{\{s, n_i\}, \{n_i, N_1\}, \{n_i, N_2\} | n_i \in N\big\}$$

Pour tout $n_i \in N$, on pose $capa(\{s, n_i\}) = 2 \times n_i$, $capa(\{n_i, N_1\}) = n_i$ et $capa(\{n_i, N_2\}) = n_i$. L'instance ne comporte qu'une source $S = \{s\}$, et $T = V$. La demande de chaque nœud n_i est égale à n_i. Nous posons également $dem(N_1) = dem(N_2) = \frac{1}{2} \times \sum_{n_i \in N} n_i$. Il convient dans notre réduction que tous les sommets soient des clients. Aussi la demande du nœud source s doit être non nulle, mais sa valeur n'est pas pertinente puisqu'auto-satisfaite. Enfin nous posons $k = 2 \times |N|$.

Nous affirmons que s'il existe une solution S à l'instance I_{ECF}, alors elle est nécessairement de coût égal à k et permet de déduire une solution à l'instance N du problème BIPARTITION. Le coût de S est ici le nombre d'arêtes de G transitant un flot de valeur non nulle. Dans toute solution réalisable, chacune des arêtes de la forme $\{s, n_i\}$ doit obligatoirement faire transiter un flot égal à sa capacité. Ces arêtes sont au nombre de $|N|$. Pour chaque arête $\{s, n_i\}$ et sur les $2 \times n_i$ unités de flot qui transitent par elle, la moitié est consommée par le sommet n_i. S'il est possible de satisfaire les demandes de N_1 et N_2 avec $k = 2 \times N$ arêtes, alors le flot non-consommé par chaque sommet n_i ne se divise pas, et est intégralement routé en direction de N_1 ou N_2. On construit alors une bipartition $N = \{N_A, N_B\}$ en ajoutant à N_A les éléments n_i de N pour lesquels l'arête $\{n_i, N_1\}$ fait transiter un flot non nul (précisément de n_1 unités), et à N_B sinon. La réciproque est immédiate.

Le point 2 de la proposition 3 est immédiat. Soit une instance I de ICECF. S'il n'y a qu'une source s, n'importe quel arbre couvrant le graphe en entrée décrit une solution pour I. S'il y a plusieurs sources, nous les fusionnons pareillement à la preuve du point 1 de la proposition 2. Nous obtenons une instance I' dont la résolution en temps polynomial donne une solution de même coût pour I.

□

10.2 NetworkDesign avec un seul type de contraintes

Nous étudions la complexité de min-NetworkDesign lorsque seulement un type de contraintes est envisagé. Nous déterminons lorsqu'elle existe la limite entre la polynomialité et la NP-complétude du problème en fonction de la valeur du paramètre régissant la contrainte.

10.2.1 Seul le degré est borné

Le problème [*deg*,∞,0]-min-NetworkDesign est NP-difficile en général, puisqu'il constitue une généralisation du problème ECF présenté précédemment.

La première remarque énonce quelques résultats pour des instances particulières. La seconde remarque conclut sur la limite entre la polynomialité et la NP-complétude de [*deg*,∞,0]-NetworkDesign en fonction de la valeur de *deg*.

Proposition 4 *:*
Considérons le problème [deg,∞,0]-min-NetworkDesign réduit aux instances où les arêtes sont de capacités infinies, et le nombre de sources de 1*. Alors :*

1. *ce problème est NP-difficile,*
2. *il reste NP-difficile même en supposant que tous les nœuds sont des clients,*
3. *il est polynomial s'il n'y a qu'un client.*
4. *les trois premiers points restent exacts lorsque l'instance possède plusieurs sources.*

Preuve :

1. Le problème est exactement le problème DEGREE-CONSTRAINED MINIMUM STEINER TREE (DCMStT) [BV95, Bau96], qui consiste à trouver un arbre de Steiner avec un nombre minimum d'arêtes et dont le degré des nœuds est borné. Ce problème est connu pour être NP-difficile.
2. Lorsque tous les nœuds sont des clients, l'objectif consiste à trouver un arbre couvrant de degré borné par deg. Déterminer l'existence d'un tel arbre correspond au problème DEGREE-CONSTRAINED SPANNING TREE (DCSpT) [GJ79], connu pour être NP-complet pour tout $deg \geq 2$ (voir preuve de la proposition 5).
3. Lorsqu'il n'y a qu'un seul client, il suffit de rechercher un plus court chemin de la source vers le client.
4. S'il y a plusieurs sources dans une instance I, nous les fusionnons pareillement à la preuve du point 1 de la proposition 2. Nous obtenons une instance I' construite en temps polynomial dont la résolution donne une solution de même coût sur I.

□

Proposition 5 *:*
Le problème de décision [deg,∞,0]-NetworkDesign est :
- *polynomial lorsque* $deg = 1$,
- *NP-complet pour tout* $deg \geq 2$.

Preuve :
Soit $I = (G, capa, S, T, dem, k)$ une instance de [deg,∞,0]-NetworkDesign.

Lorsque $deg = 1$, le problème consiste à trouver pour chaque client c_i une source dédiée s_j dans son voisinage, telle que la capacité de l'arête $\{c_i, s_j\}$ soit supérieure à la demande $dem(c_i)$. Sa résolution consiste à trouver un couplage saturant T dans le graphe biparti $H = (S, T, \{x, y\} | x \in S, y \in T, \{x, y\} \in E(G), capa(\{x, y\}) \geq dem(y))$.

Comme annoncé dans la preuve de la proposition 4 et dans les instances avec une seule source, où tous les nœuds sont des clients et les arêtes sont de capacité infinie, le problème est exactement celui du DCSpT. En fait lorsque $deg = 2$, il s'agit du problème HAMILTONIAN PATH (HP).

Il existe également une réduction vers problème HP qui prouve la NP-complétude pour toute valeur $deg \geq 2$: depuis G une instance de HP. On construit une instance $(H, capa, S, T, dem, k)$ de [deg,∞,0]-NetworkDesign telle que :
- $V(H) = V(G) \cup \{(x, i)\}_{x \in V(G), i \in [1, deg-2]}$ et $E(H) = E(G) \cup \{\{x, (x, i)\}\}_{x \in V(G), i \in [1, deg-2]}$,
- $S = \{s\}$, où s est un sommet quelconque de $V(G)$, et $T = \{(x, i)\}_{x \in V(G), i \in [1, deg-2]}$,

- $dem(t) = 1 | t \in T, capa(e) = \infty | e \in E(H)$,
- $k \geq |V(H)|$.

S'il existe une solution sur l'instance $(H, S, T, dem, capa, k)$, alors l'ensemble des arêtes de la forme $\big\{\{x,(x,i)\}|x \in V(G), i \in [1, deg-2]\big\}$ appartient à la solution. Le degré des sommets de H ne peut excéder deg dans la solution. Les sommets de $V(G)$ sont alors connectés entre eux dans la solution par un ensemble d'arêtes formant un chemin hamiltonien sur G.

□

10.2.2 Seul le nombre de sauts est borné

Le problème [∞,*hop*,0]-NetworkDesign s'intéresse à la recherche d'un flot satisfaisant un ensemble de demandes, sous la contrainte que le flot émis depuis chaque source n'effectue pas plus de *hop* sauts pour atteindre sa destination. L'objectif reste la minimisation du nombre d'arêtes mobilisées.

Le théorème 25 regroupe les principaux résultats de complexité de ce problème :

Théorème 25 *:*
Le problème [∞,hop,0]-NetworkDesign est :

1. *polynomial lorsque* $hop = 1$,
2. *NP-complet pour tout* $hop \geq 2$.

Preuve :

Posons $hop = 1$. Soit $I = (G, capa, S, T, dem, k)$ une instance du problème [∞,*hop*,0]-NetworkDesign.

Lorsque le nombre de sauts autorisé *hop* est d'au plus 1, une solution optimale en nombre d'arêtes se calcule polynomialement comme suit : pour chaque client $t \in T$, nous sélectionnons un ensemble E_t d'arêtes incidentes à t et à n'importe quel sommet source assurant à t la distribution du flot. L'ensemble E_t de cardinalité minimale se calcule en sélectionnant d'abord les arêtes $\{t, s\}|s \in S$ de plus grande capacité, et en les ajoutant successivement à E_t tant que la somme des capacités des arêtes de E_t reste inférieure à la demande de t. Les arêtes du graphe solution sont alors l'union des $E_t|t \in T$. La distribution du flot est évidente : il suffit de vérifier que la solution possède un coût inférieur ou égal à k. Ceci conclut la démonstration du point 1 du théorème 25[2].

Lorsque $hop \geq 2$ nous proposons une réduction au problème NP-complet SET-COVER (SC) :

Le problème [∞,*hop*,0]-NetworkDesign est dans NP : soit $(G, capa, S, T, dem, k)$ une instance de ce problème. Étant donné un ensemble de chaînes C de taille au plus *hop* et d'un flot chaînes-contraint $f : C \to R^+$, il est possible de vérifier en un temps polynomial que (C, f) est bien une solution pour l'intance $(G, capa, S, T, dem, k)$.

Soit une instance $I_{SC} = (X, T, k)$ une instance de SC, où T est un ensemble de n éléments $T = \{t_1, t_2, \ldots, t_n\}$, et $X = \{X_1, X_2, \ldots, X_m\}$ est composé de m sous-ensembles de T. Nous construisons depuis I_{SC} une instance I de [∞,*hop*,0]-NetworkDesign comme suit :

Nous définissons le graphe $G = (V, E)$ tel que :

$$V = \big\{X_i\big\}_{0 \leq i \leq m} \cup \big\{s_i\big\}_{0 \leq i \leq hop-2}$$

[2]Cet algorithme est repris de manière plus détaillée dans la preuve du théorème 29, avec l'ajout de la contrainte de robustesse.

et

$$E = \left\{\{X_i, t_j\} | t_j \in X_i\right\}_{0 \leq i \leq m, 0 \leq j \leq n} \cup \left\{s_i, s_{i+1}\right\}_{0 \leq i \leq hop-3}$$

Nous posons $S = \{s_0\}$, $dem : T \to R^+$ une fonction quelconque, et $capa : E \to \infty$ la fonction qui associe une capacité infinie aux arêtes de G. Soit $I = (G, capa, S, T, dem, k)$ une instance de [∞,*hop*,0]-NetworkDesign. La construction de I est clairement polynomiale en fonction de la taille de I_{SC}. La figure 10.1(b) présente l'instance I construite à partir de l'instance de SC de la figure 10.1(a), avec $hop = 4$.

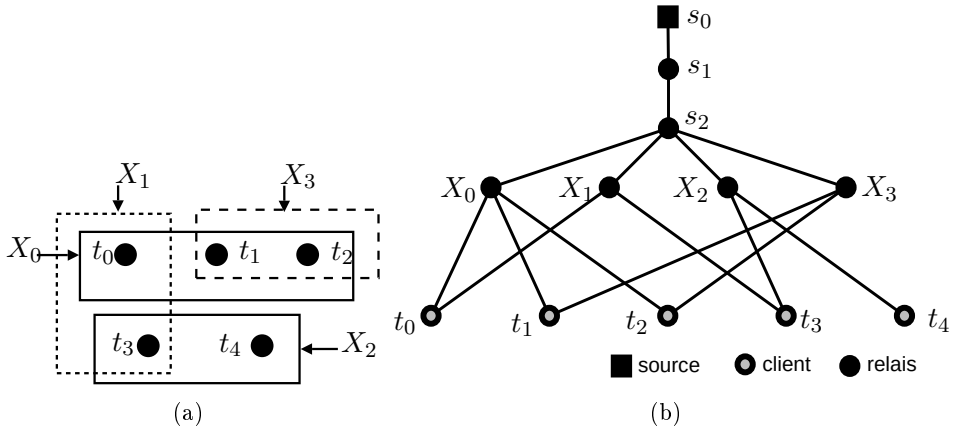


FIG. 10.1 – Construction d'une instance de [∞,*hop*,0]-NetworkDesign depuis une instance de SET-COVER et un entier $hop = 4$

Considérons une solution (C, f) de coût $a \leq k$ sur l'instance I, où C est l'ensemble des chaînes c reliant s_0 clients de T, et telles que $f(c) > 0$. La construction proposée implique que toute chaîne $c \in C$ est de taille *hop*. Appelons U l'ensemble des arêtes utilisées par les chaînes de C. Par définition $|U| = a$, il y a au moins n arêtes incidentes aux sommets de T, et au plus $a - n$ arêtes relient s_0 aux éléments de X. Soit $W \subseteq U$ l'ensemble des arêtes de la forme $\{s_{hop-2}, X_i\} | X_i \in X$. Alors $|W| \leq a - n - (hop - 2)$. En sélectionnant les éléments de X incidents à une arête de W, nous obtenons une solution à l'instance I_{SC} de coût au plus $a - n - (hop - 2)$.

Réciproquement, si $Y \subseteq X$ est une solution à I_{SC} de coût b, alors on peut en déduire une solution de coût égal à $b + n + (hop - 2)$ à I : nous posons U l'ensemble des arêtes $\{s, X_i\} | X_i \in Y$, et pour chaque client t_i nous ajoutons à U une arête $\{t_i, X_j\} | X_j \in Y$. Pour chaque client, il existe donc une chaîne de taille *hop* reliant tout client à la source, et que nous sélectionnons dans notre solution. Les capacités des arêtes sont infinies, aussi un seule chaîne suffit pour satisfaire une demande. Les arêtes de la forme $\{s_i, s_{i+1}\} | 0 \leq i \leq hop - 3$ appartiennent nécessairement à la solution. La cardinalité de U est clairement de $b + n + (hop - 2)$.

Rappelons enfin que SC est NP-complet en général, ce qui conclut la démonstration du point 2. □

Corollaire 12 *:*
La réduction de la preuve du théorème 25 conserve le rapport d'approximation entre les solutions.

La version optimisation de SET-COVER, nommée MINIMUM-SET-COVER, est connue pour être NP-difficile et non approximable à un facteur constant près. Ainsi [∞,hop,0]-min-NetworkDesign est un problème NP-difficile et non approximable à un facteur constant près.

Corollaire 13 *:*
Le problème [∞,hop,0]-NetworkDesign reste NP-complet pour tout hop ≥ 3*, même lorsque l'instance ne possède qu'une source et une destination.*

Preuve :

Nous reprenons l'instance I construite dans la démonstration du point 2 du théorème précédent. Nous ajoutons un super client, par une construction déjà présentée dans la preuve de la proposition 2. Nous obtenons une instance I' n'ayant qu'une source s_0 et un client t, et dans laquelle la distance entre ces deux nœuds a été augmentée de 1. Nous affirmons que résoudre [∞,$hop+1$,0]-NetworkDesign sur l'instance I' est aussi difficile que résoudre [∞,hop,0]-NetworkDesign sur I : s'il existe une solution S' à I' de coût k, alors elle mobilise chacune des $|T|$ arêtes ajoutées lors de l'ajout du super client, et ce à pleine capacité. On peut en déduire une solution à I de coût $k-|T|$ en sélectionnant les arêtes de S' non incidentes au super client. La réciproque est immédiate. On en conclut la NP-complétude du problème lorsque le nombre de sauts maximum est 3, et que l'instance ne contient qu'un client.

La figure 10.2.2 présente l'ajout d'un super client t à partir de l'instance de la figure 10.1(b). Les arêtes $\{c, t_i\}$ sont de capacité égale à la demande de t_i dans I.

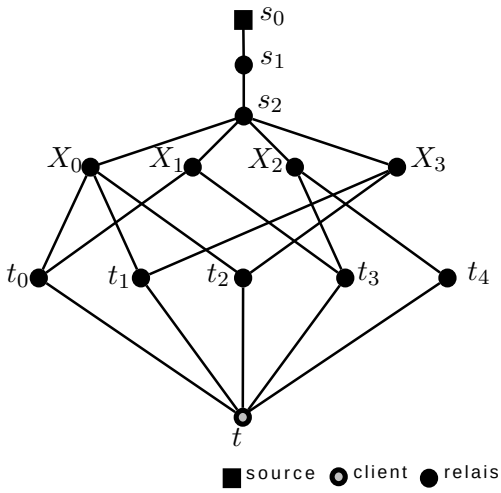


FIG. 10.2 – Ajout d'un super client t à la construction de la figure 10.1(b)

□

La preuve du dernier théorème annonce que le problème est NP-complet pour $hop \geq 2$ même lorsque l'instance ne comporte qu'une source. Le corollaire de ce théorème prouve la NP-complétude pour $hop \geq 3$ lorsqu'il n'y a qu'un client. La construction présentée dans la preuve du point 1 de la proposition 3 permet d'établir que le problème reste NP-complet lorsque tous les nœuds sont des clients. Signalons enfin que lorsque les arêtes sont de capacité infinie, et que tous les nœuds sont

clients, le problème est exactement celui du MINIMUM DIAMETER SPANNING TREE (MDSpT), connu pour être polynomial lorsque le coût des arêtes est uniforme.

10.2.3 Seule la robustesse est imposée.

Même lorsque $r = 0$, le problème $[\infty,\infty,r]$-min-NetworkDesign est NP-difficile, puisqu'il décrit exactement le problème min-NetworkDesign sans contraintes, déjà étudié en 10.1. La variation de ce paramètre n'a qu'une influence limitée sur la complexité du problème. Le nombre de sources (au moins égal à la robustesse du réseau) et de clients ne change en rien la complexité du problème, puisque la preuve de NP-complétude présentée dans la section 10.1 s'applique pour des instances ne comportant qu'une source et un client.

Nous citerons quelques résultats connus lorsque la capacité des arêtes est infinie. En pareil cas, une solution consiste à définir un sous-graphe du graphe de départ dans lequel chaque client peut être connecté à l'aide de $r + 1$ chaînes sommet-disjoints à $r + 1$ sources différentes. Une solution optimale requiert alors un nombre d'arêtes minimal. Le problème de l'existence même d'une solution (sans qu'elle soit nécessairement optimale en nombre d'arêtes) s'apparente un peu à celui du UNDIRECTED VERTEX-DISJOINT PATHS (UVDP) consistant à se demander s'il existe des chaînes sommets-disjoints entre k paires de nœuds sources et clients dans un graphe non orienté. Le problème a été montré NP-complet [Kar75] en général, mais polynomial lorsque k est borné [RS95] ou si chaque chemin recherché possède une extrémité commune. Nous mentionnerons enfin le théorème énoncé par Menger en 1927, qui a ouvert la voie à de nombreux travaux sur la connexité :

Définition 49 (un vw-séparateur) *:*
Soient v et w deux sommets d'un graphe G.
Un vw-séparateur de G est un ensemble S de sommets distincts de v et w tel que toute chaîne entre v et w passe par un des sommets de S.

Théorème 26 (Menger, 1927) *:*
Le nombre maximal de chaînes sommets-disjoints connectant deux sommets distincts non adjacents v et w est égal à la taille minimum d'un vw-séparateur.

Par une réduction appropriée, ces résultats permettent de définir l'existence ou non de chaînes disjointes dans un graphe entre un client et plusieurs sources pour une robustesse donnée r. Aucun ne détermine en revanche la solution mobilisant un nombre minimum d'arêtes.

10.3 NetworkDesign avec deux types de contraintes

Nous poursuivons l'étude de la complexité de NetworkDesign lorsque deux types de contraintes sont envisagés.

10.3.1 Lorsque le degré et le nombre de sauts sont bornés

Résoudre [*deg*,*hop*,0]-NetworkDesign est ici encore un problème NP-complet en général. Rappelons tout de même que [1,*hop*,∅]-min-NetworkDesign est polynomial pour toute valeur de *hop*, en accord avec la proposition 5. Ce problème à fait l'objet de nombreux travaux sous l'hypothèse que les arêtes sont de capacité infinie. Signalons par exemple qu'il est très voisin du DEGREE-BOUNDED

MINIMUM DIAMETER SPANNING TREE (DBMDSpT) [KJL+04] lorsque tous les nœuds sont clients, qu'il n'existe qu'une seule source et que les arêtes sont de capacité infinie.

Dans la suite de cette sous-section, les résultats énoncés supposent tous que les arêtes sont de capacité infinie. Malgré cette hypothèse réductrice, nous allons voir que certains résultats de complexité restent encore à trouver.

Les deux théorèmes suivants annonçent la complexité de [*deg*,*hop*,0]-NetworkDesign pour certaines valeurs de *deg* et *hop*.

Théorème 27 *:*
Le problème [deg,1,0]-min-NetworkDesign est polynomial pour tout deg ≥ 1 *lorsque les arêtes sont de capacité infinie.*

Preuve :

Soit $I = (G, capa, S, T, dem)$ une instance de [*deg*,*hop*,0]-min-NetworkDesign. Le nombre de sauts maximum est de 1, et les capacités des arêtes sont infinies. La demande de débit de chaque client est satisfaite à la condition qu'une arête le relie directement à une source. Toute solution se peut se réduire à trouver un ensemble d'arêtes connectant tout client à une source, de sorte qu'une source ne connecte pas plus de *deg* clients. Le coût d'une solution lorsqu'elle existe est toujours de $|T|$.

Le graphe $G = (V, E)$ de l'instance peut se réduire ici à un graphe biparti $G_B = (S, T, E')$ où $E' = \{\{x, y\} \in E | x \in S, y \in T\}$ puisque seules les arêtes reliant une source et un client sont exploitables.

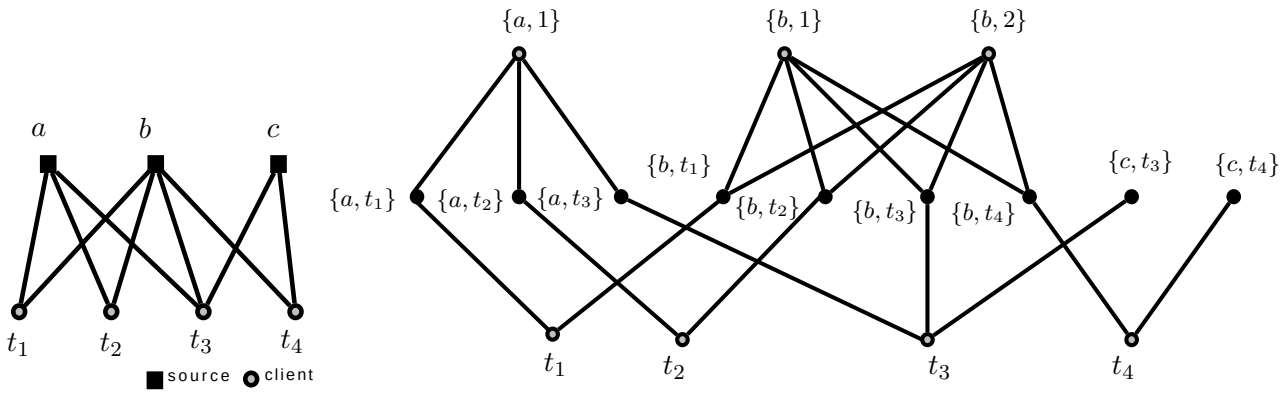
Nous définissons alors le graphe biparti $H = (X_H, Y_H, E_H)$ tel que :

$$X_H = E', \qquad Y_H = T \cup \big\{(x, i) | x \in S, d(x) > deg, i \in [1, d(x) - deg]\big\}$$

et

$$E_H = \big\{\{\{x, y\}, y\} | x \in S, y \in T\big\} \cup \big\{\{x, (x, i)\} | x \in S, i \in I\!N^*\big\}$$

La figure 10.3(b) présente un graphe biparti H construit à partir du graphe biparti G_B de la figure 10.3(a), lorsque le paramètre *deg* est fixé à 2.



(a) Un graphe biparti G_B (b) Le graphe biparti H

FIG. 10.3 – Construction du biparti H (b) à partir du biparti G_B de (a). Ici $deg = 2$.

Soit S un ensemble d'arêtes formant un couplage maximum dans H. Si S sature l'ensemble des sommets de Y_H, alors on peut en déduire une solution à l'instance I de [*deg*,1,0]-min-NetworkDesign. Nous sélectionnons dans $G_B = (S, T, E)$ les éléments de E associés à des éléments de T dans le couplage S (par exemple dans la figure 10.3(b), si l'arête $\{\{a, t_3\}, t_3\}$ appartient au couplage, alors $\{a, t_3\}$ fait partie de la solution de l'instance I sur la figure 10.3(a)). Puisque S est un couplage saturant, les éléments sélectionnés dans E touchent un nombre maximum d'éléments de T.

Le fait que le couplage sature les sommets du type $(x, i), x \in S, i \in \mathbb{N}^*$ permet de maintenir le degré de la source x inférieur ou égal à *deg* dans la solution de l'instance I. □

Théorème 28 *:*
Le problème [deg,hop,0]-min-NetworkDesign est NP-complet pour tout deg ≥ 4 *et tout hop* ≥ 2 *lorsque les arêtes sont de capacité infinie.*

Preuve :

Nous proposons une réduction au problème K-SET-COVER (KSC) qui décrit les instances de SC dans lesquelles la cardinalité de chaque sous-ensemble est d'au plus K. Le problème est connu pour être NP-complet à partir de $K = 3$.

Le problème [*deg*,*hop*,0]-NetworkDesign est dans NP : soit $(G, capa, S, T, dem, k)$ une instance de ce problème. Étant donné un ensemble de chaînes C de taille au plus *hop* et d'un flot chaînes-contraint $f : C \to R^+$, il est possible de vérifier en un temps polynomial que (C, f) est bien une solution pour l'intance $(G, capa, S, T, dem, k)$ dans laquelle le degré des nœuds est bien inférieur à *deg*.

Soit une instance $I_{KSC} = (X, T, k)$ une instance du problème K-SET-COVER, où T est un ensemble de n éléments $T = \{t_1, t_2, \ldots, t_n\}$, et $X = \{X_1, X_2, \ldots, X_m\}$ est composé de m sous-ensembles de T, de cardinalité au plus K.

Nous construisons depuis I_{KSC} une instance I du problème [*deg*,*hop*,0]-NetworkDesign comme suit :

Nous définissons le graphe $G = (V, E)$ tel que :

$$V = X \cup T \cup \left\{s_j^{X_i}\right\}_{1 \leq i \leq m, 0 \leq j \leq hop-2}$$

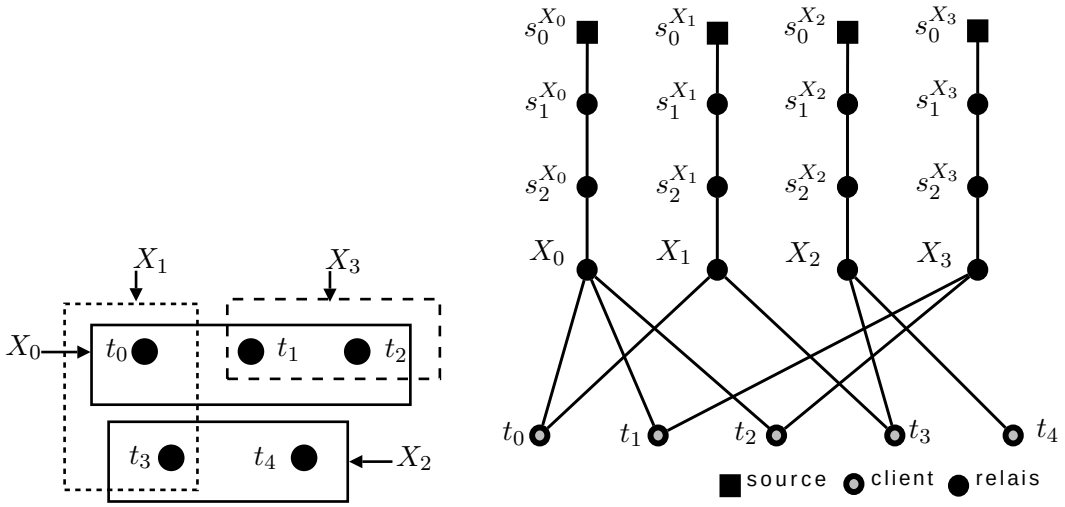
et

$$E = \left\{\{X_i, t_j\} | t_j \in X_i\right\}_{0 \leq i \leq m, 0 \leq j \leq n} \cup \left\{s_j^{X_i}, s_{j+1}^{X_i}\right\}_{1 \leq i \leq m, 0 \leq j \leq hop-3}$$

La construction de G est polynomiale en fonction de la taille de I_{KSC}, et le degré dans G de tous les sommets de $V - T$ est clairement inférieur ou égal à $K + 1$. Nous considérons l'instance $I = (G, capa, S, T, dem, k)$, dans laquelle $S = \left\{s_0^{X_i}\right\}_{1 \leq i \leq m}$ et $capa(e) = \infty, \forall e \in E$. La figure 10.4(b) présente l'instance I construite depuis l'instance de 3-SET-COVER de la figure 10.4(a).

Nous affirmons que s'il existe une solution à l'instance I sur [*deg*,*hop*,0]-NetworkDesign, alors il existe une solution de coût proportionnel sur l'instance I_{KSC} de K-SET-COVER et réciproquement.

Soit (C, f) une solution de coût a sur l'instance I, où C est l'ensemble des chaînes c reliant une source à un client de T, et telles que $f(c) > 0$. D'après la construction proposée et les conditions sur les capacités des arêtes, on peut supposer que toute chaîne $c \in C$ est de taille *hop*, et qu'il n'y a qu'une seule chaîne par client (dans le cas contraire on peut se ramener à une solution de coût inférieur). Appelons U l'ensemble des arêtes utilisées par les chaînes de C. Alors exactement $|T|$



(a) Une instance de 3-SET-COVER

(b) Une instance de min-NetworkDesign

FIG. 10.4 – -
min-NetworkDesign]Construction d'une instance de [*deg*,*hop*,0]-min-NetworkDesign depuis une instance de 3-SET-COVER

arêtes de U sont incidentes aux sommets clients. Les autres arêtes assurent la connexion entre les différentes sources et les sommets de X. On peut en déduire une solution de coût $(a-|T|)/(hop-2)$ sur l'instance I_{KSC}, en sélectionnant les ensembles incidents aux arêtes de U. Rappelons que le degré des clients dans le graphe induit par les arêtes U est de 1, et que celui des autres sommets est d'au plus $K+1$.

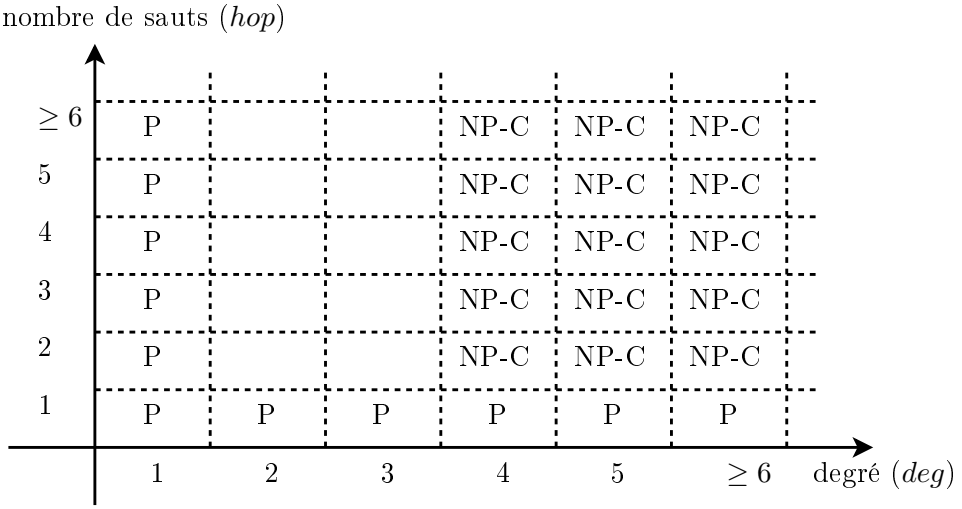
Réciproquement, s'il existe une solution de coût b pour l'instance I_{SC}, on en déduit une solution de coût $(hop-2)\times b+|T|$ sur I, en sélectionnant comme nœuds relais les ensembles de la solution, et en assurant la connexité avec la source la plus proche.

Rappelons enfin que le problème est dans NP, et que K-SET-COVER est NP-complet pour $K \geq 3$. □

Le tableau 10.1 récapitule les résultats de cette sous-section, selon les valeurs de *deg* et *hop*.

À la vue de ce dernier, certains résultats demeurent encore à trouver. On peut se demander quelle est la complexité de [*deg*,*hop*,0]-min-NetworkDesign lorsque $deg \in [2,3]$, $hop \geq 2$ et que les arêtes sont de capacité infinie.

Il est par ailleurs admis que [*deg*,*hop*,0]-min-NetworkDesign avec arêtes de capacité infinie est au moins aussi difficile à résoudre que [*deg*,*hop*,0]-min-NetworkDesign lorsque les arêtes sont de capacité finie. La NP-complétude du premier problème annonce la NP-complétude du second. En revanche, rien ne permet d'affirmer que [*deg*,*hop*,0]-min-NetworkDesign est bien polynomial lorsque $deg \geq 2$ et $hop = 1$: le fait qu'un client puisse recevoir un flot depuis plusieurs sources ne permet pas d'étendre la preuve du théorème 27 à cette configuration. La complexité de ce problème est ici ouverte.



TAB. 10.1 – complexité de [*deg*,*hop*,0]-NetworkDesign selon le nombre de sauts et le degré maximum autorisés, lorsque les arêtes sont de capacité infinie

10.3.2 Lorsque le degré est borné et la robustesse imposée

La prochaine configuration étudiée consiste à assurer une r-robustesse aux pannes lorsque le degré du graphe est borné par deg. Il ne fait aucun doute de la NP-complétude du problème, même pour $r = 0$ puisque cette valeur a été considérée par défaut dans les travaux précédents. La remarque suivante est immédiate :

Remarque 27 *:*
Si $deg \leq r$ ou si $|S| \leq r$, alors [deg,0,r]-NetworkDesign est trivialement polynomial puisque sans solution réalisable.

En accord avec la proposition 5, le problème est polynomial lorsque $deg = 1$ et $r = 0$, et NP-complet pour tout $deg \geq 2$. Nous énonçons la proposition suivante :

Proposition 6 *:*
Le problème [2,0,1]-NetworkDesign est NP-complet.

Preuve :
Nous proposons une réduction depuis le problème HAMILTONIAN PATH (HP), connu pour être NP-complet même lorsque les extrémités du chemin recherché sont fixées.

Le problème [2,∅,1]-NetworkDesign est dans NP : étant donné une instance de ce problème, on peut vérifier en temps polynomial que toute solution à cette instance assure une robustesse de 1 à chaque client, sans que le degré des nœuds n'excède 2.

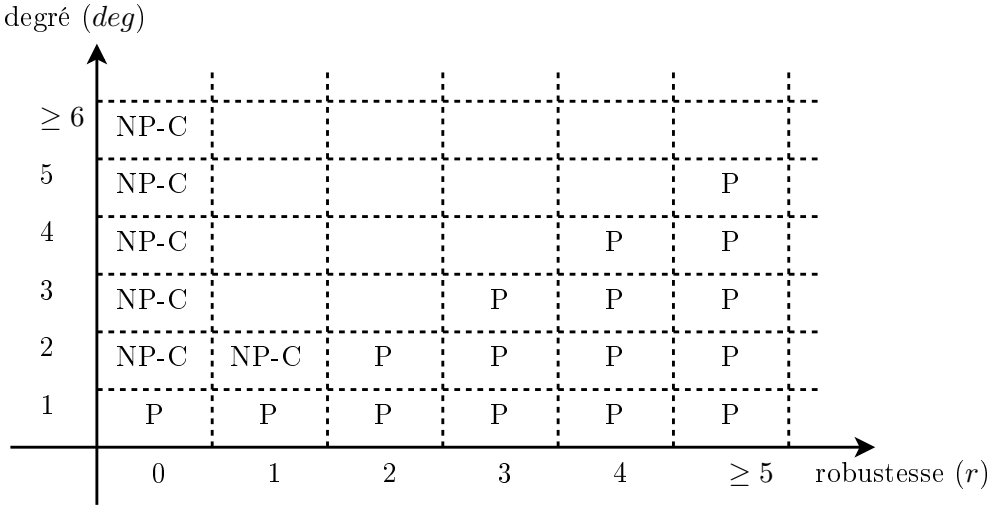
Soit G une instance de HP avec extrémités fixées, ici $s_1, s_2 \in V(G)$. Nous construisons une instance $(H, capa, S, T, dem)$ de [2,∅,1]-NetworkDesign dans laquelle :
– $V(H) = V(G) \cup \{z\}$,

- $E(H) = E(G) \cup \big\{\{s_1, z\}, \{s_2, z\}\big\}$,
- $S = \big\{\{s_1, s_2\} | s_1, s_2 \in V(G)\big\}$,
- $T = V(G)$,
- $capa(e) = \infty | e \in E(G)$.

Résoudre l'instance de NetworkDesign revient exactement à résoudre l'instance de HP avec extrémités fixées.

Dans toute solution à $(H, capa, S, T, dem)$, le degré de chaque client est de 2 exactement (au moins 2 pour assurer la robustesse, au plus 2 par la contrainte de limitation de degré). Soit U l'ensemble des arêtes utilisées dans toute solution. Alors $\big\{\{s_1, z\}, \{s_2, z\}\big\} \subseteq U$. De plus, chaque client doit être connecté aux deux sources s_1 et s_2 par deux chemins disjoints. Le degré de ces sources ne peut excéder 2, et les arêtes $\{s_1, z\}, \{s_2, z\}$ appartiennent à la solution. Il en résulte que les arêtes de $U - \big\{\{s_1, z\}, \{s_2, z\}\big\}$ décrivent un chemin hamiltonien dont les extrémités sont s_1 et s_2. La réciproque est immédiate. □

Le tableau 10.2 résume les résultats de complexité de [*deg*,∅,*r*]-NetworkDesign selon la valeur des variables *deg* et *r*.



TAB. 10.2 – complexité de [*deg*,∅,*r*]-NetworkDesign selon la robustesse minimale et le degré maximum autorisé.

10.3.3 Lorsque le nombre de sauts est borné et la robustesse imposée

Depuis les résultats énoncés dans la section précédente, il est possible de conclure immédiatement à la NP-complétude du problème [∞,*hop*,*r*]-NetworkDesign lorsque *hop* et *r* sont quelconques. Les deux théorèmes suivants étudient la frontière entre la polynomialité et la NP-complétude en fonction de ces deux paramètres. Le premier théorème annonce la polynomialité du problème pour tout couple (hop, r) tel que $hop = 1$ et $r \geq 0$. Le second précise la complexité du problème lorsque $hop \geq 2$ et $r \geq 0$.

Théorème 29 :
Le problème de décision [∞,1,r]-NetworkDesign est polynomial pour tout $r \geq 0$.

Preuve :

La preuve est une extension de la démonstration du point 1 du théorème 25.

Nous proposons un algorithme polynomial qui, depuis une instance I du problème [∞,*hop*,r]-NetworkDesign, renvoie une solution avec un nombre minimum d'arêtes lorsque $hop = 1$.

Le nombre de sauts maximum étant 1, le graphe de l'instance peut se limiter au graphe biparti G_B, sous-graphe de G induit par les sommets $S \cup T$. Notons que dans une solution, chaque arête sélectionnée est alors dédiée à un couple $(s_i, t_j), s_i \in S, t_j \in T$ et ne fera passser de flux qu'a destination de t_j. Les deux conditions sont nécessaires et suffisantes pour obtenir une solution à I :

1. chaque client $t \in T$ doit être incident à un ensemble d'arêtes $E_s(t)$ tel que $\sum_{e \in E_s(t)} capa(e) \geq dem(t)$,
2. l'ensemble $E_s(t)$ doit être de cardinalité au moins r pour chaque client $t \in T$.

Pour chaque client t, nous définissons un ensemble $E^*(t)$ de cardinalité minimum comme suit :
- nous ajoutons successivement à $E^*(t)$ les arêtes $\{t, s_i\} | s_i \in S$ en privilégiant celles de plus grande capacité, et tant que la somme des capacités des arêtes de E_t reste inférieure à la demande de t.
- Si $|E^*(t)| < r$, nous ajoutons des arêtes incidentes à t jusqu'à ce que $|E^*(t)| = r$.

Appelons $E^* = \bigcup_{t \in T} E^*(t)$. Soit (C, f), avec C l'ensemble des chaînes c de longueur 1 telles que $E(c) \in E^*$, et $f : C \to R$ une fonction telle que $f(c) = capa(E(c))$. Alors (C, f) constitue une solution de coût minimum pour I.

□

Théorème 30 :
Le problème de décision [∞,hop,r]-NetworkDesign est NP-complet pour tout $hop \geq 2$ *et* $r \geq 0$.

Preuve :

Nous montrons que [∞,*hop*,r]-NetworkDesign est NP-complet, par une réduction au problème de décision SET COVER (SC), pour tout $hop \geq 2$ et $r \geq 0$.

Le problème est dans NP : soit $(G, capa, S, T, dem, k)$ une instance de ce problème. Étant donné un ensemble de chaînes C et d'un flot chaînes-contraint $f : C \to R^+$, il est possible de vérifier en un temps polynomial que (C, f) est bien une solution pour l'intance $(G, capa, S, T, dem, k)$.

Soit une instance $I_{SC} = (X, T, k)$ une instance de SC, où T est un ensemble de n éléments $T = \{t_1, t_2, \ldots, t_n\}$, et $X = \{X_1, X_2, \ldots, X_m\}$ est composé de m sous-ensembles de T. Nous construisons depuis I_{SC} une instance I de [∞,*hop*,r]-NetworkDesign comme suit :

Nous construisons $r + 1$ copies distinctes de X nommées $X^0, X^1, \ldots X^r$, en posant $\forall 0 \leq i \leq r, X^i = \{X_1^i, X_2^i, \ldots X_m^i\}$, où X_k^i est la copie de X_k dans X^i. Notons $s_i^j = (j, i), \forall (j, i) \in [0, r] \times [0, hop - 2]$.

Nous définissons le graphe $G = (V, E)$ tel que :

$$V = \left\{X_i^j\right\}_{0 \leq j \leq r, 0 \leq i \leq m} \cup \left\{s_i^j\right\}_{0 \leq j \leq r, 0 \leq i \leq hop-2}$$

et

$$E = \left\{\{X_i^j, t_k\} | t_k \in X_i\right\}_{0\leq j\leq r, 0\leq i\leq m, 0\leq k\leq n} \cup \left\{s_i^j, s_{i+1}^j\right\}_{0\leq j\leq r, 0\leq i\leq hop-3}$$

Soient $S = \left\{s_0^j\right\}_{0\leq j\leq r}$, $dem : T \to R^+$ une fonction quelconque, et $capa : E \to \infty$ la fonction qui associe une capacité infinie aux arêtes de G. Posons $I = (G, capa, S, T, dem, l)$ une instance au problème [∞,*hop*,*r*]-NetworkDesign. La construction de I est clairement polynomiale en fonction de la taille de I_{SC}. La figure 10.3.3 présente l'instance I construite à partir de l'instance SC de la figure 10.5 et des paramètres $hop = 4$ et $r = 2$. L'instance comporte $r + 1$ sources auxquelles tout client doit être connecté.

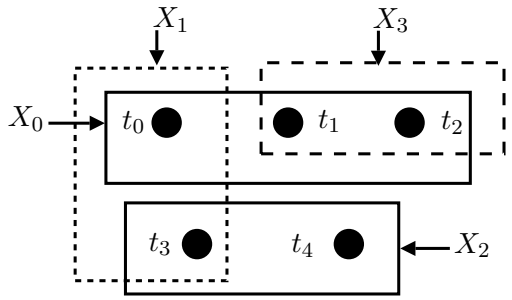


FIG. 10.5 – Une instance de SET-COVER

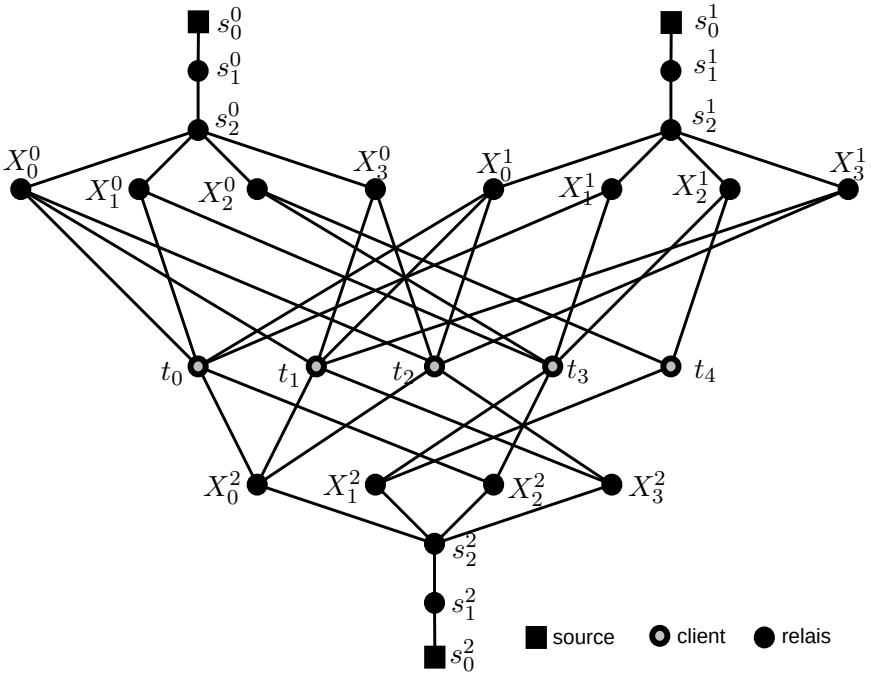


FIG. 10.6 – Une instance construite à partir de l'instance SET-COVER de la figure 10.5 et des paramètres $hop = 4$ et $r = 2$.

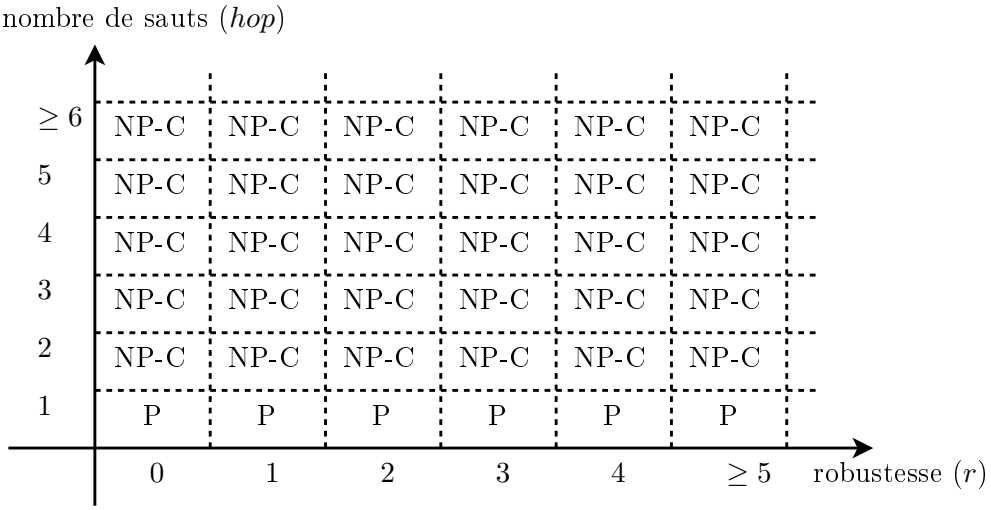
Considérons une solution (C, f) de coût inférieur ou égal à l sur l'instance I, où C est l'ensemble des chaînes c reliant une source et un client, telles que $f(c) > 0$. La construction proposée implique que toute chaîne $c \in C$ est de taille *hop*, et que deux chaînes contenant 2 sources différentes ne partagent aucune arête.
Pour une source s_0^i donnée, appelons $p_i = \left|\bigcup E(\{c\}), c \in C | s_0^i \in ext(c)\right|$ le nombre d'arêtes utilisées pour connecter s_0^i à l'ensemble des clients T. Soit $p = \min_{0 \leq i \leq r} p_i$. Alors, p arêtes suffisent pour connecter n'importe quelle source à T (évident par construction), et on peut se ramener à une solution de coût a avec $a = (r+1) \times p$.
Considérons alors une source s_0^i, et l'ensemble des arêtes $U = \bigcup E(\{c\}), c \in C | s_0^i \in ext(c)$ la connectant à T, et notons $p = |U|$. Alors exactement $p - |T| - (hop - 2)$ sommets de X_i sont utilisés pour connecter s_0^i à T. Ces sommets sont les ensembles de $X^i \cap \left\{\bigcup V(\{c\}), c \in C | s_0^i \in ext(c)\right\}$, qui définissent alors une solution au problème SC sur I_{SC} de coût $p - |T| - (hop - 2)$, encore égal à $a/(r+1) - |T| - (hop - 2)$.

Réciproquement, soit $X^S \subseteq X$ une solution sur l'instance I_{SC} de coût $b = |X_S| \leq k$. On peut raisonnablement penser que X^S est une couverture minimale, c'est-à-dire qu'il existe pour chaque ensemble X_i de X^S un élément de T qui n'appartient à aucun autre ensemble de X^S. Pour chaque couple $(s_0^i, t_j), 0 \leq i \leq r, 0 \leq j \leq n$, nous considérons la chaîne c telle que $V(c) = \{s_0^i, s_1^i, \ldots, s_{hop-2}^i, X_\alpha^i, t_j\}$ avec $X_\alpha^i \in X^S$ et $t_j \in X_\alpha^i$, et posons $f(c) = \infty$. Soit C l'union de ces chaînes, et H le graphe pour lequel $V_H = \bigcup_{c \in C} V(c)$ et $E_H = \bigcup_{c \in C} E(c)$. Alors (C, f) est bien une solution de I, puisque C assure la robustesse de T dans (G, S), et que les chaînes de C sont de longueur *hop*. Le nombre d'arêtes de H est alors de $((hop - 2) + b + |T|) \times (r + 1)$.

Ainsi pour toute solution sur I de coût a, on peut déduire une solution sur I_{SC} de coût b, où $b = a/(r+1) - |T| - (hop - 2)$, et réciproquement. Rappelons enfin que SC est un problème NP-complet. □

Corollaire 14 *:*
La réduction présentée conserve le rapport d'approximation. Rappelons que MSC est NP-difficile et non approximable à un facteur constant près. Ainsi [∞,hop,r]-min-NetworkDesign est un problème NP-difficile et non approximable à un facteur constant près pour tout hop ≥ 2 et r ≥ 0.

Le tableau 10.3 résume les résultats présentées dans cette sous-section.



TAB. 10.3 – Complexité de [∞,*hop*,*r*]-NetworkDesign en fonction de *hop* et r

Chapitre 11

Comparaison de méthodes de résolution exacte et approchée

Nous proposons dans ce chapitre trois stratégies destinées à résoudre rapidement des instances du problème NetworkDesign. Deux d'entre elles sont des méthodes exactes présentées dans les deux premières sections. Ces méthodes apportent une solution optimale au problème, mais nécessitent en contrepartie un temps d'exécution exponentiel en la taille de l'instance. Résoudre des instances plus grandes ne peut pas se faire efficacement par des approches optimales, et nécessite l'utilisation d'heuristiques. Nous proposons une heuristique basée sur la programmation génétique dans la troisième section de ce chapitre. Nous confrontons ces trois stratégies dans des jeux de simulations et présentons les résultats obtenus dans une quatrième section.

11.1 Formulation en programmation linéaire en nombres entiers

La programmation linéaire en nombres entiers est un outil de modélisation pour la résolution de nombreux problèmes d'optimisation.

Nous proposons dans cette section une formulation du problème NetworkDesign sous la forme d'un programme linéaire en nombres entiers (PLNE), dans lequel toutes les variables entières sont bivalentes[1]. La première sous-section introduit une formulation pour ce problème. La seconde sous-section instaure une discussion sur la pertinence de certaines composantes de la formulation, et introduit des coupes supplémentaires en vue d'optimiser les temps d'exécution du programme.

Nous utilisons la méthode du simplexe pour résoudre les programmes linéaires en nombre réels. Le principe de résolution d'un problème en nombres entiers P avec variables bivalentes utilise généralement la méthode du Branch-And-Bound[2] (B&B) dont le principe récursif est brièvement rappelé :

1. Le problème P est résolu dans une version relaxée où toutes les variables sont considérées réelles.

[1] Le domaine d'une variable bivalente ne comporte que deux valeurs

[2] La méthode de résolution est similaire lorsque le programme comporte des variables entières non bivalentes

2. S'il existe au moins une variable bivalente ayant une valeur fractionnaire et si P est réalisable, l'une de ces variables est choisie principalemment en fonction de sa valeur dans la solution de P.
3. Deux problèmes fils P_1 et P_2 sont définis par copie de P, à ceci près que la variable v est désormais affectée dans ces deux problèmes fils par l'une ou l'autre des deux valeurs de son domaine[3].
4. Les deux sous-problèmes P_1 et P_2 sont résolus successivement sur le même principe.

L'exploration d'un fils est récursive sur les variables entières non affectées. L'ensemble des relations entre un problème et ses fils forme une ***une arborescence de problèmes*** dont la hauteur n'excède pas le nombre de variables bivalentes[4], et dans laquelle toute feuille est soit un problème sans solution réalisable, soit une solution au problème original.

11.1.1 Formulation minimale du problème

Soit $I = (G, capa, S, T, dem, k)$ une instance du problème [*deg*,*hop*,*r*]-NetworkDesign, avec $G = (V, E)$. Une solution au problème NetworkDesign peut être décrite par un couple (C, f) où C est un ensemble de chaînes tel que $ext(c) = \{s_c, t_c | s_c \in S, t_c \in T\}, \forall c \in C$, et f est un flot chaînes-contraint sur C. Le coût de la solution est la cardinalité de l'ensemble $W = \bigcup_{c \in C} E(c)$.

Nous considérons D l'ensemble de toutes les chaînes c de G de taille inférieure ou égale à *hop* telles que $ext(c) = \{s_c, t_c | s_c \in S, t_c \in T\}$. Clairement $C \subseteq D$. Nous définissons la fonction $w : D \rightarrow R^+$ telle que :

$$w(c) = \min_{e \in E(c)} capa(e)$$

La fonction $w(c)$ définit le flot maximum qui peut transiter sur la chaîne c : dans toute solution (C, f) valide on a $f(c) \leq w(c)$ pour tout $c \in C$. Nous proposons un PLNE dans lequel nous utilisons 3 types de variables :

1. $u_e \in \{0, 1\}, \forall e \in E(G)$
2. $u_c \in [0, 1], \forall c \in D$
3. $f_c \in [0, w(c)], \forall c \in D$

Les variables u_e sont les seules variables entières, et bivalentes, de notre formulation. La variable u_e indique pour chaque arête e si cette dernière est présente ($u_e = 1$) ou absente ($u_e = 0$) de la solution. Pour une chaîne c donnée, la variable u_c indique dans une solution si c peut être utilisée (auquel cas $u_c > 0$) ou non ($u_c = 0$). Ne pas définir u_c comme variable binaire est dû au fait qu'augmenter le nombre de variables binaires augmente les temps de résolution du PLNE. Lorsque la chaîne est utilisée, f_c indique quelle quantité de flot transite sur c.

L'objectif est la minimisation du nombre d'arêtes utilisées :

$$\min \sum_{e \in E(G)} u_e$$

[3]Lorsque la variable entière n'est pas bivalente, son domaine est partitionné en deux sous-domaines, qui deviennent respectivement les domaines de v dans P_1 et P_2

[4]Lorsque toutes les variables entières sont bivalentes, ce qui est le cas dans notre formulation

Cette minimisation est sujette aux contraintes suivantes décrivant une solution :

L'inéquation 11.1 signifie que tout flot transitant sur une chaîne implique l'utilisation de cette dernière.

$$f_c \leq w(c) \times u_c, \quad \forall c \in D \tag{11.1}$$

L'inéquation 11.2 assure la cohérence entre les variables u_c et u_e : si une arête e n'est pas sélectionnée ($u_e = 0$), alors les chaînes c contenant e ne peuvent pas être utilisées ($u_c = 0$). La fonction $R(e)$ retourne le nombre de chaînes de D contenant l'arête e.

$$\sum_{c \in D | e \in E(c)} u_c \leq R(e) \times u_e, \quad \forall e \in E(G) \tag{11.2}$$

L'inéquation 11.3 exprime les contraintes de capacité sur les arêtes : la somme des flots transitant sur une arête ne peut dépasser sa capacité. Cette équation assure de même une cohérence entre les variables f_c et u_e : si une arête e n'est pas sélectionnée ($u_e = 0$) alors le flot de chaque chaîne c contenant e ne peut qu'être nul.

$$\sum_{c \in D | e \in E(c)} f_c \leq u_e \times capa(e), \quad \forall e \in E(G) \tag{11.3}$$

L'équation 11.4 décrit la contrainte de degré : pour chaque nœud x, le nombre d'arêtes incidentes à x est borné par le degré maximum deg.

$$\sum_{e \in E | x \in e} u_e \leq deg, \quad \forall x \in V(G) \tag{11.4}$$

L'équation 11.5 modélise les demandes de flot. La somme des flots reçus par chaque nœud client doit être supérieure à sa demande.

$$\sum_{c \in D | t \in ext(c)} f_c \geq dem(t), \quad \forall t \in T - S \tag{11.5}$$

Dans les inéquations suivantes, nous appelons V^k l'ensemble des parties à k éléments de V. L'inéquation 11.6 définit les contraintes de robustesses. Pour tout sommet client t, et toute partie σ de r éléments (correspondant à r pannes simultanées) il doit exister une chaîne connectant t à un nœud source quelconque et ne contenant aucun sommet de σ. Lorsque t est également une source, on peut ne considérer que les parties de $r - 1$ éléments (équation 11.7).

$$\sum_{c \in D | t \in ext(c), \sigma \cap V(c) = \emptyset} u_c \geq 0, \quad \forall t \in T - S, \forall \sigma \in V^r | t \notin \sigma \tag{11.6}$$

$$\sum_{c \in C | t \in ext(c), \sigma \cap V(c) = \emptyset} u_c \geq 0, \quad \forall t \in T \cap S, \forall \sigma \in V^{r-1} | t \notin \sigma \tag{11.7}$$

La contrainte relative au nombre de sauts est directement intégrée dans la définition de D : le flot ne transite que sur des chaînes de taille inférieure ou égale à hop.

11.1.2 Coupes supplémentaires

Lors de la résolution entière d'un PLNE, une méthode d'arrondis est utilisée, par exemple en remplaçant dans la solution optimale continue chaque composante fractionnaire par l'entier le plus proche. L'arrondi produit sur ces variables ainsi que l'ordre dans lequel sont traitées ces variables, peut jouer un rôle important dans le temps de résolution. L'inéquation 11.2 définit autant de contraintes qu'il y a d'arêtes. Ces contraintes sont peu nombreuses, mais ne sont pas pertinentes lors de la résolution entière du problème : chaque contrainte contient un nombre élevé de variables u_c, dont la somme doit être inférieure à une variable u_e, elle même pondérée par un coefficient R. Plus ce coefficient est grand, plus les méthodes d'arrondi proposées se révèlent peu efficaces. Nous proposons une alternative à l'inéquation 11.2 par l'inéquation suivante :

$$u_c \leq u_e, \quad \forall c \in D, e \in E(c) \tag{11.8}$$

Le nombre de contraintes définies par cette inéquation est de l'ordre de $|D| \times hop$, ce qui est bien supérieur à son homologue 11.2, et allonge le temps de recherche d'une solution optimale en nombres réels. En contre-partie, les méthodes d'arrondis lors de la résolution entière produisent de meilleurs résultats.

11.2 Une seconde méthode de résolution exacte : le B&B-dédié

Nous suggérons dans cette section une seconde méthode de résolution optimale du problème NetworkDesign. La définition de cette stratégie naît des critiques imputées à l'utilisation de la méthode B&B pour résoudre le PLNE de la section 11.1. Le temps de recherche d'une solution optimale est lié non seulement au nombre de variables et contraintes, mais également au temps de résolution de la relaxation linéaire de chaque sous-problème de l'arborescence. L'exploration de cette arborescence est donc très ciblée (les coupes générées font que de nombreux sous-arbres ne sont pas explorés), mais lente en raison de l'ensemble des contraintes à vérifier pour chaque nœud. Lorsque le nombre maximum de sauts autorisés est élevé, le nombre de variables et contraintes grandit très vite sur des topologies denses. Ceci risque d'avoir une répercution négative sur les temps de calcul du PLNE. Nous proposons ici une stratégie de Branch-and-Bound dédié (B&B-dédié), qui se veut plus robuste à la variation de certaines contraintes de déploiement comme le nombre de sauts maximum.

11.2.1 Idée générale

L'originalité de notre approche consiste tout d'abord à partitionner l'ensemble des contraintes en deux catégories. Nous recherchons l'ensemble des solutions satisfaisant la première catégorie de contraintes au moyen d'une méthode exacte type B&B. À chaque solution trouvée, nous vérifions les contraintes de la seconde catégorie et validons ou non la solution. Les contraintes sont dissociées comme suit :

1. La première catégorie contient les contraintes liées au degré des nœuds, et à la distance entre une source et un client. Ces contraintes sont dites « topologiques ».
2. La seconde catégorie rassemble les contraintes de robustesse, et de satisfaction de flot.

Dit autrement, la recherche de solutions répondant aux contraintes de première catégorie permet de fixer une topologie. Étant donné un ensemble de nœuds et de liens fixés, les contraintes de seconde

catégorie vérifient alors l'existence ou non d'un flot chaînes-contraint dont les chaînes assurent la robustesse demandée.

11.2.2 Recherche d'une topologie valide par B&B

Considérons $I = (G, capa, S, T, dem, k)$ une instance du problème [*deg*,*hop*,*r*]-NetworkDesign, avec $G = (V, E)$.

La recherche d'une topologie en accord avec les contraintes de première catégorie s'opère par une technique classique B&B, grâce auquel nous définissons une arborescence de topologies. Chacune de ces topologies est décrite par trois ensembles partitionnant E :

1. l'ensemble des arêtes sélectionnées E_S,
2. l'ensemble des arêtes interdites E_I,
3. l'ensemble des arêtes éligibles E_E.

Par la suite, et pour un nœud quelconque N de l'arborescence des topologies, nous désignons ces trois ensembles par les notations $E_S(N)$, $E_I(N)$ et $E_E(N)$. Chaque nœud N désigne le sous-graphe de G induit par les arêtes de $E_S(N)$. La racine de l'arborescence est le nœud r tel que $E_E(r) = E$. Pour un nœud N, on définit deux nœuds fils N_1 et N_2 comme suit :

- On extrait de N une arête e parmi l'ensemble $E_E(N)$,
- On définit deux nœuds N_1 et N_2 par copie de N,
- On ajoute e à $E_S(N_1)$, et e à $E_I(N_2)$.

Dit autrement, N_1 et N_2 réfèrent respectivement à deux topologies, selon que l'arête e a été ajoutée ou interdite à la topologie décrite par N.

Contrairement à la résolution entière du PLNE de la section 11.1, il n'y a pas ici de résolution relaxée sur les nœuds de l'arborescence. Les énumérations suivantes précisent le mode opératoire utilisé permettant de vérifier les contraintes de première catégorie.

1. Les arêtes sont successivement extraites de E_E en fonction de leur éloignement d'une source de S[5]. Cette stratégie permet d'établir pour chaque nœud N un tableau de distances indicé par les sommets de G, qui renseigne sur la distance entre un sommet et la source la plus proche sur le sous-graphe de G induit par $E_S(N)$. Les arêtes dont les deux extrémités sont à une distance au moins égale au nombre de sauts maximum autorisés sont immédiatement déclarées interdites, puisqu'aucune chaîne entre une source et une extrémité ne peut à la fois transiter par elles et satisfaire la contrainte de longueur.
2. Pour chaque nœud N, nous maintenons à jour deux tableaux indicés par les sommets de G. Le premier tableau contient les degrés des sommets du sous-graphe de G induit par $E_S(N)$, et le second ceux du sous-graphe de G induit par $E_E(N)$. Ces éléments nous permettent de définir des coupes relatives aux degrés : en un seul parcours, on peut vérifier si l'ajout ou la suppression de chacune des arêtes de $E_E(N)$ risque de violer la contrainte de degré maximum, ou de degré minimum (imposée par la robustesse). L'arête est alors directement déplacée dans l'ensemble adéquat.

Nous pouvons ainsi établir rapidement si une topologie décrite par un nœud N vérifie les contraintes de première catégorie. Si oui, nous utilisons la programmation linéaire (voir sous-section

[5] D'autres paramètres viennent s'ajouter au processus d'élection des arêtes. Mais dans tous les cas, si e_1 est choisi avant e_2, alors e_1 est au moins aussi proche d'une source que e_2.

11.2.3) pour vérifier s'il l'on peut ou non définir un flot chaînes-contraint et garantir la robustesse de ce flot. Si la solution est conforme aux contraintes de seconde catégorie, nous avons une solution à l'instance I de coût $|E_E(N)|$. Autrement, l'exploration reprend depuis N si $E_E(N)$ n'est pas vide.

Remarque 28 *:*
Nous pouvons établir de nouvelles coupes au fur et à mesure que des solutions réalisables sont trouvées : le coût d'une topologie décrite par le nœud N est clairement de $|E_S(N)|$. L'instauration d'un ***coût de référence*** *permet de limiter l'exploration en profondeur de l'arbre. Ce coût de référence est initialisé par $deg \times |V|$ qui désigne une borne supérieure du nombre d'arêtes d'une solution. Ce coût de référence est mis à jour chaque fois qu'une topologie valide vérifie les contraintes de seconde catégorie. L'exploration en profondeur d'un nœud est stoppée lorsque son coût atteint le coût de référence.*

La remarque 29 caractérise le fonctionnement de notre B&B par rapport à celui opéré dans la résolution entière du PLNE de la section précédente.

Remarque 29 *:*
Les traitements opérés sur chacun des nœuds de l'arborescence sont des opérations simples qui ne requièrent que peu de ressources. L'exploration de l'arborescence est donc plus rapide que pour l'arborescence de problèmes de la PLNE. Cependant certaines branches sont explorées en profondeur, alors que ces dernières auraient été écartées si l'on avait considéré l'ensemble des contraintes.

11.2.3 Recherche d'un flot chaînes-contraint robuste sur une topologie valide

Pour chacune des solutions renvoyées par le B&B, nous utilisons la programmation linéaire pour vérifier l'intégrité des contraintes de flots et de robustesse. Pour ce faire, nous reprenons la formulation de la section 11.1. Appelons H une topologie retournée par le B&B. Nous appliquons donc cette méthode de résolution exacte sur l'instance $I = (H, capa, S, T, dem)$. L'objectif n'est plus de trouver une solution minimisant le nombre d'arêtes, mais de déterminer l'existence ou non d'une solution. Toutes les arêtes de H peuvent appartenir à cette solution et les variables u_e sont fixées à 1. La formulation ne comporte plus de nombres entiers, et sa résolution est linéaire en fonction du nombre de variables. Les contraintes de degré définies par l'inéquation 11.4 disparaissent (traitées précédemment par le branch-and-bound).

Remarque 30 *:*
Toutes les chaînes de D peuvent ici être utilisées. Conserver les variables de type u_c ne sert qu'à déterminer si la robustesse est assurée sur l'instance I. Par contre le lien entre les variables f_c et u_c peut être rompu sans problème, par la suppression de l'inéquation 11.1.

De même, chacune des variables u_c peut être initialisée à 1, puisque les chaînes de D sont réalisables dans toute solution. Cette optimisation réduit la durée d'exécution du programme linéaire. Cependant, maintenir le domaine de u_c dans $[0, 1]$ permet de visualiser plus facilement dans une solution les chaînes participant à la robustesse du réseau.

11.3 Une heuristique issue de l'algorithmique génétique

Les méthodes exactes appliquées sur des problèmes NP-complets ne supportent généralement pas le passage à l'échelle. Nous proposons dans cette section une heuristique sous forme d'un algorithme

génétique. Notre choix s'est porté sur ces algorithmes, car leur utilisation est courante dans les problèmes de conception de réseaux avec contraintes de déploiement [GCO01, CC01, RG99]. Ils offrent généralement de bons résultats sur ce genre de problèmes, bien qu'à notre connaissance ils n'ont jamais été envisagés pour englober simultanément autant de types de contraintes.

Les algorithmes génétiques sont des procédures inspirées des mécanismes de sélection naturelle et des phénomènes génétiques. Un algorithme génétique est un algorithme stochastique itératif qui opère sur une population initiale, et qui fait évoluer cette population grâce à trois opérateurs :

1. croisement,
2. mutation,
3. sélection.

Les deux premiers sont des opérateurs d'exploration de l'espace des états. Le dernier fait évoluer la population vers les optima d'un problème. Une population est constituée de N individus. Chaque individu s'apparente à un ou plusieurs chromosomes, eux-mêmes constitués de plusieurs gènes qui contiennent les caractères héréditaires de l'individu. Les principes de sélection, de croisement, de mutation s'inspirent des processus naturels du même nom.

Notre idée consiste à reprendre la stratégie introduite dans la définition du B&B-dédiée :

1. diviser les contraintes en deux catégories,
2. déterminer des topologies satisfaisant la première catégorie de contraintes (topologies valides),
3. vérifier la seconde catégorie de contraintes sur les topologies valides.

L'utilisation d'un algorithme génétique remplace ici la technique de B&B utilisée lors de la recherche de topologies valides. La bipartition des contraintes selon deux catégories est identique à celle opérée dans le B&B-dédié.

11.3.1 Définition d'un individu

Pour un problème d'optimisation donné, un individu représente un état à explorer qui peut conduire ou non à une solution. Formuler un problème d'optimisation sous forme d'un algorithme génétique consiste en premier lieu à définir les fonctions attribuées à chacun des gènes d'un individu.

Définissons une telle formulation pour le problème [*deg*,*hop*,*r*]-NetworkDesign. Soit une I une instance $(G = (V, E), capa, S, T, dem, k)$ de ce problème, où $E = \{e_1, e_2, \ldots, e_m\}$. (les arêtes de E sont indicées de 1 à m). Un individu est un chromosome comportant m gènes, chacun associé à une arête. Un individu est alors un mot binaire M dont la i^{eme} lettre correspond à l'activité du gène associé à l'arête d'indice i. Un gène actif implique que l'arête est considérée, un gène inactif écarte cette arête.

Tout individu M décrit donc un graphe $H(M)$ dans lequel :

$$V(H(M)) = V(G)$$

et

$$E(H(M)) = \big\{\{e_i\}, 1 \leq i \leq m | M[i] = 1\big\}$$

11.3.2 Notation d'un individu

L'évolution d'une population requiert la notation des individus la composant. La note attribuée à un individu est d'autant plus forte que l'état qu'il représente s'approche d'une solution réalisable du problème NetworkDesign. Soit $H(M)$ le graphe décrit par l'individu M. Par la suite, appelons $r_H(x)$ le nombre de sources distantes d'un nœud x d'au plus *hop* dans H. Nous définissons également une fonction décroissante f_1 et une fonction croissante f_2, respectivement bornées supérieurement par deux constantes $const_1$ et $const_2$. La note attribuée à M, désignée par $note(M)$, est déterminée par l'algorithme suivant :

- $note(M)$ est initialisé à 0 ;
- Pour tout nœud x de $V(H(M))$, si $d_H(x) > deg$, alors $note(M) = note(M) + f_1(d_H(x))$, sinon $note(M) = note(M) + const_1$.
- Pour tout client $x \in T$, si $r_H(x) \leq r$, alors $note(M) = note(M) + f_2(r_H(x))$, sinon $note(M) = note(M) + const_2$.

Remarque 31 *:*
La condition $r_H(x) > r, \forall x \in T$ est nécessaire à la contrainte de robustesse, mais pas suffisante, puisque rien ne garantit que les $r_H(x)$ sources sont atteignables depuis x par des chemins disjoints. Cependant, vérifier l'intégrité de cette contrainte est algorithmiquement plus rapide que vérifier l'existence de $r+1$ chaînes disjointes reliant x à $r+1$ sources.

Cette condition pourrait être ajoutée à la recherche par B&B-dédié de la section précédente afin de limiter le nombre d'appels à la résolution linéaire.

Si $note(M)$ atteint la valeur $V(H(M)) \times const_1 + |T| \times const_2$, alors $H(M)$ est une topologie vérifiant la première catégorie de contraintes. Une résolution linéaire utilisée en sous-section 11.2.3 nous permet de vérifier si les contraintes de flot et de robustesse sont validées. Si oui, la note de l'individu est incrémentée d'un bonus (de l'ordre de |V(H(M))|), auquel nous retranchons le coût de la solution, soit $|E(H(M))|$. Les individus les mieux notés représentent donc des états plus proches des solutions réalisables de l'instance I que les individus mal notés.

11.3.3 Génération et évolution

Nous générons une population initiale de $N = 20000$ individus dont les gènes sont tirés aléatoirement dans $\{0, 1\}$. Nous lui attribuons le rang 0. Le processus de génération d'une population fille est une opération à mener avec attention : les individus doivent pouvoir évoluer vers les états-solutions du problème, sans pour autant converger vers un état unique. La diversité des individus doit être assurée, facilitant ainsi l'émergence de nouvelles combinaisons de gènes. À partir d'une population de rang n, nous générons une population de rang $n+1$ de même cardinalité comme suit :

1. Les individus sont classés au sein de la population selon leur note. Chaque individu se voit attribuer un potentiel d'enfants, inversement proportionnel à son rang dans le classement. Des points de potentiels d'enfants sont également distribués aléatoirement dans la population
2. Des couples sont formés entre individus dont le potentiel d'enfants est non nul. Chaque couple enfante deux individus, selon un processus d'enfantement décrit par la suite. Le potentiel d'enfants de chaque parent est alors décrémenté de 1. Cette étape se répète tant qu'il existe au moins deux parents dont le potentiel est non nul.
3. Chacun des enfants est sujet à une probabilité de mutation, dont l'effet est l'activation d'un gène inactif ou l'inhibition d'un gène actif.

4. Les individus les mieux classés sont intégrés à la population suivante, de même que certains individus tirés aléatoirement.

Le processus d'enfantement entre deux individus M_1 et M_2 consiste à tirer une variable aléatoire t entre 1 et m, et à définir deux fils F_1 et F_2 tels que :

$$F_1[i] = M_1[i] | 1 \leq i \leq t, \quad F_1[i] = M_2[i] | t+1 \leq i \leq m,$$
$$F_2[i] = M_2[i] | 1 \leq i \leq t, \quad F_2[i] = M_1[i] | t+1 \leq i \leq m.$$

Lorsque $M_1 = M_2$, nous instaurons une forte probabilité de mutation sur plusieurs des gènes de F_1 et F_2.

11.4 Simulations et résultats

Nous présentons une étude comparative des trois stratégies définies dans la section précédente. La première sous-section s'intéresse à la résolution d'une instance réelle par nos deux méthodes exactes. La seconde sous-section étudie l'impact des contraintes de déploiement sur les temps d'exécution des deux méthodes exactes. La dernière sous-section conclut par une étude des performances de l'heuristique par rapport au B&B-dédié.

11.4.1 Résolution d'une instance réelle

Le premier objectif de ces travaux est de résoudre une instance réelle proposée par l'entreprise qui nous a présenté le problème. L'instance est décrite par un graphe de 22 sommets et 65 arêtes. Tous les sommets du graphe sont des clients et possèdent une demande à satisfaire. L'instance compte 4 sommets sources. La capacité de chaque arête, non communiquée, a été déterminée en fonction de la qualité du signal et de la distance séparant ses deux nœuds. Notons que la taille de cette instance est considérable par rapport aux déploiements habituels que l'entreprise est amenée à réaliser. La solution recherchée doit respecter un degré d'au plus 3, un nombre de sauts d'au plus 3, et garantir une robustesse de 1. Ces valeurs de paramètres sont utilisées dans la majorité des déploiements réalisés, et nous les supposons par défaut dans la suite.

L'exécution des stratégies optimales a renvoyé chacun une solution en 27 arêtes. L'une de ces solution est présentée sur la figure 11.1.

Le temps de calcul nécessaire a été d'environ deux jours pour chaque stratégie et se décompose comme présenté dans le tableau 11.1. Ce tableau résume pour chaque stratégie le temps au bout duquel une solution réalisable et une solution optimale ont été atteintes, et le temps total d'exécution.

Outil	solution réalisable	solution optimale	Temps d'exécution
B& B dédié	0 sec	350 sec.	$\approx$ 2 jours
Solveur PLNE	0 sec	1253 sec.	$\approx$ 2 jours

TAB. 11.1 – Temps de calcul sur une instance réelle de 22 sommets et 65 arêtes

Nous retrouvons dans ces temps d'exécutions un comportement souvent observé en recherche opérationnelle : un solveur atteint très rapidement une solution optimale, mais utilise la majeure

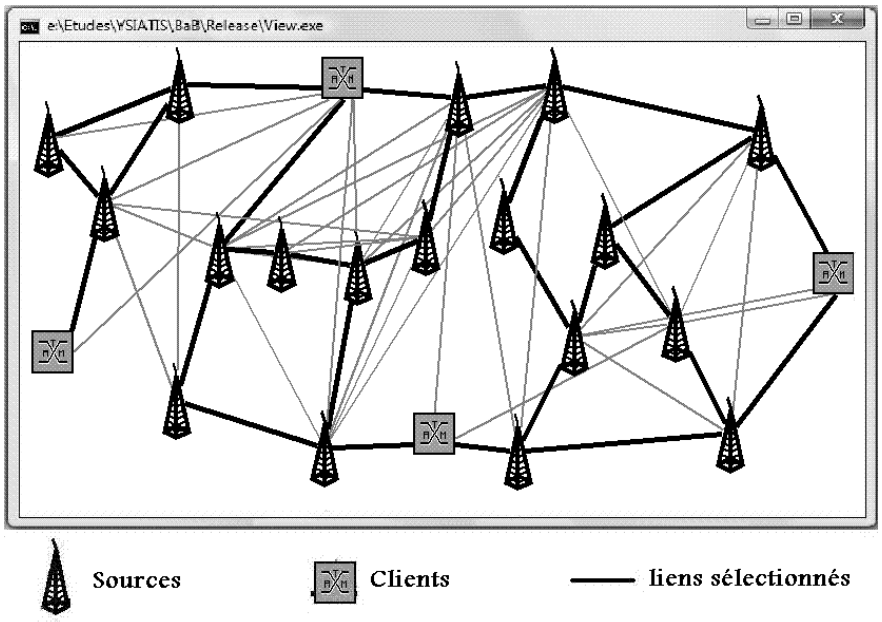


FIG. 11.1 – Résolution d'une instance réelle de [3,3,1]-min-NetworkDesign

partie du temps d'exécution à prouver son optimalité. Ce phénomène peut être atténué notamment par l'ajout de bornes inférieures précises. Le calcul de ces bornes constitue une perspective intéressante et laisse espérer des temps de calculs encore plus faibles.

Si nous comparons nos deux stratégies optimales en terme de temps d'exécution total, aucune ne semble dominer l'autre sur cette simple instance. Dans les sous-sections suivantes, nous montrons qu'elles sont pourtant complémentaires. En particulier nous montrons comment la variation de paramètres d'instance affecte différemment chacune de ces stratégies.

11.4.2 Comparaison des deux stratégies optimales

Nous avons comparé les performances de nos trois stratégies par de nombreux jeux de simulations. Une première partie des simulations vise à déterminer comment les facteurs de déploiement influent sur le temps nécessaire à la résolution d'une instance. La seconde partie met en évidence les performances de l'heuristique, essentiellement sur son efficacité à découvrir rapidement une solution réalisable par rapport à une méthode exacte.

Génération d'instances

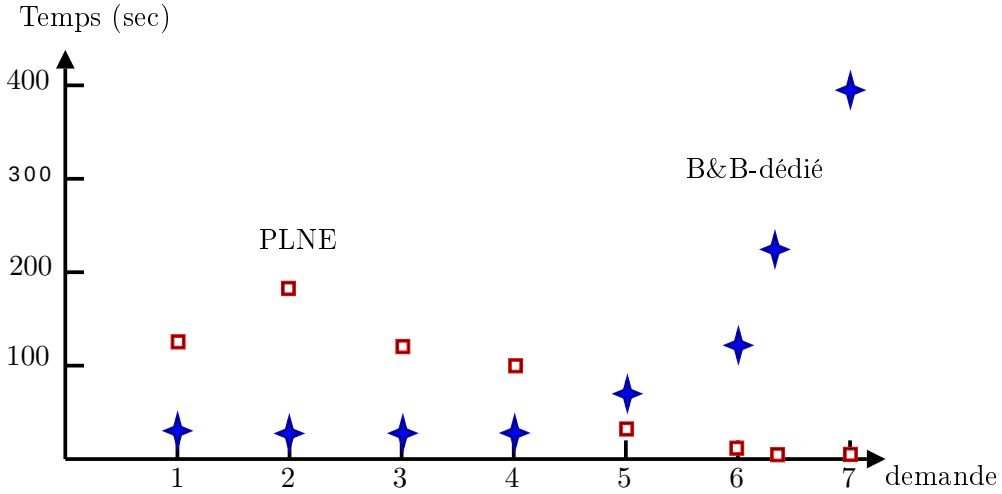
Nous avons généré aléatoirement des classes de graphes selon un schéma utilisé dans [CCS05]. Pour générer un graphe de n sommets, nous attribuons à chaque sommet une coordonnée dans un carré de dimensions $\sqrt(n) \times \sqrt(n)$. La probabilité d'existence d'une arête entre deux sommets a et b suit une loi de puissance de degré $\frac{-dist(a,b)}{\alpha \times \sqrt(2n)}$ où dist(a,b) est la distance euclidienne entre a et b, et α un paramètre donné. La variation de ce paramètre permet d'obtenir des graphes plus ou moins denses. Lorsque l'existence d'une arête est avérée, nous lui attribuons une capacité tirée

aléatoirement entre 5 et 20 unités par la fonction *capa* définie comme suit :

$$capa(\{a,b\}) = 5.0 + 15.0 * (1.0 - dist(a,b)/\sqrt{2n}) \times rand[0,1]$$

Impact de la demande :

Nous étudions l'impact de la demande des clients sur les performances de nos deux stratégies optimales. Nous générons une instance $I = (G, capa, S, T, dem)$ de [*deg*,*hop*,*r*]-min-NetworkDesign où G est un graphe aléatoire de 15 sommets et 35 arêtes. L'instance comporte 2 sources et tous les nœuds sont clients. Nous posons une demande uniforme par client $dem(x) = i | x \in T$, ou i est une variable donnée. Nous avons considéré les paramètres de déploiement par défaut, à savoir $deg = 3$, $hop = 3$ et $r = 1$. Le tableau 11.2 présente les temps d'exécution des deux stratégies[6]. La variation de la demande s'établit de 1 à 7 dans la première ligne. La seconde ligne désigne le coût optimal de la solution en fonction de la demande. Les deux dernières lignes renseignent sur le temps nécessaire à chaque stratégie pour garantir l'optimalité de la dernière solution trouvée.



demande i	1	2	3	4	5	6	6.3	7
coût de la solution optimale	18	18	18	18	18	18	19	inf
PLNE (sec)	123	182	121	100	33	6	3	0
B&B dédié (sec)	29	29	29	31	72	116	224	402

TAB. 11.2 – Variation des temps d'exécution en fonction de la demande.

Nous faisons les observations suivantes :

– Lorsque la demande est faible, l'approche par B&B dédié donne de meilleurs résultats que le PLNE. Les contraintes de flot jouent ici un rôle secondaire : dès qu'une topologie valide les contraintes de degré et garantit une connectivité entre tout client et une source, il y a de fortes chances qu'un flot valide puisse être trouvé, en raison de la faible demande. Les performances du PLNE sont en revanche moins bonnes, du fait que ce dernier vérifie pour chaque nœud exploré qu'un flot est effectivement réalisable.

[6] la notation *inf* du tableau signifie que l'instance n'a pas de solution réalisable.

– Si la demande est élevée, de nombreuses topologies validées par la première phrase du B&B dédié ne satisfont plus les contraintes de flot, ce qui le rend moins efficace. Le solveur linéaire est souvent mobilisé pour traiter des solutions potentielles qui se révèlent fausses. À l'inverse le PLNE cible plus efficacement son exploration et génère des coupes d'autant plus pertinentes que la demande est élevée (Par exemple le PLNE peut immédiatement conclure à l'absence de solution réalisable si la demande d'un nœud excède la somme des capacités des arêtes adjacentes).

Ces observations, présentées sur une instance, ont pu être vérifiées sur d'autres instances et avec d'autres paramètres de déploiement. L'écart entre les temps d'exécution des deux solutions varie selon les valeurs de *deg*, *hop* et r (la variation des temps d'exécution selon ces paramètres est analysée par la suite), mais le phénomène reste pleinement observable. Ces observations sont cohérentes avec nos stratégies : l'augmentation de la demande rend les contraintes de flot très fortes. Notre B& B-dédié les considère comme secondaires et cherche d'abord à établir une topologie respectant les contraintes de degré.

Sur des instances réelles, il a été remarqué que la capacité des liens est importante par rapport à la demande des nœuds, les opérateurs ayant tendance à surdimentionner leur réseau pour anticiper des demandes futures.

Impact du nombre de sauts :

Nous étudions l'impact du nombre de sauts sur les temps d'exécution. Nous générons une instance $I = (G, capa, S, T, dem, k)$ de [*deg*,*hop*,*r*]-min-NetworkDesign où G est un graphe de 10 sommets et 30 arêtes. L'instance comporte 2 sources et tous les nœuds sont clients. Nous avons dans un premier temps choisi un graphe suffisamment dense (le degré moyen est de 6, ce qui est relativement grand pour une instance de NetworkDesign). Nous étudions par la suite l'impact du nombre de sauts sur des densités moyennes et faibles. Nous posons $deg = 3$, $r = 1$, et faisons varier le nombre de sauts maximum autorisés *dist*. La demande de chaque client est fixée de sorte que, pour $deg = 3$, $r = 1$, et *hop* le plus petit que possible tel qu'il existe une solution réalisable, les temps de résolution de chaque stratégies sont relativement proches (voir section précédente). Le tableau 11.3 présente les temps d'exécution des deux stratégies pour *hop* variant de 1 à 7.

nombre de sauts i	1	2	3	4	5	6	7
coût de la solution optimale	inf	inf	11	11	11	11	11
PLNE (sec)	0	6	44	110	212	2809	2815
B&B dédié (sec)	1	14	9	7	27	62	64

TAB. 11.3 – Temps d'exécution en fonction du nombre de sauts : graphe fortement dense

Nous constatons que lorsque le nombre de sauts *dist* est trop faible (de 1 à 2) les deux stratégies déterminent rapidement qu'aucune solution n'est réalisable. Lorsque la valeur de *dist* atteint le minimum nécessaire à l'existence d'une solution réalisable, ici 3, les deux stratégies calculent efficacement une solution optimale. Du fait de la densité du graphe, l'augmentation de *dist* n'est pas pertinente, car elle ne permet pas d'atteindre de meilleures solutions : le coût de la solution optimale reste ici de 11, que le nombre de sauts soit minimal (égal à 3) ou maximal (la plus grande chaîne dont les sommets sont deux à deux disjoints est de 6). Aussi plus nous augmentons le nombre

de sauts, plus nous ajoutons des variables et des contraintes supplémentaires dont l'intérêt est très réduit, ce qui alourdit les temps d'exécution de nos stratégies, mais à différentes échelles :

- Le PLNE est très sensible à l'élévation du nombre de sauts : le simple fait d'autoriser un saut supplémentaire peut multiplier le nombre chaînes de longueur bornée, et donc le nombre de variables et contraintes de la formulation linéaire en nombres entiers. Les temps d'exécutions augmentent de manière exponentielle.
- Le B&B dédié voit également ses temps d'exécution augmenter, mais de manière bien plus modérée. Un nombre supérieur de nœuds sont explorés, mais l'appel au programme linéaire se fait exclusivement sur une topologie de degré borné par *deg*. Ceci limite le nombre de chaînes et donc de variables et contraintes générées.

La définition même du B&B-dédié a été motivée par les craintes de tels comportements de la part du PLNE. Ces observations s'avèrent conformes à nos prévisions. Le B&B-dédié fait ici preuve d'une robustesse appréciable face à la variation du nombre de sauts. Notons qu'à partir d'une certaine valeur de *dist*, le nombre de chaînes se stabilise - puisque chaque chaîne ne peut contenir deux fois le même sommet - et les temps de calcul n'évoluent plus. Nous avons validé ces constatations sur plusieurs instances pour lesquelles le graphe du réseau était dense, et avec d'autres valeurs pour les paramètres *deg* et *hop*.

Nous avons étendu notre protocole expérimental sur des graphes moyennement et faiblement denses. Il apparaît que :

- Lorsque l'instance est réalisable pour une valeur de *hop* la plus petit possible, les deux stratégies calculent une solution optimale dans des temps de référence.
- L'augmentation légère de *hop* entraîne une diminution du coût optimal de la solution. Cette diminution est d'autant plus marquée que le graphe est peu dense. Elle est pleinement profitable au B&B-dédié, car elle permet de limiter plus efficacement l'exploration de l'arborescence de recherche une fois la solution optimale atteinte.
- En revanche sur le PLNE, le gain engendré par une exploration moins profonde ne compense pas l'augmentation du temps de résolution de la relaxation linéaire de chaque sous-problème de l'arborescence d'exploration, principalement dopé par l'augmentation du nombre de variables et contraintes.
- Enfin, si *dist* s'éloigne trop d'un minimum, nous retrouvons un phénomène identique à celui observé sur les graphes à forte densité, avec augmentation des temps d'exécution des deux stratégies jusqu'à stabilisation.

Nous illustrons l'augmentation des temps d'exécution du PLNE et la diminution de ceux du B&B-dédié par deux jeux de simulation présentés dans les tableaux 11.4 et 11.5. Le tableau 11.4 présente les temps d'exécution pour une instance composé d'un graphe de 15 sommets et 35 arêtes. Tous les sommets sont clients, et trois d'entre eux sont également des sources. Le graphe est moyennement dense.

Le tableau 11.4 présente les temps de calcul pour un de densité plus faible, composé de 20 sommets pour 30 arêtes. Les sommets sont tous clients, dont 5 sources.

Remarque 32 *:*
Il était raisonnable de penser que sur le PLNE, l'augmentation du temps de calcul global serait moins prononcée lorsque le graphe est de faible densité : en effet lorsque dist augmente, le nombre de chaînes de longueur bornée par dist augmente d'autant moins vite que la densité du graphe est

nombre de sauts *hop*	1	2	3	4	5	6	7	8
coût de la solution optimale	inf	inf	18	16	15	15	14	14
PLNE (sec)	0	0	118	1128	2029	2298	17884	≥ 20.000
B&B-dédié (sec)	0	5	33	8	2	3	2	2

TAB. 11.4 – Temps d'exécution en fonction du nombre de sauts : graphe moyennement dense

nombre de sauts *hop*	1	2	3	4	5	6	7	8	9
coût de la solution optimale	inf	inf	inf	21	19	18	18	18	18
PLNE (sec)	0	0	0	6	37	176	1578	5083	≥ 20.000
B&B-dédié (sec)	0	0	1	6	1	0	1	1	2

TAB. 11.5 – Temps d'exécution en fonction du nombre de sauts : graphe faiblement dense

faible. Or dans cette configuration, nos observations nous indiquent que la valeur de la solution optimale décroit moins vite à mesure que dist augmente, que dans des graphes de densité moyenne. Ainsi lorsque dist augmente :

- *dans les graphes de faible densité : le coût de la solution optimale diminue peu, et le nombre de chaînes de longueur au plus dist augmente raisonnablement.*
- *dans les graphes de densité moyenne : le coût de la solution optimale diminue de manière un peu plus prononcée, mais le nombre de chaînes de longueur dist subit une augmentation plus forte.*

Ces phénomènes se complétant, l'augmentation du temps de calcul du PLNE semble relativement homogène pour les graphes de faible ou de moyenne densité.

Impact du degré :

Il ressort de nos simulations qu'il est difficile d'évaluer avec précision l'impact du degré sur les temps d'exécution de nos stratégies. Borner le degré est une contrainte très forte, qui agit a priori sur plusieurs points :

- Si *deg* est très faible, l'espace des solutions est très réduit. De nombreuses coupes sont générées par nos stratégies ce qui limite l'exploration des solutions réalisables. En revanche, la découverte de la première solution réalisable est plus longue. Atteindre rapidement des solutions réalisables permet de définir des bornes supérieures. Afin de limiter la profondeur de l'exploration et donc d'accélérer les temps de calculs, il est préférable de déterminer ces dernières le plus vite possible.
- Si *deg* est très grand, l'espace des états est beaucoup plus vaste. L'exploration de l'ensemble des solutions est donc plus coûteuse en temps. Cependant une solution réalisable peut être déterminée plus rapidement, et le côut de la solution optimale peut être diminué par rapport à celui obtenu avec une une valeur de *deg* très faible. Nous limitons alors l'exploration aux états de coûts strictement inférieurs.

Ces observations portent essentiellement sur les graphes de moyenne et forte densité. Sur des graphes faiblement denses, augmenter le degré maximum autorisé n'influence pas beaucoup les temps de calculs, puisque *deg* va rapidement atteindre et dépasser le degré moyen des sommets.

Sur des tailles d'instance raisonnables, il apparaît qu'un degré maximum élevé n'a qu'une influence très limitée sur le coût de la solution optimale, et joue plutôt sur la distribution des degrés dans une solution. Sur les simulations effectuées, peu d'instances voient leur coût diminuer lorsque *deg* augmente, quelque soit la proportion de clients dans la solution. Aussi un fort degré ne permet généralement pas une résolution plus rapide : l'espace des solutions est beaucoup plus vaste, tandis que la profondeur d'exploration n'est pas réduite par l'existence d'une solution optimale de meilleur coût.

Nous illustrons ces observations en présentant le comportement du B&B-dédié sur une instance composée de 15 sommets et 35 arêtes. Chaque nœud est client, et l'instance comporte 3 sources. Le tableau 11.6 décrit quelques statistiques d'exécution lorsque le degré maximum autorisé varie de 2 à 8, pour un nombre de sauts d'au plus 3 et une robustesse de 1. La première colonne présente la variation du degré maximum autorisé *deg*. Les colonnes 2 et 3 donnent le coût de la solution optimales, et le nombre de solutions réalisables trouvées, qui avaient un meilleur coût que les précédentes lors de leur découverte[7]. Les colonnes suivantes indiquent le nombre d'états qui ont été explorés (colonne 4) après coupes, ainsi que la proportion de ceux qui ont été explorés avant (colonne 5) et après (colonne 6) la découverte d'une solution optimale. La colonne 7 indique le nombre total de coupes générées , et sa proportion par rapport au nombre d'états explorés. La dernière colonne donne le temps d'exécution global en secondes.

deg	**Coût**	**Sols**	**État explorés**	**avant opt.**	**après opt.**	**Coupes ($\times 10^6$)**	**Temps**
2	inf	0	6339	100.0%	0.00%	0.26 (412%)	0
3	18	4	2.670.578	0.34%	99.66%	3.8 (142%)	30
4	18	8	14.452.442	2.37%	97.63%	21.9 (151%)	241
5	18	12	19.429.336	4.46%	95.54%	28.9 (1.49%)	398
6	18	12	19.758.607	3.46%	96.54%	29.2 (1.47%)	390
7	18	15	19.756.198	3.22%	96.78%	29 (1.46%)	375
8	18	15	19.756.198	3.22%	96.78%	29 (1.46%)	375

TAB. 11.6 – Temps d'exécution du B&B-dédié en fonction du degré.

En étudiant les résultats présentés dans le tableau 11.6, on voit que la solution optimale, lorsqu'elle existe, est déterminée très rapidement, et que l'essentiel de l'exploration est consacrée à prouver son optimalité. Une solution réalisable n'est atteignable qu'à partir d'un degré de 3. Augmenter le degré ne permet pas d'obtenir une solution de meilleur coût. En revanche l'augmentation du nombre d'états à explorer est considérable : +541% lorsque le degré maximum passe de 3 à 4, et +134% entre 4 et 5. Remarquons que le temps de résolution est proportionnel au pourcentage d'états explorés avant la découverte de la solution optimale. Ceci signifie que le B&B-dédié s'exécute d'autant plus rapidement qu'une solution optimale est vite trouvée. La recherche de cette solution optimale est facilitée par la découverte à la volée de solutions réalisables, qui génèrent des coupes de profondeur lors du processus d'exploration des états. Le nombre de solutions réalisables explorées est naturellement très petit, puisque dès qu'une solution de coût k est trouvée, nous cherchons uniquement les solutions de coût strictement inférieur à k. Nous observons que ce nombre augmente avec le degré. Tant que le degré augmente en restant inférieur ou égal à 5, les coupes générées par ces

[7]Si une solution de coût k a été trouvée à un instant t, alors ce nombre est incrémenté si et seulement si une solution de coût $k' < k$ est trouvée, et k' devient le nouveau coût de référence.

solutions ne permettent pas de compenser l'importante augmentation du nombre d'états. Ce phénomène atteint son apogée pour $deg = 5$, la solution optimale n'est trouvée qu'après avoir parcouru 4.46% du nombre total d'états explorés. Les temps d'exécution atteignent un maximum. À partir d'un degré autorisé de 6, nous assistons au processus inverse : le nombre d'états explorés a certes augmenté, mais la solution optimale est atteinte plus rapidement (au bout de 3.46% du nombre total d'états explorés). Cette observation se confirme pour un degré de 7 : la solution optimale est déterminée plus vite, ce qui génère une coupe efficace contribuant à diminuer le nombre d'états total explorés (qui devient inférieur à celui annoncé pour un degré de 6).

À titre de comparaison, les temps d'exécution du PLNE sont beaucoup moins prévisibles. Le tableau 11.4.2 présente les temps d'exécution du PLNE sur la même instance. La colonne 3 présente le nombre d'itérations effectuées par le PLNE. La colonne 3 désigne le pourcentage d'itérations effectuées avant la découverte d'une solution optimale.

deg	**Coût**	**itérations**	**avant s. opt.**	**après s. opt**	**temps**
2	inf	74	100.00%	0.00%	0
3	18	766070	0.41 %	99.59 %	168
4	18	2420079	0.27 %	99.73 %	515
5	18	2406927	0.25 %	99.75 %	482
6	18	2329243	0.32 %	99.68 %	383
7	18	2479197	0.27 %	99.73 %	524

TAB. 11.7 – Temps d'exécution du PLNE en fonction du degré.

Nous constatons ici aussi une augmentation importante du temps de calcul lorsque le degré passe de 3 à 4, relatif à l'augmentation de l'espace des états. En revanche, les variations lorsque le degré dépasse 4 sont plus délicates à expliquer. Nous supposons que ce comportement est propre au solveur linéaire, et plus précisément à l'ordre dans lequel les variables binaires sont considérées : au terme de la résolution de la relaxation linéaire, les variables $useEdge(e)$ doivent satisfaire les contraintes de l'inéquation 11.4, incluant le paramètre *deg*. L'ordre de branchement sur ces variables binaires dépend des valeurs que prennent ces variables dans la relaxation linéaire, elles même sujettes aux variations de *deg*. L'augmentation de *deg* laisse plus de liberté dans l'affectation des $useEdge(e)$. L'ordre des branchements se révèle alors plus aléatoire : efficace dans certains cas, et pénalisant dans d'autres. Notons que la rapidité à découvrir une solution optimale est ici encore un élément qui influence les temps d'exécution de nos stratégies.

Impact de la robustesse :

Le dernier paramètre étudié est celui de la robustesse. À priori, plus le coefficient de robustesse r est élevé, plus le nombre de contraintes dans le PLNE va être élevé. Le nombre de contraintes générées par l'inéquation 11.6 est de l'ordre de n^{r+1}. Il est difficile d'évaluer l'impact de ce coefficient lorsque ce dernier prend de grandes valeurs, à cause des raisons suivantes :

- Un coefficient de robustesse de r implique au moins $r + 1$ sources dans le réseau,
- Dans toute solution, le degré des clients est minoré par $r + 1$ (par r lorsque le client est également une source). Les sommets clients doivent donc avoir un degré suffisamment fort dans le graphe de départ.

– Les chaînes connectant tout client à $r+1$ sources distinctes doivent être deux à deux nœuds-distinctes. Ceci est difficilement réalisable pour de grandes valeurs de r lorsque le graphes est de faible ou moyenne densité.

Ces contraintes limitent fortement le nombre d'instances de taille raisonnable sur lesquelles on peut espérer obtenir un fort coefficient. Seules quelques rares instances générées aléatoirement ont une solution réalisable avec un coefficient de 3. Pour atteindre des coefficients supérieurs ou égaux à 4 il est nécessaire de diminuer dramatiquement le nombre de clients de l'instance, d'augmenter le nombre de sources, et d'autoriser un degré maximum important. Notre étude se limite donc au domaine compris entre 0 et 2 :

– Les résultats de nos simulations montrent que lorsque la robustesse est à 0, le coût de la solution optimale est généralement faible. Les coupes générées une fois la solution optimale trouvée sont pertinentes, et amènent à une résolution relativement rapide du problème, pour les deux stratégies.
– Lorsque le coefficient de robustesse est à 1, le coût optimal augmente, et l'espace des solutions reste étendu. Les temps de résolution sont corrects pour des instances de taille raisonnable.
– Lorsque le coefficient de robustesse atteint 2, le coût optimal augmente encore, mais l'espace des solutions se réduit considérablement. Les temps de résolution du problème sont très brefs, autant pour le B&B-dédié que pour le PLNE.

Le tableau 11.8 présente les temps d'exécution pour une instance de 10 sommets,40 arêtes et 3 sources, lorsque tous les nœuds sont clients. Le degré maximum est de 3, et le nombre de sauts d'au plus 3.

robustesse r	0	1	2
coût de la solution optimale	7	10	14
PLNE (sec)	87	337	6
B&B-dédié (sec)	15	96	1

TAB. 11.8 – Temps d'exécution en fonction de la robustesse

Notons que lors de l'exécution du PLNE, la résolution de la relaxation linéaire du problème semble plus longue à l'initialisation lorsque $r = 2$. Mais les branchements sont plus ciblés, et les coupes nombreuses.

11.4.3 Performances de l'algorithme génétique

Nous terminons ce chapitre par l'analyse de l'heuristique définie sous la forme d'un algorithme génétique.On ne peut pas établir à quel moment notre heuristique aura découvert la solution optimale sur de grandes instances. Le temps d'exécution ne dépend pas ici du nombre d'états explorés, mais du nombre de générations durant lesquelles nos solutions vont être amenées à évoluer. Notre évaluation consiste, pour une instance donnée, à comparer les temps utilisés par l'heuristique et le B&B-dédié pour atteindre une solution réalisable de même coût (ce coût n'est pas forcément optimal). Le tableau 11.9 présente un échantillon des résultats obtenus. Les coûts indicés d'une étoile ont été prouvés optimaux par une méthode exacte. Pour les autres instances, le coût optimal n'a pu être déterminé en moins de 24 heures.

Nos simulations montrent que sur les instances de faible taille, le B&B-dédié offre de meilleures performances : d'une part, quelques générations sont nécessaires à notre algorithme génétique pour

Instance				Algorithme génétique			B&B-dédié
Graphe	**deg**	**hop**	**r**	**Coût**	**Temps**	**Génération**	**temps**
$\|V\| = 15$ $\|E\| = 35$	3	3	1	18*	4 sec.	6	2 sec.
$\|V\| = 15$ $\|E\| = 50$	3	3	1	19*	7 sec.	10	272 sec.
$\|V\| = 22$ $\|E\| = 65$	3	3	1	27*	4 sec.	11	203 sec.
$\|V\| = 20$ $\|E\| = 70$	3	3	1	24*	13 sec.	13	63 sec.
$\|V\| = 20$ $\|E\| = 70$	3	3	1	26	17 sec.	16	26 sec.
				25	22 sec.	20	557 sec.
$\|V\| = 35$ $\|E\| = 112$	3	3	1	48	51 sec.	22	>= 24 heures
$\|V\| = 50$ $\|E\| = 160$	3	4	1	63	84 sec.	18	>= 24 heures
				62	143 sec.	20	>= 24 heures

TAB. 11.9 – Comparaison des temps de recherche d'une solution réalisable

établir des individus décrivant une solution réalisable. Le temps perdu à faire évoluer une population initiale est suffisant au B&B-dédié pour atteindre une solution réalisable. D'autre part le B&B-dédié est une stratégie optimale, et trouvera l'optimum même si l'espace des solutions optimales est très restreint. Si l'espace des solutions réalisables est très grand par rapport à celui des solutions optimales, les individus de l'algorithme génétique ont tendance à évoluer dans ce premier et éprouvent des difficultés à converger vers le second.

En revanche sur des instances de grandes tailles, notre algorithme génétique atteint plus facilement des solutions réalisables. Les écarts de temps sont proportionnels au nombre d'arêtes du graphe de l'instance. Notons que la génération des individus est telle que dans la première génération, la distribution des arêtes est uniforme dans les individus. À l'inverse dans le B&B-dédié, les premières arêtes sélectionnées se retrouvent plus souvent dans les premiers nœuds évalués. Si ces arêtes ne peuvent faire partie d'une solution, le B&B-dédié ne pourra pas déterminer rapidement l'existence une solution réalisable. L'algorithme génétique offre une certaine robustesse face à cela, alors que le B&B-dédié doit brancher dès le départ sur les bonnes arêtes pour être le plus efficace possible. Le choix des arêtes sur lesquelles brancher au départ est d'autant plus critique que le nombre d'arêtes est grand.

Remarque 33 *:*
L'utilisation d'un algorithme génétique offre un intérêt certain sur le point suivant : lorsqu'une instance ne possède pas de solution réalisable pour un triplet (deg, hop, r), alors l'algorithme génétique peut retourner une solution très proche d'une solution réalisable. La notation d'un individu que nous avons proposés suppose que plus le degré d'un sommet est proche de deg plus le nombre de points ajoutés à la note finale est grand. Il en est de même pour le nombre de sources distantes d'au plus hop d'un client donné. En généralisant cette approches sur l'ensemble des contraintes, nous pouvons définir un algorithme génétique qui retourne des solutions proches de la solution espérée lorsque celle ci ne peut être atteinte. Ceci s'avère particulièrement intéressant si l'instance ne peut être satisfaite à cause d'un seul client (auquel cas dans un déploiement réel on relâcherait une contrainte sur ce client uniquement). Les deux stratégies optimales proposées ne permettent pas une telle souplesse.

Conclusion

L'étude menée dans cette partie nous a permis d'introduire le problème de conception de réseaux radio multi-sauts robustes NetworkDesign. Les contraintes de déploiement sont liées à la satisfaction de bande passante, au degré maximal autorisé, au nombre de sauts entre une source et une destination, et au nombre de pannes simultanées que peut supporter le réseau. L'objectif est de minimiser le nombre de liens du réseau à déployer.

Nous avons étudié dans le chapitre 10 l'impact de chacune des contraintes de déploiement sur la difficulté du problème. Nous avons notamment établi les bornes entre polynomialité et NP-complétude pour des familles de NetworkDesign dans lesquelles certaines contraintes sont ignorées. Certaines de ces familles réfèrent parfois à des problèmes algorithmiques bien connus. Le problème min-NetworkDesign a été montré NP-difficile même lorsque seule la contrainte de flot est considérée. Dans ce cas particulier, on peut aisément vérifier l'existence d'une solution réalisable en considérant toutes les arêtes comme faisant partie d'une solution (Il suffit alors de savoir si les demandes de flot sont réalisables pour obtenir une solution, ou au contraire affirmer que le problème ne possède pas de solution). Cette observation se vérifie également si nous considérons également la contrainte de robustesse (la recherche de r chemins disjoints à origine ou destination unique se réduit à un problème de flots). En revanche l'introduction des contraintes de degré ou de distance ne permet plus d'assurer l'existence d'une solution, car elles rendent la sélection simultanée de certains liens incompatible avec les critères d'une solution.

Nous avons montré que le problème est polynomial lorsque le nombre de sauts ou le degré autorisé ont pour maximum 1. Résoudre certaines instances devient NP-complet dès que l'un ou l'autre de ces paramètres atteint la valeur 2. Mais nous n'avons pas pu établir avec précision la frontière entre polynomialité et NP-complétude lorsque ces deux paramètres interagissent simultanément, principalement lorsque le degré est compris entre 2 et 3, et le nombre de sauts entre 2 et *hop*, avec *hop* constant.

Nous avons proposé dans le chapitre 11 deux outils de résolution exacte,un PLNE et un B&B-dédié, pour traiter des instances de taille raisonnable. Nous avons étudié l'impact des trois paramètres de degré, nombre de sauts et robustesse sur les temps d'exécution de ces stratégies. Les observations qui en ressortent attestent que nos deux stratégies sont complémentaires. Par exemple lorsque la demande des nœuds est faible par rapport aux capacités des liens le B&B aboutit plus rapidement à une solution optimale. Le B&B-dédié est de plus robuste à un nombre élevé de sauts. En revanche, l'utilisation du PLNE est préconisée lorsque la demande est élevée car les coupes générées sont plus performantes. Par ailleurs le PLNE détermine plus rapidement si une instance peut assurer un facteur de robustesse r donnée ou non.

Déterminer comment agit chacun de ces paramètres sur les temps d'exécution de nos outils permet d'avoir une meilleure vision sur la façon d'aborder une instance donnée.

La prise en compte d'instances plus larges a nécessité de définir une heuristique basée sur les algorithmes génétiques. L'analyse de cette heuristique montre son efficacité à trouver rapidement des solutions réalisables lorsque les méthodes exactes se révèlent inefficaces. Les méthodes exactes restent cependant globalement plus performantes sur les instances de faible taille, notamment pour trouver les solutions de coût optimal.

Les perspectives que nous donnons à ce travail consistent essentiellement à définir de meilleures bornes sur le coût des solutions optimales de NetworkDesign. L'utilisation de nouvelles normes et technologies (Wi-max, antennes directionnelles plus performantes, ...) nous laissent penser que les tailles d'instances de ce problème seront plus importantes dans le futur. Savoir calculer des bornes pertinentes permettrait d'améliorer considérablement les temps de calcul de nos stratégies, validant ainsi leur utilisation dans des problèmes de conception plus étendus géographiquement.

Conclusion générale

Le travail présenté dans ce manuscrit a porté sur l'étude de trois problèmes algorithmiques inspirés des contraintes de fonctionnement des réseaux sans fil multi-sauts. La communication au sein de ces réseaux s'opère au moyen d'antennes omnidirectionnelles ou directionnelles. L'utilisation d'antennes directionnelles est privilégiée par exemple dans des réseaux très localisés géographiquement ou des réseaux dynamiques où les nœuds sont mobiles. La réception d'un message par ces antennes est sensible aux interférences et rend difficile l'accomplissement de certaines opérations : dans un tel environnement, un nœud peut émettre vers ses voisins en une seule émission, mais ne peut pas recevoir simultanément plusieurs transmissions. Cette contrainte rend difficile la résolution de certains problèmes algorithmiques. Soulignons que la complexité des problèmes de communication dans les réseaux sans-fil est parfois masquée par l'utilisation de la norme 802.11, qui assure à l'utilisateur un accès exclusif au média radio. Les phénomènes d'interférences sont cependant bien présents au niveau de la couche physique, et leur traitement requiert des retransmissions multiples et l'utilisation d'accusés de réception. Nous avons étudié dans les deux premières parties de ce mémoire deux problèmes de communication, sous l'hypothèse qu'aucun protocole bas-niveau n'assure de couche MAC idéale. En supposant que la topologie du réseau est connue (par un superviseur ou par l'ensemble des nœuds du réseau), nous avons su définir et étudier des stratégies de résolution sans accusé de réception pour les deux problèmes algorithmiques considérés.

L'étude du problème de la diffusion a fait l'objet de la première partie de ce mémoire. Différencier les modèles synchrones et asynchrones dans lesquels nous avons étudié ce problème a nécessité la mise en place d'une classification. Nous avons pu ordonner les différentes approches et travaux réalisés grâce à cette dernière. L'objet de notre travail a consisté à définir des stratégies de résolution au problème de la diffusion sur ces différents modèles. Sur les modèles synchrones, et en complément aux stratégies traditionnelles d'ordonnancement d'émissions, nous avons défini une nouvelle stratégie de diffusion, dite en arborescence, dont un des intérêts est sa déclinaison dans un modèle asynchrone.

Nos travaux se sont par la suite orientés sur l'étude de cette stratégie. Nous avons montré l'existence systématique d'une stratégie de diffusion en arborescence. Malheureusement, minimiser les coûts en temps de diffusion ou en nombre d'émissions par cette stratégie sont des problèmes NP-difficiles, aussi bien dans les cas synchrones qu'asynchrones.

Nous avons montré l'évolution de la complexité du problème visant à minimiser le nombre d'émissions en fonction de la topologie. Lorsque le graphe du réseau est triangulé, le nombre d'émissions minimum d'une stratégie de diffusion en arborescence est exactement la taille d'un ensemble dominant connexe minimum. Ce résultat aboutit à un algorithme polynomial minimisant le nombre d'émissions dans les graphes d'intervalles. Mais le problème devient NP-complet sur la classe des path-graphes, une classe de graphes comprise entre les graphes triangulés et les graphes d'intervalles. Sa version optimisation devient même NP-difficile et non approximable à un facteur constant

près sur les graphes triangulés. Lorsque l'on s'intéresse à la minimisation du temps de diffusion, nous avons montré l'existence d'un algorithme approximant le temps de diffusion par un facteur Δ lorsque le graphe est triangulé, où Δ est le degré maximum du graphe.

Parallèlement, nous poursuivi l'étude de stratégies d'ordonnancement synchrones sur des graphes dont l'excentricité de la source est 2. Des travaux menés dans [CW91] ont montré l'existence de stratégies de diffusion en $O(\ln^2 n)$ étapes. L'introduction d'un nouvel outil, la mv-décomposition, a permis de définir une stratégie de diffusion également en $O(\ln^2 n)$ étapes, mais dans des temps de calculs raisonnablement meilleurs. La mv-décomposition est un outil intéressant, que nous avonségalement utilisé pour définir une stratégie minimum en nombre d'étapes et d'émissions sur les graphes convexes circulaires.

L'étude de notre stratégie en arborescence dans un modèle synchrone a mené à définir des heuristiques de diffusion sur la grille et le tore. Le principal résultat est la mise en place d'une stratégie de diffusion en arborescence sur ces topologies. Les comparaisons menées entre les performances des heuristiques et le coût optimal ont conclus à la pertinence de nos heuristiques.

Nous avons montré que notre stratégie de diffusion en arborescence offre théoriquement des solutions optimales moins bonnes que les stratégies classiques de temporisation et d'ordonnancement d'émissions. Cependant, des jeux de simulation ont montré que les coûts obtenus sont généralement du même ordre. Les simulations ont été réalisées sur des graphes de disques unitaires, qui modélisent efficacement les réseaux radios.

Résoudre efficacement le problème de la diffusion constitue un enjeu majeur dans le fonctionnement des réseaux radios. Un second problème tout aussi important et celui de la satisfaction d'un ensemble de requêtes de communications donné. L'étude, cantonnée à un modèle synchrone, a considéré le problème sous deux familles, qui se différencient uniquement par la connaissance ou non de la fonction de routage. Les deux familles nommées DAWN-paths et DAWN-requests sont déclinées en plusieurs sous-problèmes, dont une version online. Nous avons étudié dans la seconde partie du mémoire les problèmes de chacune de ces familles lorsque l'objectif recherché est la minimisation du temps de diffusion.

Les résultats de complexité présentés attestent de la difficulté de ces problèmes même lorsque des conditions supplémentaires sont envisagées sur la topologie. Le principal résultat est la NP-complétude de DAWN-paths lorsque le graphe du réseau est un arbre binaire. Le résultat se décline pour prouver la NP-complétude sur les graphes de disques unitaire. D'autres résultats annoncent la limite entre la polynomialité et la NP-complétude des problèmes en fonction du nombre d'étapes maximum autorisé. Ces résultats montrent une frontière très basse pour des valeurs de 3 ou 2, selon que les routes de communication sont connues ou pas.

Nous apportons cependant un résultat encourageant, puisque nous proposons des algorithmes polynomiaux dès lors que le nombre de requêtes est borné par une constante K. Cette polynomialité est garantie même sur des réseaux dynamiques, qui modélisent efficacement des réseaux sur lesquels un ensemble de requêtes est en cours de satisfaction. Ceci permet d'envisager un processus de satisfaction de requêtes dans des conditions d'exécution plus réalistes et ouvre une réflexion sur la mise en œuvre d'heuristiques lorsque le nombre de requêtes n'est pas borné.

Les travaux menés dans ces deux premières parties ont apporté un ensemble de résultats théoriques et d'outils de résolution optimale, auxquels peuvent se comparer les algorithmes utilisés dans la résolution pratique de ces problèmes.

Nous avons montré par deux exemples que l'utilisation d'antennes omnidirectionnelles dans les réseaux sans-fil augmente la difficulté de certains problèmes algorithmiques, que l'on sait pourtant résoudre facilement dans les réseaux filaires. Cette difficulté est notamment dûe à la présence d'interférences prévenant certaines émissions simultanées.

L'utilisation d'antennes directionnelles pour établir les liens d'un réseau sans-fil offre un tout autre contexte d'exécution : les communications issues de l'emploi de cette technologie sont beaucoup moins sensibles aux interférences. Les antennes directionnelles permettent une communication avec des sites géographiquement plus éloignés. Elles se retrouvent dans le déploiement de réseaux sans-fil à dimension pluri-urbaine, voire départementale. Mais la mise en œuvre de tels réseaux se heurte à d'autres types de contraintes.

Dans la dernière partie de ce mémoire, nous avons étudié le problème de conception de réseaux radio multi-sauts robustes, dont le rôle est d'assurer la distribution d'un flux depuis un ensemble de sources vers un ensemble de destinations. Les contraintes de déploiement sont liées à la satisfaction de bande passante, au degré maximal autorisé, au nombre de sauts entre une source et une destination, et au nombre de pannes simultanées que peut supporter le réseau. L'objectif est de minimiser le nombre de liens du réseau à déployer.

Notre étude s'est intéressée à l'impact de chacune des contraintes de déploiement sur la difficulté à concevoir un réseau validant toutes les contraintes, et qui minimise le nombre de liens utilisés. Nous avons notamment établi les bornes entre polynomialité et NP-complétude lorsque certaines contraintes sont ignorées. Parmi les résultats énoncés, nous avons montré que le problème est polynomial lorsque le nombre de sauts ou le degré autorisé ont pour maximum 1. Résoudre certaines instances devient NP-complet dès que l'un ou l'autre de ces paramètres atteint la valeur 2.

Nos travaux nous ont conduits a essayer de résoudre rapidement le problème sur des instances de taille raisonnable. Nous avons pour cela proposé deux outils de résolution exacte définis par un PLNE et un B&B-dédié. Nous avons étudié l'impact des trois paramètres de degré, nombre de sauts et résistance aux pannes sur les temps d'exécution de ces stratégies, grâce à quoi nous avons pu définir dans quelles circonstances une stratégie peut s'avérer meilleure que l'autre. Par exemple lorsque la demande des nœuds est faible par rapport aux capacités des liens, le B&B aboutit plus rapidement à une solution optimale. De plus, les temps d'exécution du B&B-dédié ne sont que peu sensibles à la variation du nombre de sauts. En revanche, le PLNE offre de meilleurs résultats lorsque la demande est élevée, et détermine plus rapidement le nombre maximum de pannes simultanées que peut supporter un réseau déployé.

La prise en compte d'instances plus larges a nécessité la définition d'une heuristique basée sur les algorithmes génétiques. Ces méta-heuristiques ont prouvé par le passé leur efficacité sur des problèmes de conception de réseaux. Une des originalités de notre algorithme génétique est son interaction avec un programme linéaire, interaction également présente dans la stratégie de B&B-dédié précédemment introduite. L'analyse de cette heuristique montre son efficacité à trouver rapidement des solutions réalisables lorsque les méthodes exactes se révèlent inefficaces. Les méthodes exactes restent cependant globalement plus performantes sur les instances de faible taille, notamment pour trouver les solutions de coût optimal.

Les outils et formulations définis dans cette troisième partie sont pleinement exploitables pour un opérateur dont la mission est d'assurer la couverture en haut-débit de zones rurales. En outre, l'étude théorique du problème permet d'aborder la conception de réseaux multi-sauts robustes avec des

connaissances intéressantes sur la variation du nombre de liens minimum en fonction des politiques de déploiment adoptées.

Bibliographie

[ABNLP91] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. A lower bound for radio broadcast. *J. Comput. Syst. Sci.*, 43(2) :290–298, 1991.

[AHU74] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1974.

[Bau96] F. Bauer. Multicast routing in point-to-point networks under constraints, 1996.

[BCB99] S. Basagni, I. Chlamtac, and D. Bruschi. A mobility-transparent deterministic broadcast mechanism for ad hoc networks. *IEEE/ACM Trans. Netw.*, 7(6) :799–807, 1999.

[BGK+06a] J.-C. Bermond, J. Galtier, R. Klasing, N. Morales, and S. Pérennes. Gathering in specific radio networks. In *8èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (AlgoTel06), Trégastel*, May 2006.

[BGK+06b] J.-C. Bermond, J. Galtier, R. Klasing, N. Morales, and S. Pérennes. Hardness and approximation of gathering in static radio networks. In *FAWN06, Pisa, Italy*, March 2006.

[BP94] J. R. Blair and B. Peyton. On finding minimum-diameter clique trees. *Nordic J. of Computing*, 1(2) :173–201, 1994.

[BP97] D. Bruschi and M. Del Pinto. Lower bounds for the broadcast problem in mobile radio networks. *Distrib. Comput.*, 10(3) :129–135, 1997.

[BP05] J.-C. Bermond and J. Peters. Efficient gathering in radio grids with interference. In *AlgoTel'05*, Presqu'île de Giens, May 2005.

[BV95] F. Bauer and A. Varma. Degree-constrained multicasting in point-to-point networks, 1995.

[BXFJ03] B. Bui-Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(2) :267–285, Apr 2003.

[BYGI87] R. Bar-Yehuda, O. Goldreich, and A. Itai. On the time-complexity of broadcast in radio networks : an exponential gap between determinism randomization. In *PODC '87 : Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 98–108, New York, NY, USA, 1987. ACM Press.

[BYGI92] R. Bar-Yehuda, O. Goldreich, and A. Itai. On the time-complexity of broadcast in multi-hop radio networks : An exponential gap between determinism and randomization. *J. Comput. Syst. Sci.*, 45(1) :104–126, 1992.

[CC01] H. Chou, G. Premkumar ., and C.-H. Chu. Genetic algorithms for communications network design - an empiricalstudy of the factors that influence performance. *Evolutionary Computation, IEEE Transactions on*, 5 :236–249, 2001.

[CCJ90] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Math.*, 86(1-3) :165–177, 1990.

[CCS05] G. Chakraborty, D. Chakraborty, and N. Shiratori. A heuristic algorithm for optimum transmission schedule in broadcast packet radio networks. *Computer Communications*, 28 :74–75, 2005.

[CF94] I. Chlamtac and A. Faragó. Making transmission schedules immune to topology changes in multi-hop packet radio networks. *IEEE/ACM Trans. Netw.*, 2(1) :23–29, 1994.

[CGG+00] B. Chlebus, L. Gąsieniec, A. Gibbons, A. Pelc, and W. Rytter. Deterministic broadcasting in unknown radio networks. In *SODA '00 : Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 861–870, Philadelphia, PA, USA, 2000. Society for Industrial and Applied Mathematics.

[CGOR00] B. Chlebus, L. Gąsieniec, A. Östlin, and J.M. Robson. Deterministic radio broadcasting. In *ICALP '00 : Proceedings of the 27th International Colloquium on Automata, Languages and Programming*, pages 717–728, London, UK, 2000. Springer-Verlag.

[CGR02] M. Chrobak, L. Gasieniec, and W. Rytter. Fast broadcasting and gossiping in radio networks. *J. Algorithms*, 43(2) :177–189, 2002.

[Che04] G. Chelius. *Architectures et Communications dans les réseaux spontanés sans fil.* PhD thesis, INSA de Lyon, INRIA Rhône Alpes, France, April 2004.

[CHLD] X. Cheng, X. Huang, D. Li, and D. Du. Polynomial time approximation scheme for minimum connected dominating set in ad hoc wireless networks.

[CK85] I. Chlamtac and S. Kutten. On broadcasting in radio networks - Problem analysis and protocol design. *IEEE Transactions on Communications*, 33 :1240–1246, December 1985.

[CK87] I. Chlamtac and S. Kutten. Tree-based broadcasting in multihop radio networks. *IEEE Transactions on Computers*, C–36(10) :1209–1223, 1987.

[CMS03] A. Clementi, A. Monti, and R. Silvestri. Distributed broadcast in radio networks of unknown topology. *Theor. Comput. Sci.*, 302(1-3) :337–364, 2003.

[CR02] D. Coudert and H. Rivano. Lightpath assignment for multifibers WDM optical networks with wavelength translators. In *IEEE Globecom*, volume 3, pages 2686–2690, Taiwan, November 2002. OPNT-01-5.

[CW91] I. Chlamtac and O. Weinstein. The wave expansion approach to broadcasting in multihop radio network. *IEEE Transaction Communication*, (39) :426–433, 1991.

[CWY00] Y. Caro, D. B. West, and R. Yuster. Connected domination and spanning trees with many leaves. *SIAM Journal on Discrete Mathematics*, 13(2) :202–211, 2000.

[DM78] Y. Dalal and R. Metcalfe. Reverse path forwarding of broadcast packets. *Commun. ACM*, 21(12) :1040–1048, 1978.

[EKS05] G. Even, G. Kortsarz, and W. Slany. On network design problems : fixed cost flows and the covering steiner problem. *ACM Trans. Algorithms*, 1(1) :74–101, 2005.

[Fer02] A. Ferreira. On models and algorithms for dynamic communication networks : The case for evolving graphs. In 4^e *rencontres francophones sur les Aspects Algorithmiques des Telecommunications (ALGOTEL'2002)*, Mèze, France, May 2002.

[FKMS06] S. Funke, A. Kesselman, U. Meyer, and M. Segal. A simple improved distributed algorithm for minimum CDS in unit disk graphs. *ACM Transactions on Sensor Networks*, 2(3) :444–453, 2006.

[FL94] P. Fraigniaud and E. Lazard. Methods and problems of communication in usual networks. In *Proceedings of the international workshop on Broadcasting and gossiping 1990*, pages 79–133, New York, NY, USA, 1994. Elsevier North-Holland, Inc.

[Gav74] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *J. Combinatorial Th.*, 16(B) :47–56, 1974.

[GCO01] M. Gen, R. Cheng, and S. S. Oren. Network design techniques using adapted genetic algorithms. *Adv. Eng. Softw.*, 32(9) :731–744, 2001.

[GHP95] P. Galinier, M. Habib, and C. Paul. Chordal graphs and their clique graphs. In *Workshop on Graph-Theoretic Concepts in Computer Science*, pages 358–371, 1995.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

[GK96] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. In *European Symposium on Algorithms*, pages 179–193, 1996.

[GM95] I. Gaber and Y. Mansour. Broadcast in radio networks. In *SODA '95 : Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 577–585, Philadelphia, PA, USA, 1995. Society for Industrial and Applied Mathematics.

[Gra66] R. L. Graham. Bounds for certain multiprocessor anomalies. *Bell System Technical Journal*, 45 :1563–1581, 1966.

[HHL86] S.M. Hedetniemi, S.T. Hedetniemi, and A.L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18 :319–349, 1986.

[Kar75] R. M. Karp. On the computational complexity of combinatorial problems. *Networks*, 5 :45–68, 1975.

[KE05] G. Kortsarz and M. Elkin. An improved algorithm for radio broadcast (submitted), 2005.

[KJL+04] Konemann, Jochen, Levin, Asaf, Sinha, and Amitabh. Approximating the degree-bounded minimum diameter spanning tree problem. *Algorithmica*, 41(2) :117–129, November 2004.

[KP04] D. R. Kowalski and A. Pelc. Centralized deterministic broadcasting in undirected multi-hop radio networks. In *APPROX-RANDOM*, pages 171–182, 2004.

[LPHH84] R. Laskar, J. Pfaff, S. M. Hedetniemi, and S. T. Hedetniemi. On the algorithmic complexity of total domination. *SIAM Journal on Algebraic and Discrete Methods*, 5(3) :420–425, 1984.

[MBH+95] M. V. Marathe, H. Breu, H. B. Hunt III, S. S. Ravi, and D. J. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25 :59–68, 1995.

[MP01] G. De Marco and A. Pelc. Faster broadcasting in unknown radio networks. *Inf. Process. Lett.*, 79(2) :53–56, 2001.

[Pel00] D. Peleg. Deterministic radio broadcast with no topological knowledge, 2000.

[RG99] F. Rothlauf and D. Goldberg. Tree network design with genetic algorithms - an investigation in the locality of the pruefernumber encoding. In Scott Brave and Annie S. Wu, editors, *Late Breaking Papers at the 1999 Genetic and Evolutionary Computation Conference*, pages 238–244, Orlando, Florida, USA, 13 1999.

[RR88] G. Ramalingam and C. Pandu Rangan. Total domination in interval graphs revisited. *Inf. Process. Lett.*, 27(1) :17–21, 1988.

[RS95] N. Robertson and P. D. Seymour. Graph minors. xiii : the disjoint paths problem. *J. Comb. Theory Ser. B*, 63(1) :65–110, 1995.

[TG95] A. Turing and J.-Y. Girard. *La machine de Turing*. Seuil, Paris, France, 1995. Translated from English by Julien Basch et Patrice Blanchard.

[TNCS02] Y. Tseng, S. Ni, Y. Chen, and J. Sheu. The broadcast storm problem in a mobile ad hoc network. *Wirel. Netw.*, 8(2/3) :153–167, 2002.

[Tur36] A. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42) :230–265, 1936.

[WAF02] P. Wan, K. Alzoubi, and O. Frieder. Distributed construction of connected dominating set in wireless ad hoc networks, 2002.

Table des figures

Liste des tableaux

Résumé : Ces dernières années ont connu l'avènement des réseaux sans fil, dopés par leur facilité de déploiement et par leur usage dans de multiples domaines : réseaux domestiques Wi-Fi, téléphonie mobile, réseaux ad-hoc, réseaux de capteurs, ... L'objet de cette thèse porte sur l'étude de problèmes algorithmiques directement inspirés des contraintes de fonctionnement rencontrées dans de tels réseaux, et se découpe en trois parties. La première partie de nos travaux s'intéresse à l'étude du problème de la diffusion d'un message émis depuis un nœud source unique vers l'ensemble des nœuds participant au réseau. Ce problème est abordé dans plusieurs modèles de communication, qui supposent tous des émissions omnidirectionnelles à portée fixée et l'existence de phénomènes d'interférences. Il en résulte l'incapacité pour un nœud donné de garantir la réception correcte de deux transmissions voisines simultanées. Nous étudions la complexité de ce problème et proposons des stratégies de résolution exactes ou avec garantie de performance. Dans une seconde partie, l'un des modèles de communication précédemment introduits sert de support à l'étude d'un autre problème algorithmique, dont l'objet est la satisfaction de requêtes de communications. Les travaux menés sur ce problème visent à établir sa complexité ainsi que les facteurs dont elle dépend. La dernière partie nous amène au problème de conception de réseaux sans fil. L'objectif est d'assurer une distribution de flux depuis des nœuds sources vers des nœuds clients, en minimisant le coût de l'infrastructure déployée. Les communications établies ici à l'aide d'antennes directionnelles ne sont pas sujettes aux phénomènes d'interférences. La difficulté du problème réside dans la satisfaction de contraintes de déploiement (nombre d'antennes limitées par nœud, résistance aux pannes, ...). Nous étudions la complexité de ce problème, et proposons plusieurs méthodes de résolution exactes et approchées pour des instances de taille raisonnable.

Abstract : The last couple of years have seen the advent of wireless networks, doped by their ease of deployment and their use in multiple fields : personal WiFi networks, mobile telephony, ad hoc networks, sensors networks, ... The subject of this thesis relates to the study of algorithmic problems directly inspired by operating constraints which can be found in such networks. This manuscrit is divided into three parts. The first part of our work is devoted to the study of the problem of broadcasting a single source node message to all the other nodes of a network. This problem is tackled in various communication models. All the considered model suppose range-fixed omnidirectional transmissions subject to interference phenomena. It results from that, that any given node is unable to retrieve simultaneously two incoming transmissions. We study the complexity of this problem and propose some strategies in order to solve it.

In a second part, We study another algorithmic problem in the same communication model, whose object is to satisfy a given set of communication requests. Our work consists in etablishing the complexity of this problem, and studying the impact of various factors on this complexity.

The last part considers the problem of designing survivable radio networks . The objective is to ensure a distribution of bandwidth from source nodes to customers nodes, by minimizing the cost of the deployed infrastructure. Communications are made via directional antennas, and are not subjects to interferences. The difficulty of the problem lies in the satisfaction of deployment constraints (limited number of antennas per node, robustness against failures of nodes, ...). We study the complexity of this problem, and propose exact and approximated resolution methods to solve reasonable size instances.

Discipline : Informatique

Mots Clés : Théorie, Complexité, Réseaux sans-fil, Communication.

LIRMM UMR 5506 - 161 rue Ada, 34392 Montpellier Cedex 5 - France