Connexité dans les réseaux de télécommunications

Aubin Jarry

Université de Nice - Sophia Antipolis
UFR SCIENCES

École Doctorale STIC

THÈSE

pour obtenir le titre de
Docteur en Sciences
de l'Université de Nice - Sophia Antipolis

Spécialité: INFORMATIQUE

présentée et soutenue par
Aubin JARRY

Connexité dans les réseaux de télécommunications

Thèse dirigée par **Afonso FERREIRA**

soutenue le 14 Janvier 2005

Jury

M.	Igor	LITOVSKY	Professeur à l'UNSA	Président
M.	Éric	FLEURY	Professeur à l'INSA	Rapporteur
M.	Philippe	MAHEY	Professeur à l'ISIMA	Rapporteur
M.	Joseph	PETERS	Professeur à Simon Fraser University	Rapporteur
M.	Jean-Claude	BERMOND	Directeur de recherches CNRS	Examinateur
M.	Alexandre	LAUGIER	Ingénieur France Télécom	Examinateur
M.	Afonso	FERREIRA	Directeur de Recherches CNRS	Directeur

Table des matières

Table des figures

Introduction

Contexte

Cette thèse a été réalisée au sein du projet Mascotte.[1]. Le principal objectif du projet Mascotte est de développer des méthodes et des outils algorithmiques pour les réseaux de télécommunication. Ses principaux thèmes de recherche recouvrent ;

- l'algorithmique, les mathématiques discrètes et l'optimisation combinatoire
- les algorithmes de routage
- le dimensionnement de réseaux
- la simulation orientée objet distribuée
- le calcul parallèle et les réseaux d'interconnexions.

Au cours de cette thèse, j'ai étudié certains problèmes liés à la connexité et au routage dans les réseaux de télécommunication.

Objectifs à atteindre

Notre étude s'est tout d'abord concentrée sur l'étude des réseaux statiques, et plus particulièrement des réseaux optiques, sur lesquels nous avons travaillé en collaboration avec France Télécom. Les problèmes de télécommunication sur les réseaux statiques recouvrent les problèmes de dimensionnement de réseaux avant leur déploiement, puis les problèmes de routage sur les réseaux déployés. Ces problèmes ont été et sont étudiés extensivement durant ces cinquante dernières années et forment une branche importante du domaine de l'optimisation combinatoire [37]. A partir de caractéristiques souhaitées d'un réseau (nombre de clients, bande passante mise à disposition, délai de connexion, etc ...) et à partir d'une fonction de coût sur les réseaux, les problèmes de dimensionnement consistent à trouver des réseaux à coût minimum et offrant les caractéristiques souhaitées. Les problèmes de routage répondent quant à eux au besoin d'établir des communications. Étant donnés des paires de clients souhaitant communiquer entre eux, existe-t-il des routes permettant de réaliser simultanément ces connexions ? Ces problèmes se décomposent en trois catégories suivant le type de communication : problèmes de chemins disjoints, problèmes de multiflots entiers, et problèmes de multiflots fractionnaires.

Notre travail s'est basé sur deux caractéristiques des réseaux optiques. D'une part, la très grande

[1] Mascotte est un projet commun Cnrs / Inria / Université de Nice - Sophia Antipolis.

bande passante offerte par la fibre optique a amené les opérateurs de télécommunication à vouloir exploiter une partie des ressources des réseaux optiques pour assurer une continuité du service à l'épreuve des pannes. Les pannes sont le plus fréquemment provoquées par des coupures accidentelles de lignes de télécommunication enterrées. La cassure des fibres peut résulter de phénomènes naturels (comme les glissements de terrain) ou à l'activité humaine (travaux publics, erreurs de manipulation). Idéalement, les utilisateurs du réseau ne doivent pas être affectés. Pour cela, la connexité du réseau doit être suffisante. Dans le cadre de cette problématique, je me suis intéressé au dimensionnement de réseaux 2-connexes (réseaux qui restent fonctionnels après n'importe quelle panne).

D'autre part, les fibres optiques sont en général conçues comme des collections de paires de liens opposés. Les réseaux optiques peuvent ainsi être modélisés par des graphes orientés symétriques. Ces graphes ont des propriétés topologiques fortes, dont les algorithmes de routage peuvent tirer profit. J'ai étudié les problèmes de chemins disjoints et les problèmes de multiflots entiers sur les graphes symétriques.

Ensuite, notre étude a porté sur le concept de connexité pour les réseaux dynamiques. Les réseaux dynamiques ont d'abord été introduits dans le domaine des transports : les problèmes classiques consistent à acheminer des marchandises sur un réseau routier dont les caractéristiques changent en raison de la périodicité des feux de signalisation, de la densité du trafic, etc... D'autre évènements temporels interviennent (horaires de bateaux, de trains ...).

L'apparition des réseaux dynamiques dans le domaine des télécommunications provient de l'étude des réseaux satellitaires en orbite basse, puis des réseaux radio (type senseurs, ad-hoc, blue-tooth, etc..) C'est pour ces derniers réseaux que la connexité est devenu un enjeu. En effet, un convoyeur s'attend a ce qu'un réseau routier soit toujours connecté, c'est-à-dire que l'existence d'une route entre le départ et la destination ne soit jamais mise en question. A contrario, les réseaux radio subissent des pannes ou des ruptures intempestives de communication qui peuvent déconnecter le réseau.

Le modèle des Graphes Évolutifs a été récemment proposé par notre équipe pour fournir un cadre combinatoire à l'étude des réseaux dynamiques. Dans une première approche, cette modélisation contient implicitement une connaissance a priori de tous les évènements qui ont lieu sur un réseau. Cette connaissance, réaliste dans des réseaux dits "programmés", ne peut pas être utilisée lorsque les évènements sont inconnus ou peu connus a priori (pannes, état du trafic, etc..). Cependant, l'existence d'algorithmes exacts dans le cas d'une connaissance totale du réseau permet de comparer des protocoles réels à un optimum théorique, et établir ainsi un étalonnage des protocoles existants. Dans le cadre de cette thèse, nous avons continué à développer le modèle des graphes évolutifs avec trois objectifs : du point de vue applicatif, fournir des algorithmes efficaces et dont la complexité est prouvée pour une vaste gamme de réseaux dont les états futurs sont connus ; fournir un référentiel absolu pour pouvoir comparer des algorithmes sur des réseaux dont les états futurs ne sont pas connus ; du point de vue théorique, développer un cadre combinatoire pour comprendre quelle est l'influence du facteur temps sur les concepts de routage, de connexité et de flot.

Résultats décrits dans ce document

Nous résumons ici les résultats obtenus dans ce travail de thèse. Ces résultats sont divisés en deux grandes parties : réseaux statiques et réseaux dynamiques.

1. **Réseaux statiques**
   - **connexité :** nous avons déterminé le coût minimum d'un réseau résistant à une panne, c'est à dire le nombre minimum d'arêtes des graphes 2-connexes et des graphes 2-arête-connexes de diamètre fixé, prouvant ainsi une conjecture de Bolobás. Ce résultat à fait l'objet d'une communication à la rencontre ROADEF [50].
   - **graphes symétriques(chemins) :** le problème des chemins disjoints est NP-complet pour une instance quelconque de communication. Nous avons démontré que lorsque le nombre de requêtes est borné, il existe un algorithme polynomial pour résoudre ce problème. La preuve de cette démonstration repose en grande partie sur l'étude des "mineurs" effectuée par Robertson et Seymour, ce qui implique malheureusement que nous n'avons pas d'algorithmes réellement efficaces pour résoudre ce problème à cause de la grande complexité des algorithmes de Robertson et Seymour. Nous ne savons pas si cette difficulté est inhérente au problème lui-même, ou provient de la méthode employée. Ces résultats ont fait l'objet d'une communication au workshop SIROCCO [52].
   - **graphes symétriques(multiflots) :** Nous avons montré que la présence d'un seul flot est suffisante pour rompre la symétrie du problème des multiflots entiers sur les graphes symétriques. Ceci fait que les problèmes de multiflots sur les réseaux orientés symétriques sont presque équivalents aux problèmes de multiflots dans les réseaux orientés "à un flot près". Aussi nous avons montré que le problème des multiflots entiers est NP-difficile pour trois paires de requêtes. Nous avons ensuite étudié une instance particulière du problème ou les requêtes sont elles-mêmes symétriques. Ceci peut se justifier d'un point de vue applicatif par la résolution de communications type pair-à-pair ou type cablées. D'un point de vue théorique, ce problème est entièrement symétrique. Dans ce cas, nous avons proposé dans le cas de deux commodités un algorithme exact et rapide (correspondant à 6 calculs de flots simples). Il est remarquable qu'il s'agisse d'un des rares cas particuliers de multiflot entier ou il existe un algorithme polynomial pour deux commodités. Nous avons aussi montré, sans surprise, que le problème est NP-difficile pour un nombre indéterminé de requêtes. Le résultat sur les multiflots à deux commodités à fait l'objet d'une communication à la conférence STACS [49]. Les questions suivantes sont restées ouvertes : (1) Existe-t-il un algorithme polynomial pour la résolution du problème du multiflot entier pour deux requêtes ? (2) A partir de combien de requêtes (trois ou plus) le problème des multiflots symétriques ne peut plus être résolu en temps polynomial ?
2. **réseaux dynamiques**
   - **trajets optimaux :** la notion de trajet, qui modélise une communication ayant une dimension temporelle a été formalisée dans le cadre des graphes évolutifs par Ferreira [26]. Nous avons proposé des algorithmes polynomiaux pour calculer des trajets optimaux pour minimiser la date d'arrivée, le temps de trajet, ou le nombre de liens utilisés. Nous avons également montré qu'en présence d'une fonction arbitraire de coût sur les liens, trouver un trajet de coût minimum est un problème NP-difficile. Ces résultats sur les trajets ont fait l'objet d'une communication au workshop WIOPT'03 [10] et d'un article du journal IJFCS [9].

- **composantes connexes :** le problème des composantes connexes maximales avait été étudié dans le cadre des graphes évolutifs pour le calcul d'arbres de multicast [5], et a été montré NP-difficile. Nous avons étendu ce résultat dans le cas où l'ensemble des liens forme une double grille (obtenue en ajoutant des liens entre des sommets à distance 2 sur une grille classique). Nous avons proposé un algorithme polynomial pour résoudre le problème dans le cas où l'ensemble des liens forment un arbre. Ces résultats ont fait l'objet d'une communication au workshop DIALM-POMC [51].
- **arbre couvrant de poids minimum :** du point de vue applicatif, nous avons montré que pour les réseaux radio, calculer un arbre couvrant de coût énergétique minimal est NP-difficile, contrairement au cas statique. Ce résultat implique entre autres que les heuristiques actuelles de communication sur les réseaux radio, qui sont fondées sur le calcul d'arbre couvrant ne peuvent pas s'appliquer directement au cas dynamique. Nous avons cependant fourni un algorithme permettant de minimiser la consommation d'énergie du noeud le plus vulnérable. Ces résultats on fait l'objet d'une communication au workshop WIOPT'04 [28].
- **flot maximum :** Enfin, nous avons travaillé sur le problème du flot maximum. Nous avons formalisé la correspondance entre le problème du flot maximum et celui de la coupe minimum dans le cadre des graphes évolutifs, qui repose sur la modélisation espace-temps de Ford et Fulkerson. Le problème dual de coupe minimum s'exprime de manière très simple dans les graphes évolutifs, et peut être résolu dans certains cas particuliers. Nous ne connaissons pas la classe de complexité du problème dans le cas général.

Plan de lecture

La première partie de la thèse concerne les réseaux statiques. Nous étudions au chapitre 1 un problème lié à la résistance aux pannes : quel est le coût minimum d'un réseau qui ne sera pas déconnecté quelle que soit la panne qui survient. Ce problème se traduit en théorie des graphes par l'étude des graphes 2-connexes extrémaux. Nous démontrons une conjecture qui stipule que le nombre d'arêtes d'un graphe 2-connexe de diamètre D et contenant n sommets est $\lceil \frac{nD-(2D+1)}{D-1} \rceil$

Les chapitres 2 et 3 sont consacrés aux problèmes de routage dans les réseaux optiques. Nous exploitons une propriété des réseaux optiques : deux nœuds adjacents sont reliés par une paire de liens opposés. Nous modélisons un réseau optique par un graphe orienté symétrique qui reflète cette propriété. Au chapitre 2, nous étudions le problème des chemins disjoints, et nous montrons qu'il existe un algorithme polynomial pour résoudre ce problème lorsque le nombre de requêtes est borné. Au chapitre 3, nous étudions le problème des multiflots entiers. Nous montrons que ce problème est NP-difficile pour 3 requêtes. Nous donnons un algorithme polynomial dans le cas de deux paires de requêtes symétriques.

La deuxième partie de la thèse concerne les réseaux dynamiques. Nous présentons le modèle des graphes évolutifs au chapitre 4. Ce chapitre comprend une définition détaillée du modèle ; sa complexité combinatoire dépend du nombre de liens et du nombre d'évènements du réseau. Nous introduisons le concept de trajet qui représente une communication entre deux nœuds du réseau au cours du temps.

Au chapitre 4, nous présentons trois mesures de distances sur les graphes évolutifs. Nous proposons des algorithmes pour optimiser les trajets d'après ces distances. Nous montrons également

que lorsque l'on utilise une mesure de distance arbitraire, le problème d'optimisation des trajets est NP-difficile.

Au chapitre 5, nous étudions les composantes connexes des graphes évolutifs. Une composante connexe représente un ensemble de nœuds qui peuvent s'échanger des messages entre eux. Nous montrons que le problème de trouver une composante connexe de taille maximale est NP-difficile, même lorsque les liens entre les nœuds suivent une structure très régulière. Nous donnons un algorithme polynomial dans le cas particulier où l'ensemble des liens forment un arbre.

Au chapitre 7, nous abordons le problème du flot maximum dans les réseaux dynamiques. Nous montrons que le problème dual de la coupe minimum peut s'exprimer de façon concise sous forme de contraintes linéaires. Nous résolvons ce problème dans le cas où tous les liens du réseau ont des délais similaires.

Première partie

Réseaux statiques

Chapitre 1

Résistance aux pannes : 2-connexité

Sommaire

1.1 Introduction

La fibre optique est un support privilégié des réseaux de communication modernes. La bande passante, le taux de pertes, le délai et beaucoup de critères rendent ce support préférable aux anciens réseaux électriques et dans la plupart des cas aux réseaux satellitaires.

Alors que la première utilisation de la fibre optique permettait le transport d'un seul flux de données, la technologie de multiplexage en longueur d'ondes (Wavelength Division Multiplexing) a permis d'augmenter considérablement le débit offert par une fibre optique. Le principe en est assez simple[1] : le signal est transporté par un rayon de lumière monochromatique d'une certaine

[1] la technologie mise en oeuvre est quant à elle très complexe

longueur d'onde. Un prisme permet de séparer physiquement un rayon lumineux polychromatique en ses composants monochromatiques sans toucher à l'information transportée par ces rayons. Il est donc possible de transporter dans une même fibre plusieurs flux de données simultanément sans qu'il y ait d'interférences.

Le très grand débit offert par les fibres optiques implique que le coût d'un réseau de télécom -munication dépend beaucoup moins qu'autrefois du débit offert sur chaque lien. De plus, une partie de la surcapacité des fibres optiques peut être utilisée pour créer des canaux de secours qui garantiront la survie du réseau en cas de pannes ponctuelles.

Dans ce chapitre, nous étudions un des problèmes liés à la résistance aux pannes dans les réseaux optiques, c'est à dire à la capacité de réagir lors de la défaillance d'une infrastructure. Les pannes sont provoquées le plus fréquemment par la détérioration accidentelle des lignes de télécommunications enterrés. La cassure des fibres peut résulter de phénomènes naturels (glissement de terrains ...) ou à l'activité humaine (travaux de construction, erreurs de manipulation...). D'autres types de pannes surviennent également aux niveaux des serveurs centraux. Idéalement, les utilisateurs qui ne sont pas directement concernés par la panne ne doivent pas être affectés. Pour cela, la connexité et la capacité du réseau physique doivent être suffisantes pour pouvoir suppléer aux éléments défectueux, et des politiques de redistribution du trafic doivent être mises en jeu. Les problèmes d'optimisation se situent à trois niveaux :

- le réseau physique doit contenir suffisamment de liens pour que la rupture de l'un ou plusieurs d'entre ces liens ne sépare pas le réseau en plusieurs parties déconnectées.
- au niveau du réseau physique, il faut trouver de manière préventive une route alternative pour chaque lien de communication qui pourrait être rompu [20].
- lorsqu'une panne survient, on peut être tenté de modifier globalement l'ensemble des routes attribuées aux utilisateurs pour s'adapter aux nouvelles contraintes de communication [18, 88].

Nous avons effectué notre étude dans le cadre du premier niveau : assurer que le réseau physique puisse subir plusieurs pannes sans être déconnecté. Les surcapacités offertes par les liens optiques permettent de ne pas avoir à doubler tous les équipements d'un réseau pour le rendre résistant aux pannes. Cependant, assurer cette protection induit tout de même un surcoût en équipements et ne doit pas se faire au détriment d'autres caractéristiques du réseau dont dépend la qualité du service offert aux clients. Nous nous intéressons à un problème d'optimisation lié à la conception de réseaux optiques : déterminer le coût minimum d'un réseau qui puisse résister à une panne. Les paramètres du réseau sont le nombre de nœuds du réseau et la distance maximale qui sépare deux nœuds. Les nœuds représentent les équipements émettant ou recevant des communications (clients, serveurs, etc ...). La distance maximale entre les nœuds conditionne le temps mis pour établir une communication et est un élément essentiel de la qualité des communications. Ce problème nous a été proposé par France Télécom R&D [62].

Contenu du chapitre

La **section 1.2** présente la formulation du problème en théorie des graphes, et un état de l'art sur ce sujet.

La **section 1.3** contient des remarques préliminaires sur le problème ainsi que des exemples de graphes extrémaux (i.e. les meilleurs graphes pour le problème posé).

La **section 1.4** introduit un nouveau concept, la *double excentricité*, qui est une mesure sur

les graphes similaire à la mesure classique de l'excentricité, et qui permet de déduire le rapport du nombres d'arêtes sur le nombre de sommets du graphe.

Dans la **section 1.5** nous déterminons le nombre d'arêtes d'un graphe 2-connexe, et dans la **section 1.6** nous déterminons le nombre d'arêtes d'un graphe 2-arête-connexe.

1.2 Formulation et État de l'art

Nous effectuons notre approche dans le cadre de la théorie des graphes, qui est un des outils les mieux adaptés à la modélisation des réseaux de télécommunication. Un réseau est donc modélisé par un graphe, les sommets du graphe correspondent aux nœuds du réseau, et les arêtes du graphe correspondent aux liens optiques. En première approximation, minimiser le coût du réseau revient à minimiser le nombre d'arêtes du graphe. De manière également approchée, la distance maximale entre deux nœuds du réseau sera représentée par le diamètre du graphe.

Un graphe $G = (V, E)$ est dit k-connexe si $G - S$ reste connexe pour tout sous-ensemble $S \subset V$ tel que $|S| \leq k - 1$. Le diamètre d'un graphe est D si pour chaque paire de sommets $\{x, y\}$ il existe un chemin contenant au plus D arêtes les reliant, et si il existe au moins une paire de sommets pour laquelle le plus court chemin contient exactement D arêtes.

On dit qu'un graphe satisfait la propriété (n, k, D) si son ordre est n ($|X| = n$), s'il est k-connexe et si son diamètre est D. Un graphe $G = (V, E)$ est minimal pour la propriété P s'il n'existe aucun sous-ensemble d'arêtes $F \subset E$, $F \neq E$ tel que $G' = (V, E)$ vérifie la propriété P. Si G est 2-connexe, $\{x, y\} \subset V$ est appelée paire séparatrice si la suppression de x et y crée au moins deux composantes connexes non vides.

Deux problèmes liés à la résistance aux pannes et au coût des liens de communication sont principalement étudiés :

- trouver un graphe $G = (V, E)$ tel que E soit de cardinalité minimale et tel que G satisfait la propriété (n, k, D).
- trouver un graphe $G = (V, E)$ qui satisfait la propriété (n, k, D) tel que $< w.E >$ soit minimum (problème pondéré par un vecteur de coût sur les arêtes w).

En 1964, Murty et Vijayan [74] ont posé le problème de déterminer le nombre minimal $f_v(n, D, D', k-1)$ d'arêtes d'un graphe d'ordre n, de diamètre D et tel que son diamètre augmente au plus de D' si l'on supprime $k - 1$ sommets. Ce dernier problème revient au problème $P_1(n, k, D)$ lorsque $D' = n - 1$. En effet un graphe solution de $f_v(n, D, n - 1, k - 1)$ reste connexe (de diamètre au plus n) si on enlève n'importe quel ensemble de $k - 1$ arêtes.

En 1968, Bollobás [6] conjecturait que les graphes obtenus en joignant deux sommets par p chemins de longueur D étaient des graphes extrémaux (i.e minimaux en nombre d'arêtes) dans la classe $(n, D, D', 1)$ où n et D vérifient $n \equiv 2(mod.D - 1)$ et $D' = 2D - 2$. Cette conjecture a été prouvée en 1976 par Cacetta [12] dans le cas $(n, 4, D', 1)$, $D' \geq 6$. Le problème a été complètement résolu par Murty [72] [73] dans le cas $D = 2$, $d' \geq 3$ et $k = 2$. Il montre que $f_v(n, 2, D', 1) = 2n - 5$.

Un résultat de Usami est mentionné dans [4] : $f_v(n, D, 6, 2) = Max\{n, \lceil \frac{4n-8}{3} \rceil\}$.

Le problème et la conjecture générale ont aussi été proposés et étudiés par Enomoto et Usami [21] et par A. Laugier, F. Boyer et U. Goldschmidt [63]. Les deux ont étudié les paires séparatrices (voir ci-après) mais remarqué que cet outil était insuffisant pour résoudre la conjecture. Nous présentons ici une preuve de la conjecture (Théorème 1) qui reprend cette idée mais utilise la notion nouvelle d'aplatissement qui permet de conclure.

Théorème 1 *Soit $G = (V, E)$ un graphe 2-connexe de diamètre $D \geq 2$. Alors $|E| \geq \lceil \frac{|V|D-(2D+1)}{D-1} \rceil$. Cette borne est la meilleure possible.*

1.3 Préliminaires

1.3.1 Cas particuliers

Si le diamètre de G est égal à 1, G est nécessairement un graphe complet ; il contient $\frac{n(n-1)}{2}$ arêtes. Ce cas extrême n'est pas représentatif de l'ensemble des graphes 2-connexes. On pourra aussi remarquer que le quotient $\frac{D}{D-1}$ n'est pas défini pour $D = 1$. Nous n'étudions par conséquent que les graphes de diamètre au moins 2.

Si un graphe est 2-(sommet)-connexe, tous ses sommets sont de degré au moins deux. E contient donc au moins $|V|$ arêtes. Un cycle contenant moins de $2D+1$ sommets est de diamètre inférieur à D. Noter que pour $|V| \leq 2D+1$, $|V| \geq \lceil \frac{|V|D-(2D+1)}{D-1} \rceil$. Les cycles sont donc des graphes extrémaux.

Nous présentons maintenant trois classes connues de graphes 2-connexes dont le nombre d'arêtes est exactement $\lceil \frac{|V|D-(2D+1)}{D-1} \rceil$.

– Les graphes de la première classe sont formés par un ensemble de chaînes de longueur D s'appuyant sur un cycle de longueur $(2D + 1)$.

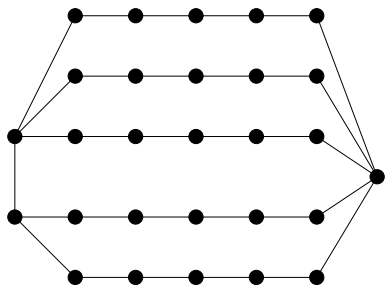


FIG. 1.1 – Graphe 2-connexe extrémal obtenu par induction.

– Les graphes de la deuxième classe sont formés par des chaînes de longueur $(2k+1)$ s'appuyant sur quatre sommets $(D = 4k + 1)$.

– Enfin, le graphe de Petersen est lui aussi un graphe 2-connexe extrémal. $(D = 2; |V| = 10; |E| = 15)$

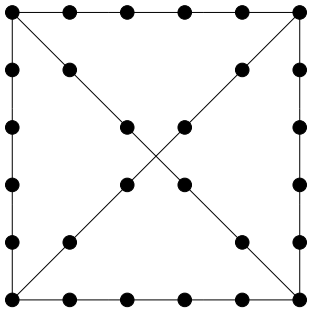


FIG. 1.2 – Graphe 2-connexe extrémal obtenu à partir d'un graphe complet à quatre sommets.

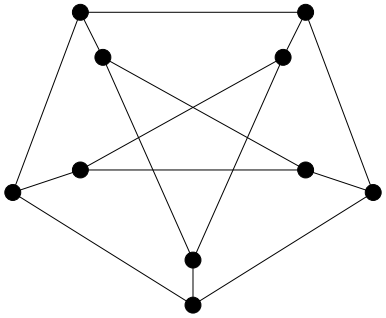


FIG. 1.3 – Graphe de Petersen

1.3.2 Notations

Soit $G = (V, E)$ un graphe non orienté. Pour tout sous-graphe $G' = (V', E')$ de G, on appelle $G - G'$ ou $G - V'$ le sous-graphe de G engendré par l'ensemble de sommets $(V - V')$.

Définition 1 (oreille) *Soit μ un chemin de G. μ est une oreille de G si tout sommet intérieur $x \in V_\mu^{int}$ de μ est de degré deux, et si les sommets extérieurs $y \in V_\mu^{ext}$ sont de degré au moins trois.*

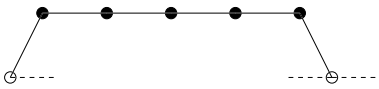


FIG. 1.4 – Exemple d'oreille.

Définition 2 (Décomposition sur un ensemble de sommets) *Soit S un ensemble de sommets de V et soit $V_1, V_2, \ldots$ les composantes connexes de $G - S$. Soit P_i le sous-graphe de G généré par $S \bigcup V_i$. La suite $P_1, P_2, \ldots$ est appelée décomposition de G sur l'ensemble S.*

1.4 Partition en chaînes d'un graphe G et double excentricité

Nous allons montrer ici comment un graphe 2-arête-connexe G peut se décomposer à partir d'un de ses sous-graphes et de chaînes dont la longueur est bornée par un entier k. Nous utiliserons le concept de *double excentricité* pour déterminer cet entier k. Le rapport entre le nombre d'arêtes et

le nombre de sommets des chaînes de la partition est supérieur à $\frac{k}{k-1}$. Nous utiliserons ce rapport pour déduire le coût en arête du graphe G.

Soit H_0 un graphe non orienté. Pour tout entier $k > 2$, on définit l'ensemble de graphes $\mathbb{E}_{(H_0,k)}$ par induction :

- H_0 est un élément de $\mathbb{E}_{(H_0,k)}$
- pour tout élément H de $\mathbb{E}_{(H_0,k)}$, si G est un graphe composé par H et par une chaîne de longueur inférieure ou égale à k reliant deux sommets de H, alors $G \in \mathbb{E}_{(H_0,k)}$

L'ensemble $\mathbb{E}_{(H_0,k)}$ que nous venons de définir contient tous les graphes qui peuvent se décomposer à partir de H_0 et de chaînes de longueur bornée par k.

La donnée de l'excentricité de G par rapport à un sous-graphe H_0 (voir la Définition 36 en page 146 de l'annexe) n'est pas suffisante pour déterminer l'entier k. Nous allons utiliser une grandeur dérivée de l'excentricité, la *double excentricité* pour déterminer k.

Définition 3 (double excentricité) *Soit S un ensemble de sommets non vide de V. On appelle* double excentricité de G par rapport à S, *notée $\beta(G,S)$ la grandeur suivante, définie à partir de l'excentricité $\alpha(G,S)$:*

- *si l'ensemble S contient un seul sommet, alors $\beta(G,S) = 2 \times \alpha(G,S) + 1$.*
- *si il existe un sommet s de S tel que $G - \{s\}$ est déconnecté, alors $\beta(G,S) = 2 \times \alpha(G,S) + 1$.*
- *si il existe deux sommets distincts s_1 et s_2 de S et deux sommets distincts x_1 et x_2 de V tels que $D(x_1,s_1) = \alpha(G,S)$, $D(x_1, S\backslash\{s_1\}) > \alpha(G,S)$, $D(x_2,s_2) = \alpha(G,S)$ et $D(x_2, S\backslash\{s_2\}) > M$, alors $\beta(G,S) = 2 \times \alpha(G,S) + 1$.*
- *enfin, si aucun de ces trois premier cas est vérifié, $\beta(G,S) = 2 \times \alpha(G,S)$.*

Les deux derniers cas sont illustrés à la Figure 1.5. Par abus de langage, si H est un sous-graphe de G ayant S pour ensemble de sommets, on notera $\beta(G,H) = \beta(G,S)$.

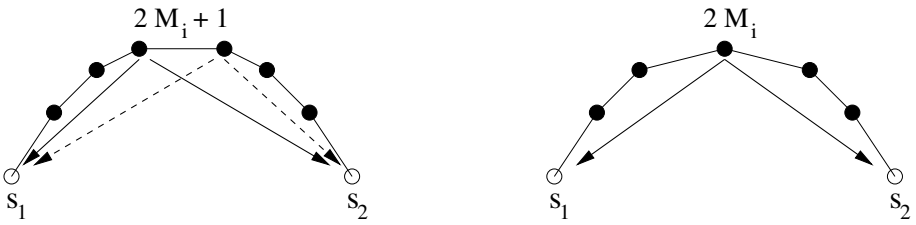


FIG. 1.5 – Définition de la double excentricité β d'une chaîne par rapport à ses deux extrémités s_1 et s_2. Le paramètre α représente l'excentricité de la chaîne par rapport à s_1 et s_2. La double excentricité de cette chaîne est égale à sa longueur.

Lemme 1 (borne sur la double excentricité) *Soit $G = (V,E)$ un graphe non orienté de diamètre D. Soit S un ensemble de sommets de V contenant au moins deux éléments. Soit $P_1 = (V_1,E_1), P_2 = (V_2,E_2)$... une décomposition de G sur l'ensemble S. Alors pour tout $i \neq j$, $\beta(P_i,S) + \beta(P_j,S) \leq 2D + 1$*

Preuve

Considérons deux des graphes P_i et P_j avec $i \neq j$. D'après la définition de la double excentricité, il existe deux sommets s_1 et s_2 dans S, un sommet x de V_i et un sommet y de V_j tels que $D(x,s_1) \geq$

$\lfloor\frac{\beta(P_i,S)}{2}\rfloor$, $D(y,s_2) \geq \lfloor\frac{\beta(P_j,S)}{2}\rfloor$, pour tout sommet $s \in S\backslash\{s_1\}$, $D(x,s) \geq \lceil\frac{\beta(P_i,S)}{2}\rceil$, et pour tout sommet $s \in S\backslash\{s_2\}$, $D(y,s) \geq \lceil\frac{\beta(P_j,S)}{2}\rceil$.

Ceci implique que

$$D(x,y) \geq \min(\lfloor\frac{\beta(P_i,S)}{2}\rfloor + \lceil\frac{\beta(P_j,S)}{2}\rceil, \lceil\frac{\beta(P_i,S)}{2}\rceil + \lfloor\frac{\beta(P_j,S)}{2}\rfloor)$$

$$D(x,y) \geq \lfloor\frac{\beta(P_i,S)+\beta(P_j,S)}{2}\rfloor.$$

Comme $D(x,y) \leq D$, on a $\lfloor\frac{\beta(P_i,S)+\beta(P_j,S)}{2}\rfloor \leq D$, d'où $\beta(P_i,S)+\beta(P_j,S) \leq 2D+1$. □

Lemme 2 (partition en chaînes) *Soit $G=(V,E)$ un graphe non orienté 2-arête-connexe. Soit S un sous-ensemble non vide de G, et soit $H_0=(S,A)$ le sous-graphe de G induit par S. Alors G fait partie de l'ensemble $\mathbb{E}_{(H_0,k)}$ avec $k=\beta(G,S)$.*

Preuve
Nous allons montrer par récurrence sur $\beta(G,S)$ et sur le nombre d'arêtes de $G-S$ que $G \in \mathbb{E}_{(H_0,k)}$ avec $k=\beta(G,S)$. Si $G=H_0$, alors $G \in \mathbb{E}_{(H_0,k)}$. Si $\beta(G,S)=0$ alors $G=H_0$. Nous supposons désormais que tout sous-graphe H de G contenant H_0, disjoint de G, tel que $\beta(H,S) \leq \beta(G,S)$ et tel que tous les sommets de $H-S$ sont de degré au moins deux appartient à l'ensemble $\mathbb{E}_{(H_0,k)}$.

- premier cas : on suppose que $\beta(G,S)$ est impair. Soit x un sommet de V tel que $D(x,S)=\beta(G,S)$. Soit y un voisin de x. On note μ la plus grande oreille de G contenant l'arête (x,y) telle que $V_\mu \bigcap S = \emptyset$. Chaque extrémité de μ est soit un sommet de degré au moins trois, soit un sommet de l'ensemble S. Le sous-graphe $H=G-\mu$ contient H_0, est disjoint de G, et est tel que tous les sommets de $H-S$ sont de degré au moins deux. De plus, pour tout sommet $z \in H-S$, le plus court chemin de z vers S dans G ne passe pas par x car $D(x,S) \geq D(z,S)$. Le plus court chemin de z vers S dans G ne passe donc pas par l'arête (x,y), ni par l'oreille μ. Aussi, $\alpha(H,S) \leq \alpha(G,S)$. D'après l'hypothèse de récurrence, H appartient à $\mathbb{E}_{(H_0,k)}$. Comme μ est une chaîne de longueur inférieure à k, G appartient également à $\mathbb{E}_{(H_0,k)}$.
- deuxième cas : on suppose que $\beta(G,S)$ est pair. On appelle V_α l'ensemble des sommets x de G tels que $D(x,S)=\alpha(G,S)$. Soit s un sommet de S tel que pour tout sommet $x \in V\backslash S$, $D(x,s) \leq \alpha(G,S)$. Comme $\beta(G,S)$ est pair, par définition $G-\{s\}$ est connexe. Soit x un sommet de V_α tel que la distance de x à $S\backslash\{s\}$ soit maximale dans $G-\{s\}$. Soit y un voisin de x. On note μ la plus grande oreille de G contenant l'arête (x,y) telle que $V_\mu \bigcap S = \emptyset$. Chaque extrémité de μ est soit un sommet de degré au moins trois, soit un sommet de l'ensemble S. Le sous-graphe $H=G-\mu$ contient H_0, est disjoint de G, et tous les sommets de $H-S$ sont de degré au moins deux. De plus, pour tout sommet $z \in H-S$, le plus court chemin de z vers le sommet s dans G ne passe pas par x car $D(x,s) \geq D(z,s)$. Le plus court chemin de z vers le sommet s dans G ne passe donc pas par l'arête (x,y), ni par l'oreille μ. Pour les mêmes raisons, le plus court chemin de z vers l'ensemble $S\backslash\{s\}$ dans $G-\{s\}$ ne passe pas par l'oreille μ. Aussi, $\beta(H,S) \leq \beta(G,S)$. D'après l'hypothèse de récurrence, H appartient à $\mathbb{E}_{(H_0,k)}$. Comme μ est une chaîne de longueur inférieure à k, G appartient également à $\mathbb{E}_{(H_0,k)}$.

□

1.5 Graphes 2-sommet-connexes

Soit $G = (V, E)$ un graphe 2-connexe de diamètre $D \geq 2$. On appelle $\ell_G = \max(0, |V| - (2D+1))$ et $k_G = |E| - |V|$. La propriété que nous voulons prouver, notamment $|E| \geq \lceil \frac{|V|D-(2D+1)}{D-1} \rceil$, est équivalente à la propriété $k_G \geq \lceil \frac{\ell_G}{D-1} \rceil$. Nous allons prouver par récurrence que $k_G \geq \lceil \frac{\ell_G}{D-1} \rceil$. Si G est un cycle, alors $\ell_G = 0$, et donc la propriété est vraie. Nous utilisons pour hypothèse de récurrence que tout graphe G' contenant moins d'arêtes que G et de diamètre inférieur ou égal à D vérifie la propriété $k_{G'} \geq \lceil \frac{\ell_{G'}}{D-1} \rceil$.

Soit C un plus petit cycle de G. Tous les cycles de G contiennent au moins autant de sommets que C. Nous étudierons le cas où $G - C$ n'est pas connexe, puis le cas où $G - C$ est connexe et C contient $2D$ sommets ou moins, et enfin le cas où C contient $2D + 1$ sommets.

1.5.1 Cas où un plus petit cycle de G déconnecte G

Lemme 3 *Soit C un plus petit cycle de G. Soit P_1, P_2, ... une décomposition de G sur les sommets de C. Le sous-graphe P_1 est 2-connexe et son diamètre est inférieur ou égal à D.*

Preuve
Soit x un sommet de P_1. Si x est aussi un sommet de C, $(P_1 - C)$ est connecté à C à travers deux sommets de C, donc $P_1 - \{x\}$ est connexe. Si x n'est pas un sommet de C, chaque composante connexe de $(P_1 - C) - \{x\}$ est connectée à C, donc $P_1 - \{x\}$ est connexe. Ceci prouve que P_1 est 2-connexe.

Soit y un sommet de P_1 distinct de x. Soit μ un plus court chemin de x vers y dans G. Si μ est entièrement inclus dans P_1, alors la distance entre x et y est la même dans G que dans P_1. Sinon, $\mu = \mu_1\mu_2\mu_3$ est tel que μ_1 et μ_3 sont entièrement dans P_1 et tel que μ_2 relie deux points a et b du cycle C. Comme C est un plus petit cycle de G, il existe un plus court chemin μ'_2 de a vers b entièrement inclus dans C. Le chemin $\mu_1\mu'_2\mu_3$ est un plus court chemin de x vers y entièrement inclus dans P_1 : la distance entre x et y est la même dans G que dans P_1. Ceci prouve que le diamètre de P_1 est inférieur ou égal au diamètre de G. □

Proposition 1 (C déconnecte G) *Soit $C = (V_C, E_C)$ un plus petit cycle de G. Si $G - C$ n'est pas connexe, alors $k_G \geq \lceil \frac{\ell_G}{D-1} \rceil$.*

Preuve
Soit $P_1 = (V_1, E_1), P_2 = (V_2, E_2), ..$ une décomposition de G sur les sommets de C. On appelle $\beta_i = \beta(P_i, C)$, $\ell_i = |V_i| - |V_C|$ et $k_i = |E_i| - |V_i|$. Le Lemme 1 indique qu'un seul des P_i (par exemple P_1) vérifie $\beta_i \geq D + 1$. G contient $|V_1| + \ell$ sommets et $|V_1 + \ell + k$ arêtes, avec $\ell = \Sigma_{i\neq 1}\ell_i$ et $k = \Sigma_{i\neq 1}k_i$. D'après le Lemme 2, pour tout $i \neq 1$, $k_i \geq \lceil \frac{\ell_i}{\beta_i - 1} \rceil$, ce qui implique que $k \geq \lceil \frac{\ell}{D-1} \rceil$. D'après le Lemme 3, P_1 est 2-connexe et son diamètre est inférieur ou égal à D. Par hypothèse de récurrence, $k_{P_1} \leq \lceil \frac{\ell_{P_1}}{D-1} \rceil$, ce qui implique que $k_G \leq \lceil \frac{\ell_G}{D-1} \rceil$. □

1.5.2 Cas où la taille d'un plus petit cycle est inférieure à $2D$

Lemme 4 *Soit C un plus petit cycle de G. Si C contient une oreille de taille supérieure ou égale à $D+1$, alors $k_G \geq \lceil \frac{\ell_G}{D-1} \rceil$.*

Preuve Soit $e_1 e_2 ... e_{D+1}$ une oreille de taille $D+1$ de C, reliant x_0 à x_{D+1}. Comme G est de diamètre D, quel que soit le sommet x de V on a : $D(x, x_{lfloor \frac{x_{D+1}}{2} \rfloor}) \leq D$ et $D(x, x_{\lceil \frac{x_{D+1}}{2} \rceil}) \leq D$. Aussi, pour tout x en dehors de C, on a $D(x, x_0) \leq \lceil \frac{D-1}{2} \rceil$ ou bien $D(x, x_{D+1}) \leq \lceil \frac{D-1}{2} \rceil$. On a donc $\alpha(G, C) \leq \lceil \frac{D-1}{2} \rceil$, ce qui implique $\beta(G, C) \leq D$. D'après le Lemme 2, G contient $V_C + \ell$ sommets et $|E_C| + \ell + k$ arêtes avec $k \geq \lceil \frac{\ell}{D-1} \rceil$. Comme C est 2-connexe et de diamètre inférieur ou égal à D, on a $k_G \leq \lceil \frac{\ell_G}{D-1} \rceil$. □

Lemme 5 *Soit C un cycle de G tel que $G - C$ est connexe. Soit μ une oreille de C reliant deux sommets de degré au moins 3. Alors $G - \mu$ est 2-connexe.*

Preuve
Soit s_1 et s_2 les deux sommets de C reliés par μ. Soit $x \in V \backslash V_\mu$. Si x appartient à C, alors $(C - \mu) - x$ contient au plus 2 composantes connexes. Ces deux composantes connexes sont reliées par s_1 et s_2 à $G - C$. $G - \mu\{x\}$ est donc connexe. Si x n'appartient pas au cycle C, et comme G est 2 connexe, chaque composante connexe de $G - C - x$ est reliée au cycle C (en dehors de la chaîne μ). Aussi, $G - \mu - \{x\}$ est connexe. □

Proposition 2 (grand oreille) *Soit C un plus petit cycle de G tel que $G - C$ est connexe. Si C contient une oreille couvrant plus de la moitié du cycle, alors $k_G \geq \lceil \frac{\ell_G}{D-1} \rceil$.*

Preuve
Soit μ une plus grande oreille de C couvrant plus de la moitié de C. Si μ est de longueur supérieure ou égale à $D+1$, alors d'après le Lemme 4, $k_G \geq \lceil \frac{\ell_G}{D-1} \rceil$. Sinon, on sait d'après le Lemme 5 que $G - \mu$ est 2-connexe. Comme μ couvre plus de la moitié du cycle C, le diamètre de $G - \mu$ est inférieur ou égal à D. Le fait que μ soit de taille inférieure à D implique que $G \in \mathbb{E}_{(G-\mu, D)}$ et donc $k_G \geq \lceil \frac{\ell_G}{D-1} \rceil$. □

1.5.3 Aplatissement d'un cycle

Nous allons décrire ici comment nous transformons un cycle minimal en arbre.

Définition 4 (aplatissement) *Soit C un cycle. Un aplatissement de C est un couple $(T, \sim)$ où T est un arbre contenant $\lceil \frac{|V_C|+1}{2} \rceil$, et $\sim$ est une relation d'équivalence telle que $T = \frac{C}{\sim}$.*

Lemme 6 (aplatissement correct) *Soit C un cycle et S un ensemble de sommets de C. Si la longueur de la plus grande oreille de C sans sommet dans S est inférieure ou égale à $\lfloor \frac{V_C}{2} \rfloor$, alors il existe un aplatissement $(T, \sim)$ de C tel que chaque feuille de T est une classe pour la relation d'équivalence $\sim$ contenant un sommet de degré 3 ou plus de G.*

Preuve
Si C contient un nombre pair de sommets, alors on nomme s_1 et s_2 deux sommets de S tels que $D(s_1, s_2)$ est maximal.

– si $D(s_1, s_2) = \frac{V_C}{2}$ alors on définit la relation $\sim$ sur les couples (x, y) de C de telle sorte que $(x \sim y) \Leftrightarrow D(x, s_1) = D(y, s_1)$. L'aplatissement correspondant est illustré à la Figure 1.6.

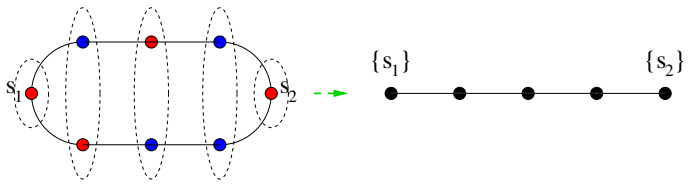


FIG. 1.6 – Aplatissement suivant deux sommets opposés

– sinon, il existe un sommet s_3 de S tel que chaque composante connexe de $C - \{s_1, s_2, s_3\}$ contient au plus $(\frac{|V_C|}{2} - 2)$ sommets. On note $D_1 = \frac{D(s_1,s_2)+D(s_1,s_3)-D(s_2,s_3)}{2}$, $D_2 = \frac{D(s_1,s_2)+D(s_2,s_3)-D(s_1,s_3)}{2}$ et $D_3 = \frac{D(s_1,s_3)+D(s_2,s_3)-D(s_1,s_2)}{2}$, et on a $D_1 + D_2 + D_3 = \frac{|V_C|}{2}$. On définit la relation $\sim$ sur les couples (x, y) de C de telle sorte que $(x \sim y)$ si et seulement s'il existe $i \in \{1, 2, 3\}$ tel que $D(x, s_i) = D(y, s_i) \leq D_i$. L'aplatissement correspondant est illustré à la Figure 1.7 Si $|V_C|$ est impair, on fusionne deux sommets de C (suivant une relation $\sim_f$)

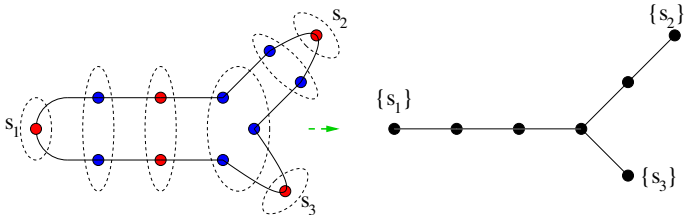


FIG. 1.7 – Aplatissement suivant trois sommets.

pour obtenir un cycle $C' = \frac{C}{\sim_f}$ contenant un nombre pair de sommets. La longueur maximale d'une oreille de C' sans sommet dans $\frac{S}{\sim_f}$ est au plus $\frac{|V_C|-1}{2}$. Il y a donc un aplatissement $(T, \sim)$ de C' tel que chaque feuille de T contient un élément de $\frac{S}{\sim_f}$. Aussi, $(T, \sim \circ \sim_f)^2$ est un aplatissement de C tel que chaque feuille de T est une classe contenant au moins un sommet de S.

□

1.5.3.1 Aplatissement d'un plus court cycle de G

Nous allons montrer qu'il existe un aplatissement d'un plus court cycle C G qui ne détruit pas la 2-connexité de G. Lorsque le cycle C contient moins de $2D$ sommets, nous en déduirons que $k_G \geq \lceil \frac{\ell_G}{D-1} \rceil$.

[2] deux sommets x et y de C sont en relation $x \sim \circ \sim_f y$ si les classes d'équivalence de x et de y pour la relation $\sim_f$, $\overline{x}$ et $\overline{y}$, vérifient $\overline{x} \sim \overline{y}$

Lemme 7 *S'il existe in cycle C de G tel que toute oreille de C soit de longueur inférieure ou égale à la moitié du cycle C, alors il existe une relation d'équivalence $\sim$ sur G tel que $\frac{G}{\sim}$ est un graphe 2-connexe contenant $|V| - \ell$ sommets et $|E| - \ell - k$ arêtes, avec $\ell = \lceil \frac{|V_C|-1}{2} \rceil$ et $k \geq 1$.*

Preuve
Soit S l'ensemble des sommets de degré supérieur ou égal à 3 de C. D'après le Lemme 6 (aplatissement correct), il existe un aplatissement $(T, \sim)$ de C tel que chaque feuille de T contient un sommet de S. Nous étendons la relation $\sim$ sur G de telle sorte que pour tout sommet x n'appartenant pas à C, et tout sommet y, $(x \sim y) \Leftrightarrow (x = y)$. Nous allons montrer à présent que $\frac{G}{\sim}$ est 3-connexe, et que G contient $|V| - \ell$ sommets et $|E| - \ell - k$ arêtes avec $\ell = \lceil \frac{|V_C|-1}{2} \rceil$ et $k \geq 1$. Soit x un sommet de $\frac{G}{\sim}$. Si x n'appartient pas à T, alors x est une classe qui ne contient qu'un seul sommet de G, et donc $\frac{G}{\sim} - \{x\}$ est connexe. Si x appartient à T, alors $T - \{x\}$ est une réunion d'arbres. Chacun de ces arbres est connecté par une feuille à $\frac{G-C}{\sim}$. Comme $G - C$ est connexe, $\frac{G}{\sim} - \{x\}$ est connexe également. □

Proposition 3 (plus petit cycle de taille inférieure à $2D$) *Soit C un plus petit cycle de G tel que $G - C$ est connexe. Si C contient au plus $2D$ sommets, alors $k_G \geq \lceil \frac{\ell_G}{D-1} \rceil$.*

Preuve
D'après la Proposition 2 (grande oreille), si une des oreilles de C couvre plus de la moitié de C alors $k_G \geq \lceil \frac{\ell}{D-1} \rceil$. Dans le cas contraire, et d'après le Lemme 7, il existe une relation d'équivalence $\sim$ sur G telle que $\frac{G}{\sim}$ est un graphe 2-connexe contenant $|V| - \ell$ sommets et $|E| - \ell - k$ arêtes, avec $\ell = \lceil \frac{|V_C|-1}{2} \rceil$ et $k \geq 1$. De plus, comme $\frac{G}{\sim}$ est obtenu par fusion de sommets de G, le diamètre de $\frac{G}{\sim}$ est inférieur ou égal à D. Si $|V_C| \leq 2D$, alors $k \geq \lceil \frac{\ell}{D-1} \rceil$. Aussi, on a $k_G \geq \lceil \frac{\ell_G}{D-1} \rceil$. □

1.5.4 Cas où la taille d'un plus petit cycle est $2D + 1$

Un plus petit cycle de G ne peut pas contenir $2D + 2$ sommets car cela violerait la contrainte imposée par le diamètre de G. Si un plus petit cycle de G contient exactement $2D + 1$ sommets, on peut en déduire des propriétés très fortes sur G.

Lemme 8 *Si la taille d'un plus petit cycle de G est $2D + 1$, alors chaque sommet de G appartient à un cycle de taille $2D + 1$.*

Preuve
Soit x un sommet de G. Soit y un sommet de G tel que $D(x, y)$ est maximal. On choisit un plus court chemin μ_1 de x vers y. Comme y est de degré au moins 2, il existe un voisin z de y qui n'est pas sur le chemin μ_1. On choisit un plus court chemin μ_2 de z à x. Le chemin $\mu = \mu_1(y, z)\mu_2$ est un chemin de longueur au plus $2D + 1$ reliant x à lui-même. Comme l'arête (y, z) n'apparaît qu'une seule fois dans ce chemin, μ contient un cycle de taille inférieure ou égale à $2D + 1$. Par hypothèse sur la taille d'un plus court cycle, μ ne peut être qu'un cycle de taille exactement $2D + 1$, et x appartient à ce cycle. □

Lemme 9 *Soit C un plus petit cycle de G de taille $2D + 1$. Si G est plus grand que ce cycle, alors tous les sommets de G sont de degré au moins 3.*

Preuve
On appelle $a_1a_2...a_{2D+1}$ les sommets du cycle C de telle sorte que pour tout i, (a_i, a_{i+1}) soit une arête du cycle (la dernière arête est (a_{2D+1}, a_0)). Comme G contient des sommets en dehors de C, et comme G est connexe, il existe au moins un sommet de C de degré au moins 3. On suppose sans perte de généralité que a_D est de degré au moins 3. C est un plus petit cycle de G, ce qui veut dire que $D(a_0, a_D) = D$. Soit x un voisin de a_D qui ne soit pas dans C et soit μ un plus court chemin de x vers a_0. Comme $D(x, a_0) \leq D$, le chemin μ ne peut pas passer par a_D. Comme G ne contient pas de cycle de taille inférieure à $2D$, le chemin μ ne peut passer par aucun sommet du cycle C. Ceci implique que a_0 est de degré au moins 3. De façon symétrique, a_{2D+1} est aussi de degré au moins 3. On peut refaire tout ce raisonnement après avoir fait une rotation de C de D sommets. Comme D et $2D + 1$ sont premiers entre eux, on peut montrer que tous les sommets de C sont de degré au moins 3. □

Proposition 4 (plus petit cycle de taille $2D + 1$) *Si un plus petit cycle de G contient $2D + 1$ sommets, alors $k_G \geq \lceil \frac{\ell_G}{D-1} \rceil$.*

Preuve
D'après les deux lemmes précédents, chaque sommet de G appartient à un cycle de longueur $2D+1$, et si G n'est pas réduit à un cycle, chaque sommet de G est de degré au moins 3. Si G est un cycle, alors on a immédiatement $k_G \geq \lceil \frac{\ell_G}{D-1} \rceil$. Sinon, on distingue deux cas :

- si $D \geq 3$, on a $2|E| \geq 3|V|$.
- si $D = 2$, on choisit un sommet s de plus haut degré dans G, et on divise les autres sommets en deux ensembles : $D_1 = \{x \in G \text{ tel que } D(s, x) = 1\}$ et $D_2 = \{y \in G \text{ tel que } D(s, y) = 2\}$. G ne contient pas de cycle de taille inférieure à 4. Ceci implique
$$\forall \{x_1, x_2\} \subset D_1, x_1 \neq x_2, \quad (x_1x_2) \notin E$$
$$\forall y \in D_2 \ \exists!\ x \in D_1, \qquad (xy) \in E$$

  On peut donc diviser D_2 en une union disjointe :
$$D_2 = \bigcup_{x \in D_1} D_2^x \quad \text{avec} \quad D_2^x = \{y\ in D_2 \text{ — } D(x, y) = 1\}$$

  Si s est de degré 3, G contient exactement 10 sommets et 15 arêtes (G est le graphe de Petersen), et donc $k_G \geq \ell_G$. Si s est de degré supérieur ou égal à 4, on définit $\forall x \in D_1, r_x = deg(x)-3$ et $r = \Sigma_{x \in D_1} r_x$. On a $|V| = 1+deg(s)+(2deg(s)+r)$ c'est à dire $|V|-1 = 3deg(s)+r$. Comme
$$\forall x_1 \in D_1, \forall y_1 \in D_2^{x_1}, \forall x_2 \in D_1 \backslash \{x_1\}, \quad D(y_1, x_2) \leq 2,$$

  le degré de chaque sommet de D_2 est de degré supérieur ou égal à $|D_1| \geq 4$. On somme les degrés des sommets de G et on obtient :
$$\begin{array}{rcccccc} 2|E| & = & deg(s) & + & \Sigma_{x \in D_1} deg(x) & + & \Sigma_{y \in D_2} deg(y) \\ 2|E| & \geq & deg(s) & + & deg(s) \times 3 & + & (deg(s) \times 2 + r) \times 4 \\ 2|E| & \geq & 4(deg(s) \times 3 + r) & & & & \\ 2|E| & \geq & 4(|V| - 1) & & & & \\ |E| - |V| & \geq & |V| - 2 & & & & \end{array}$$

Aussi, $k_G \geq \lceil \frac{\ell_G}{D-1} \rceil$. □

1.5.5 Récapitulation de la preuve

Théorème 2 *Soit $G = (V, E)$ un graphe non orienté 2-connexe de diamètre $D \geq 2$. Alors $|E| \geq \lceil \frac{|V|D-(2D+1)}{D-1} \rceil$.*

Preuve
Si G est un cycle alors $k_G = \ell_G = 0$. Sinon, on suppose que chaque graphe $G' = (V', E')$ 2-connexe de diamètre inférieur ou égal à D et contenant strictement moins que $|E|$ arêtes vérifie $k_{G'} \geq \lceil \frac{\ell_{G'}}{D-1} \rceil$. Soit C un plus petit cycle de G :

- si $G - C$ n'est pas connnexe, la Proposition 1 (C déconnecte G) implique que $k_G \geq \lceil \frac{\ell_G}{D-1} \rceil$.
- si $G - C$ est connexe et si C contient $2D$ sommets ou moins, alors la Proposition 3 (plus petit cycle de taille inférieure à $2D$) implique que $k_G \geq \lceil \frac{\ell_G}{D-1} \rceil$.
- si C contient $2D + 1$ sommets la Proposition 4 (plus petit cycle de taille $2D + 1$) implique que $k_G \geq \lceil \frac{\ell_G}{D-1} \rceil$.

L'inégalité $k_G \geq \lceil \frac{\ell_G}{D-1} \rceil$ est équivalente à $|E| \geq \lceil \frac{|V|D-(2D+1)}{D-1} \rceil$. □

1.6 Graphes 2-arête connexes

Nous allons étudier maintenant les graphes 2-arêtes connexes. Un graphe 2-arête connexe reste connexe même lorsqu'une de ses arêtes disparaît. Soit $G = (V, E)$ un graphe non orienté 2-arête-connexe de diamètre $D \geq 2$. Nous savons d'après le Théorème 2, que nous avons démontré à la section précédente, que si G est également 2-sommet-connexe, $|E| \geq \lceil \frac{|V|D-(2D+1)}{D-1} \rceil$. Il nous reste à étudier le cas où un sommet s déconnecte G. Nous allons d'abord exhiber une classe de graphes $G = (V, E)$ 2-arête connexes de diamètre pair et tels que $|E| = \lceil \frac{(n-1)(D+1)}{D} \rceil$.

Théorème 3 *Soit $D \geq 2$ un entier pair et $n \geq 1$. Il existe un graphe 2-arête connexe $J_{D,n}$ de diamètre D qui contient exactement n arêtes et $\lceil \frac{(n-1)(D+1)}{D} \rceil$ arêtes.*

Preuve
On appelle k et m le quotient et le reste de la division euclidienne de $n - 1$ par D. On a $n = 1 + kD + m$. On appelle $J_{D,n}$ le graphe défini de la manière suivante :

- x est le sommet central
- k oreilles de longueur $D + 1$ relient x à lui-même
- si $m > 0$, une oreille de longueur $m + 1$ relie x à lui-même.

$J_{D,n}$ contient n sommets et $\lceil \frac{(n-1)(D+1)}{D} \rceil$ arêtes. $J_{D,n}$ est 2-arête-connexe et de diamètre D. □

Nous allons à présent démontrer qu'un graphe $G = (V, E)$ 2-arête connexe et de diamètre D contient au minimum $\min(\lceil \frac{|V|D-(2D+1)}{D-1} \rceil, \lceil \frac{(|V|-1)(D+1)}{D} \rceil)$ arêtes.

Théorème 4 *Soit $G = (V, E)$ un graphe non orienté 2-connexe de diamètre $D \geq 2$. Si D est pair alors $|E| \geq \lceil \frac{|V|D-(2D+1)}{D-1} \rceil$. Si D est impair alors $|E| \geq \min(\lceil \frac{|V|D-(2D+1)}{D-1} \rceil, \lceil \frac{(|V|-1)(D+1)}{D} \rceil)$.*

Preuve
Si G est 2-connexe alors, d'après le Théorème 2, on a $|E| \geq \lceil \frac{|V|D-(2D+1)}{D-1} \rceil$. Sinon, il existe $S \in V$

tel que $G-\{s\}$ n'est pas connexe. Soit $P_1=(V_1,E_1)$, $P_2=(V_2,E_2)$, .. une décomposition de G sur $\{s\}$. Soit $\beta_i=\beta(P_i,\{s\})$, $\ell_i=|V_i|-1$ et $k_i=|E_i|-\ell_i$. On suppose sans perte de généralité que β_1 est maximal parmi les β_i.

- si $\lfloor\frac{\beta_1}{2}\rfloor<\frac{D}{2}$, d'après le Lemme 2, pour tout i on a $k_i\geq\lceil\frac{\ell_i}{D-1}\rceil$ ce qui implique que $|E|\geq\lceil\frac{|V|D-(2D+1)}{D-1}\rceil$.
- si $\lfloor\frac{\beta_1}{2}\rfloor=\frac{D}{2}$ (D doit être pair), alors pour tout i on a $\beta_i\leq(D+1)$ et d'après le Lemme 2, $\forall i$ $k_i\geq\lceil\frac{\ell_i}{D}\rceil$ d'où $|E|\geq\lceil\frac{(|V|-1)(D+1)}{D}\rceil$.
- enfin, si $\lfloor\frac{\beta_1}{2}\rfloor>\frac{D}{2}$, alors $\forall i\neq 1$, $\beta_i\leq D$. G contient $|V_1|+\ell$ sommets et $|E_1|+\ell+k$ arêtes avec $\ell=\Sigma_{i\neq 1}\ell_i$ et $k=\Sigma_{i\neq 1}k_i$. D'après le Lemme 2, pour tout $i\neq 1$, $k_i\geq\lceil\frac{\ell_i}{D-1}\rceil$, ce qui implique que $k\geq\lceil\frac{\ell}{D-1}\rceil$. On peut voir que P_1 est 2-arête connexe et de diamètre inférieur ou égal à D, ce qui implique que les bornes sur $|E|$ sont vérifiées pour P_1 :
  - si D est impair, alors $|E_{P_1}|\geq\lceil\frac{|V_{P_1}|D-(2D+1)}{D-1}\rceil$ d'où $|E|\geq\lceil\frac{|V|D-(2D+1)}{D-1}\rceil$.
  - si D est pair, alors $|E_{P_1}|\geq\min(\lceil\frac{|V_{P_1}|D-(2D+1)}{D-1}\rceil,\lceil\frac{(|V_{P_1}|-1)(D+1)}{D}\rceil)$, d'où $|E|\geq\min(\lceil\frac{|V|D-(2D+1)}{D-1}\rceil,\lceil\frac{(|V|-1)(D+1)}{D}\rceil)$.

□

1.7 Conclusion et perspectives

Dans ce chapitre nous avons résolu un problème posé par France Télécom R&D concernant la résistance aux pannes des réseaux optiques. Nous avons déterminé le coût minimum d'un réseau qui résiste à une panne, et dont les paramètres sont le nombre de nœuds et la distance maximale entre deux nœuds.

Du point de vue théorique, nous avons démontré une conjecture de Bollobás [6] de 1968 sur le nombre d'arêtes d'un réseau 2-connexe et de diamètre fixé. Les techniques que nous avons employées sont essentiellement des techniques classiques pour prouver des propriétés sur des graphes, nous avons cependant introduis les notions de double excentricité et d'aplatissement. La notion de double excentricité peut éventuellement s'étendre à d'autres problèmes concernant les graphes deux-connexes, tandis que la notion d'aplatissement est un morphisme surjectif, ce qui élargit les notions classiques d'isomorphisme pour les graphes (que l'on retrouve par exemple en [22]).

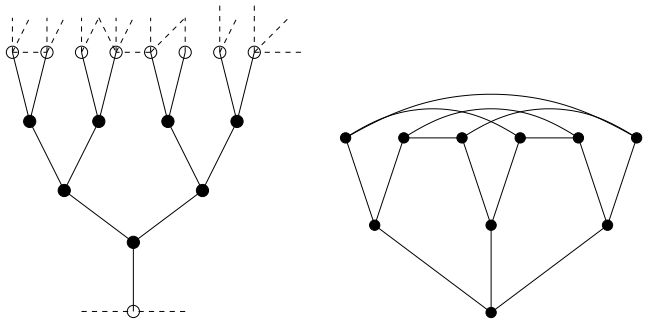
Il serait intéressant de pouvoir énumérer l'ensemble des réseaux optimaux pour ce problème. Au cours de la démonstration de la conjecture, on peut s'apercevoir que les graphes extrémaux ne s'obtiennent pas uniquement par induction en ajoutant des chaînes à un cycle. Les graphes extrémaux qui étaient déjà connus suivent cette construction. Nous avons découvert que les graphes extrémaux peuvent aussi s'obtenir en créant des cycles (contraire de l'aplatissement) à partir de graphes extrémaux plus petits. Les seuls graphes extrémaux irréductibles sont les cycles et le graphe de Petersen.

Il reste également à résoudre le problème plus général sur les réseaux pouvant résister à plusieurs pannes. Les techniques de preuve que nous avons employé ne s'étendront pas sans difficulté au cas général, car les notions élémentaires mises en jeu se transforment en des objets beaucoup plus complexes dans le cas de la k-connexité :

- l'équivalent d'une chaîne serait un arbre de degré k. Un arbre de degré 3 et de profondeur 3

est représenté à la Figure 1.8 (a). Des arbres de ce type seraient élémentaires pour la partition des graphes extrémaux.

– les graphes extrémaux irréductibles seraient formés par la réunion de plusieurs arbres. Ceci est illustré par le Figure 1.8 (b).



(a) (b)

FIG. 1.8 – Équivalents de la chaîne et du cycle lorsque le graphe est 3-connexe : (a) arbre de degré 3 et de profondeur 3, brique de base de graphes extrémaux 3-connexes de diamètre 6. (b) équivalent du cycle lorsque le graphe est 3 connexe de diamètre 2. Il s'agit du graphe de Petersen.

Chapitre 2

Routage dans les réseaux optiques : chemins disjoints

Sommaire

2.1 Introduction

Le rôle principal d'un réseau de télécommunications est de permettre des connexions entre différents nœuds du réseau. Celles-ci s'expriment par des requêtes que le réseau s'efforce de satisfaire. Une requête est la donnée d'un émetteur, d'un récepteur et d'un débit à transmettre. Réaliser les requêtes consiste à trouver pour chaque requête un ensemble de chemins sur lesquels la connexion va s'établir. La réalisation des requêtes doit être compatible avec les contraintes de capacité du

réseau. On représente usuellement un réseau par un graphe orienté (ou par un graphe non orienté), les sommets du graphe représentent les nœuds du réseau, et les arcs (ou les arêtes) représentent les liens entre le nœuds du réseau. De plus, chaque sommet et chaque arc (ou chaque arête) est muni d'une capacité qui représente la quantité maximale d'information qui peut transiter sur celui-ci à un instant donné. Le fait de pouvoir ou non réaliser des requêtes sur un réseau correspond au problème du *routage* sur un graphe muni de capacités. Il s'agit d'un problème très classique de théorie des graphes [37]. Plusieurs cas de figures se présentent pour le problème du routage :

- si les requêtes correspondent à un échange d'information, et si cette information peut être librement divisée en morceaux qui circuleront indépendamment sur le réseau, on parle de *multiflot fractionnaire*. Ce type de requêtes correspond par exemple au téléchargement de fichiers sur internet.
- si les requêtes correspondent à l'établissement d'un canal de communication réservé sur le réseau, on parle de *chemins disjoints*. Autrefois, ce type de requêtes était utilisé pour les communications téléphoniques. Aujourd'hui, il est toujours utilisé pour établir des chemins réservés dans un réseau. Ces chemins seront ensuite utilisés pour former un réseau logique de type ATM[1], par exemple pour une application spécifique.
- enfin, entre les deux configurations mentionnées plus haut, chaque requête peut correspondre à l'établissement de plusieurs chemins, et on parle alors de *multiflot entier*.

Le problème du multiflot fractionnaire peut s'écrire sous forme de programme linéaire [54], et de nombreux algorithmes polynomiaux existent pour résoudre ce problème [15]. Le problème du multiflot entier et celui des chemins disjoints font quand à eux partie de la classe des problèmes NP-difficiles ; des algorithmes fournissant des solutions approchées existent, le plus souvent basés sur des solutions fractionnaires [39, 55, 64].

Dans ce chapitre, nous allons étudier le problème des chemins disjoints, alors que le chapitre 3 est consacré au problème du multiflot entier. Dans les deux chapitres, nous allons utiliser une propriété spécifique au réseaux optiques. Les équipements de télécommunication optiques sont directionnels, c'est à dire que lorsqu'un réseau optique est installé, chaque élément permet l'envoi ou la réception d'informations dans une direction donnée. Cependant, les équipements optiques sont constitués de paires d'éléments opposés, ce qui implique que la quantité d'information que l'on peut envoyer dans une direction est égale à la quantité d'information que l'on peut recevoir depuis cette direction. On peut donc représenter un réseau optique par un graphe orienté symétrique. Les sommets du graphe représentent les nœuds du réseau, les arcs représentent les liens optiques. La symétrie du graphe signifie que l'existence d'un arc implique l'existence de l'arc opposé.

Nous nous intéressons au problème suivant : réaliser simultanément un certain nombre de requêtes dans un réseau de télécommunications en respectant les contraintes de capacité des liens. Dans le cas où les requêtes sont de débit unitaire, et les liens de capacité unitaire, ce problème revient à chercher dans un graphe orienté (ou non orienté) des chemins arc-disjoints (ou arête-disjoints). Si le nombre de requêtes n'est pas borné, le problème est NP-complet. Dans le cas orienté, c'est un problème NP-complet même pour 2 requêtes [36]. Si le graphe est non orienté et si le nombre de requêtes ℓ est fixé, il existe un algorithme en temps polynomial pour résoudre le problème [86]. Nous montrons ici que sous les mêmes conditions (nombre de requêtes fixé) le problème est polynomial dans le cas d'un graphe orienté symétrique. Ceci répond à une question posée par le P. Chanas [13].
Remarque : l'identité de chaque requête est fondamentale : en effet, si on admet une permutation sur les sommets émetteurs ou récepteurs, c'est à dire si on veut réaliser des chemins disjoints entre

[1] Asynchronous Transfer Mode [13, 14]

deux ensembles de sommets, on se ramène à un problème de flot maximum ou à un problème de connexité (voir la Figure 2.1).

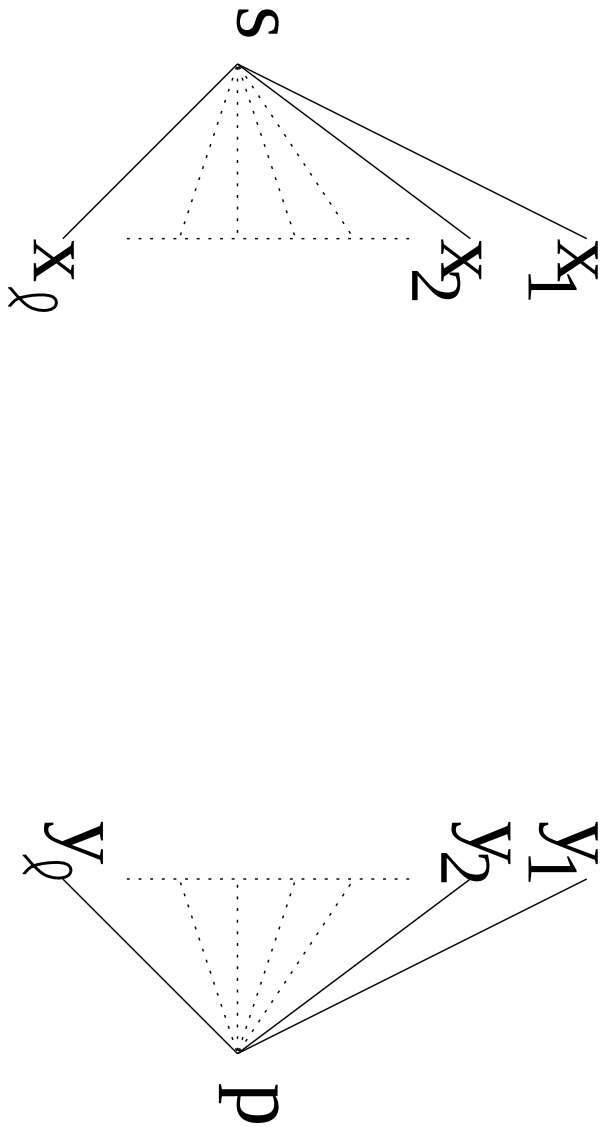


FIG. 2.1 – Satisfaire les requêtes (x_1, y_1), (x_2, y_2), ... (x_ℓ, y_ℓ) ne revient pas à trouver un flot maximum entre une source virtuelle s et un puits virtuel p ; en effet, les destinataires ne doivent pas être intervertis.

Contenu du chapitre

Nous présentons d'abord une synthèse des résultats antérieurs sur les diverses variantes du problème des chemins disjoints, puis nous évoquerons les concepts de mineurs, homéomorphismes et motifs concernant le problème des chemins arc-disjoints dans un graphe orienté symétrique, et enfin nous montrerons que le problème des chemins arc-disjoints dans un graphe symétriques est polynomial pour un nombre borné de requêtes.

2.2 Résultats antérieurs sur le problème des chemins disjoints

Le problème des chemins disjoints a été largement étudié dans la littérature scientifique pour deux raisons. D'abord, de nombreux problèmes de chemins disjoints surgissent naturellement en théorie des graphes et optimisation combinatoire, notamment dans le cadre de la théorie des flots. D'autre part (et c'est le cas dans cette thèse), il y a diverses applications pratiques qui posent des problèmes de chemins disjoints.

Les problèmes de chemins disjoints recouvrent en fait 4 grandes catégories, selon que les chemins doivent être sommet ou (arc/arête) disjoints, et selon que le graphe est orienté on non. Nous allons d'abord présenter les relations entre problèmes de chemins sommet-disjoints et leur les chemins (arc/arête)-disjoints. Nous présenterons ensuite les résultats concernant les chemins arc-disjoints dans les graphes orientés, puis les résultats concernant les chemins arête-disjoints dans les graphes non-orientés, et enfin le problème des chemins arc-disjoints dans les graphe orientés symétriques.

2.2.1 Chemins sommets disjoints et chemins (arc/arêtes) disjoints

Le problème des chemins sommets disjoints est naturellement proche du problème des chemins (arc/arêtes) disjoints.

Dans le cadre des graphes orientés, ces deux problèmes sont presque identiques [61].

Théorème 5 *Soit un graphe orienté G et un ensemble de requêtes R sur G. Alors il existe un graphe orienté G' et un graphe orienté G''' que l'on peut construire à partir de G en temps polynomial, tel que :*

- *si G est acyclique, alors G' et G'' sont acycliques.*
- *il existe une solution au problème des chemins sommet-disjoints sur G pour les requêtes R si et seulement si il existe une solution au problème des chemins arc-disjoints sur G' pour les requêtes R.*
- *il existe une solution au problème des chemins arc-disjoints sur G pour les requêtes R si et seulement si il existe une solution au problème des chemins sommet-disjoints sur G'' pour les requêtes R.*

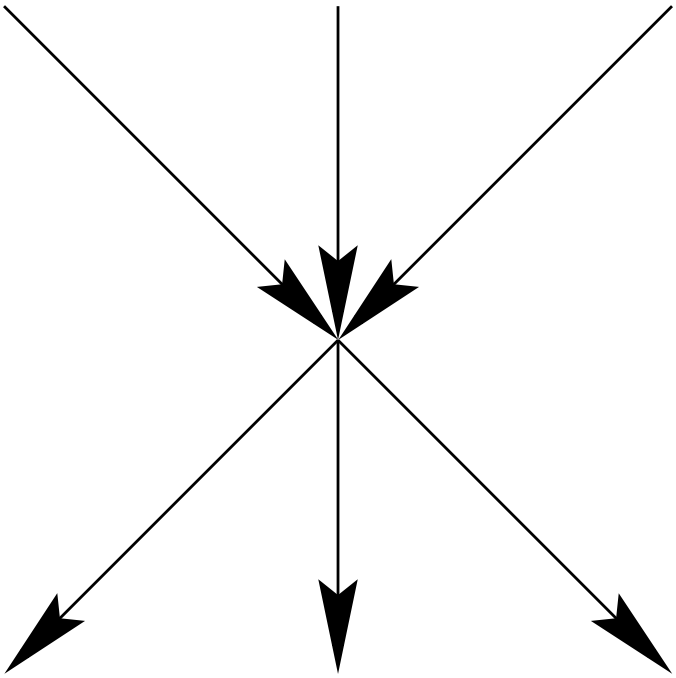
Preuve

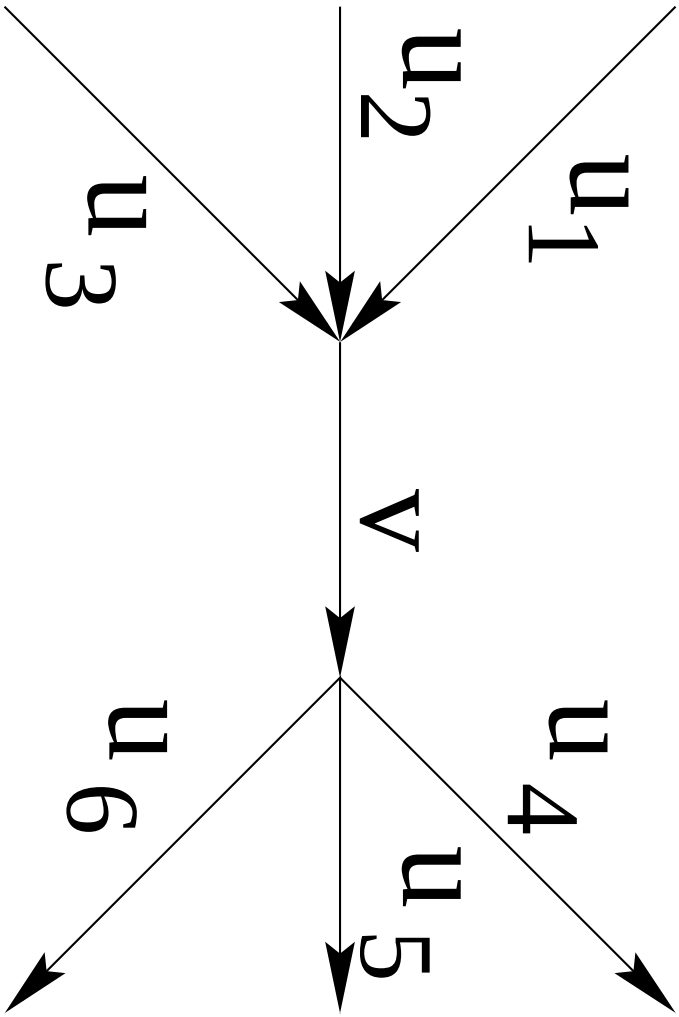
- **Réduction chemins sommet-disjoints → arc-disjoints :** chaque sommet de G ayant un degré entrant et un degré sortant supérieurs à un est dédoublé dans G'.
- **Réduction chemins arc-disjoints → chemins sommet-disjoints :** On construit le graphe G'' des arcs de G contenant $(\mathrm{E}_G + |R|)$ sommets.

□

On peut remarquer qu'en plus de l'acyclicité, de nombreuses propriétés de graphes sont identiques sur G, G', et G''.

Pour les graphes non orientés, la réduction chemins arête-disjoints → chemins sommet-disjoints, est identique au cas orienté. La réduction inverse (voir l'article [86]) est légèrement plus délicate, et




u2
u3
u1
v
u6
u4
u5

impose d'augmenter le nombre de requêtes (de manière polynomiale), ce qui fait que le problème des chemins sommet-disjoints est légèrement plus difficile.

2.2.2 Chemins arc-disjoints dans les graphes orientés

En 1980, Fortune, Hopcroft et Wyllie ont montré que trouver deux chemins arc-disjoints dans un graphe orienté est un problème NP-complet [36]. Voici un tableau regroupant quelques principales variantes du problème des chemins disjoints pour les graphes orientés.

Requêtes	Autres hypothèses	Difficulté du problème	Références
deux	-	NP-complet	[36]
nombre non borné	G acyclique	NP-complet	[23]
nombre borné	G acyclique	polynomial	[36]
trois	$(G+R)$ eulérien	polynomial	[47]
sur 5 sommets	$(G+R)$ eulérien	NP-complet	[92]
nombre non borné	$(G+R)$ planaire	NP-complet	[94]
nombre non borné	G planaire et acyclique	NP-complet	[94]
destinations sur une face	G planaire et acyclique, $(G+R)$ eulérien	polynomial	[75]

2.2.3 Chemins arête-disjoints dans les graphes non orientés

En 1975, Karp [53] a montré que le problème des chemins disjoints est NP-complet dans les graphes non orientés. En 1993, Middendorf et Pfeiffer [68] ont montré que le problème est NP-complet même si $(G+R)$ est planaire[2]. En 2001, Nishizeki, Vygen et Xiao Zhou [76] ont montré que le problème est NP-complet même si G est série parallèle. De façon positive, si le nombre de requêtes est borné, Robertson et Seymour [86] donnent en 1995 un algorithme polynomial (en $O(|V|)^3$) en réduisant ce problème à celui du *δ-folio containment* qui consiste à énumérer tous les *mineurs étiquetés* d'une certaine taille d'un graphe.

Théorème 6 (Robertson & Seymour, 1986) *Si le nombre de requêtes est borné, il existe un algorithme en $O(|V|^3)$ pour le problème des chemins sommet-disjoints dans un graphe non-orienté.*

2.2.4 Résultats antérieurs sur les graphes orientés symétriques

Le problème des chemins arc-disjoints dans les graphes orientés symétriques a été abordés par Chanas [13] dans le cadre d'une étude sur les réseaux ATM. Le problème est NP-difficile lorsque le nombre de requêtes n'est pas borné. Il existe par contre un algorithme assez simple pour résoudre deux requêtes.

[2] $G+R$ est le graphe formé par G plus les arêtes correspondant aux requêtes de R.

Théorème 7 *Si le nombre de requêtes fait partie de l'instance, le problème des chemins disjoints dans un graphe orienté symétrique est NP-complet.*

Preuve
La preuve se fait à partir d'une réduction du cas orienté général :
Soit G un graphe orienté et $(x_1, y_1)...(x_\ell, y_\ell)$ un ensemble de requêtes. On note G^* le symétrisé de G (xy ou $yx \in E_G \Rightarrow xy$ et $yx \in E_{G^*}$). $\forall xy \in E_{G^*} \setminus E_G$ on ajoute la requête (x, y). Notons qu'une solution sur G en induit une sur G^*. Réciproquement, on peut forcer toute solution sur G^* à utiliser l'arc xy pour router la requête (x, y). Le coût de la réduction correspond à celui de la symétrisation, c'est à dire $O(|V_G| + |E_G|)$. □

Théorème 8 *Il existe un algorithme polynomial permettant de résoudre le problème des 2 chemins arc-disjoints dans un graphe orienté symétrique.*

Preuve
Soit G un graphe orienté symétrique et $(x_1, y_1), (x_2, y_2)$ deux requêtes.
On cherche un chemin (élémentaire) de x_1 à y_1 (complexité en $O(|E| + |V|)$) et on construit le graphe des écarts (Ford & Fulkerson). On cherche un chemin de x_2 à y_2 dans le graphe des écarts ($O(|E| + |V|)$). Si ce dernier chemin n'existe pas, il n'y a pas de 2-flot de (x_1, x_2) vers (y_1, y_2) , et le problème n'a pas de solution (Ford & Fulkerson). Si ce chemin existe, il correspond à un chemin réel de G disjoint du précédent ; en effet, la symétrie du graphe assure que le graphe des écarts a la même connexité que le graphe originel privé du premier chemin. □

Récapitulation

Voici deux tableaux récapitulatifs sur la complexité du problème des chemins disjoints, suivant le type du graphe et le nombre de requêtes.
Chemins sommet-disjoints :

	Graphe	
Nombre de requêtes	orienté	non orienté
une requête	P	P
deux requêtes et plus	NP	P
nombre indéterminé de requêtes	NP	NP

Chemins arc-disjoints :

	Graphe		
Nombre de requêtes	orienté	non orienté	orienté symétrique
une requête	P	P	P
deux requêtes et plus	NP	P	P
nombre indéterminé de requêtes	NP	NP	NP

La classe de complexité du problème pour un graphe orienté symétrique lorsque le nombre de requêtes est borné et supérieur ou égal à trois était un problème ouvert (cf [13]). Nous déterminons cette classe en montrant qu'il existe un algorithme polynomial dans ce cas.

	Graphe		
Nombre de requêtes	non-orienté	orienté	symétrique
1	P	P	P
2	P	NP	P
3	P	NP	P
borné	P	NP	P
non borné	NP	NP	NP

2.3 Problème des chemins disjoints, mineurs, motifs

Dans leur article intitulé *Graph Minors, XIII, the disjoint path problem* [86], Robertson et Seymour relient le problème des chemins sommets disjoints dans les graphes non-orienté au problème des *mineurs*, et plus généralement au problème du *δ-folio containment*. Nous définirons une notion voisine, les *motifs*, puis nous allons expliciter leur relation avec le problème des chemins arc-disjoints.

2.3.1 Mineurs et δ-folio containment

De façon concise, un graphe non orienté H est un mineur d'un autre graphe G si on peut l'obtenir en supprimant des sommets de G, en supprimant des arêtes de G, et en contractant des arêtes[3]. Le problème de savoir si un graphe H est le mineur d'un graphe G est NP-complet : il s'agit d'une généralisation du problème d'isomorphisme de graphe.

Un graphe H est un mineur *étiqueté* de G sur un ensemble de sommets S si H est un mineur de G et si certains sommets de H (ceux appartenant à S) correspondent exactement à certains sommets de G. Par exemple, le fait de trouver un chemin entre deux sommets x et y dans un graphe G correspond à décider si le graphe composé des sommets x, y et de l'arête (x, y) est un mineur de G étiqueté en x et y. De façon naturelle, il a été montré que le problème des chemins sommet-disjoints dans un graphe G revenait à décider si un graphe particulier (le graphe des demandes) était un mineur étiqueté de G.

Le problème du *delta-folio containment* est une généralisation du problème des mineurs étiquetés[4]. Robertson et Seymour ont montré qu'on pouvait le résoudre en temps polynomial lorsque la taille des mineurs est bornée. En fait, la méthode proposée consiste (entre autres) à énumérer tous les mineurs de taille k du graphe G, et à les comparer aux mineurs proposés.

Ce résultat sur les mineurs a montré que le problème des chemins sommet-disjoints est polyno-

[3] contracter une arête (x, y) signifie fusionner les sommets x et y.

[4] il s'agit de résoudre le problème des mineurs étiquetés pour une famille de candidats. Pour plus de détails, voir [86].

mial pour un nombre borné de requêtes, et partant, que le problème des chemins arêtes disjoints est lui aussi polynomial pour un nombre borné de requêtes.

2.3.2 Homéomorphismes de sous-graphes

La notion d'homéomorphisme de sous-graphe est très similaire à la notion de mineur. Un graphe H est homéomorphe à un graphe G si les sommets de H et de G sont les mêmes, et si chaque arête (ou arc) de H correspond à un chemin de G, et que chaque arête (ou arc) de G appartient exactement à un de ces chemins. Le problème des chemins arête (ou arc) - disjoints dans un graphe G revient à déterminer si un certain graphe H est homéomorphe à un des sous-graphes de G. Dans le cadre des graphes non orientés, ce problème peut se réduire au problème du δ-*folio containment* [86]. Dans le cadre des graphes orientés, Fortune, Hopcroft et Wyllie [36] ont montré que ce problème était polynomial si H est une étoile (équivalent à un problème de flot) et NP-complet sinon.

2.3.3 Motifs d'un graphe

La notion de motifs reprend les idées d'homéomorphisme de graphe, et fait le lien entre les graphes orientés symétriques et le graphe correspondant non orienté.

Définition 5 (Symétrisé,squelette) *Soit G=(V,E) un graphe non-orienté.*
- *On note $E^* = \{arcs\ xy, yx$ — $l'arête\ x, y \in E\}$.*
- *G^*=(V,E^*) est appelé **symétrisé** de G.*
- *G est appelé **squelette** de G^*.*

Définition 6 (Motif, réduction)
- *Soit G=(V,E) et $\hat{G}$=($\hat{V}$,$\hat{E}$) deux graphes non-orientés.*
- *Soit R et $\hat{R}$ deux ensembles de couples de sommet de G et $\hat{G}$.*

*($\hat{G}$,$\hat{R}$) est appelé **motif** de (G,R) si*

- *il existe une injection i de $\hat{V}$ dans V telle que i($\hat{R}$) = R ; il existe un ensemble de chemins arête-disjoints dans G reliant i(x) à i(y) pour tout $xy \in \hat{E}$*
- *$\hat{R}$ est un ensemble de requêtes réalisables dans $\hat{G}^*$, le symétrisé de $\hat{G}$.*

La relation que nous avons définie est transitive, et nous la notons ($\hat{\text{G}}$,$\hat{\text{R}}$) < (G,R). Si card $\hat{\text{E}}$ < card E, on dira que ($\hat{\text{G}}$,$\hat{\text{R}}$) est une **réduction** de (G,R).

Définition 7 (Motif minimal) *Soit G=(V,E) un graphe non-orienté, et R un ensemble de couples de V.*
*(G,R) est appelé **motif minimal** si*

- *R est un ensemble de requêtes réalisables dans G^*, le symétrisé de G.*
- *(G,R) n'admet pas de réduction.*

Un motif de (G, R) est donc un couple $(\hat{G}, \hat{R})$ tel que $\hat{G}$ est homéomorphe à un sous-graphe de G, et tel que l'image de $\hat{R}$ par ce morphisme est R. Une condition supplémentaire impose qu'il

existe des chemins arc-disjoints pour les requêtes $\hat{R}$ dans le graphe symétrique $\hat{G}^*$. L'idée est qu'un motif minimal représente les *nœuds*[5] que forment des chemins arc-disjoints dans un graphe orienté symétrique.

2.3.4 Relation entre les motifs et le problème des chemins arc-disjoints

Nous allons nous attacher ici aux motifs minimaux pour ℓ requêtes. On appelle F_ℓ l'ensemble des motifs minimaux à ℓ requêtes. Cet ensemble reprèsente toutes les formes de *nœuds* possibles que l'on peut avoir avec ℓ chemins arc-disjoints. Nous allons d'abord montrer que décider si un couple $(\hat{\mathrm{G}},\hat{\mathrm{R}})$ est un motif de (G,R) est polynomial en la taille de G (mais non en la taille de $\hat{\mathrm{G}}$), puis nous allons montrer que si l'ensemble des motifs minimaux F_ℓ est fini, alors le problème des chemins disjoints dans un graphe orienté symétrique est polynomial.

Lemme 10 *Soit (G,R) et $(\hat{G},\hat{R})$ deux graphes non-orientés. Si $\hat{R}$ est un ensemble de requêtes réalisables dans $\hat{G}^*$, il existe un algorithme en $O(c(|\hat{E}|)\times|V|^{|\hat{V}|}\times|V|^3)$ (polynomial en G) déterminant si $(\hat{G},\hat{R}) < (G,R)$, où $c(|\hat{E}|)$ est une constante dépendant de $|\hat{E}|$.*

Preuve
Pour toute injection de $\hat{V}$ dans V (il y en a $O(|V|^{|\hat{V}|})$) on teste si $i(\hat{R}) = i(R)$ et s'il y a $|\hat{E}|$ chemins arête-disjoints. Ce test coûte $c(|\hat{E}|) \times O(|V|^3)$ d'après le théorème 6. □

Lemme 11 *Soit G^* un graphe symétrique, et R un ensemble de ℓ requêtes sur G, le squelette de G^*. Ces requêtes sont réalisables si et seulement si il existe un motif minimal $(\hat{G},\hat{R})$ tel que $(\hat{G},\hat{R}) < (G,R)$.*

Preuve
Si les requêtes sont réalisables sur G^*, G est un motif de lui-même, et contient donc un motif minimal. Si (G,R) contient un motif minimal $(\hat{\mathrm{G}},\hat{\mathrm{R}})$, les chemins arc-disjoints réalisant $\hat{\mathrm{R}}$ dans $\hat{\mathrm{G}}^*$ induisent des chemins arc-disjoints réalisant R dans G^*. □

Théorème 9 *Si F_ℓ est fini, il existe un algorithme polynomial pour trouver ℓ chemins arc-disjoints dans un graphe symétrique.*

Preuve
Soit G^* un graphe symétrique, et R un ensemble de ℓ requêtes sur G, le squelette de G^*. D'après le lemme 11, les requêtes sont réalisables si et seulement si (G,R) contient un motif minimal : on énumère donc tous les motifs minimaux de F_ℓ (il y en a $|F_\ell$—). Pour chacun d'entre eux, on teste si $(\hat{G},\hat{R}) < (G,R)$ (d'après le lemme 10, en temps $O(c \times |V|^{(n+3)})$, où $c = max(|c(\hat{E})|)_{\hat{G}\in F_\ell}$ et $n = max(|\hat{V}|)_{\hat{G}\in F_\ell}$). □

[5] imaginer que chaque chemin représente un fil

2.4 Motifs minimaux

Nous avons vu dans la section précédente la relation entre le problème des chemins arc-disjoints dans les graphes orientés symétriques et les motifs minimaux à ℓ requêtes. Dans cette section, nous allons montrer que l'ensemble des motifs minimaux à ℓ requêtes est fini, et donc que le problème des chemins arc-disjoints est polynomial dans les graphes orientés symétriques.

2.4.1 Propriétés élémentaires des motifs minimaux

Soit (G,R) un motif minimal à ℓ requêtes, et $\mu_1...\mu_\ell$ des chemins réalisant ces requêtes dans G^*.

- toutes les arêtes de G sont *occupées* par un chemin.
  (sinon on pourrait en supprimer)
- deux arêtes adjacentes ne sont pas *occupées* de la même façon (i.e. occupées par un μ_i seul, ou occupées par μ_i et μ_j).
  (sinon on pourrait les fusionner)

Multigraphes

Théorème 10 *Si (G,R) est un motif minimal, et si G est un multigraphe, on peut obtenir un motif minimal (G',R') tel que G' soit un graphe simple en dupliquant les sommets de G, et sans ajouter d'arête.*

Preuve :
Soit (G,R) un motif minimal et (μ_i) des chemins de G^* réalisant R, et soit x et y deux sommets de G reliés par k arêtes.

- on suppose que tous les arcs de G^* sont occupés. On suppose que μ_1 contient les arcs xy et yz_1. On note μ_2 le chemin contenant les arcs z_1y et $yz_2...\mu_i$ le chemin contenant les arcs $z_{i-1}y$ et yz_i. Tous les arcs étant occupés, il existe I tel que $z_I = x$.
- si des arcs sont inoccupés, ou si un des chemins finit sur y, le procédé précédent nous amène soit sur y, soit sur un z_J. Si on est sur z_J, un arc libre nous amène sur y. Si on est sur y, on emprunte soit un chemin débutant sur y non déjà emprunté, soit un arc libre non déjà emprunté vers z_{J+1}.

On ajoute à G le sommet y' et on transforme les arêtes (une de chaque s'il y en a plusieurs) xy, $yz_1...yz_{I-1}$ en xy', $y'z_1...y'z_{I-1}$: on obtient ainsi G'. Une réduction de G' induit immédiatement une réduction de G : G' est minimal. En itérant moins de $|E|$ fois le procédé, on obtient un motif simple minimal.

Conséquences
Les motifs minimaux correspondant à des multigraphes s'obtiennent donc en fusionnant des sommets des motifs minimaux correspondant à des graphes simples : on ne s'intéressera par la suite qu'aux graphes simples, sans perte de généralité.

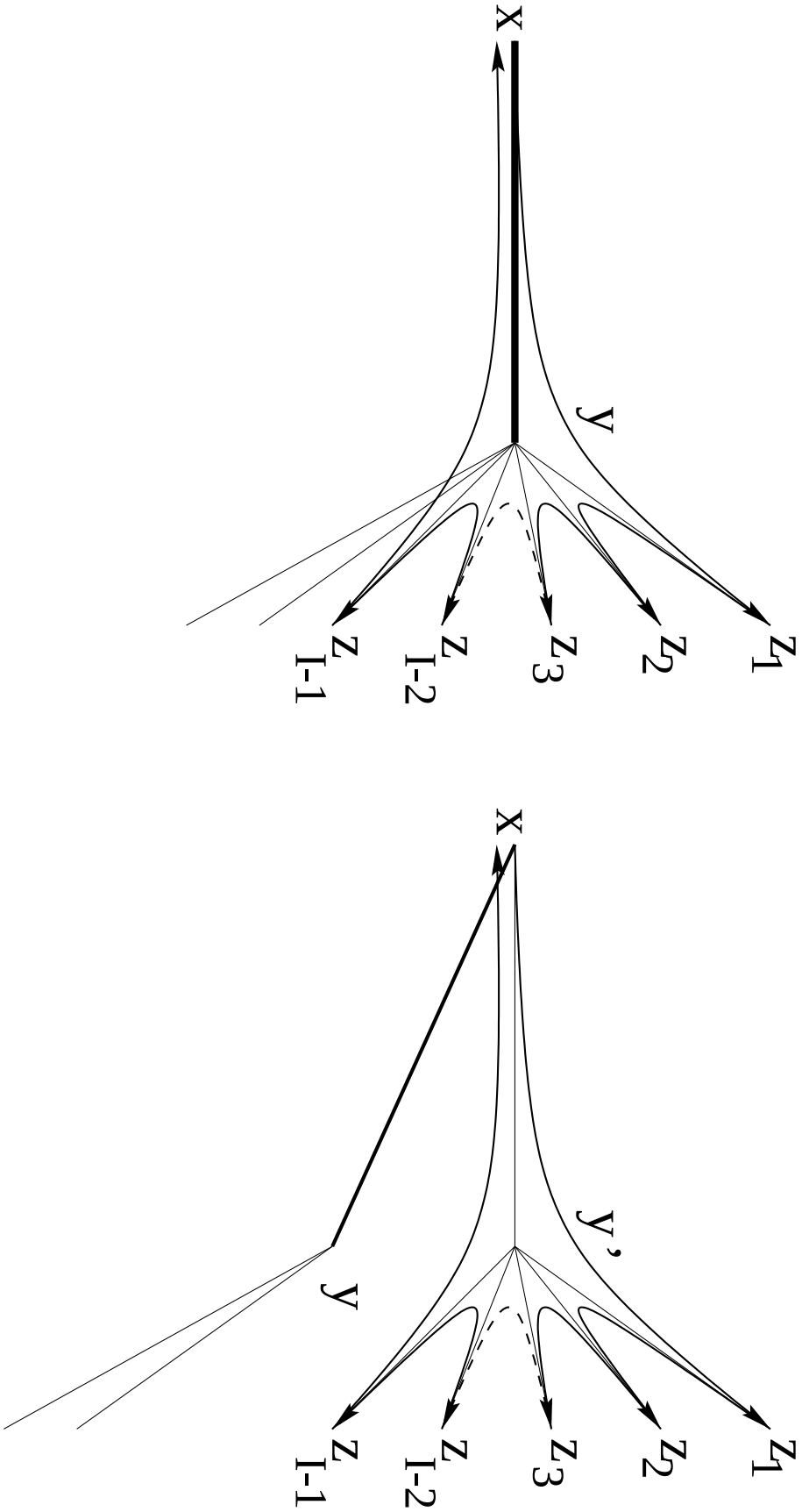


FIG. 2.4 – Duplication du sommet d'un multigraphe, avec conservation des solutions au problème des chemins disjoints.

2.4.2 Cas de deux requêtes

Le but ici est d'expliciter F_2, l'ensemble des motifs minimaux à deux requêtes.
Soit (G,R) un motif à deux requêtes, réalisées dans G^* par μ_1 et μ_2.

Définition 8 (se croiser) *Si μ_1 contient l'arc ab et μ_2 contient l'arc ba, on dit que μ_1 et μ_2 se croisent en (ab).*
- *si μ_1 est de la forme ...ab...cd... et μ_2 de la forme ...ba...dc... on dit que les chemins se croisent dans le même sens.*
- *si μ_1 est de la forme ...ab...cd... et μ_2 de la forme ...dc...ba... on dit que les chemins se croisent à contresens.*

Première réduction :
Si μ_1 et μ_2 se croisent deux fois *dans le même sens* alors on peut réduire G.

Cette réduction est toujours valable quel que soit le nombre de chemins, on l'appellera toujours *Première réduction*. Si deux chemins se croisent plusieurs fois dans un motif minimal, ce sera toujours à contresens.

Deuxième réduction :
Si μ_1 et μ_2 se croisent deux fois à contresens alors on peut réduire G en privilégiant l'un des deux chemins.
Cette réduction suppose que les arcs retour du chemin privilégié soient inoccupés, ce qui n'est pas toujours vrai quand il y a plus de deux chemins.

Dans un motif minimal à deux requêtes, les deux chemins se croisent au plus une fois. F_2 est donc constitué de deux graphes, illustrés par la Figure 2.8

2.4.3 Cas de trois requêtes

Nous allons montrer que l'ensemble F_3 des motifs minimaux à trois requêtes est fini, et nous allons le décrire en détail.
Soit (G,R) un motif à trois requêtes réalisées par μ_1, μ_2 et μ_3.

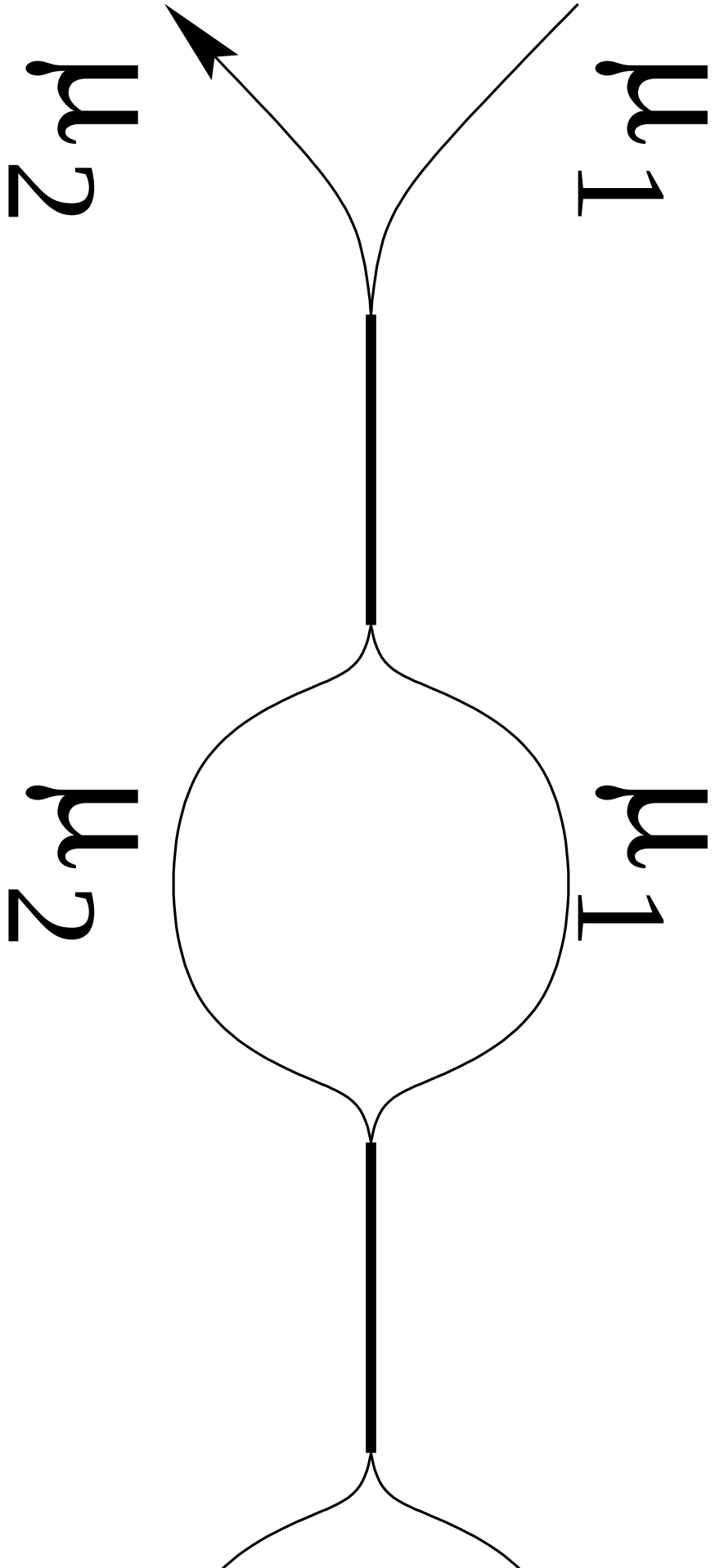
Première et deuxième réductions : La première réduction est identique au cas $\ell = 2$ (page 45). La deuxième réduction est quant à elle très proche du cas $\ell = 2$: si μ_1 et μ_2 se croisent deux fois à contresens, ils forment une *boucle*. Si μ_3 ne coupe pas μ_1 dans cette boucle, il y a une réduction.

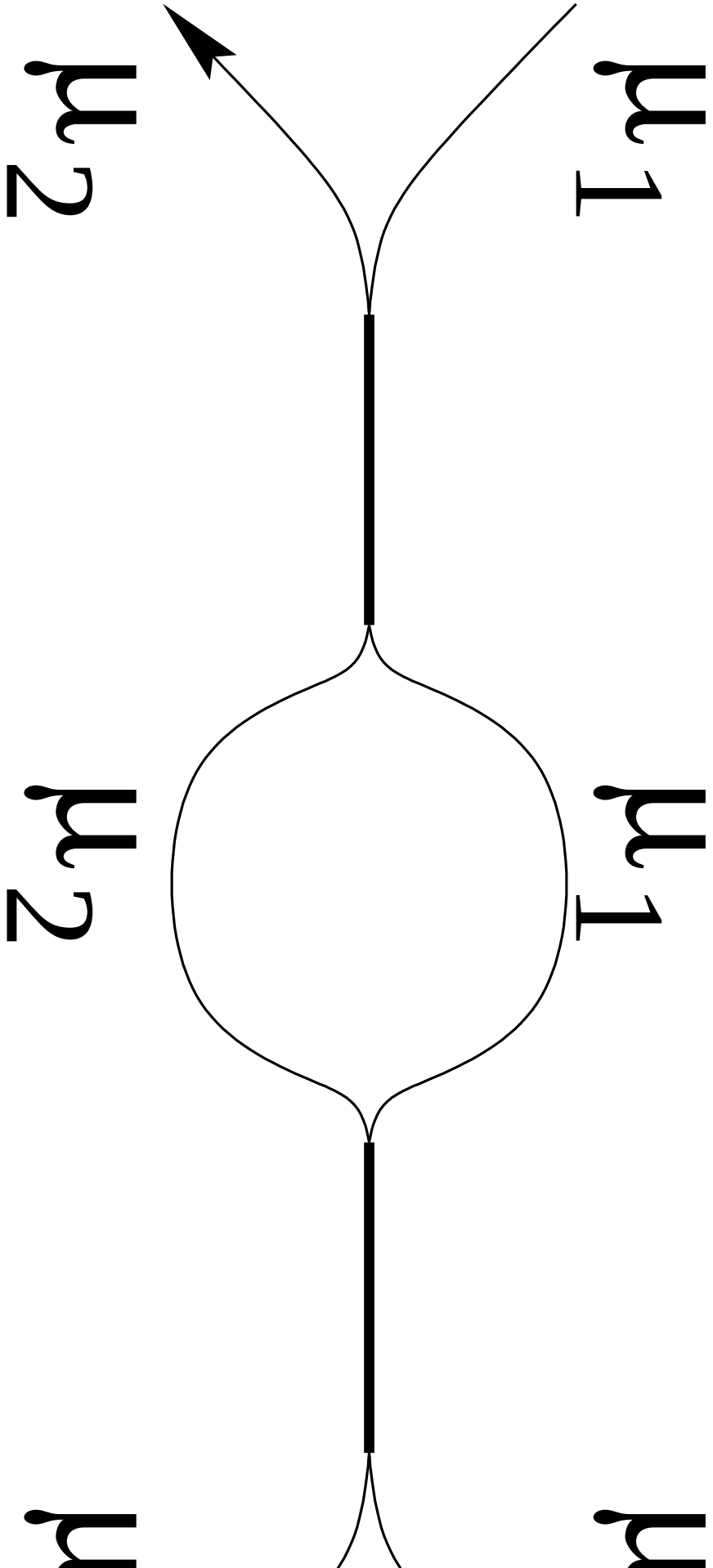
Aussi, dans un motif minimal, μ_3 coupe μ_1 et μ_2 sur chacune des boucles qu'ils forment.

Troisième réduction : On considère une boucle formée par μ_1 et μ_2. Si μ_3 pénètre deux fois dans cette boucle, il y a réduction.

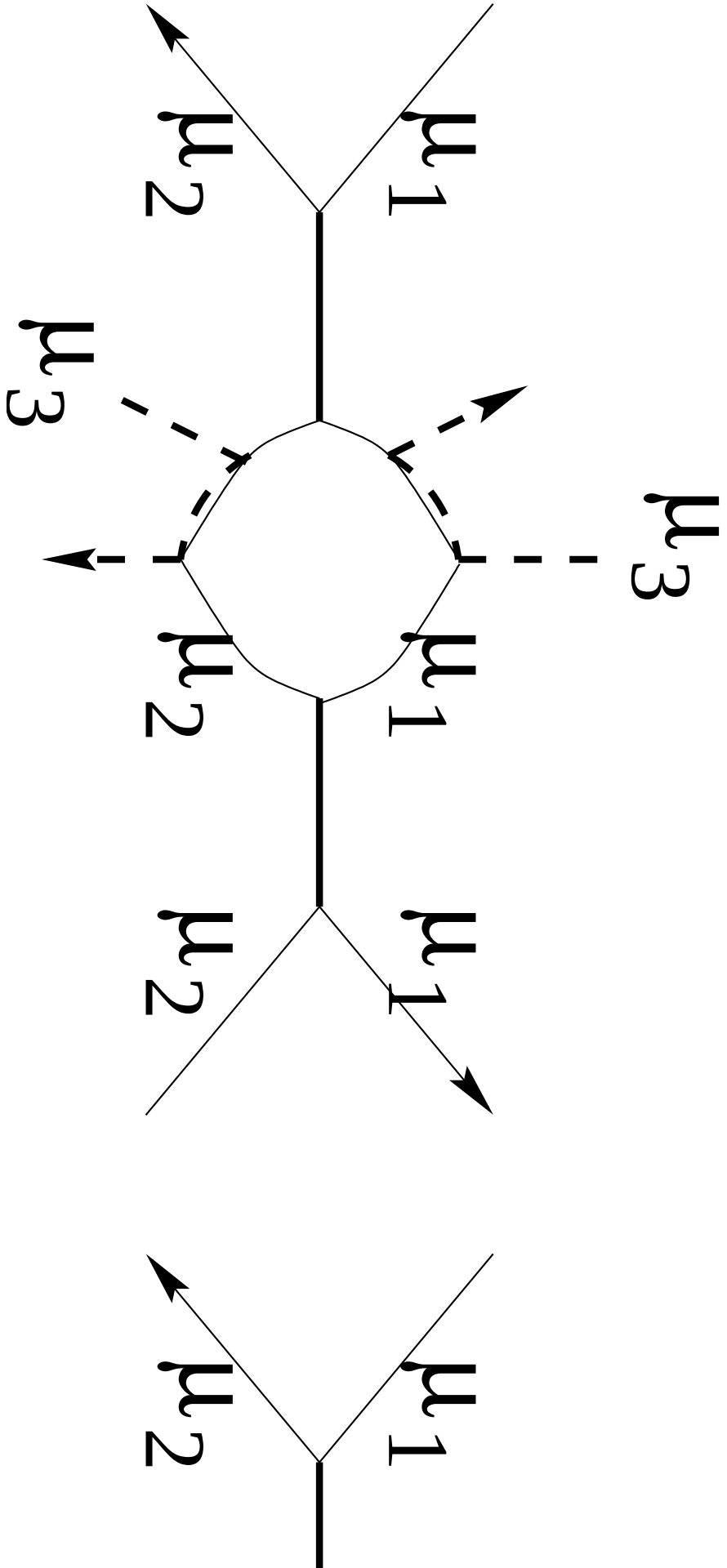
Cette réduction suppose qu'un seul chemin vienne interférer avec la boucle ; elle s'applique difficilement lorsqu'il y a plus de trois chemins.

u_2
u_1
u_1
u_2


μ_2
μ_1
μ_2
μ_1


μ_2
μ_1
μ_2
μ_1

λ2
λ1


μ_2
μ_1
μ_3
μ_3
μ_2
μ_1
μ_2
μ_1
μ_2
μ_1

Dans un motif minimal, μ_3 ne croise qu'une fois chaque boucle créée par μ_1 et μ_2.

Étude des motifs minimaux :

Soit (G,R) un motif minimal à trois requêtes réalisées par μ_1, μ_2, μ_3. On veut montrer que les chemins se croisent deux à deux au plus deux fois. Supposons que μ_1 et μ_2 se croisent trois fois (voir Figure 2.10) :

- la première réduction étant impossible, μ_1 et μ_2 forment deux boucles,
- la deuxième réduction étant impossible, μ_3 croise μ_1 et μ_2 dans chacune de ces boucles,
- la troisième réduction étant impossible, μ_3 coupe chaque boucle en une seule fois

Soit μ_3 croise deux fois μ_1 *dans le même sens* (1ère boucle avant la 2ème), soit μ_3 croise deux fois μ_2 *dans le même sens* (2ème boucle avant la 1ère).
Dans les deux cas, la première réduction s'applique, il y a donc contradiction.

Illustration des motifs de $\mathbf{F}_3$

- **cas a : réutilisation de $\mathbf{F}_2$**
  On considère le cas où μ_3 n'interfère ni avec μ_1, ni avec μ_2.
- **cas b : un seul croisement**
- cas b.1 : un chemin coupe les deux autres (voir Figure 2.13).
- cas b.2 :
  Si chaque chemin coupe les deux autres une fois : on choisit la numérotation de sorte que μ_1 commence par couper μ_2.
  Soit μ_2 commence par couper μ_3 et μ_3 commence par couper μ_1 (voir Figure 2.14).
- cas b.3 :
  Soit μ_2 commence par couper μ_1 et μ_3 commence par couper μ_1 (3 possibilités) illustrées par la Figure 2.15
  - si μ_2 commence par couper μ_1 et μ_3 commence par couper μ_2, on effectue la permutation (1→2,2→1,3→3).
  - si μ_2 commence par couper μ_3 et μ_3 commence par couper μ_2, on effectue la permutation (1→3,2→1,3→2).
- **cas c : boucle**
  Si μ_1 coupe deux fois μ_2, μ_3 ne peut pas couper μ_1 juste après ou juste avant avoir coupé la boucle (voir Figure 2.16). On a donc 6 possibilités illustrées par la Figure 2.17.

2.4.4 Motifs minimaux pour ℓ requêtes

On déjà résolu les cas $\ell = 2$ et $\ell = 3$. Nous allons maintenant généraliser ces résultats et montrer que l'ensemble F_ℓ de motifs minimaux à ℓ requêtes est fini pour tout ℓ. Pour cela, on recherche des propriétés sur les motifs minimaux de manière à borner leur taille en fonction de ℓ.

Première réduction :
Nous rappelons que si μ_1 et μ_2 se croisent deux fois *dans le même sens* alors on peut réduire G. Aussi, si deux chemins se croisent plusieurs fois dans un motif minimal, ce sera toujours à contresens.

Deuxième réduction :

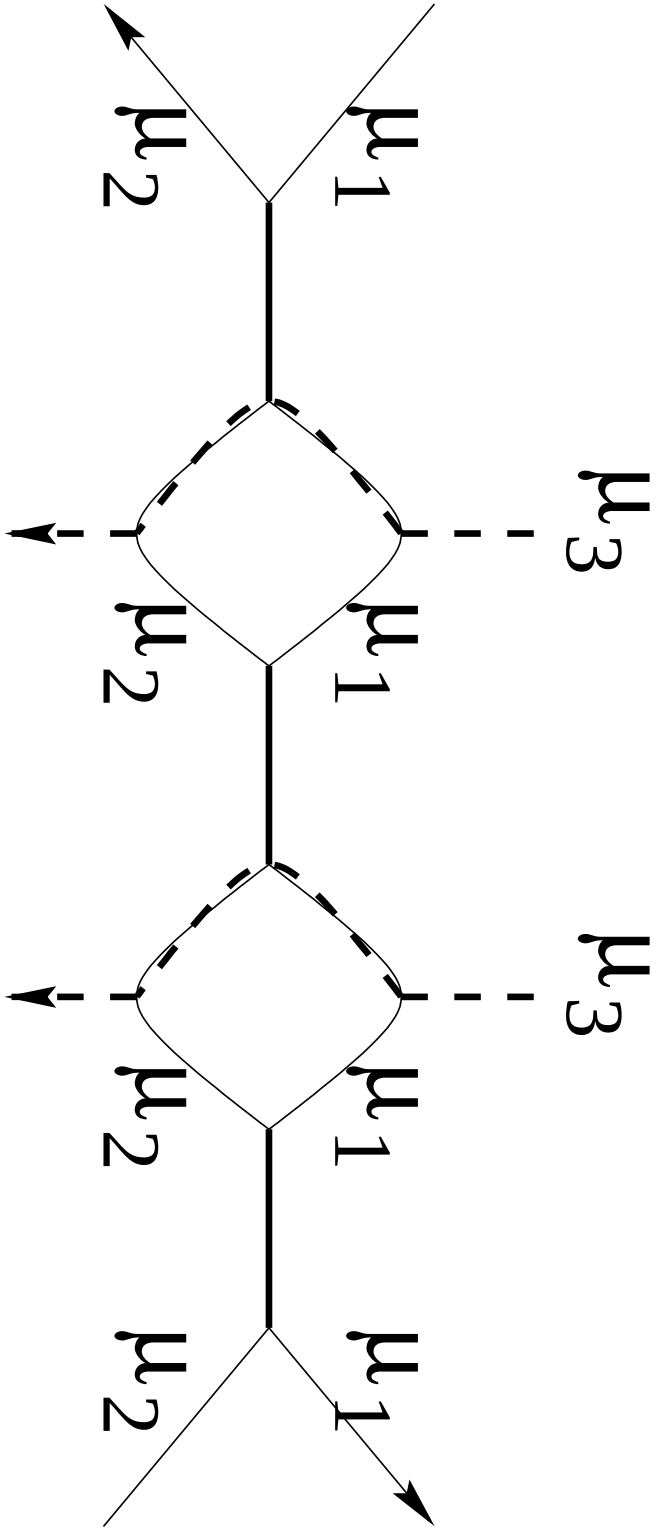


FIG. 2.10 – Les chemins μ_1 et μ_2 se croisent trois fois

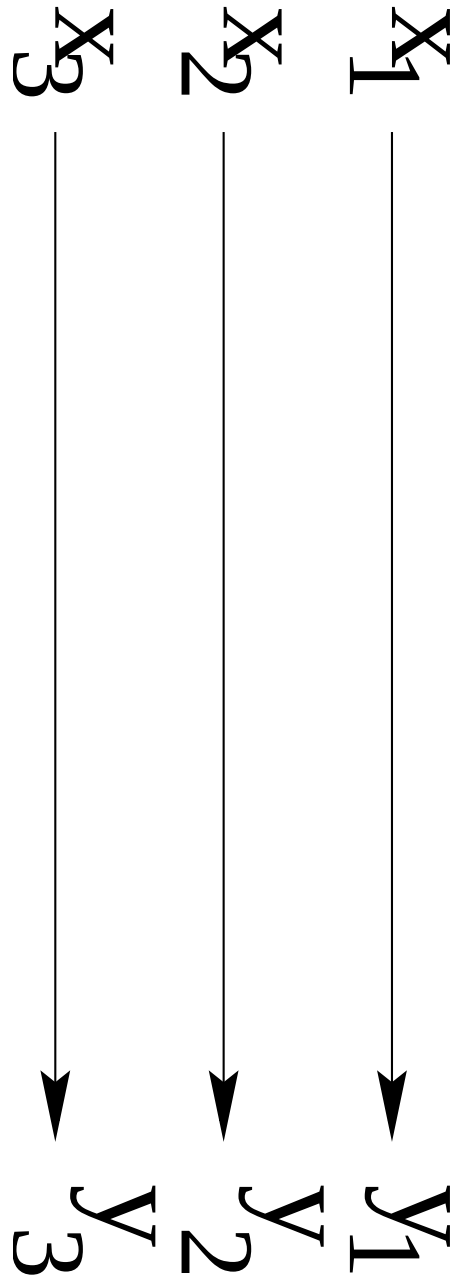


FIG. 2.11 – Motif où tous les chemins sont isolés

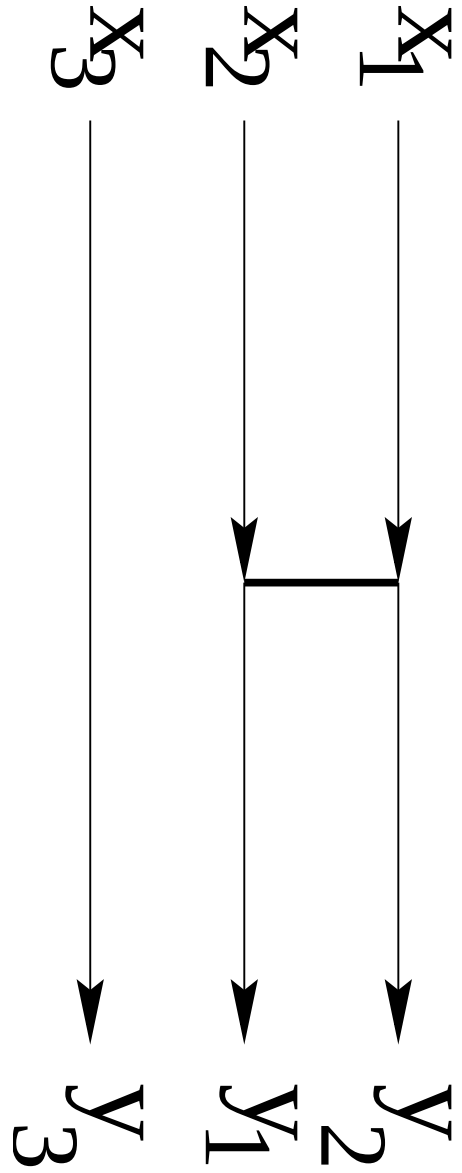
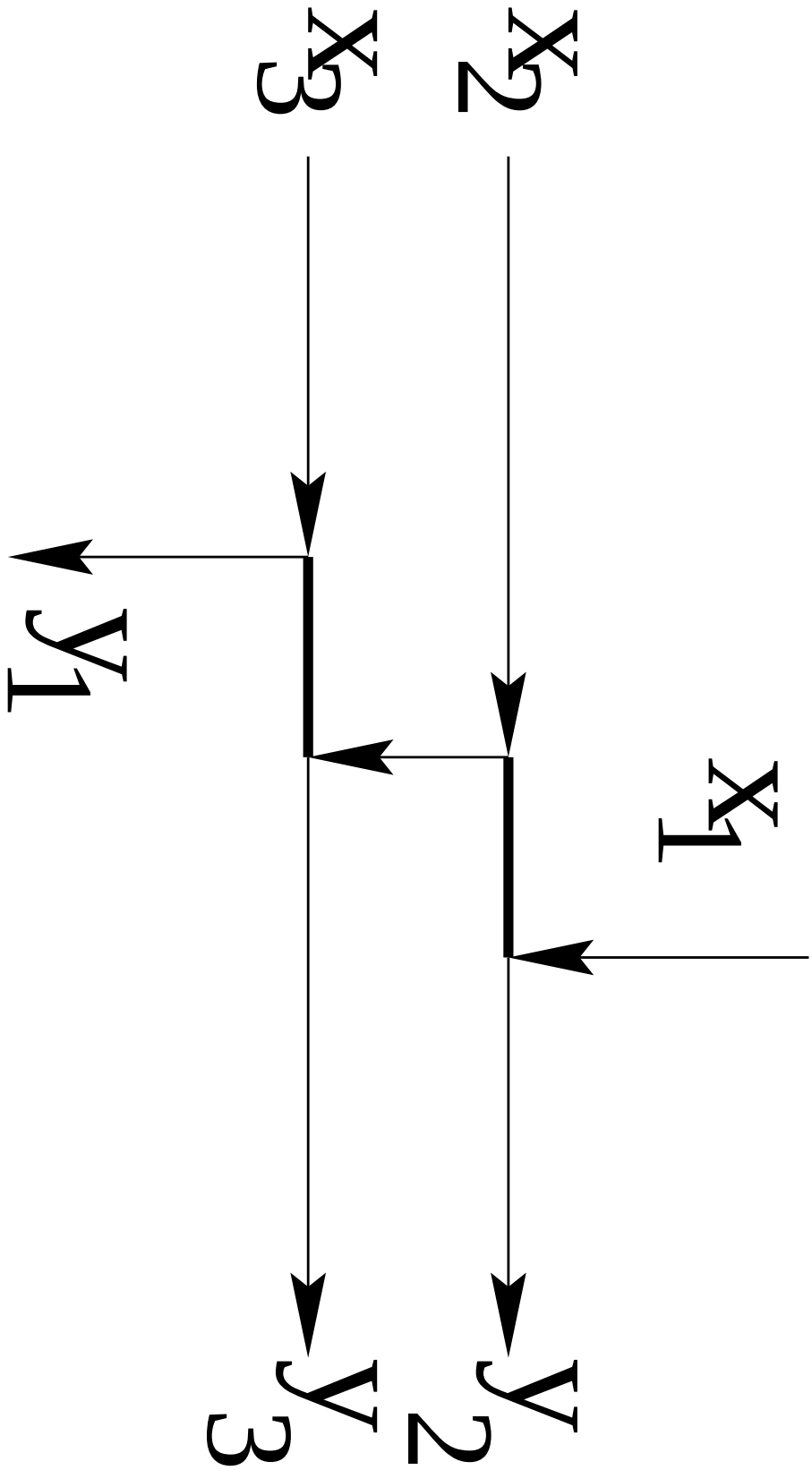


FIG. 2.12 – Motif contenant un chemin isolé


x3
x2
y1
x1
y3
y2

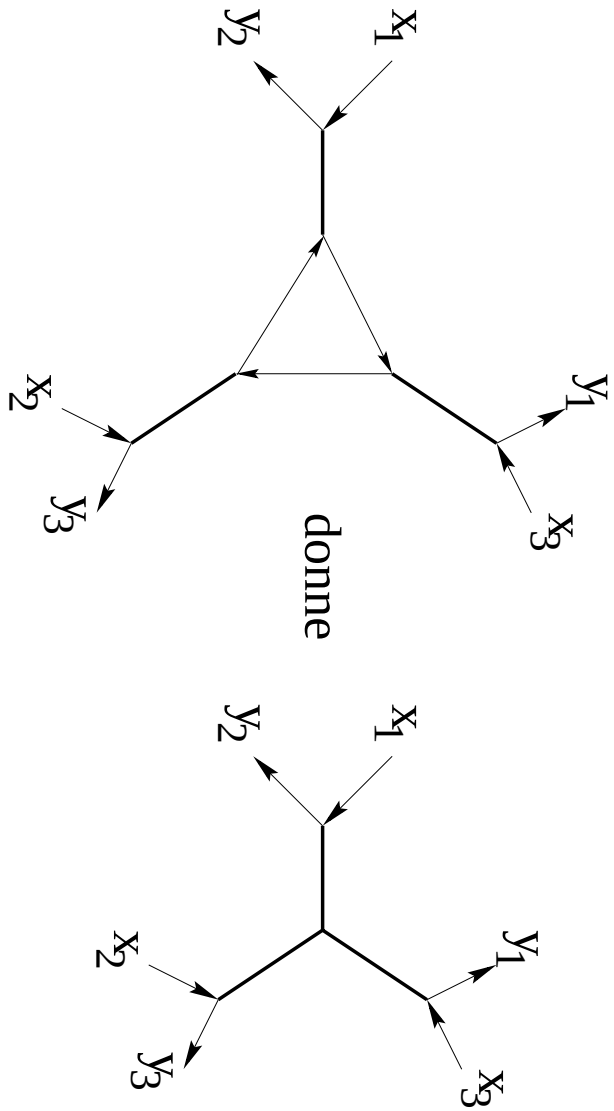


FIG. 2.14 – Cas où le chemin μ_2 commence par couper le chemin μ_3 et où le chemin μ_3 commence par couper le chemin μ_1. Motif correspondant.

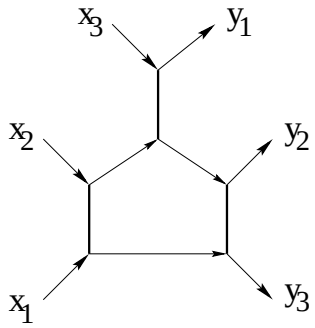


FIG. 2.15 – Motif où le chemin μ_2 commence par couper le chemin μ_3 et où le chemin μ_3 commence par couper le chemin μ_1.

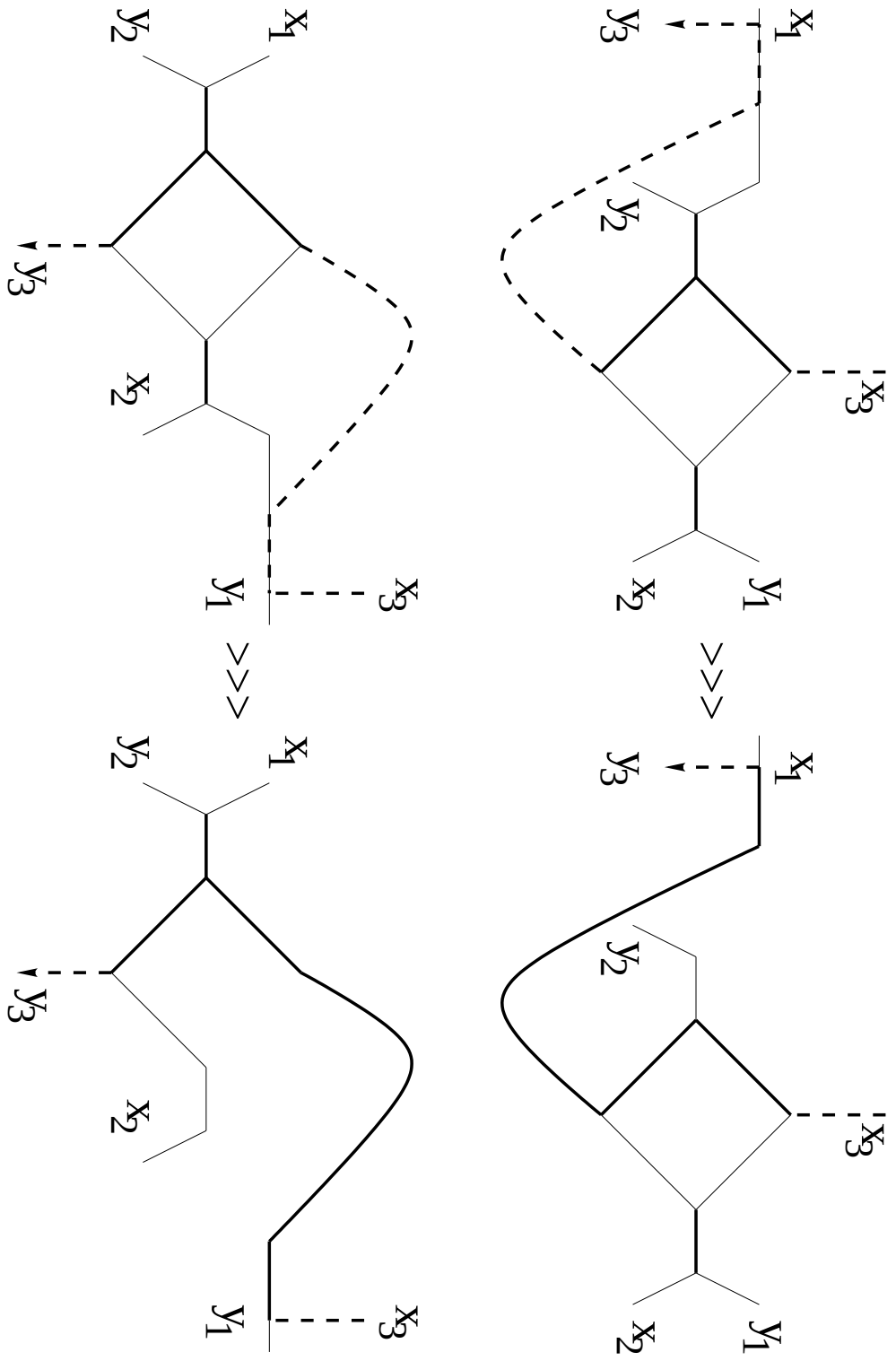


FIG. 2.16 – Cas où μ_1 coupe deux fois μ_2 : μ_3 ne peut pas couper μ_1 juste après ou juste avant avoir coupé la boucle.

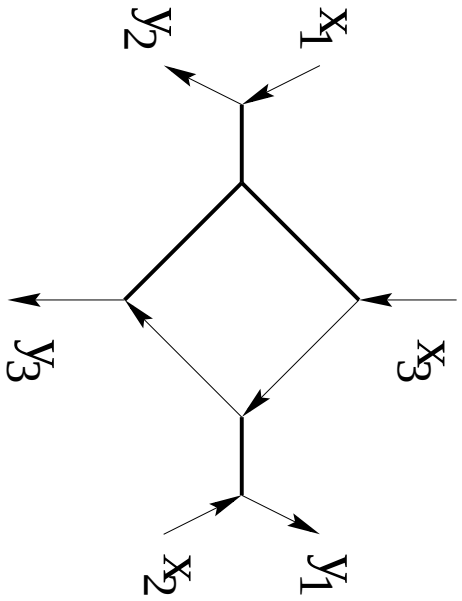


FIG. 2.17 – Motif où le chemin μ_1 coupe deux fois le chemin μ_2

On étudie le cas où μ_2 et μ_3 croisent deux fois μ_1 l'un à la suite de l'autre, à contresens.
Notation :
Si $abc \in mu_i$ est une paire d'arcs telle que $ba \in \mu_j$ et $cb \in \mu_k$, b est appelé *point double* de μ_i.
Nous en déduisions le lemme suivant :

Lemme 12 *Soit G un motif minimal réalisé par $\mu_1 ... \mu_\ell$. μ_i a donc au plus $(\ell-1)^2$ points doubles.*

Troisième réduction :

Définition 9 (Graphe des écarts) *Soit G=(V,E), R un motif réalisé par $\mu_1 ... \mu_\ell$, et $G^*=(V,E^*)$ le symétrisé de G.*
Soit E_e l'ensemble des arcs inoccupés de E^*, $G_e=(V,E_e)$ est appelé ***graphe des écarts*** *de G.*

Pour tout sommet x du graphe des écarts, on note $\mathrm{d}^+(x)$ son degré sortant, $\mathrm{d}^-(x)$ son degré entrant, $r^+(x) = |\{i | x_i = x\}|$ et $r^-(x) = |\{y_i = x\}|$. Par la propriété de conservation du flot, on a On a $\mathrm{d}^+(x) - \mathrm{d}^-(x) = \mathrm{r}^+(x) - \mathrm{r}^-(x)$..

Proposition 5 *Si (G,R) est motif minimal réalisé par $\mu_1 ... \mu_\ell$, son graphe des écarts est composé d'au plus ℓ chemins. En ajoutant une requête par chemin, on obtient un motif minimal réalisé par au plus $2 \times \ell$ chemins dont le graphe des écarts est vide.*

Preuve

– Si le graphe des écarts admet un cycle, il s'agit d'un cycle libre dans G.
On peut l'utiliser pour allonger un chemin et supprimer une arête de G (voir Figure 2.21).

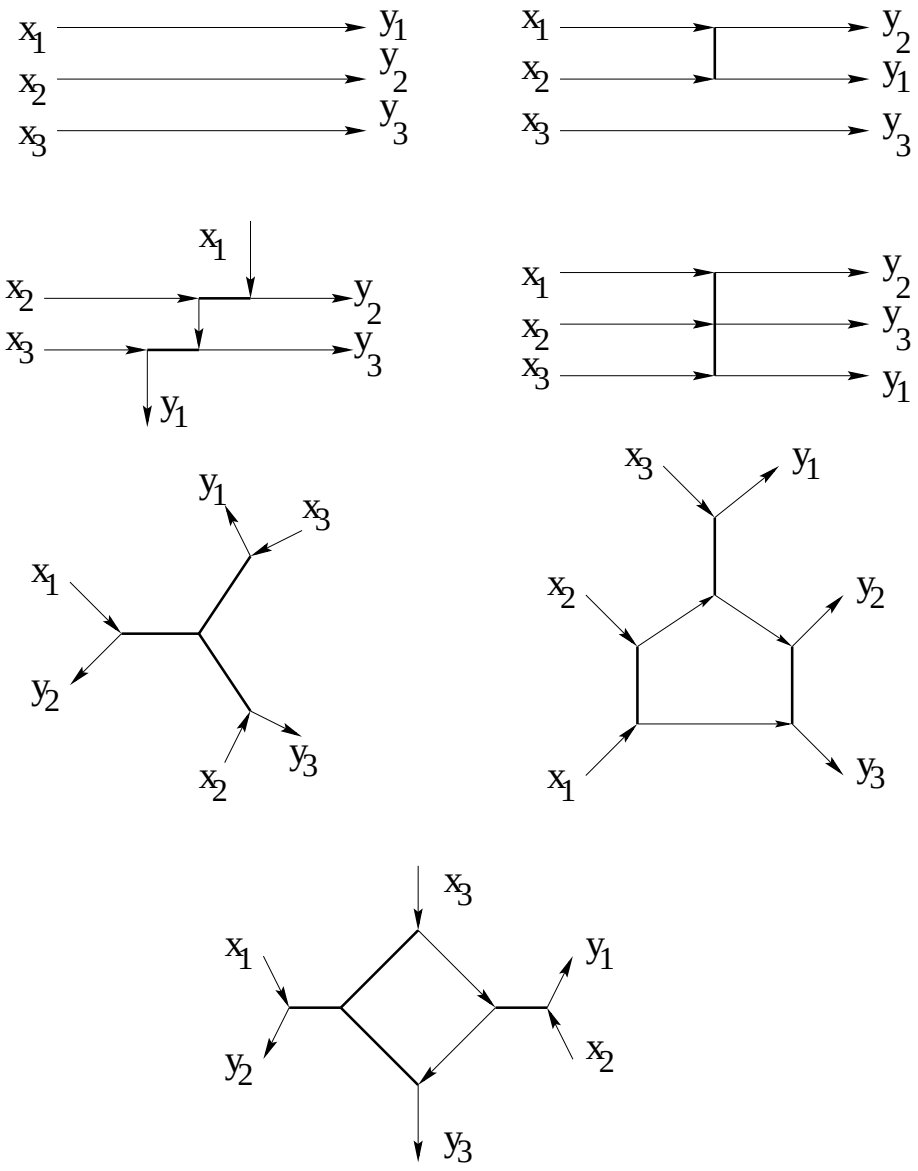


FIG. 2.18 – Ensemble F_3 des motifs minimaux à trois requêtes

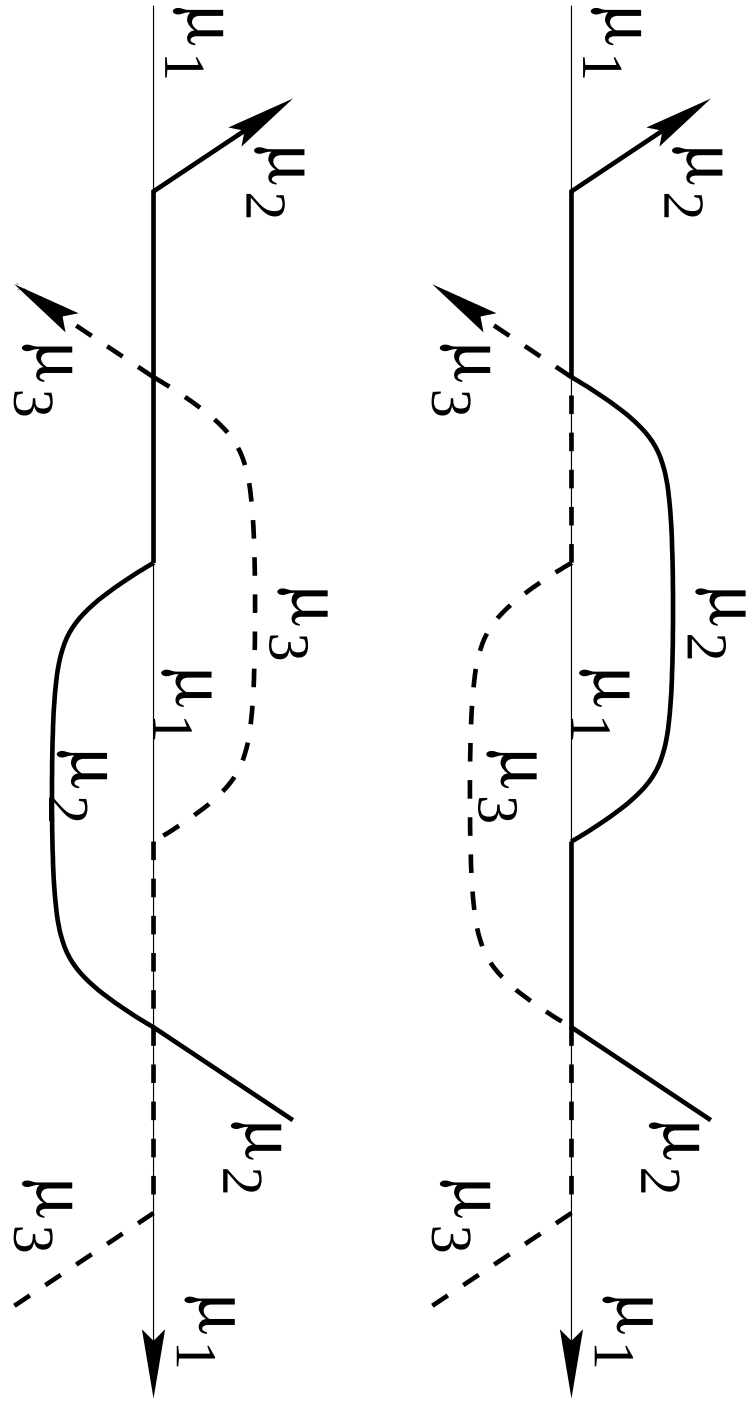


FIG. 2.19 – Deuxième réduction sur les motifs d'un graphe, pour un nombre indéterminé de requêtes.

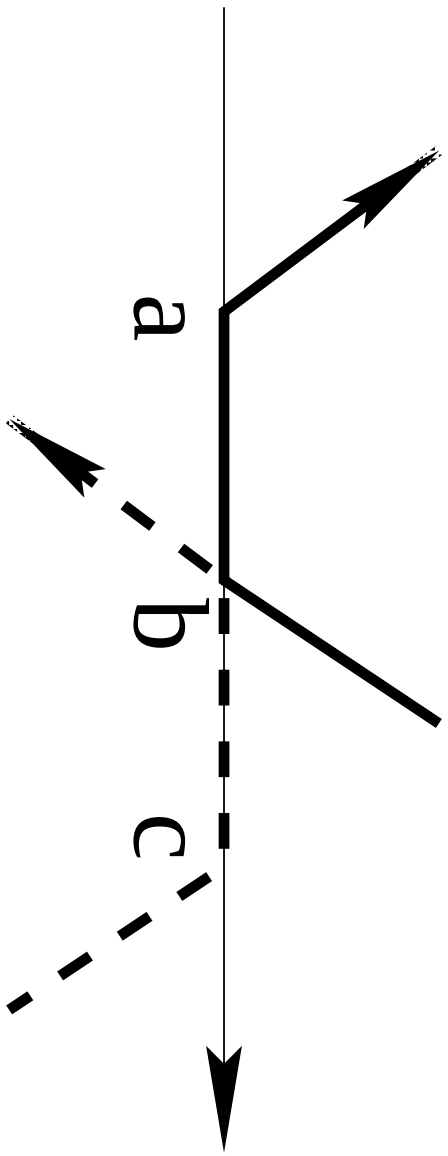


FIG. 2.20 – Sommet appartenant à trois chemins distinct, nommé point double..

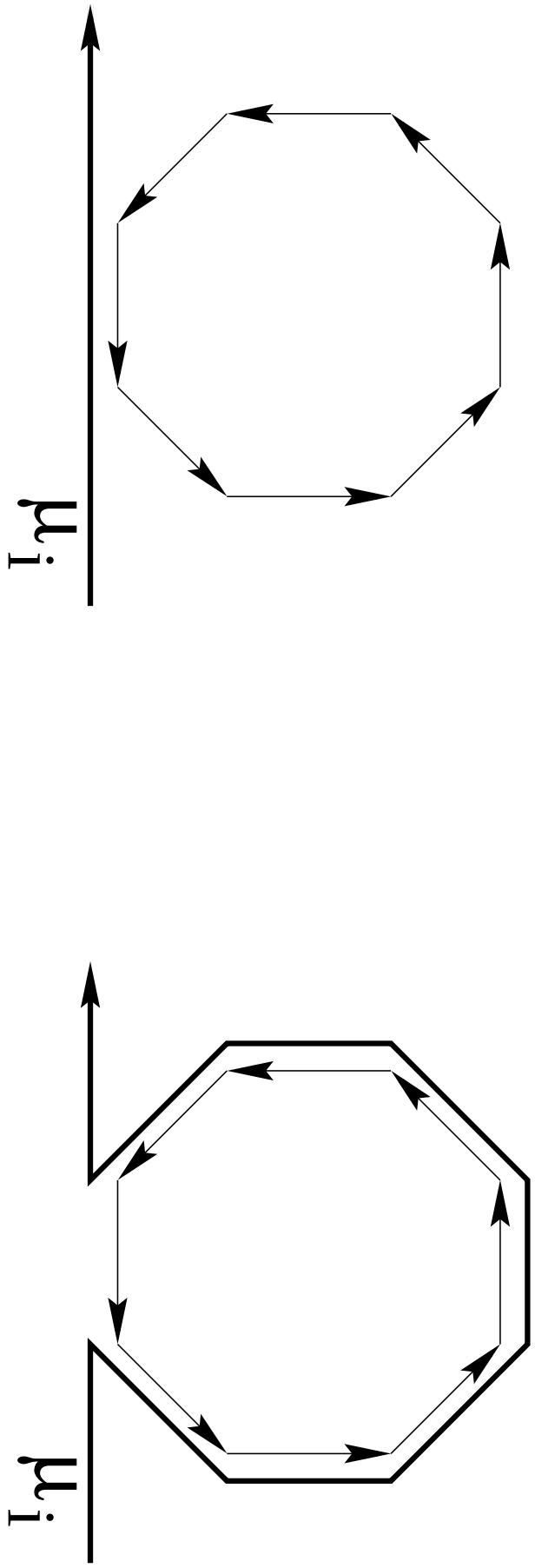


FIG. 2.21 – Troisième réduction sur les motifs d'un graphe, pour un nombre indéterminé de requêtes.

– Si le graphe des écarts n'admet pas de cycle, on peut le décomposer en chemins d'extrémités $y_i\ x_j$, où $(x_i, y_i), (x_j, y_j) \in$ R, soit en au plus ℓ chemins.

□

Théorème 11 *Il existe un algorithme polynomial permettant de trouver ℓ chemins dans un graphe symétrique.*

Preuve
Soit (G,R) un motif minimal réalisé par $\mu_1 ... \mu_{2\times\ell}$, dont le graphe des écarts est vide. Chaque μ_i n'est composé que de points doubles (sauf les extrémités), et d'après la *deuxième réduction* (12) il au plus $((2 \times \ell - 1)^2$ points doubles. Aussi, G contient au plus $\ell \times ((2 \times \ell - 1)^2 + 1)$ arêtes. F_ℓ est donc fini. D'après le théorème 9, le problème des chemins disjoints dans un graphe symétrique est polynomial pour un nombre borné de requêtes. □

2.5 Conclusion et perspectives

Au cours de ce chapitre, nous avons montré qu'il existe un algorithme polynomial permettant de répondre au problème des chemins disjoints dans un graphe orienté symétrique lorsque le nombre de requêtes est borné. Ce résultat est à mettre en relation avec le théorème de Fortune, Hopcroft et Wyllie [36] qui stipule que trouver seulement deux chemins disjoints dans un graphe orienté est un problème NP-difficile.

D'autre part, notre algorithme s'appuie essentiellement sur celui du *δ-folio containment* de Robertson et Seymour [86]. Robertson et Seymour avaient ainsi donné une solution polynomiale au problème des chemins disjoints dans un graphe non orienté lorsque le nombre de requêtes est borné. D'un point de vue technique, nous avons montré que les communications dans un réseau optique pouvaient toujours être routées de manière à former un certain *motif*. Le nombre de motifs possibles dépend uniquement du nombre de requêtes et non de la forme du réseau ou de la localisation des requêtes. Le problème de trouver un ou plusieurs motifs donnés correspond au problème du *δ-folio containment* que nous avons cité. Malheureusement, l'algorithme de Robertson & Seymour est réputé impraticable même pour un petit nombre de requêtes car il manipule des constantes très grandes. Aussi, il serait intéressant de trouver des algorithmes plus efficaces pour un petit nombre de requêtes.

Récapitulation : le tableau suivant résume les différentes classes de complexité des problèmes de chemins arc-disjoints en fonction du nombre de requêtes et du type de graphe. Nous avons montré que le cas orienté symétrique se comporte de la même manière que le cas non-orienté, avec des algorithmes polynomiaux pour un nombre de requêtes borné.

	Graphe		
Nombre de requêtes	non-orienté	orienté	symétrique
1	P	P	P
2	P	NP	P
3	P	NP	**P**
borné	P	NP	**P**
non borné	NP	NP	NP

Dans le cadre des réseaux optiques, il est également intéressant d'étudier une généralisation du problème des chemins disjoints : le problème du multiflot entier. Cette étude fait l'objet du chapitre suivant.

Chapitre 3

Routage dans les réseaux optiques : multiflots

Sommaire

3.1 Introduction

Dans ce chapitre, nous étudions une généralisation au problème des chemins disjoints : le problème du multiflot entier. De la même manière qu'au chapitre précédent, nous étudions ce problème dans le cadre du routage dans les réseaux optiques. Nous modélisons un réseau optique par un graphe orienté symétrique muni d'une fonction de capacité. La fonction de capacité sur les arcs du graphe correspond à la quantité maximale d'information qui peut transiter par un lien du réseau optique. La fonction de capacité que nous considérons est également symétrique, c'est à dire que la capacité d'un arc est égale à la capacité de l'arc opposé.

Le problème du multiflot entier à deux commodités est NP complet dans le cas général ([36] et [53]) et de nombreuse variantes ont été étudiées, suivant les propriétés du graphe G, de l'ensemble des requêtes R, ou du graphe composé $G + R$. Ces variantes se divisent en problèmes NP-complets

ou polynomiaux. Une de ces variantes est le problème des chemins disjoints, traité au chapitre précédent. Les principales autres variantes concernent les cas où G ou $G+R$ sont eulériens, planaires, etc...

Lorsque G est planaire et eulérien, et lorsque les requêtes R sont sur une seule face de G, le problème est polynomial [78], et peut même être résolu en temps linéaire [95].

Dans le cas symétrique, le graphe G est également eulérien, ce qui fait du cas symétrique un cas particulier du cas eulérien. On note cependant que les principaux résultats du cas Eulérien concernent en fait la structure $G+R$. Sur cette variante, Nash-Williams a prouvé en 1965 (le lecteur trouvera une preuve de ce théorème en [93]) que pour deux requêtes le problème est polynomial, alors que par ailleurs il a été prouvé que le problème est NP-complet pour 3 requêtes ou plus [94]. Dans le cas le plus général, le problème du multiflot entier est NP-difficile pour seulement 2 requêtes [36].

Contenu du chapitre

Dans la section 3.2, nous introduisons les notations sur les flots et les multiflots, puis nous présentons deux conditions classiques, nécessaires au problème du multiflot, et enfin nous présentons une preuve de la NP-complétude du problème du multiflot entier à deux commodités.

Dans la section 3.3, nous nous attachons au cas des graphes symétriques. Nous montrons que le problème du multiflot à 3 commodités est NP-complet dans le cas des graphes symétriques, puis nous montrons que le problème du multiflot symétrique est lui aussi NP-complet.

Dans la section 3.4, nous étudions plus spécifiquement le cas où les requêtes sont symétriques. Dans le cadre des réseaux optiques, ceci correspond aux applications de type *video conférence*, où la quantité d'information échangée entre les utilisateurs est équilibrée dans les deux sens. Les applications de type *diffusion d'informations* ne rentrent pas dans ce cadre, car l'information y est surtout émise depuis une source, et la communication n'est donc pas symétrique. On peut observer que la donnée d'un graphe orienté symétrique et de requêtes symétriques sur ce graphe est identique à la donnée d'un graphe non orienté avec des requêtes non orientées. De plus, une solution au problème non orienté est parfaitement transposable au problème orienté. Bien que cela soit vrai, la variante symétrique du problème de multiflot entier admet plus de solutions est également plus facile à résoudre : Even, Itai et Shamir [23] ont prouvé que le problème du multiflot entier dans un graphe non orienté est NP-complet pour seulement 2 requêtes, de valeur $(1, v)$. Nous allons prouver quant à nous que le problème symétrique est polynomial lorsqu'il n'y a que deux commodités. Pour arriver à ce résultat, nous utiliserons des coupes pour construire ou prouver l'existence de certains flots. Nous exploiterons aussi la symétrie du problème pour intervertir des flots opposés sans enfreindre les contraintes des capacité.

Voici un tableau des différentes complexités pour les problèmes de flot et de chemins disjoints, pour différentes variantes. Nous notons N le nombre de sommets d'un graphe, et M le nombre d'arcs ou d'arêtes.

	Graphe		
Problème : déterminer l'existence de...	orienté	non orienté	orienté symétrique
un chemin	$O(M + NlogN)$ [16]		
chemins disjoints pour k requêtes	NP-difficile [36]	O(N^3) [86]	O(N^3)
chemins disjoints	NP-difficile [53]	NP-difficile [53]	NP-difficile [13]
flot	C_{flot} [35]		C_{flot} [35]
multiflot à 2 commodités (biflot)	NP-difficile [23]		non résolu
multiflot à $k \geq 3$ commodités	NP-difficile		**NP-difficile**
flot symétrique	NP-difficile	C_{flot}	C_{flot}
biflot symétrique	NP-difficile	NP-difficile	$\mathbf{6C_{flot} + O(M)}$
multiflot symétrique à $k \geq 3$ commodités	NP-difficile	NP-difficile	non résolu
multiflot symétrique	NP-difficile	NP-difficile	**NP-difficile**

3.2 Notations et résultats antérieurs sur le problème du multiflot entier

Dans cette section nous présentons d'abord les notations que nous utilisons à travers ce chapitre, puis nous abordons quelques résultats classiques sur le critère de coupe. Enfin, nous présentons la preuve de Even, Itai et Shamir [23] sur le problème du multiflot entier à deux commodités.

3.2.1 Fonctions sur les arcs, flots, multiflots, coupes

Étant donné un graphe orienté $G = (V, A)$, nous allons utiliser des fonctions de l'ensemble des arcs A vers l'ensemble des entiers naturels $\mathbb{N}$. Dans ce chapitre, nous considérons que les flots (usuellement notés f, g ou h) font partie de l'ensemble des fonctions de A vers $\mathbb{N}$. La capacité maximale des arcs d'un graphe (usuellement notée κ) fait également partie des fonctions de A vers $\mathbb{N}$. Voici les opérations que nous effectuons sur les fonctions :

- comparaison : soit f_1 et κ deux fonctions de A vers $\mathbb{N}$, $f_1 \leq \kappa$ signifie que pour tout $a \in A$, $f_1(a) \leq \kappa(a)$.
- somme : soit f_1 et f_2 deux fonctions de A vers $\mathbb{N}$, $f_1 + f_2 : A \to \mathbb{N}$ est la fonction définie par $\forall a \in A$, $(f_1 + f_2)(a) = f_1(a) + f_2(a)$.
- différence : soit f_1 et κ deux fonctions de A vers $\mathbb{N}$ telles que $f_1 \leq \kappa$, $(\kappa - f_1) : A \to \mathbb{N}$ est la fonction définie par $\forall a \in A$, $(\kappa - f_1)(a) = \kappa(a) - f_1(a)$.

Étant données trois fonctions f_1, f_2, κ de A vers $\mathbb{N}$, remarquons que si $f_1 \leq \kappa$, alors $(f_1 + f_2) \leq \kappa$ est équivalent à $f_2 \leq (\kappa - f_1)$.

Nous définissons ici un flot entier comme fonction de A vers $\mathbb{N}$ qui satisfait la contrainte de conservation du flot.

Définition 10 (flot) *Soit $G = (V, A)$ un graphe orienté. Soit $s, t \in V$ et $v \in \mathbb{N}$. Un flot f de source s, de destination t et de valeur v est une fonction $f : A \to \mathbb{N}$ telle que*

- $\forall y \notin \{s, t\}, \Sigma_{x \in \Gamma^-(y)} f(x, y) = \Sigma_{z \in \Gamma^+(y)} f(y, z)$
- $\Sigma_{x \in \Gamma^-(s)} f(x, s) + v = \Sigma_{z \in \Gamma^+(s)} f(s, z))$

– $\Sigma_{x\in\Gamma^-(t)} f(x,t) = \Sigma_{z\in\Gamma^+(t)} f(t,z)) + v$

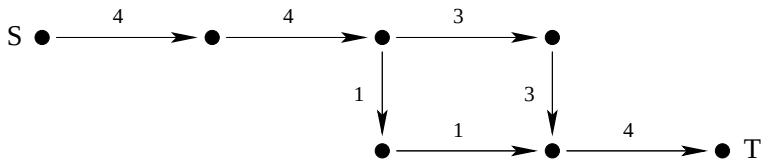


FIG. 3.1 – Un flot f_1 de s vers t de valeur 4.

Cette définition des flots permet l'existence de circuits. Aussi, nous préciserons "flot sans circuit" lorsque le besoin s'en fera sentir. En général, les problèmes de flot imposent une capacité maximale κ sur les arêtes, ce qui entraîne pour un flot f la contrainte supplémentaire $f \leq \kappa$. Cette dernière contrainte ne fait cependant pas partie de la définition du flot.

Nous introduisons maintenant des définitions spécifiques aux fonctions et graphes symétriques ; fonctions de retour et opération de simplification sur les fonctions,

Définition 11 (f^r, fonction retour de f) *Soit $G = (V, A)$ un graphe symétrique. Soit $f : A \to \mathbb{N}$. La fonction retour de f, $f^r : A \to \mathbb{N}$ est la fonction définie par $\forall (x, y) \in A$, $f^r(x, y) = f(y, x)$*

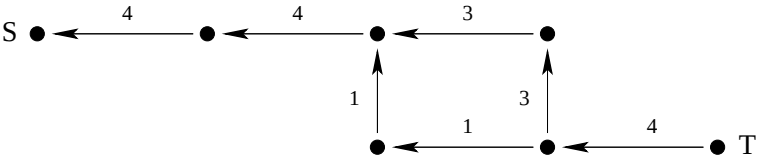


FIG. 3.2 – Un flot retour f_1^r de t vers s de value 4.

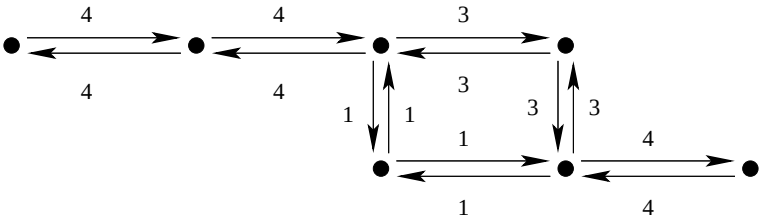


FIG. 3.3 – Une fonction symétrique $(f_1 + f_1^r)$ sur A.

Définition 12 (simplification) *étant donnée une fonction f de A vers $\mathbb{N}$, $|f| : A \to \mathbb{N}$ est la fonction définie par $\forall (x, y) \in A$, $|f|(x, y) = f(x, y) - \min\{f(x, y), f(y, x)\}$*

Définition 13 (Multiflot à k commodités) *Soit $G = (V, A)$ un graphe orienté. Pour tout $i \in \{1, ..k\}$, Soit $s_i, t_i \in V$, $v_i \in \mathbb{N}$ et soit f_i un flot de s_i vers t_i de valeur v_i. $(f_1, ..f_k)$ est un multiflot à k commodités de $(s_1, ..s_k)$ vers $(t_1, ..t_k)$ de valeur $(v_1, ..v_k)$.*

Nous considérons que les multiflots sont des collections de flots. Étant donnée une capacité maximale κ, la contrainte de capacité s'écrira $(\Sigma_i f_i) \leq \kappa$. Les multiflots à 2 commodités sont aussi appelés **biflots**.

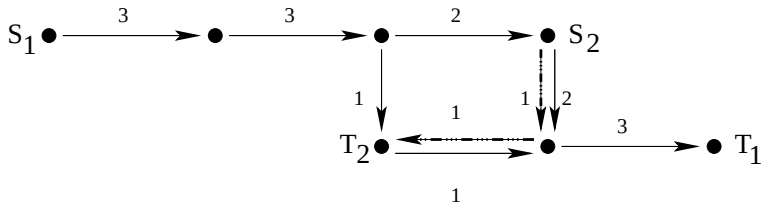


FIG. 3.4 – Multiflot (f_1, f_2) de (s_1, s_2) vers (t_1, t_2) de valeur $(3, 1)$.

3.2.2 Coupes, distances, et multiflots.

Les notions de coupes et de distance fournissent des conditions nécessaires pour l'existence des multiflots. Nous allons d'abord présenter la notion de coupe sur un graphe, de critère de coupe, et le théorème de Menger [67]. Puis nous allons présenter le théorème japonais[48, 79], qui en est une généralisation.

Pour des raisons de simplicité d'écriture, nous définissons une coupe sur un graphe comme un ensemble d'arcs séparant un sous ensemble de sommets du reste du graphe.

Définition 14 (coupe induite) *Soit C un sous-ensemble de V tel que $C \neq \emptyset$ et $C \neq V$. L'ensemble d'arcs sortants $(C \times (V \backslash C)) \bigcap A$ est appelé coupe induite par C. Pour toute fonction $f : A \to \mathbb{N}$, nous appelons $f(C)$ la somme des valeurs des arcs sortants $\Sigma_{(a \in A(C))}\ f(x, y)$.*

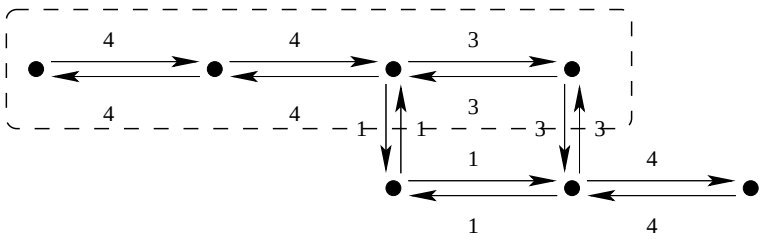


FIG. 3.5 – Une coupe induite de valeur 4.

Définition 15 (critère de coupe) *Soit $\kappa : A \to \mathbb{N}$ une capacité sur G. Soit $s_1, ..s_k$, $t_1, ..t_k \in V$. Soit $v_1, ..v_k \in \mathbb{N}$. Le critère de coupe pour le $(k+1)$-uplet $(\kappa, (s_1, t_1, v_1), .. (s_k, t_k, v_k))$ est : pour toute coupe induite par $C \subset V$, pour tout $I \subset \{1, ..k\}$: $(\forall i \in I,\ s_i \in C$ et $t_i \notin C) \Rightarrow \kappa(C) \geq (\Sigma_{i \in I} v_i)$.*

Le théorème de Menger [67] spécifie que le critère de coupe est une condition nécessaire et suffisante pour le problème du flot entier. Cela a pour conséquence que le critère de coupe est une condition nécessaire pour le problème du multiflot entier.

Théorème 12 (Menger) *Soit $\kappa : A \to \mathbb{N}$ une capacité sur G. Soit $s_1,..s_k$, $t_1,..t_k \in V$. Soit $v_1,..v_k \in \mathbb{N}$. Le critère de coupe pour $(\kappa,(s_1,t_1,v_1),..\ (s_k,t_k,v_k))$ est une condition nécessaire pour l'existence d'un multiflot de $(s_1,s_2,...)$ vers $(t_1,t_2,...)$ de valeur $(v_1,v_2,...)$.*

Dans le cadre de l'étude des multiflots, le critère de coupe a été généralisé au début des années 70 par Iri, Onaga et Kakusho [48, 79], pour donner une condition nécessaire et suffisante sur l'existence d'un multiflot (fractionnaire), connue sous le nom de théorème japonais :

Définition 16 (critère de distance) *Soit $\kappa : A \to \mathbb{N}$ une capacité sur G. Soit $s_1,..s_k$, $t_1,..t_k \in V$. Soit $v_1,..v_k \in \mathbb{N}$. Le critère de distance pour le $(k+1)$-uplet $(\kappa,(s_1,t_1,v_1),..\ (s_k,t_k,v_k))$ est : pour toute fonction de coût sur les arcs $w : A \to \mathbb{N}$ on a*

$$\sum_{1\leq i\leq k} D_w(s_i,t_i)\times v_i \leq \sum_{a\in A} w(a)\times\kappa(a),$$

où $D_w(x,y)$ est la coût minimum d'un chemin de x à y.

Théorème 13 (japonais) *Soit $\kappa : A \to \mathbb{N}$ une capacité sur G. Soit $s_1,..s_k$, $t_1,..t_k \in V$. Soit $v_1,..v_k \in \mathbb{N}$. Le critère de distance pour $(\kappa,(s_1,t_1,v_1),..\ (s_k,t_k,v_k))$ est une condition nécessaire pour l'existence d'un multiflot entier de $(s_1,s_2,...)$ vers $(t_1,t_2,...)$ de valeur $(v_1,v_2,...)$.*

Le critère de distance n'est qu'une condition nécessaire dans le cas des multiflots entiers. On peut interpréter la partie gauche de l'inégalité comme le coût minimum du multiflot, et la partie droite de l'inégalité comme la valeur totale contenue dans le graphe G. Nous verrons page 67 un cas où le critère de coupe est vérifié mais où le critère de distance ne l'est pas.

3.2.3 Problème du biflot dans un graphe orienté

Le problème du biflot est un cas particulier du problème du multiflot ou il n'y a que deux commodités. Nous présentons ici une preuve similaire à celle de Even, Itai & Shamir [23], montrant que le problème du biflot entier est NP-complet.

Théorème 14 *Soit n un entier naturel, G un graphe orienté dont les arêtes sont munies d'une capacité unitaire, et $(s_1,p_1)m\ (s_2,p_2)$ 2 couples de sommets . Déterminer s'il existe un biflot entier de (s_1,s_2) vers (p_1,p_2) de valeur $(1,n)$ est un problème NP-complet.*

Preuve
Le problème du biflot entier est un cas particulier du problème du multiflot entier, et fait donc partie de la classe NP. Nous allons présenter une réduction de 3-SAT au problème du biflot entier. On se donne une instance de 3-SAT, avec des variables booléennes $b_1,b_2...b_m$ et des clauses $C_1,C_2...C_n$.
Construction du graphe G :

- Pour chaque variable booléenne b_i on construit deux chaînes $x_{(i,0)},x_{(i,1)},\dots x_{(i,n)}$ et $\overline{x}_{(i,0)},\overline{x}_{(i,1)},\dots \overline{x}_{(i,n)}$. Ces deux chaînes sont reliées par leurs extrémités, c'est à dire que $x_{(i,0)} = \overline{x}_{(i,0)}$ et que $x_{(i,n)} = \overline{x}_{(i,n)}$.

– On ajoute ensuite l'arc $(s_1, x_{(1,0)})$, pour tout $i \in \{2, m\}$ l'arc $(x_{(i-1,n)}, x_{(i,0)})$ et l'arc $(x_{(m,n)}, p_1)$. La colonne ainsi construite est illustrée à la figure 3.6. On peut remarquer qu'un chemin P de cette colonne, de s_1 vers p_1, détermine chaque variable booléen b_i suivant qu'il passe par les sommets x_i ou $\overline{x}_i$.
– Pour chaque clause C_j de 3-SAT on ajoute les sommets y_j et y'_j. Si la variable booléenne b_i apparaît positivement dans la clause C_j, on ajoute les arcs $(y_j, \overline{x}_{(i,j-1)})$ et $(\overline{x}_{(i,j)}, y'_j)$. Si la variable booléenne b_i apparaît négativement dans la clause C_j, on ajoute les arcs $(y_j, x_{(i,j-1)})$ et $(x_{(i,j)}, y'_j)$. Cette construction est illustrée à la figure 3.7.
– Enfin, pour tout $j \in \{1, n\}$, on ajoute enfin les arcs (s_2, y_j) et (y'_j, p_2). Cette construction est illustrée à la figure 3.8.

3-SAT a une solution ⇒ le biflot existe :
La solution de 3-SAT s'exprime par une détermination des variables booléennes b_i à vrai ou faux : elle induit un chemin P entre s_1 et p_1 dans la colonne du graphe représentant les variables booléennes. Chaque clause C_i étant satisfaite, il existe un chemin $y_j x_{(i,j-1)} x_{(i,j)} y'_j$ ou $y_j \overline{x}_{(i,j-1)} \overline{x}_{(i,j)} y'_j$ disjoint de P. L'ensemble de ces chemins créée un flot de s_2 à p_2 de valeur n.
Le biflot existe ⇒ 3-SAT a une solution :
Un chemin P de s_1 vers p_1 ne peut quitter la colonne des variable booléennes. Il détermine ainsi une affectation des variables booléennes. Le fait qu'il existe un flot de s_2 à p_2 de valeur n implique que pour chaque $j \in \{1, n\}$, il existe un chemin de y_j vers y'_j et donc que la clause C_j est satisfaite. □

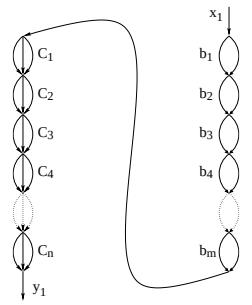


FIG. 3.6 – Colonne des variables booléennes

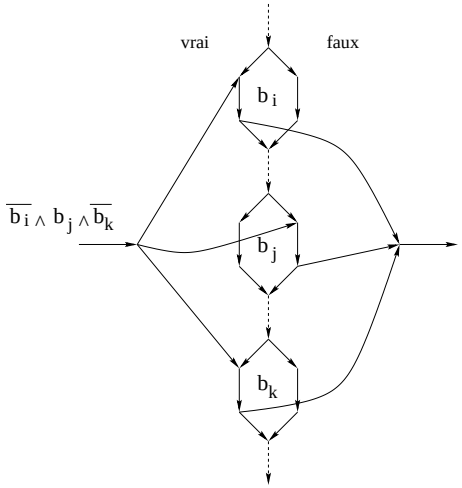


FIG. 3.7 – Une clause de 3-SAT

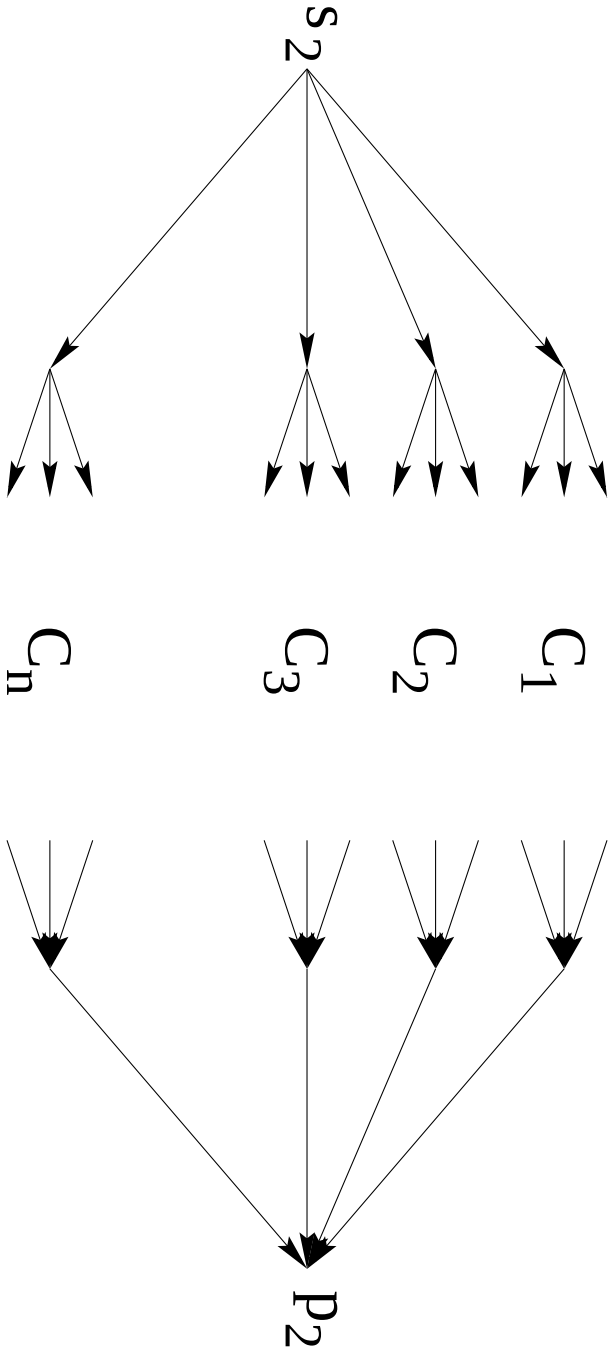


FIG. 3.8 – Les clauses de 3-SAT sont mises en parallèle entre s_2 et p_2

3.3 Multiflots dans les graphes symétriques

Dans cette section, nous présentons des résultats préliminaires sur les multiflots dans les graphes symétriques. Dans un premier temps, nous allons montrer qu'une des conséquences du résultat d'Even, Itai et Shamir [23] sur les multiflots à 2 commodités dans les graphes orientés acycliques est que le problème du multiflot à 3 commodités est NP-complet dans les graphes symétriques. Nous présentons ensuite quelques résultats sur le problème à deux commodités, qui n'est pas encore résolu. Dans un deuxième temps, nous introduisons formellement ce qu'est un multiflot symétrique, et montrons sans surprises que le problème général des multiflots symétriques est NP-difficile. Enfin, nous présentons un résultat préliminaire sur les flots demi-entiers, qui sera utilisé à la section 3.4, consacrée au problème du biflot symétrique.

3.3.1 Multiflot à 3 commodités

En 1976, Even, Itai et Shamir [23] on montré que le problème du multiflot entier à deux commodités était NP-complet. Leur démonstration utilise un graphe acyclique, ce qui fait que le problème est également NP-complet sur la classe des graphes acycliques. En fait, un graphe orienté acyclique est presque équivalent à un graphe orienté symétrique dont la moitié des arcs serait saturée par un flot. Nous allons montrer de cette manière que le problème du multiflot à 3 commodités est NP-complet dans les graphes symétriques.

Théorème 15 *Le problème du multiflot entier à 3 commodités est NP-complet dans les graphes symétriques.*

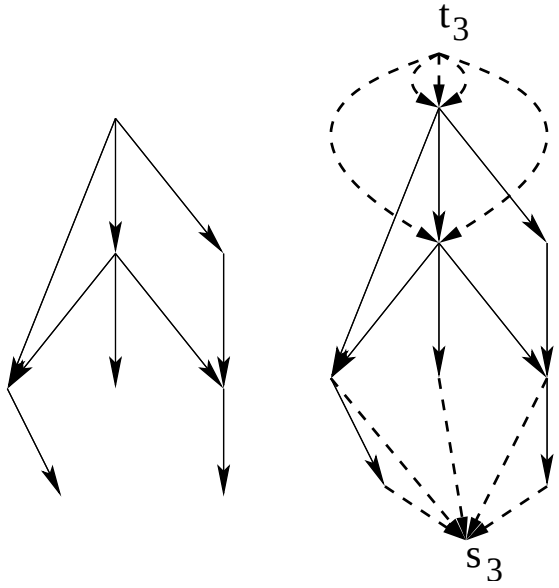
Preuve
Soit $G = (V, A)$ un graphe orienté acyclique. Soient (s_1, s_2) et (t_1, t_2) deux sources et deux puits. Soient v_1 et v_2 deux entiers positifs. Nous allons construire un graphe orienté symétrique G', définir deux sommets s_3 et t_3 et une valeur v_3, et nous allons montrer que déterminer s'il existe un multiflot à deux commodités de (s_1, s_2) vers (t_1, t_2) de valeur (v_1, v_2) dans le graphe G muni d'une capacité 1 sur tous ses arcs est équivalent à déterminer s'il existe un multiflot de (s_1, s_2, s_3) vers (t_1, t_2, t_3) de valeur (v_1, v_2, v_3) dans le graphe G^* muni d'une capacité symétrique κ^* sur tous ses arcs. Nous construisons un graphe acyclique $G' = (V', A')$ comme suit (construction illustrée à la figure 3.9) :

- soit $V' = V \bigcup \{s_3, t_3\}$
- soit $A' = A \bigcup (\cup_{x \in V}(x, s_3)) \bigcup (\cup_{x \in V}(t_3, x))$
- soit $\kappa : A' \to \mathbb{N}$ tel que pour tout arc $(x, y) \in A$, $\kappa(x, y) = 1$ et tel que pour tout $x \in V$ de degré entrant $\delta^-(x)$ et de degré sortant $\delta^+(x)$,
  - si $\delta^-(x) > \delta^+(x)$, alors $\kappa(x, s_3) = \delta^-(x) - \delta^+(x)$.
  - si $\delta^+(x) > \delta^-(x)$, alors $\kappa(t_3, x) = \delta^+(x) - \delta^-(x)$.

Les seules différences entre les graphes G' et G sont l'ajout des sommets t_3 et t_3, l'ajout d'arcs vers le sommet s_3 et l'ajout d'arcs depuis le sommet t_3. De plus, la capacité κ est unitaire sur les arcs de G. Aussi, un multiflot de (s_1, s_2) vers (t_1, t_2) de valeur (v_1, v_2) existe dans G muni d'une capacité unitaire si et seulement si il existe dans G' muni de la capacité κ.

Le graphe G' que nous avons défini est orienté acyclique. La définition de κ assure que pour tout sommet $x \in V$, la somme des capacités entrantes est égale à la somme des capacités sortantes. En

conséquence, la somme des capacités entrantes de s_3 est égale à la somme des capacités sortantes de t_3. On définit $v_3 = \Sigma_{x \in V} \kappa(s_3, x)$. On définit le graphe symétrique $G^* = (V', A'^*)$ muni de la capacité κ^*. Un flot de valeur v_3 de s_3 vers t_3 dans G^* sature nécessairement tous les arcs (x, y) tels que $(y, x) \in A'$. Aussi, il existe un multiflot de (s_1, s_2) vers (t_1, t_2) de valeur (v_1, v_2) dans G muni d'un capacité unitaire si et seulement si il existe un multiflot de (s_1, s_2, s_3) vers (t_1, t_2, t_3) de capacité (v_1, v_2, v_3) dans G^* muni de la capacité κ^*. Le problème du multiflot à 3 commodités est donc NP-complet dans un graphe orienté symétrique. □



(a) graphe G (b) graphe G'

FIG. 3.9 – Graphe orienté acyclique G. Deux sommets s_3 et t_3 sont ajoutés. Des arcs en provenance de t_3 et des arcs à destination de s_3 sont ajoutés de telle sorte que pour chaque sommet $x \notin \{s_3, t_3\}$, les capacités entrantes sont égales aux capacités sortantes.

3.3.2 Le problème du biflot dans les graphes symétriques

Nous savons désormais que le problème du multiflot entier à 3 commodités est NP-complet dans les graphes symétriques (Théorème 15 page 62). Nous ne connaissons pas la complexité du problème pour 2 commodités. Nous allons d'abord présenter un algorithme polynomial pour une requête de valeur $(1, v)$, puis nous allons montrer que le critère de coupe n'est pas une condition suffisante pour résoudre le problème du biflot.

Théorème 16 *Le problème du multiflot entier est polynomial dans les graphes symétriques pour deux commodités de valeur* $(1, v)$.

Preuve
Nous supposons que le critère de coupe est vérifié (le test peut se faire en temps polynomial). Nous allons calculer un flot de valeur v pour la première commodité, puis un flot de valeur 1 pour la

seconde commodité.
Les données du problème sont les suivantes :
- une capacité symétrique sur les arcs de A $\kappa : A \to \mathbb{N}$;
- une source et une destination (s_1, s_2), $(t_1, t_2) \in V^2$ pour le multiflot souhaité ;
- la valeur du multiflot souhaité $(1, v) \in \mathbb{N}^2$.

Tout d'abord, nous calculons un flot simple $f_2 \leq \kappa$ de s_2 vers t_2 de valeur v. Si le critère de coupe est vérifié, alors f_2 existe, et peut être calculé par n'importe quel algorithme polynomial de flot. Nous allons maintenant prouver que le critère de coupe est encore vrai pour $((\kappa - f_2), (s_1, t_1, 1))$. Soit $C \subset V$ une coupe telle que $s_1 \in C$ et $t_1 \notin C$. Deux cas sont possibles :
- si $s_2 \in C$ et si $t_2 \notin C$, alors $\kappa(C) \geq (1 + v)$. Si $f_2(C) \geq v + 1$, alors il existe $x \in C$ et $y \notin C$ avec $(x, y) \in A$ tels que $f_2(y, x) \geq 1$ d'où $f_2(x, y) = 0$, ce qui donne $(\kappa - f_2)(C) \geq 1$
- dans le cas contraire, $\kappa(C) \geq 1$. Si $f_2(C) \geq 1$, alors il existe $x \in C$ et $y \notin C$ avec $(x, y) \in A$ tels que $f_2(y, x) \geq 1$ d'où $f_2(x, y) = 0 : (\kappa - f_2)(C) \geq 1$

Comme le critère de coupe est vérifié pour $((\kappa - f_2), (s_1, t_1, 1))$, d'après le Théorème de Menger il existe bien un flot $f_1 \leq (\kappa - f_2)$ de s_1 vers t_1 et de valeur 1. □

Nous allons maintenant montrer que le critère de coupe n'est pas une condition suffisante pour ce problème même si elle reste nécessaire. Voici un contre-exemple où le critère de coupe est valide, mais où les requêtes sont impossibles à satisfaire :
- nous prenons comme ensemble de sommets $V = \{s_1, s_2, t_1, t_2\}$;
- nous prenons comme ensemble d'arcs $A = \{(s_1, s_2), (s_2, s_1), (s_2, t_1), (t_1, s_2), (s_1, t_2), (t_2, s_1), (t_2, t_1), (t_1, t_2)\}$;
- nous considérons le graphe orienté symétrique $G = (V, A)$;
- la capacité des arcs de G est $\kappa : A \to \mathbb{N}$ et vérifie

$$\kappa(s_1, s_2) = \kappa(s_2, s_1) = 1, \quad \kappa(s_2, t_1) = \kappa(t_1, s_2) = 1,$$
$$\kappa(s_1, t_2) = \kappa(t_2, s_1) = 3, \quad \kappa(t_2, t_1) = \kappa(t_1, t_2) = 1.$$

Le critère de coupe est bien vérifié pour $(\kappa, (s_1, t_1, 2), (s_2, t_2, 2))$, mais il n'y a pas de multiflot de (s_1, s_2) vers (t_1, t_2) de valeur $(2, 2)$.

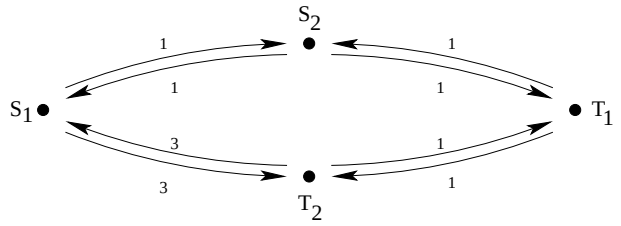


FIG. 3.10 – Il n'y a pas de multiflot de (s_1, s_2) vers (t_1, t_2) de valeur $(2, 2)$.

3.3.3 Multiflots symétriques

Si un multiflot non-orienté représente une solution pour une instance de communication entre deux paires de sommets, alors pour chaque paire de sommets les mêmes chemins sont utilisés pour la direction aller et retour. Si nous permettons à la direction "retour" d'emprunter un chemin différent du chemin aller, nous obtenons un multiflot symétrique.

Définition 17 (multiflot symétrique) *Soit $G = (V, A)$ un graphe orienté symétrique. Soient $s_1, s_2, .., t_1, t_2...$ des sommets de G et $(v_1, v_2...)$ des entiers positifs. Un multiflot de $(s_1, t_1, s_2, t_2, ...)$ vers $(t_1, s_1, t_2, s_2, ...)$ de valeur $(v_1, v_1, v_2, v_2, ...)$ est appelé multiflot symétrique de $(s_1, s_2, ...)$ vers $(t_1, t_2, ...)$ de valeur $(v_1, v_2, ...)$.*

Nous définissons le critère de coupe symétrique, écriture simplifiée du critère de coupe que nous avons défini page 59.

Définition 18 (critère de coupe symétrique) *Soit $G = (V, A)$ un graphe orienté symétrique. Soit $\kappa : A \to \mathbb{N}$ une capacité symétrique sur G. Soient $s_1, s_2, .., t_1, t_2...$ des sommets de G et $(v_1, v_2...)$ des entiers positifs. Le critère de coupe pour $(\kappa, (s_1, t_1, v_1), (t_1, s_1, v_1), (s_2, t_2, v_2), (t_2, s_2, v_2), ...)$ est appelé critère de coupe symétrique pour $(\kappa, (s_1, t_1, v_1), (s_2, t_2, v_2), ...)$*

Nous montrerons à la section 3.4 que le problème du biflot symétrique est polynomial si et seulement si le critère de coupe symétrique est vérifié. Nous ne connaissons pas la complexité du problème du multiflot symétrique à k commodités, pour $k \geq 3$. Dans le cas ou le nombre de commodités est indéterminé, nous allons montrer à présent que le problème du multiflot symétrique est NP-complet.

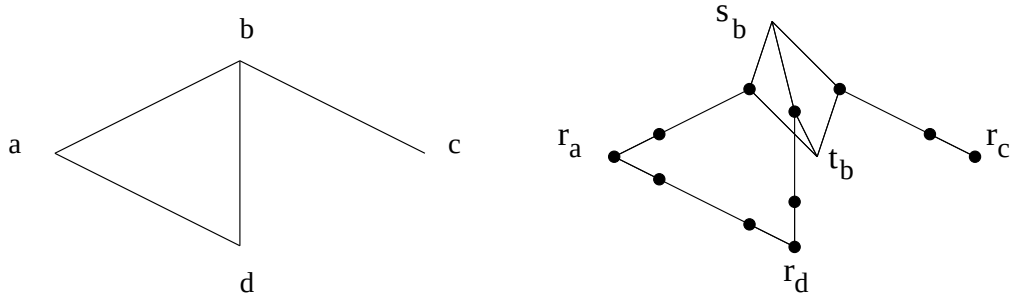
Théorème 17 *Le problème du multiflot symétrique est NP-complet.*

Preuve
Nous allons réduire le problème des chemins disjoints (voir chapitre 2) dans un graphe non-orienté au problème du multiflot symétrique. Soit $G = (V, E)$ un graphe non orienté et R un ensemble de requêtes (pour des chemins sommet-disjoints) sur G. On suppose en plus que les sommets présents dans l'ensemble de requêtes R sont de degré 1. On construit le graphe orienté symétrique $G' = (V', A)$ comme suit (voir Figure 3.11) :

- pour tout sommet x de V de degré supérieur ou égal à trois, on construit
  - les sommets s_x et t_x dans V'
  - pour toute arête (x, y) de E, on construit le sommet e_{xy} dans V', et les arcs (s_x, e_{xy}), (e_{xy}, s_x), (t_x, e_{xy}) et (e_{xy}, t_x) dans A.
- pour tout sommet x de V de degré 1 ou 2, on construit
  - le sommet r_x dans V'
  - pour toute arête (x, y) de E, on construit le sommet e_{xy} dans V', et les arcs (r_x, e_{xy}), et (e_{xy}, r_x) dans A.
- pour toute arête (x, y) de E, on construit les arcs (e_{xy}, e_{yx}) et (e_{yx}, e_{xy}).
- enfin, pour toute requête $r = (x, y)$ de R, on note $s_r = r_x$ et $t_r = r_y$.

Nous allons montrer que satisfaire l'ensemble des requêtes R dans G revient à résoudre le problème du multiflot symétrique de $(s_x)_{x \in V \cup R}$ vers $(t_x)_{x \in V \cup R}$ de valeur $(\delta^+(s_x) - 1)$ dans le graphe symétrique $G' = (V', A)$ muni d'une capacité unitaire.

Pour chaque sommet x de V, examinons la requête de s_x vers t_x de valeur $\delta(x) - 1$. Une solution pour cette requête consiste à utiliser un chemin $(s_x, e_{xy}), \mu, (e_{xz}, t_x)$, ou μ est un chemin de e_{xy} vers e_{xz}. On peut toujours remplacer ce chemin long par le chemin plus simple $(s_x, e_{xy}), (e_{xy}, t_x))$: si une autre requête utilisait l'arc $(e_{x,y}, t_y)$, elle utilisera désormais le chemin libéré $\mu, (e_{xz}, t_x)$. La solution simplifiée pour la requête de s_x vers t_x utilise $(\delta(x) - 1)$ chemins de s_x à t_x et $(\delta(x) - 1)$



(a) graphe G (b) graphe G'

FIG. 3.11 – Graphe non orienté G. Le graphe orienté symétrique G' (représenté en (b) par son squelette non orienté) est construit à partir de G' en dédoublant les sommets de degré supérieur ou égal à trois.

chemins de t_x à s_x. Il y donc quatre arcs de libres correspondant à deux voisins y et z de x : les arcs (s_x, e_{xy}), (e_{xy}, t_z), (t_x, e_{xz}) et (e_{xz}, s_x) (ceci est illustré à la figure 3.12).

Pour chaque requête $r = (x, y)$ de R, examinons maintenant la requête $(s_r, t_r) = (r_x, r_y)$ de valeur 1. Une solution pour r s'exprime sous la forme d'un chemin μ de r_x vers r_y et d'un chemin μ^{-1} de r_y vers r_x. Comme x et de degré 1 par hypothèse, r_x a un unique sommet voisin e_{xz}. Le chemin μ emprunte donc l'arc (e_{xz}, e_{zx}) et le chemin μ^{-1} emprunte l'arc opposé (e_{zx}, e_{xz}). Comme on peut le voir à la Figure 3.12, ceci implique que les chemins μ et μ^{-1} empruntent toujours des arcs opposés de type (e_{ab}, e_{ba}) et (e_{ba}, e_{ab}).

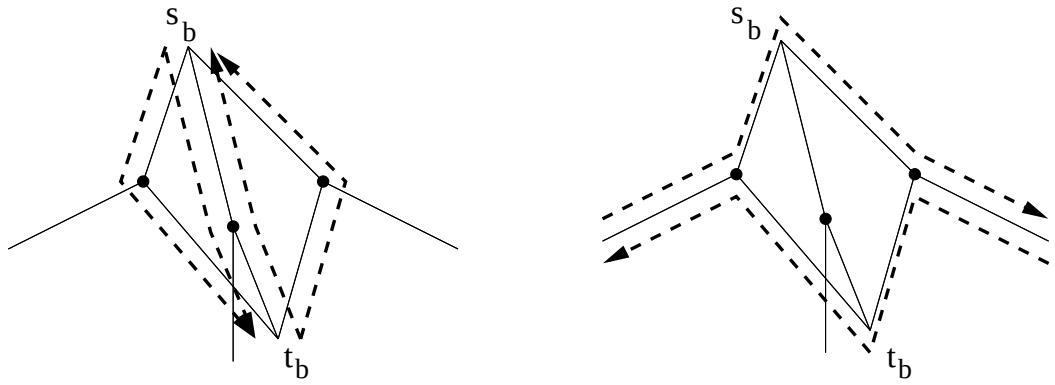
Enfin pour chaque sommet $x \in V$, une requête de type (s_x, t_x) ne permet le passage que d'une seule paire de chemins $\mu, \mu^{-}1$ (voir Figure 3.12).

Aussi on peut en déduire que la réalisation d'un multiflot symétrique dans G' correspond exactement à la réalisation de chemins sommet-disjoints dans G. □

Nous verrons à la section 3.4 que le problème du biflot symétrique a une solution si et seulement si le critère de coupe symétrique est vérifié. Cependant, le critère de coupe symétrique n'est pas une condition suffisante pour 3 commodités ou plus, comme le montre la figure 3.13.

3.3.4 Flots demi-entiers

Pour la résolution du problème du biflot symétrique, présenté à la section 3.4, nous aurons besoin d'effectuer des arrondis de flots demi-entiers. Nous traitons ici ces arrondis. Nous considérons un graphe orienté $G = (V, A)$, et un flot entier f de valeur paire sur G. Le fait que f soit de valeur paire signifie que la quantité de flot quittant la source est paire. Par contre, la quantité de flot traversant un arc donné du graphe peut être paire ou impaire. Nous allons montrer au théorème 18 que le flot f peut se décomposer en deux flots entiers g et g', tels que pour tout arc $a \in A$, $2g(a) \leq (f(a) + 1)$ et $2g'(a) \leq (f(a) + 1)$. Cette dernière propriété signifie que g et g' sont des arrondis de la fonction $\frac{f}{2}$. Autrement dit, la signification du théorème 18 est donc que l'on peut choisir un arrondi g de la



(a) chemins utilisés pour la requête (s_b, t_b) (b) chemins libres

FIG. 3.12 – Cet exemple reprend la construction à partir du sommet b de la Figure 3.11. Les deux requêtes symétriques pour le sommet b laissent libres 2 paires d'arcs.

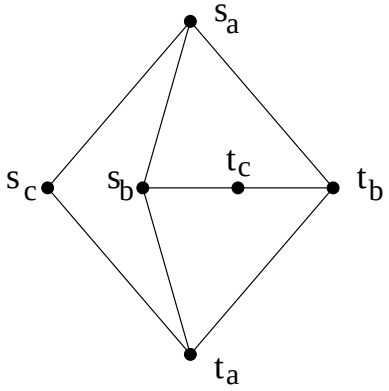


FIG. 3.13 – Graphe orienté symétrique G contenant 6 sommets et 8 paires d'arcs. Si on munit G d'une capacité unitaire, le critère de coupe symétrique est vérifié pour $(1, (s_a, t_a, 2), (s_b, t_b, 1), (s_c, t_c, 1))$. Un chemin de s_a à t_a nécessite 2 arcs, un chemin de s_b à t_b nécessite 2 arcs, et un chemin de s_c à t_c nécessite 3 arcs. Un multiflot symétrique de (s_a, s_b, t_b) vers (t_a, t_b, t_c) de valeur $(2, 1, 1)$ nécessiterait 18 arcs alors que G ne contient que 16 arcs.

fonction $\frac{f}{2}$ tel que g est un flot. Nous donnons un algorithme calculant cet arrondi (Algorithme 1) ; la preuve du théorème 18 repose sur la validité de l'Algorithme 1.

Algorithme 1 (Demi-flot)

Complexité: $O(|A|)$

Entrée: un graphe orienté $G = (V, A)$ et un flot entier f de valeur paire sur G.

Sortie: une fonction $g : A \to \mathbb{N}$

Variables: une fonction $f' : A \to \mathbb{N}$
Début

1. $\forall a \in A$, $f'(a) \leftarrow f(a)$
2. tant qu'il existe un arc $(x_{start}, y_{start}) \in A$ tel que $f'(x_{start}, y_{start})$ est impair
   - $f'(x_{start}, y_{start}) \leftarrow f'(x_{start}, y_{start}) + 1$
   - $x \leftarrow y_{start}$
   - tant que $x \neq x_{start}$ faire
     - s'il existe $y \in \Gamma^+(x)$ tel que $f'(x, y)$ est impair alors
       - $f'(x, y) \leftarrow f'(x, y) + 1$
       - $x \leftarrow y$
     - sinon soit $y \in \Gamma^-(x)$ tel que $f'(y, x)$ est impair
       - $f'(y, x) \leftarrow f'(y, x) - 1$
       - $x \leftarrow y$
3. $\forall a \in A$, $g(a) = \frac{f'(a)}{2}$

Fin

Théorème 18 (demi-flot) *Soit $G = (V, A)$ un graphe orienté et f un flot entier de valeur paire $2v$ sur G. Il existe un flot g de valeur v sur G tel que pour tout arc $a \in A$, $(f - 1)(a) \leq 2g(a) \leq (f + 1)(a)$. Ce flot peut être calculé par l'algorithme 1 en temps $O(|A|)$.*

Le théorème se démontre en deux étapes. Nous montrons d'abord que l'algorithme 1 se termine correctement, puis nous montrons que la sortie g a bien les propriétés requises.

Proposition 6 *L'Algorithme 1 se termine en temps $O(|A|)$.*

Preuve
Comme f est un flot de valeur paire, alors pour tout sommet $x \in V$ il y a un nombre pair de sommets y dans le voisinage de x tel que $f'(x, y)$ ou $f'(y, x)$ est impair. Cette propriété est vraie pour la source et pour le puits de f car la valeur de f est paire, et pour les autres sommets car la loi de conservation du flot l'impose. Aussi, lorsque l'algorithme traite un sommet x après avoir modifié f' sur un arc adjacent à x, il existe nécessairement un autre arc adjacent à x sur lequel f' est encore impair, sauf si le sommet x est celui par lequel l'algorithme a commencé x_{start}. De plus, à chaque étape le nombre d'arcs $a \in A$ tel que $f'(a)$ est impair décroît, ce qui veut dire que le nombre de pas de calculs est borné par le nombre d'arcs a tels que $f(a)$ est impair. La complexité de l'algorithme est donc en $O(|A|)$. □

Proposition 7 *Si f est un flot de valeur $2v$, la sortie g de l'Algorithme 1 est un flot de valeur v sur G tel que pour tout arc $a \in A$, $(f(a) - 1) \leq 2g(a) \leq (f(a) + 1)$.*

Preuve
Au début de l'algorithme, f' est un flot de valeur $2v$. Au début de chaque boucle de (pas de calcul 2), les équations de flot sont violées en x_{start} et en y_{start}. A chaque étape de la boucle, les équations de flots sont restaurées en x mais violées en y, ce qui fait qu'il y a toujours un déséquilibre pour les deux sommets x_{start} (une unité de flot sortante en trop) et x (une unité de flot entrante en trop). A la fin de la boucle, $x = x_{start}$, ce qui fait que les équations de flot sont vérifiées pour tout les sommets. A la fin de l'algorithme, f' est par conséquent un flot de valeur $2v$ sur G. De plus, la valeur de f' est modifiée exactement une fois pour chaque arc où f' était impair, ce qui fait que pour tout $a \in A$, $f'(a)$ est pair et $|(f'(a) - f(a)| \leq 1$. Comme g est fixé à $\frac{f'}{2}$ à la fin de l'algorithme, g est bien un flot de valeur v tel que $\forall a \in A$, $(f(a) - 1) \leq 2g(a) \leq (f(a) + 1)$. □

3.4 Biflot symétrique

Dans cette section, nous considérons un graphe symétrique $G = (V, A)$ avec une capacité symétrique $\kappa : A \to \mathbb{N}$. Nous considérons aussi deux couples de sommets (s_1, t_1) et (s_2, t_2), et deux entiers v_1 et v_2. Nous allons montrer que le problème du multiflot symétrique peut se résoudre en temps polynomial.

Le but de cette partie est d'expliciter la preuve du théorème suivant.

Théorème 19 (biflot symétrique) *Le critère de coupe symétrique est une condition nécessaire et suffisante pour l'existence d'une solution au problème du biflot symétrique. Une telle solution peut être trouvée par l'Algorithme 2 en* $6C_{flow} + O(|A|)$ *étapes.*

Voici l'Algorithme 2 qui résout le problème du biflot symétrique. L'algorithme est divisé en trois parties, qui seront expliquées et prouvées successivement.

Algorithme 2 (Biflot symétrique)

Complexité: $6C_{flow} + O(|A|)$

Entrée: Un graphe orienté symétrique $G = (V, A)$; une capacité symétrique $\kappa : A \to \mathbb{N}$; $s_1, t_1, s_2, t_2 \in V$; $v_1, v_2 \in \mathbb{N}$.

Sortie: $f_1 : A \to \mathbb{N}$; $f_{-1} : A \to \mathbb{N}$; $f_2 : A \to \mathbb{N}$; $f_{-2} : A \to \mathbb{N}$.

Variables: une fonction $f : A \to \mathbb{N}$
Début

1. calculer à l'aide de l'**Algorithme 3 (première étape)** un flot f de s_1 vers t_1 de valeur v_1 tel que le critère de coupe est vérifié pour $((\kappa - f), (s_2, t_2, v_2), (t_2, s_2, v_2))$
2. calculer à l'aide de f et de l'**Algorithme 4 (deuxième étape)** et f deux flots (f_2, f_{-2}) de (s_2, t_2) vers (t_2, s_2) de valeur (v_2, v_2) tel que $(f_2 + f_{-2}) \leq \kappa$ et tel que le critère de coupe est vérifié pour $((\kappa - f_2 - f_{-2}), (s_1, t_1, v_1))$.
3. calculer à l'aide de l'**Algorithme 5 (troisième étape)** deux flots (f_1, f_{-1}) de (s_1, t_1) vers (t_1, s_1) de valeur (v_1, v_1) tel que $(f_1 + f_{-1} + f_2 + f_{-2}) \leq \kappa$.

Fin

Tout d'abord nous présenterons la première partie de notre algorithme (Algorithme 3) qui fournit un flot pour la première commodité, et tel que les capacités restantes permettent le respect du critère de coupe pour la seconde commodité. Cependant, le flot ainsi calculé ne possède pas assez de propriétés pour faire partie de la solution au problème du biflot symétrique. Ensuite, nous présenterons le calcul de deux flots (flot aller et flot retour) pour la deuxième commodité à l'Algorithme 4, en utilisant le flot calculé avec l'Algorithme 3. Ces deux flots pour la deuxième commodité feront partie de la solution au problème du biflot symétrique. Enfin, nous calculons les deux flots (flot aller et flot retour) pour la première commodité (Algorithme 5).

Pour commencer, nous décrivons un algorithme qui calcule un flot f satisfaisant le critère de coupe pour $((\kappa - f), (s_2, t_2, v_2), (t_2, s_2, v_2))$.

Algorithme 3 (Première étape)

Complexité: $3C_{flow} + O(|A|)$

Entrée: Un graphe orienté symétrique $G = (V, A)$; une capacité symétrique $\kappa : A \to \mathbb{N}$; $s_1, t_1, s_2, t_2 \in V$; $v_1, v_2 \in \mathbb{N}$.

Sortie: $f : A \to \mathbb{N}$

Variables: $g, g', h : A \to \mathbb{N}$
Début

1. calculer un flot $h \leq \kappa$ de s_2 vers t_2 et de valeur v_2
2. soit h^r le flot retour de h :
   (a) calculer un flot simple $g \leq (\kappa + h^r - h)$ de s_1 vers t_1 de valeur v_1
   (b) calculer un flot simple $g' \leq (\kappa + h - h^r)$ de s_1 vers t_1 de valeur v_1
3. calculer à l'aide de l'**Algorithme 1** un flot f de s_1 vers t_1 de valeur v_1 tel que $2f \leq (|g + g'| + 1)$

Fin

Lemme 13 (première étape) *Soit $G = (V, A)$ un graphe symétrique. Soit $\kappa : A \to \mathbb{N}$ une capacité symétrique sur G. Soit s_1, t_1, $s_2, t_2 \in V$ et v_1, $v_2 \in \mathbb{N}$ tels que le critère de coupe symétrique soit vérifié pour le triplet $(\kappa, (s_1, t_1, v_1), (s_2, t_2, v_2))$. Alors il existe un flot $f \leq \kappa$ de s_1 vers t_1 de valeur v_1 tel que le critère de coupe est vérifié pour le triplet $((\kappa - f), (s_2, t_2, v_2), (t_2, s_2, v_2))$. Un tel flot peut être calculé par l'Algorithme 3 en temps $3C_{flow} + O(|A|)$.*

Nous allons d'abord montrer que l'Algorithme 3 se termine correctement, puis que le flot calculé a les propriétés attendues.

Proposition 8 *Si le critère de coupe symétrique est vérifé pour le triplet $(\kappa, (s_1, t_1, v_1), (s_2, t_2, v_2))$, alors l'Algorithme 3 se termine en temps $3C_{flow} + O(|A|)$.*

Preuve
Comme par le critère de coupe symétrique est vérifié par hypothèse pour $(\kappa, (s_1, t_1, v_1), (s_2, t_2, v_2))$, cela prouve qu'il existe un flot h de s_2 vers t_2. Si on considère la super source s reliée à s_1 par un arc

de capacité v_1 et à s_2 par un arc de capacité v_2 et le super puits t reliée à t_1 par un arc de capacité v_1 et à t_2 par un arc de capacité v_2, il existe un flot de valeur (v_1+v_2) reliant s à t. Du point de vue de l'algorithme de flot de Ford et Fulkerson par chemins augmentants [35], la capacité $(\kappa+h^r-h)$ est la capacité résiduelle du graphe G après le calcul du flot h. Comme le flot h est de capacité v_2, il existe un flot augmentant g de s_1 vers t_1, de valeur v_1 et soumis à la capacité $(\kappa+h^r-h)$. De la même manière, il existe également flot g' de valeur v_1 de t_1 vers s_1 tel que $g' \leq (\kappa-h^r+h)$. Ces 3 calculs de flots se font naturellement en $3C_{flow}$ $(+O(|A|)$ pour les calculs sur les capacités). Enfin, d'après le Théorème 18, le flot f peut être calculé par l'Algorithme 1 en $O(|A)$. □

Proposition 9 *Si le critère de coupe symétrique est vérifié pour le triplet* $(\kappa,(s_1,t_1,v_1),(s_2,t_2,v_2))$, *alors le flot* f *calculé par l'Algorithme 3 préserve le critère de coupe pour* $((\kappa-f),(s_2,t_2,v_2),(t_2,s_2,v_2))$.

Preuve
Comme $(g+g') \leq 2\kappa$, nous avons $f \leq \kappa$. Avant de considérer la fonction $(\kappa-f)$ nous allons donner deux bornes inférieures pour la fonction $(\kappa-|g+g'|) : A \rightarrow \mathbb{Z}$. Ces bornes nous permettront de cerner la fonction $(\kappa-f)$ sur les coupes entre s_2 et t_2

La fonction g' a été calculé de telle sorte que $g' \leq (\kappa+h-h^r)$; en d'autres termes, $(\kappa-g') \geq (h^r-h)$ (**a**), ou encore $(\kappa-(g+g')) \geq (h^r-h-g)$ (**b**). Maintenant, considérons un arc (x,y) de A tel que $g(x,y)=0$:

- si $g'(x,y) \geq g(y,x)$, alors par définition $|g+g'|(x,y) = g'(x,y)-g(y,x)$, et $(\kappa-|g+g'|)(x,y) = \kappa(x,y)-g'(x,y)+g(y,x)$. En utilisant l'inégalité (**a**) nous avons $(\kappa-|g+g'|)(x,y) \geq (h^r-h)(x,y)+g(y,x)$, ce qui veut dire $(\kappa-|g+g'|)(x,y) \geq (h^r-h+g^r)(x,y)$ (**c**).
- dans le cas contraire, $|g+g'|(x,y)=0$, d'où $(\kappa-|g+g'|)(x,y)=\kappa(x,y)$. Comme $g \leq \kappa+h^r-h$, nous avons $\kappa(y,x) \geq g(y,x)+h(y,x)-h(x,y)$, d'où $\kappa(y,x) \geq (h^r-h+g^r)(x,y)$. Comme $(\kappa-|g+g'|)(x,y)=\kappa(x,y)=\kappa(y,x)$, nous avons $(\kappa-|g+g'|)(x,y) \geq (h^r-h+g^r)(x,y)$ (**c**).

Comme g est un flot simple, pour tout arc (x,y) nous avons soit $g(x,y)=0$ et l'inégalité (**c**) s'applique, soit $g^r(x,y)=0$ et nous retournons à l'inégalité (**b**). Nous avons donc toujours l'inégalité suivante : $(\kappa-|g+g'|) \geq (h-h^r-g+g^r)$ (**d**). En substituant g' à g dans le raisonnement précédent, nous obtenons pour tout arc $(x,y) \in A$, $(\kappa-|g+g'|) \geq (h-h^r-g'+g'^r)$ (**e**).

Maintenant que nous avons déterminé les inégalités (**d**) et (**e**), considérons une coupe induite par $C \subset V$, avec par exemple $s_2 \in C$ et $t_2 \notin C$. Nous allons prouver que $(\kappa-f)(C) \geq v_2$. D'après (**e**), $(\kappa-|g+g'|)(C) \geq (h-h^r-g'+g'^r)(C)$ ce qui donne ici $(\kappa-|g+g'|)(C) \geq v_2+(g'^r-g')(C)$ (**f**).

- si $t_1 \in C$ ou si $s_1 \notin C$, alors $g'(C) \leq g'^r(C)$. Cela nous donne d'après (**f**) $(\kappa-|g+g'|)(C) \geq v_2$. Comme $f \leq |g+g'|$, nous avons $(\kappa-f)(C) \geq v_2$.
- sinon, $s_1 \in C$ et $t_1 \notin C$. Sur la coupe induite par C, cela donne $g'(C) = g'^r(C)+v_1$ et d'après (**f**), $(\kappa-|g+g'|)(C) \geq v_2-v_1$. Dans ce cas nous utilisons le fait que $|g+g'|$ est un flot de valeur $2v_1$ de s_1 vers t_1, d'où $|g+g'|(C) = |g+g'|^r(C)+2v_1$ et $f(C) = f^r(C)+v_1$. Avec $f^r \leq |g+g'|^r$, cela donne $f(C) \leq |g+g'|^r(C)+v_1$ ou encore $f(C) \leq |g+g'|(C)-v_1$. Aussi, $(\kappa-f)(C) \geq v_2$

Le critère de coupe est donc bien vérifié pour $((\kappa-f),(s_2,t_2,v_2))$. Par symétrie, il est également vérifié pour $((\kappa-f),(t_2,s_2,v_2))$. □

Bien que le critère de coupe soit vérifié pour $((\kappa-f),(s_2,t_2,v_2),(t_2,s_2,v_2))$, cela n'est pas suffisant pour garantir l'existence d'un flot f_2 de s_2 vers t_2 de valeur v_2 et d'un flot f_{-2} de t_2

vers s_2 de valeur v_2 tels que $(f + f_2 + 2{-}2) \leq \kappa$. Cependant, l'algorithme proposé (Algorithme 4) calcule deux flots f_2 et f_{-2} à partir du flot f qui feront partie de la solution. Le flot f, finalement incompatible avec f_2 et f_{-2} , est abandonné par la suite. La Figure 3.14 montre un exemple de flot f qui ne pourrait pas faire partie de la solution. Le lecteur peut remarquer que le flot f de cet exemple n'est pas celui que donnerait l'Algorithme 3. En fait, pour tous les exemples que nous avons étudié, le flot f calculé par l'Algorithme 3 fait partie de la solution, c'est à dire que $f_1 = f$. Cela signifie soit que la deuxième partie (Algorithme 4) est superflue, soit qu'un exemple approprié (que nous n'avons pas encore rencontré) existe.

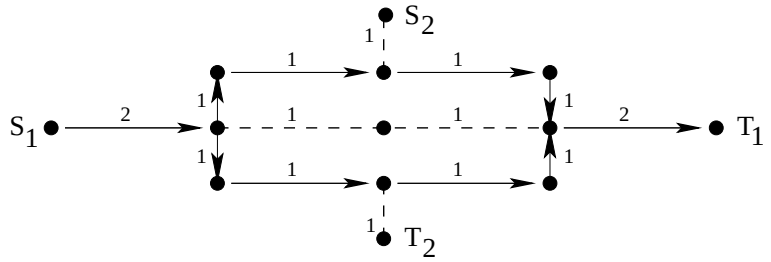


FIG. 3.14 – le critère de coupe est vérifié pour $((\kappa - f), (s_2, t_2, 1), (t_2, s_2, 1))$

Algorithme 4 (Deuxième étape)

Complexité: $2C_{flow} + O(|A|)$

Entrée: Un graphe orienté symétrique $G = (V, A)$; une capacité symétrique $\kappa : A \to \mathbb{N}$; $v_2 \in \mathbb{N}$; $f : A \to \mathbb{N}$.

Sortie: $f_2, f_{-2} : A \to \mathbb{N}$

Variables: $h, h' : A \to \mathbb{N}$
Début

1. calculer un flot $h \leq (\kappa - f)$ de s_2 vers t_2 de valeur v_2
2. calculer un flot $h' \leq (\kappa - f^r)$ de t_2 vers s_2 de valeur v_2
3. calculer à l'aide de l'Algorithme 1 un flot f_2 de s_2 vers t_2 tel que pour tout arc $a \in A$, $(|h+h'|(a)-1) \leq 2f_2(a) \leq (|h + h'|(a) + 1)$.

Fin

Lemme 14 (deuxième étape) *Soit $\kappa : A \to \mathbb{N}$ une capacité symétrique. Soit $s_1, t_1, s_2, t_2 \in V$ et $v_1, v_2 \in \mathbb{N}$. Soit $f \leq \kappa$ un flot de s_1 vers t_1 de valeur v_1 tel que le critère de coupe est vérifié pour $((\kappa - f), (s_2, t_2, v_2), (t_2, s_2, v_2))$. Alors il existe deux flots f_2 et f_{-2} de (s_2, t_2) vers (t_2, s_2) de valeur (v_2, v_2) tels que $(f_2 + f_{-2}) \leq \kappa$ et tels que le critère de coupe est vérifié pour le triplet $((\kappa - f_2 - f_{-2}), (s_1, t_1, v_1))$. Ces deux flots peuvent être calculés par l'Algorithme 4 en temps $2C_{flow} + O(|A|)$.*

La preuve de ce lemme se fera en trois temps : premièrement la complétion de l'algorithme, deuxièmement le fait que $(f_2 + f_{-2}) \leq \kappa$, et troisièmement le fait que le critère de coupe est vrai pour $((\kappa - f_2 - f_{-2}), (s_1, t_1, v_1))$.

Proposition 10 *Si le critère de coupe est vérifié pour* $((\kappa - f), (s_2, t_2, v_2), (t_2, s_2, v_2))$*, alors l'Algorithme 4 se termine en* $2C_{flow} + O(|A|)$ *pas de calcul.*

Preuve
Comme le critère de coupe est vérifié par hypothèse pour $((\kappa - f), (s_2, t_2, v_2), (t_2, s_2, v_2))$, les flots h et h' existent. $|h + h'|$ est un flot de s_2 vers t_2 de valeur $2 \times v_2$, donc d'après le Théorème 18, le flot f_2 peut être calculé par l'Algorithme 1. L'Algorithme 4 calcule deux flots, appelle l'Algorithme 1 et calcule 4 fonctions sur A : il se termine donc en temps $(2C_{flow} + O(|A|))$. □

Proposition 11 *Les flots* f_2 *et* f_{-2} *calculés par l'Algorithme 4 sont tels que* $(f_2 + f_{-2}) \leq \kappa$.

Preuve
Nous savons que pour tout arc $a \in A$, $2f_2(a) \leq (|h + h'|(a) + 1)$ et $2f^r_{-2}(a) \leq (|h + h'|(a) + 1)$. Comme $|h + h'| \leq 2\kappa$, nous avons pour tout $a \in A$, $2f_2(a) \leq (2\kappa(a) + 1)$ et $2f^r_{-2}(a) \leq (2\kappa(a) + 1)$, d'où $f_2 \leq \kappa$ et $f_{-2} \leq \kappa$. De plus, comme $f_2 + f^r_{-2} = |h + h'|$, pour tout arc $a \in A$ soit $f_2(a)$ est nul, soit $f_{-2}(a)$ est nul. Aussi, $(f_2 + f_{-2}) \leq \kappa$. □

Proposition 12 *Si* $f \leq \kappa$ *est un flot de* s_1 *vers* t_1 *de valeur* v_1 *tel que le critère de coupe soit vérifié pour* $((\kappa - f), (s_2, t_2, v_2), (t_2, s_2, v_2))$*, les flots* f_2 *et* f_{-2} *calculés par l'Algorithme 4 sont tels que le critère de coupe est vérifié pour* $((\kappa - f_2 - f_{-2}), (s_1.t_1, v_1))$.

Preuve
Nous allons étudier une coupe induite par $C \subset V$ telle que $s_1 \in C$ et $t_1 \notin C$. Pour commencer, nous allons prouver que $(|h+h'|(C)+|h+h'|(V\backslash C)) \leq (2\kappa(C)-2v_1)$. Cela implique que $((f_2+f_{-2})(C)+(f_2+f_{-2})(V\backslash C)) \leq (2\kappa(C)-2v_1)$ (**a**). Ensuite, nous prouverons que $(f_2+f_{-2})(C) \leq (\kappa(C)-v_1)$ (**b**).

Nous décomposons $|h + h'|$ en deux fonctions : g (la partie provenant de h) et g' (la partie provenant de h') de façon que $g = \min(h, |h + h'|)$ et $g' = \min(h', |h + h'|)$. Nous appelons r la fonction symétrique $r = (h+h')-|h+h'|$. Remarquons que si $g(x, y)$ est non nul, alors $g'(y, x) = 0$. Ceci implique que $(g+g'^r+r) \leq (\kappa-f)$ (**c**). Comme h et h' sont des flots allant dans des directions opposées, nous avons $(h(C) - h(V\backslash C)) = (h'^r(V\backslash C) - h'^r(C))$. Ceci implique que $(h + h'^r)(C) = (h^r + h')(C)$, et donc $(g + g'^r + 2r)(C) \geq (g^r + g')(C)$. Par conséquent, $(g + g'^r + g^r + g')(C) \leq 2(g + g'^r + r)(C)$ (**d**). D'après (**c**) et (**d**) on a $|h + h'|(C) + |h + h'|(V\backslash C) \leq (2\kappa(C) - 2v_1)$ (**a**).

Comme $(f_2+f^r_{-2}) = |h+h'|$, (**a**) implique que $(f_2+f^r_{-2})(C)+(f_2+f^r_{-2})(V\backslash C) \leq (2\kappa(C)-2v_1)$, d'où $(f_2 + f_{-2})(C) + (f_2 + f_{-2})(V\backslash C) \leq (2\kappa(C) - 2v_1)$ (**e**). Les flots f_2 et f_{-2} vont dans des directions opposées, ce qui fait que $(f_2 + f_{-2})(C) = (f_2 + f_{-2})(V\backslash C)$ (**f**). D'après (**e**) et (**f**) nous avons $(f_2 + f_{-2})(C) \leq (\kappa(C) - v_1)$ (**b**). □

Une fois que f_2 et f_{-2} sont calculés, il est très simple de calculer f_1 et f_{-1} (Algorithme 5).

Algorithme 5 (Troisième étape)

Complexité: $C_{flow} + O(|A|)$

Entrée: Un graphe orienté symétrique $G = (V, A)$; une capacité symétrique $\kappa : A \to \mathbb{N}$; $s_1, t_1 \in V$, $v_1 \in \mathbb{N}$; $f_2, f_{-2} : A \to \mathbb{N}$.

Sortie: $f_1, f_{-1} : A \to \mathbb{N}$

Variables: -
Début

1. calculer un flot $f_1 \leq (\kappa - f_2 - f_{-2})$ de s_1 vers t_1 de valeur v_1
2. calculer $f_1 = (\kappa - f_1 - f_2 - f_{-2})$

Fin

Lemme 15 (troisième étape) *Si (f_2, f_{-2}) est un biflot de (s_2, t_2) vers (t_2, s_2) de valeur (v_2, v_2) tel que $(f_2 + f_{-2}) \leq \kappa$ et tel que le critère de coupe soit vérifié pour $((\kappa - f_2 - f_{-2}), (s_1, t_1, v_1))$, alors l'Algorithme 5 se termine en $C_{flow} + O(|A|)$ pas de calcul, et les deux fonctions calculées f_1 et f_{-1} forment un biflot de (s_1, t_1) vers (t_1, s_1) de valeur (v_1, v_1) tel que $f_1 + f_{-1} + f_2 + f_{-2}) \leq \kappa$.*

Preuve
Comme par hypothèse le critère de coupe est vérifié pour $((\kappa - f_2 - f_{-2}), (s_1, t_1, v_1))$, le flot f_1 existe. L'Algorithme 5 se termine donc en temps $C_{flow} + O(|A|)$. La fonction $\kappa - f_1 - f_2 - f{-}2 : A \to \mathbb{N}$ vérifie toute les équations d'un flot de valeur v_1 entre t_1 et s_1. Les cycles éventuels du flot f_{-1} peuvent aisément être supprimés en C_{flow} pas supplémentaires. □

Récapitulons les différentes étapes :

Théorème 19 (biflot symétrique). *Le critère de coupe symétrique est une condition nécessaire et suffisante pour l'existence d'une solution au problème du biflot symétrique. Une telle solution peut être trouvée par l'Algorithme 2 en $6C_{flow} + O(|A|)$ étapes.*

Preuve
D'après le corollaire de Menger [16], le critère de coupe symétrique est une condition nécessaire pour l'existence d'une solution au problème. Les données sont :

- un graphe orienté symétrique $G = (V, A)$;
- une capacité symétrique $\kappa : A \to \mathbb{N}$;
- deux sources et deux puits (s_1, s_2), $(t_1, t_2) \in V^2$;
- deux valeurs $(v_1, v_2) \in \mathbb{N}^2$.

Si le critère de coupe symétrique est vérifié pour $(\kappa, (s_1, t_1, v_1), (s_2, t_2, v_2))$ alors d'après le Lemme 13, l'Algorithme 3 comprend $3C_{flow} + O(|A|)$ pas de calcul ; d'après le Lemme 14, l'Algorithme 4 comprend $2C_{flow} + O(|A|)$ pas de calcul et d'après le Lemme 15, l'Algorithme 5 comprend $C_{flow} + O(|A|)$ pas de calcul ; aussi, l'Algorithme 2 se termine en temps $6C_{flow} + O(|A|)$.

De plus, d'après les Lemmes 13, 14 et 15 le multiflot à 4 commodités $(f_1, f_{-1}, f_2, f_{-2})$ calculé par l'Algorithme 2 est un biflot symétrique de (s_1, s_2) vers t_1, t_2 de valeur (v_1, v_2) tel que $(f_1 + f_{-1} + f_2 + f_{-2}) \leq \kappa$. Il s'agit donc d'une solution au problème du biflot symétrique. □

3.5 Conclusion et Perspectives

Dans ce chapitre, nous avons étudié le problème du multiflot entier dans les graphes orientés symétriques. Nous avons montré que ce problème est NP-difficile dès que le nombre de requêtes

dépasse ou est égal à trois. Lorsque les requêtes sont elles aussi symétriques, nous avons proposé un algorithme polynomial pour deux paires de requêtes. Cet algorithme nécessite seulement six calculs de flots simple, ce qui le rend performant.

Récapitulation : le tableau suivant résume les différentes classes de complexité des problèmes de multiflot en fonction du type de requêtes et du type de graphe.

	Graphe		
Problème : déterminer l'existence de...	orienté	non orienté	orienté symétrique
un chemin	$O(M + NlogN)$ [16]		
chemins disjoints pour k requêtes	NP-difficile [36]	O(N^3) [86]	O(N^3)
chemins disjoints	NP-difficile [53]	NP-difficile [53]	NP-difficile [13]
flot	C_{flot} [35]		C_{flot} [35]
multiflot à 2 commodités (biflot)	NP-difficile [23]		non résolu
multiflot à $k \geq 3$ commodités	NP-difficile		**NP-difficile**
flot symétrique	NP-difficile	C_{flot}	C_{flot}
biflot symétrique	NP-difficile	NP-difficile	$\mathbf{6C}_{flot}$ **+ O(M)**
multiflot symétrique à $k \geq 3$ commodités	NP-difficile	NP-difficile	non résolu
multiflot symétrique	NP-difficile	NP-difficile	**NP-difficile**

Deux questions sont restées ouvertes au cours de cette étude. On ne sait pas si le problème du multiflot entier dans un graphe orienté symétrique admet un algorithme polynomial pour deux requêtes. Ensuite, on ne connaît pas le comportement du problème lorsque les requêtes sont symétriques, et lorsqu'il y a trois paires de requêtes ou plus.

Deuxième partie

Graphes évolutifs

Chapitre 4

Le modèle des Graphes évolutifs

Sommaire

4.1 Introduction

Nous nous intéressons dans cette partie aux environnements de communication mobiles et sans infrastructure, tels les réseaux ad-hoc, les réseaux de mobiles et les systèmes de satellites à orbite basse LEO (Low Earth Orbiting). Ceux-ci présentent un changement de paradigme par rapport aux réseaux reposant sur une infrastructure fixe, comme les réseaux optiques ou la téléphonie cellulaire : les données sont transférées de noeud à noeud via des interactions pair-à-pair, et non via une infrastructure fixe de routeurs. Naturellement, cela engendre de nouveaux problèmes concernant le routage optimal sous des conditions variables sur ces réseaux dynamiques [87]. Dans le cas dynamique, le cas généralisé de routage utilisant les plus court chemins, où des méthodes utilisant des chemins de moindre coût sont compliquées par le mouvement arbitraire des agents mobiles,

ayant pour conséquence des variations de coût et de connexité. Tout ceci motive l'étude de modèles pour cette dynamicité, et le design d'algorithmes pour la prendre en compte [90].

Il faut noter que dans le cas des réseaux de senseurs, des réseaux de satellites LEO, et de quelques autres classes de réseaux dont les agents ont des trajectoires prédestinées, l'évolution du réseau est assez déterministe. En particulier, les réseaux de satellites LEO [19, 27, 96, 97] communiquent via des liens ISL (Inter Satellite Links) entre deux satellites qui sont à portée l'un de l'autre. Les liens ISL connectant des satellites dans le même plan orbital sont fixes car ces satellites se déplacent à une vitesse relative nulle. Mais les liens ISL entre satellites de différents plans orbitaux varient alors que les satellites prennent et perdent contact. Ceci entraîne des variations sur la topologie du réseau. Comme les trajectoires des satellites sont connues à l'avance, il est possible d'exploiter ce déterminisme pour optimiser les stratégies de routage [27].

Notre travail a consisté à étudier les problèmes de communications sur ces réseaux déterministes. Pour cela, nous avons développé le modèle des graphes évolutifs, qui sera décrit en détail dans ce chapitre. En plus des réseaux déterministes, notre modèle peut également s'appliquer, dans le cadre de l'analyse compétitive [8], aux réseaux dont on ne connaît pas l'évolution future. Dans ces réseaux, la comparaison entre les protocoles de communication peut se faire par rapport à des routages optimaux théoriques, calculés en reconstituant l'historique du réseau.

Nous allons présenter brièvement les modèles de réseaux dynamiques. Les modèles se différentient principalement suivant deux concepts : la formalisation du temps et la capacité (ou non) de stockage dans les noeuds.

Réseaux dynamiques

Le domaine plus général des réseaux dynamiques est étudié dans le cadre de l'optimisation des transports. En effet, les réseaux de transport disposent d'une infrastructure fixe, mais leurs caractéristiques évoluent en fonction d'évènements temporels comme les feux de signalisation, la densité du trafic, les horaires des trains/avions/bateaux.

Les premières recherches sur les réseaux dynamiques ont été menées par Ford et Fulkerson, et présentées en 1958 [34, 35]. Les auteurs étudiaient les problèmes de flot maximum lorsque le temps est discrétisé, et ont développé une technique qui est toujours largement utilisée. Depuis lors, de nombreux autres problèmes ont été analysés impliquant par exemple les flots les plus rapides, les flots de coût minimum, les modèles où le temps est continu, ou encore les modèles dont les paramètres changent au cours du temps. Le survey de Lovetskii et Melamed [65] présente une bonne description des problèmes et techniques lorsque le temps est discret ou continu ; celui de Aronson [3] recouvre les problèmes de flot maximum et de livraisons ; celui de Powell, Jaillet et Odoni [85] s'intéresse aux problèmes de modélisation, notamment ceux liés la finitude du temps.

Le terme *dynamique* peut être utilisé lorsque les paramètres propres du réseau évoluent avec le temps. C'est la terminologie que nous employons (voir par exemple les arguments employés dans les articles [29] et [85]). Cependant on trouve également le terme *dynamique* lorsque la solution d'un problème dépend du temps même si le réseau est parfaitement statique.

La formalisation du temps

Une première disparité entre les modèles concerne la durée de vie (*time horizon*) du réseau. Certains modèles ont une durée de vie finie, et d'autres, infinie. Lorsque la durée de vie est finie, tout doit se passer avant une date T. Cette échéance implique que toute les communications considérées doivent arriver avant elle, ce qui veut dire qu'aucun message ne peut emprunter un arc s'il ne pourra pas en sortir avant la date T. La plupart des modèles théoriques utilisent une durée finie. Cependant, certains problèmes pratiques nécessitent une durée potentiellement infinie. Par exemple, les problèmes de trafic routier peuvent être modélisés avec des caractéristiques périodiques, avec des activités périodiques mais qui n'ont pas de fin. Lorsqu'il s'agit de calculer le coût d'un flot dans le cas infini, le coût pris en compte est le coût sur une période, ou bien le coût à chaque instant, et non le coût global. D'autres questions relatives au temps infini sont traitées dans [85]. Le concept de débit (*throughput*) a été introduit pour les réseaux dynamiques de durée infinie par Orlin [81, 82], qui a apporté des résultats théoriques et pratiques pour les problèmes de débit maximum et de coût minimum pour un débit fixé.

Une deuxième disparité concerne l'écoulement du temps. Celui-ci peut se faire de manière discrète ou continue. Lorsque le temps est discret, on peut se ramener sans perte de généralité à considérer le réseau aux dates $t = 0, 1, 2 \dots T$, et on peut supposer que toutes les grandeurs relatives au temps sont des entiers. Nous verrons qu'il s'agit du cas le plus facile ; les problèmes où le temps est continu présentent des difficultés supplémentaires. Dans les cas pratiques le temps peut être discrétisé, convertissant alors les problèmes à temps continu en des problèmes à temps discret. Bien sûr, le choix du pas de discrétisation à une incidence directe sur la complexité des problèmes, qui grandit lorsque le pas diminue. Pour pouvoir utiliser une notation s'appliquant à tous les cas, nous notons l'espace des dates $\mathbb{T}$ (voir Notation 2 page 83). De cette manière, $\mathbb{T}$ contient $\{0, 1, 2, \dots, T\}$ dans les modèles à temps discret, et $\mathbb{T}$ contient $[0, T]$ dans les modèles à temps continu.

Enfin, dans les modèles les plus généraux de réseaux dynamiques les paramètres des arcs sont aussi des fonctions du temps. Nous avons les fonctions $u_a : \mathbb{T} \to \mathbb{R}_+$, $c_a : \mathbb{T} \to \mathbb{R}$, $\zeta_a : \mathbb{T} \to \mathbb{T}$ qui dénotent la capacité, le coût, et le temps de traversée d'une arête.

La capacité de stockage dans les noeuds

On peut légitimement se demander si un message qui vient d'arriver sur un sommet a le droit de rester sur ce sommet en attendant le moment opportun pour emprunter un autre arc. Ceci amène directement à considérer une capacité de stockage (*storage* ou *holdover*) dans les noeuds. On peut donc appeler $s_x : \mathbb{T} \to \mathbb{R}_+$ la capacité de stockage du noeud x au cours du temps. Dans certains cas, cette capacité est supposée non nulle mais n'est cependant pas utilisée, et on peut la fixer à zéro pour tout les sommets. En fait, l'hypothèse par défaut dans la littérature est que les capacités de stockage sont illimitées, même si cette hypothèse n'est pas utilisée pour la résolution de beaucoup de problèmes. Les modèles qui supposent une capacité de stockage limitée le précisent explicitement.

Remarquons que lorsque les capacités de stockage sont limitées, une communication peut avoir le comportement suivant : en raison d'un goulet d'étranglement, le trajet parcouru peut contenir des cycles (pensez à un avion en attente d'atterrissage). Ceci s'explique par le fait qu'un message doit attendre que le goulet se libère, mais ne peut être stocké, et doit donc circuler sur une boucle. Si les capacités de stockage sont illimitées, il est toujours préférable pour les messages d'attendre sur un noeud, et les communications ne forment pas de cycle.

Du point de vue mathématique sur les flots, la possibilité de stockage implique que les équations de conservation du flot ne sont pas respectées à tout instant, car le flot entrant sur un sommet peut être différent du flot sortant. Deux techniques permettent de contenir cette difficulté. La première technique, qui se conçoit bien sur les modèles à temps discret, consiste à ajouter des boucles artificielles sur les noeuds, avec un temps de traversée unitaire. Le stockage est remplacé par un flot circulant sur ces boucles unitaires. La deuxième technique consiste à modifier les équations de flot : à tout instant t, la somme (ou intégrale) de flot qui est entrée dans un sommet avant t doit être supérieure à la somme (ou intégrale) de flot qui en est sortie. De plus, la différence entre les deux, qui représente les unités stockées sur le noeud, doit être inférieure à la capacité de stockage du noeud. Aussi nous n'avons plus une équation de conservation de flot, mais plutôt des inéquations. Usuellement, on suppose aussi que les noeuds sont 'vides' à la fin du temps imparti, ce qui signifie une égalité de la somme (intégrale) du flot entré avec celle du flot sorti au temps T.

Graphes Évolutifs

Le modèle des Graphes évolutifs est un modèle de réseau dynamique à durée finie, où tous les paramètres du réseau évoluent de manière discrète. Il existe un ensemble de dates, appelées évènements, qui correspondent à un changement dans l'état du réseau. Entre deux de ces dates, l'état du réseau est constant. Un tel réseau est illustré à la figure 4.1.

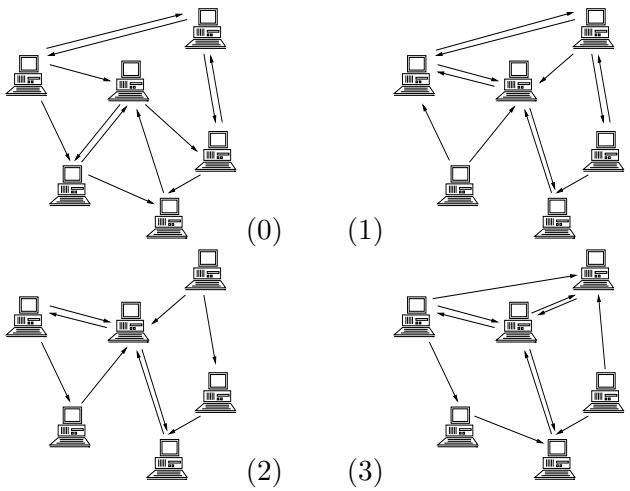


FIG. 4.1 – Un exemple de réseau dynamique évoluant de façon discrète. Les indices correspondent aux quatre états successifs du réseau.

L'originalité de notre approche consiste à prendre en compte la combinatoire [1] d'un modèle dynamique le plus général possible. Le caractère discret du modèle (par opposition à un modèle où les paramètres sont des fonctions arbitraires du temps) permet justement de contrôler cette combinatoire. Il s'agit aussi d'un choix proche de ce que l'on obtient en pratique lorsqu'on reconstitue l'historique d'un réseau à partir de ses traces. Les reconstitutions de l'historique d'un réseaux sont notamment utilisées pour analyser une expérience lorsque l'on teste un nouveau protocole [24].

[1] le coût en espace d'une représentation du réseau au cours du temps.

4.2 Le modèle des Graphes Évolutifs

Dans cette section, nous allons définir précisément quelles sont les notations et définitions fondamentales du modèle des graphes évolutifs.

4.2.1 Le graphe sous jacent

Nous utilisons l'ensemble de sommets V qui représente les noeuds du réseau, et l'ensemble d'arêtes E qui représente toutes les connexions passées présentes, futures et/ou possibles. Le graphe $G = (V, E)$ est appelé graphe sous-jacent.

Notation 1 (Graphe sous-jacent $G = (V, E)$) *Le graphe $G = (V, E)$ représente tous les noeuds et connexions possibles, passées, présentes et futures, du réseau.*

Il est vrai que l'on peut toujours considérer, sans perte de généralité, que le graphe sous-jacent est un graphe complet. Cependant, suivant les applications, et suivant la difficulté combinatoire de certains problèmes, il peut être intéressant d'émettre comme hypothèse que le graphe sous-jacent possède certaines propriétés et d'exploiter ces propriétés.

Le graphe sous-jacent peut être orienté comme il peut être non orienté. Notons que même dans le cas non-orienté, la dimension temporelle des graphes évolutifs que nous verrons par la suite confère une certaine orientation au réseau. Nous verrons par exemple que le fait qu'il existe un trajet depuis un sommet x vers un sommet y n'implique pas que la réciproque soit vraie, même lorsque tous les liens sont non orientés.

Sauf mention contraire, les algorithmes que nous présenterons seront toujours définis sur des graphes évolutifs dont les liens sont orientés mais ils pourront également s'appliquer au cas non orienté. A l'inverse, toutes les preuves établissant une difficulté combinatoire seront définies sur des graphes évolutifs dont les liens sont non orientés, et s'appliqueront également au cas non orienté.

4.2.2 Le domaine temporel

En étudiant les réseaux dynamiques, on peut utiliser les entiers, les réels, un sous-ensemble de réels, etc..., pour représenter le temps. Nous noterons $\mathbb{T}$ l'ensemble utilisé pour représenter toutes les dates, durées, délais, qui sont dans le domaine temporel. Nous utiliserons en général les dates $-\infty$ et $+\infty$ qui bien qu'étrangères à la modélisation proprement dite d'un réseau dynamique auront une signification particulière. Plus précisément, s'il est impossible d'accomplir une action, la première date possible pour accomplir cette action sera $+\infty$. A l'inverse, le dernier moment pour accomplir cette action sera $-\infty$.

Notation 2 (Domaine temporel $\mathbb{T}$) *Le domaine temporel $\mathbb{T}$ est l'ensemble de toutes les dates, durées, délais, etc... possibles. Il pourra être identifié avec $\overline{\mathbb{Z}}$, $\overline{\mathbb{R}_+}$, ..., suivant les applications. Il contiendra toujours les grandeurs $+\infty$ et $-\infty$.*

4.2.3 Suites de dates

Dans les réseaux dynamiques que nous considérons, les liens et les noeuds sont susceptibles d'apparaître ou de disparaître. Chaque apparition ou disparition constitue un évènement daté dans le temps. Nous introduisons donc la suite des dates remarquables notée $\mathcal{S}_{\mathbb{T}} = t_1, t_2, \ldots, t_i, t_{i+1}, \ldots$ et qui se situe dans le domaine temporel $\mathbb{T}$. Par exemple, la date à laquelle un des liens du réseau devient impraticable fera partie de cette suite. Nous notons G_i le graphe représentant l'état du réseau (noeuds et liens fonctionnels) durant l'intervalle $[t_i, t_{i+1}]$. G_i est un sous-graphe du graphe sous-jacent.

Notation 3 (Séquence temporelle $\mathcal{S}_{\mathbb{T}}$) *Nous appelons $\mathcal{S}_{\mathbb{T}} = t_1, t_2, \ldots$ la suite ordonnée de toutes les dates remarquables au cours de la vie du réseau.*

Notation 4 (Séquence de sous-graphes $\mathcal{S}_G$) *Nous appelons $G_i = (V_i, E_i)$ le graphe qui représente les noeuds fonctionnels (représentés par V_i) et les liens fonctionnels (représentés par E_i) durant l'intervalle de temps $[t_i, t_{i+1}]$. Nous notons $\mathcal{S}_G = G_1, G_2, \ldots$ la suite ordonnée de tous ces sous-graphes de G.*

4.2.4 Le délai

Dans notre modèle de graphe évolutif, chaque arête nécessite un temps de traversée, appelé délai, qui sera toujours positif. Notons qu'une hypothèse sera toujours vérifiée sur le délai des liens : chaque liens suit le modèle FIFO (First In First Out) ce qui signifie que une donnée ne peut pas en 'doubler' une autre sur ce lien. Cette condition se traduit mathématiquement de la façon suivante : pour toute arête et tout couple de dates (t_1, t_2) dans $\mathbb{T}$, nous avons la propriété $t_1 \leq t_2 \;\Rightarrow\; t_1 + \zeta(e, t_1) \leq t_2 + \zeta(e, t_2)$.

La difficulté combinatoire du codage et du calcul d'un délai variable n'est pas l'objet de nos études, et nous supposerons toujours que son calcul coûte $O(1)$ opérations. Spécifiquement, notre modèle combinatoire prend son sens pour un délai constant, ou à tout le moins constant à chaque apparition ou disparition d'une arête. Cependant, la plupart des algorithmes que nous présentons s'étendent aussi au cas où le délai est variable.

Notation 5 (Délai ζ) *Nous appelons $\zeta : E \to \mathbb{T}$ la fonction qui indique le temps de traversée de chaque arête de G.*

4.2.5 Le système

Nous appelons la totalité du système $\mathcal{G} = (G, \mathcal{S}_G, \mathcal{S}_{\mathbb{T}}, \zeta)$ un graphe évolutif.

Définition 19 (Graphe évolutif) *Soit un graphe $G = (V, E)$, et $\mathcal{S}_G$ une suite ordonnée de sous-graphes de G. Soit $\mathcal{S}_{\mathbb{T}}$ une suite croissante de $\mathbb{T}$ contenant un élément de plus que $\mathcal{S}_G$. Soit ζ une fonction de E vers $\mathbb{T}$. Le système $\mathcal{G} = (G, \mathcal{S}_G, \mathcal{S}_{\mathbb{T}}, \zeta)$ est appelé graphe évolutif.*

Par abus de langage, nous appellerons également graphe évolutif un système où le délai de toutes les arêtes est nul, et où $t_1 = 1, t_2 = 2 \ldots$.

Définition 20 (Graphe évolutif (atemporel)) *Soit un graphe $G = (V, E)$, et $\mathcal{S}_G$ une suite ordonnée de sous-graphes de G. Le système $(G, \mathcal{S}_G)$ est appelé graphe évolutif (atemporel).*

4.3 Trajets dans les graphes évolutifs

Nous allons maintenant présenter une application principale sur les graphes évolutifs : les trajets entre les sommets. Contrairement à d'autres modèles de réseaux dynamiques où des chemins sont établis dans un sous graphe G_i et sont mis à jour par exemple lors des pannes de liens (voir par exemple [24]), nous considérons des trajets *qui prennent du temps*, en raison non seulement du délai des arêtes, mais aussi de la possibilité d'attendre l'apparition d'un lien. A présent, voyons ces définitions.

4.3.1 Dates d'émission possibles

Soit e une arête de E, et σ une date de $\mathbb{T}$. Si le lien représenté par e peut être utilisé sans problème à la date σ, alors nous disons que σ est une date d'émission possible pour l'arête. Dans notre modèle, cela se traduit par le fait que e doit être présente dans tous les sous-graphes G_i tels que $[t_i, t_i + 1[\bigcap[\sigma, \sigma + \zeta(e)] \neq \emptyset$.

Notation 6 (Date d'émission possible) *Soit e une arête du graphe sous-jacent. Une date σ d'émission possible pour e est un élément de $\mathbb{T}$ tel que le lien représenté par l'arête e est fonctionnel durant $[\sigma, \sigma + \zeta(e)]$.*

4.3.2 Les trajets

Un trajet $\mathcal{J}$ dans un graphe évolutif $\mathcal{G}$ sera défini par un chemin $P = e_1, e_2, \ldots$ du graphe sous-jacent G, et par une suite de dates démission $\mathcal{S}_\sigma = \sigma_1, \sigma_2, \ldots$ consistante avec notre modèle. Le trajet est donc un couple $\mathcal{J} = (P, \mathcal{S}_\sigma)$. La suite des dates d'émission est consistante si les messages ne partent pas avant d'être arrivés, et si les arêtes sont bien là lorsqu'elles sont utilisées.

Définition 21 (Trajet) *Soit $\mathcal{G} = (G, \mathcal{S}_G, \mathcal{S}_\mathbb{T}, \zeta)$ un graphe évolutif. Soit $P = e_1, e_2, \ldots, e_k$ un chemin dans G et $\mathcal{S}_\sigma = \sigma_1, \sigma_2, \ldots, \sigma_k$ une suite de $\mathbb{T}$ telle que pour tout $1 \leq i < k$, $\sigma_i + \zeta(e_i) \leq \sigma_{i+1}$, et telle que pour tout $1 \leq i \leq k$, σ_i est une date d'émission possible pour e_i. Le système $\mathcal{J} = (P, \mathcal{S}_\sigma)$ est appelé trajet.*

Dans le cas des graphes évolutifs atemporels, nous avons une définition similaire, où $\mathcal{S}_\sigma = \sigma_1, \sigma_2, \ldots, \sigma_k$ est une suite croissante d'entiers.

Définition 22 (Trajet (discret)) *Soit $(G, \mathcal{S}_G)$ un graphe évolutif (atemporel). Soit $P = e_1, e_2, \ldots, e_k$ un chemin dans G et $\mathcal{S}_\sigma = \sigma_1, \sigma_2, \ldots, \sigma_k$ une suite de $\mathbb{N}$ telle que pour tout $1 \leq i < k$, $\sigma_i \leq \sigma_{i+1}$ et*

pour tout $1 \leq i \leq k$*, l'arête* e_i *est présente dans le sous-graphe* G_i*. Le système* $(P, \mathcal{S}_\sigma)$ *est appelée trajet (discret).*

Nous dirons que $\mathcal{J} = (P, \mathcal{S}_\sigma)$ est un trajet du sommet x vers le sommet y si P est un chemin du sommet x vers le sommet y.

4.4 Complexité combinatoire des graphes évolutifs

Nous allons présenter ici une formalisation des graphes évolutifs qui permet d'expliciter leur taille et de calculer la complexité des calculs élémentaires sur les graphes évolutifs : calcul des dates d'émission et de réception des messages.

4.4.1 Intervalles de présence

Si l'état d'un sommet (fonctionnel ou non fonctionnel) ne change pas entre le sous graphe G_i et le sous graphe G_{i+1}, il est inutile de répéter l'information concernant ce sommet. Par conséquent, nous choisissons de retenir uniquement les intervalles $I = [t_i, t_j]$ durant lesquels le sommet x est fonctionnel. Les différents états du sommet x sont décrits par une suite ordonnée de ces intervalles $P_V(x) = I_1, I_2, \ldots$. De même les états d'une arête de E sont décrits par la suite $P_E(e)$.

Définition 23 (Intervalles de présence d'un sommet $P_V(x)$) *Soit* x *un sommet du graphe sous-jacent* G*. Nous notons* $P_V(x)$ *la suite* $[t_{i_1}, t_{i_2}], [t_{i_3}, t_{i_4}], \ldots$ *telle que :*

- *pour tout* j*,* $i_j < i_{j+1}$
- *pour tout* j*,* $t_{i_j} \in \mathcal{S}_\mathbb{T}$
- *pour tout* i*,* x *est un sommet du sous graphe* G_i *si et seulement si il existe un temps non nul* $\epsilon > 0$ *tel que* $[t_i, t_i + \epsilon]$ *est dans un des intervalles de* $P_V(x)$

Définition 24 (Intervalles de présence d'une arête $P_E(e)$) *Soit* e *une arête du graphe sous-jacent* G*. Nous notons* $P_V(e)$ *la suite* $[t_{i_1}, t_{i_2}], [t_{i_3}, t_{i_4}], \ldots$ *telle que :*

- *pour tout* j*,* $i_j < i_{j+1}$
- *pour tout* j*,* $t_{i_j} \in \mathcal{S}_\mathbb{T}$
- *pour tout* i*,* e *est un sommet du sous graphe* G_i *si et seulement si il existe un temps non nul* $\epsilon > 0$ *tel que* $[t_i, t_i + \epsilon]$ *est dans un des intervalles de* $P_E(e)$

4.4.2 La combinatoire des graphes évolutifs

Soit $\mathcal{G} = (G, \mathcal{S}_G, \mathcal{S}_\mathbb{T}, \zeta)$ un graphe évolutif. Nous présentons ici les différentes grandeurs utilisées dans les analyses de complexité combinatoire sur $\mathcal{G}$.

Nous avons besoin d'expliciter la taille du graphe sous-jacent : nous notons N son nombre de sommets et M son nombre d'arêtes.

Notation 7 (Taille du graphe sous jacent) *Nous notons* $N = |V|$ *et* $M = |E|$.

Pour tout sommet $x \in V$, nous appelons dynamicité de x, $\delta_V(x)$ le nombre d'intervalles de présence dans la suite $P_V(x)$. De même, pour toute arête $e \in E$, nous appelons dynamicité de e, $\delta_E(e)$ le nombre d'intervalles de présence dans la suite $P_E(e)$. Le maximum de toutes les dynamicités des sommets de V est noté δ_V, et le maximum de toutes les dynamicités des arêtes de E est noté δ_E. Enfin, la plus grande de ces deux grandeurs est appelée dynamicité du graphe évolutif $\mathcal{G}$ et est notée δ.

Notation 8 (Dynamicité δ) *Pour tout $x \in V$, soit $\delta_V(x)$ le nombre d'intervalles dans $P_V(x)$. Nous notons $\delta_V = \max_{x \in V} \delta_V(x)$. De même, Pour tout $e \in E$, soit $\delta_E(e)$ le nombre d'intervalles dans $P_E(e)$. Nous notons $\delta_E = \max_{e \in E} \delta_E(e)$. Enfin, nous appelons dynamicité du graphe évolutif $\mathcal{G}$ le maximum $\delta = \max\{\delta_V, \delta_E\}$.*

Pour décrire la taille globale de la donné d'un graphe évolutif, nous additionnons les dynamicités des sommets et des arêtes.

Notation 9 (Taille d'un graphe évolutif) *Soit $\mathcal{G} = (G, \mathcal{S}_G, \mathcal{S}_\mathbb{T}, \zeta)$ un graphe évolutif. Nous appelons $\mathcal{N} = \Sigma_{v \in V} \delta_V(v)$ et $\mathcal{M} = \Sigma_{e \in E} \delta_E(e)$.*

4.4.3 Codage compact

La donnée du graphe évolutif $\mathcal{G}$ peut être fournie sous forme de listes d'adjacence chaînées, avec une liste ordonnée d'intervalles de présence pour chaque voisin. Le délai de l'arête peut également être présent à cet endroit, ainsi qu'éventuellement d'autres données relatives à l'arête. La tête de chaque liste est un sommet avec sa propre liste de présence (Figure 4.2).

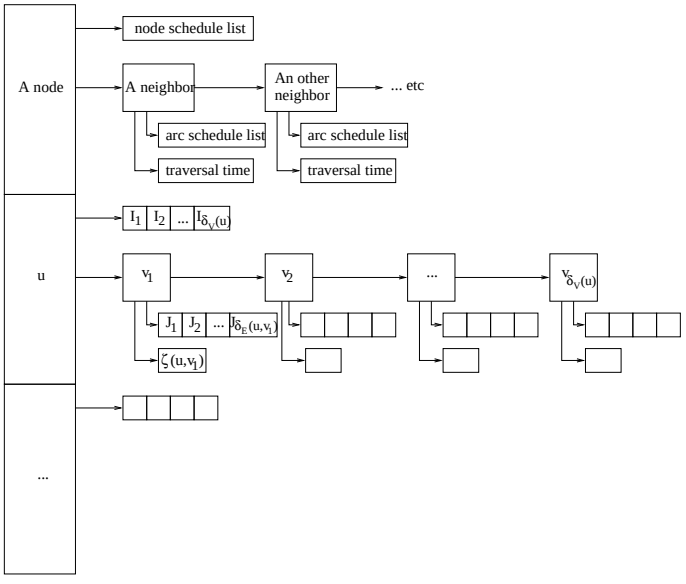


FIG. 4.2 – Structure de donnée pour une représentation compacte d'un graphe évolutif.

L'espace utilisé par la structure de données est proportionnelle à la taille des listes d'adjacence, plus le nombre d'intervalles de présence considérés. La taille totale des listes est $O(\mathcal{M} + \mathcal{N})$.

4.4.4 Mémoire et dynamique des noeuds

Une question intéressante sur les réseaux que nous modélisons est apparue lorsque nous avons étudié les problèmes de routage : que se passe-t-il lorsqu'un noeud ne fonctionne plus ? Une première conséquence est que tous les liens depuis et vers ce noeud s'arrêtent simultanément de fonctionner. Une deuxième conséquence est que les données stockées sur ce noeud peuvent éventuellement être perdues, selon le type de réseau modélisé. Comme la première conséquence peut être modélisée par l'arrêt des liens, on peut dire que lorsqu'un noeud cesse de fonctionner, cela signifie que les données du noeud sont perdues.

Dans la plupart des applications où les noeuds ne sont pas supposés perdre de l'information, les sommets seront donc permanents : $\delta_V = 1$ et $\mathcal{N} = N$. Dans les autres cas, nous pouvons transformer le graphe évolutif $\mathcal{G}$ en $\mathcal{G}'$ de telle sorte que les noeuds ayant k intervalles de présence soient dupliqués $k-1$ fois. Le graphe évolutif $\mathcal{G}'$ aura essentiellement les mêmes propriétés que le graphe $\mathcal{G}$, à l'exception près que tous ses sommets ont une dynamicité de 1. Plus précisément, nous aurons $\mathcal{N}' = \mathcal{N}$, $\mathcal{M}' = \mathcal{M}$, $N' = \mathcal{N}$, $M' \leq M \times \delta_V$, ${\delta_V}' = 1$, ${\delta_E}' \leq \delta_E$, et $\delta' \leq \delta$.

Aussi, nous considérerons uniquement des graphes évolutifs tels que $\delta_V = 1$. La dynamicité de ces graphes évolutifs est confondue avec celle de leurs arêtes ($\delta = \delta_E$). Les graphes évolutifs ayant une dynamicité de sommets δ_V strictement supérieure à 1 pourraient nécessiter d'être étudiés dans certains cas très particuliers. Par exemple, lors de l'étude des composantes connexes, il n'est pas identique d'avoir deux sommets distincts ou bien deux occurrences distinctes d'un même sommet. Notons par ailleurs que la pertinence de l'étude des composantes connexes en cas de perte de données sur les noeuds reste peu élevée. Au cours de cette thèse, nous n'avons pas rencontré de contexte où la dynamique des noeuds est réelle ($\delta_V > 1$), et où la transformation décrite au paragraphe précédent n'est pas suffisante.

4.4.5 Calcul des émissions et des réceptions de message

Une série de questions récurrentes se pose dans la quasi totalité des problèmes concernant les graphes évolutifs. Étant donnée une arête e et une date t, quand peut-on émettre sur l'arête e après la date t ? Quelle est la dernière date d'émission possible ? Quelle sera la date de réception ? Toutes ces questions ont bien sûr une réponse relativement simple. Il suffit de trouver l'intervalle de présence approprié, et d'utiliser le délai de l'arête e. Le délai se trouve immédiatement, et nous supposons que son addition ou soustraction se fait de manière atomique et coûte $O(1)$. La recherche de l'intervalle de présence approprié peut se faire rapidement si $P_V(x)$ est trié dans l'ordre croissant, et coûte $O(\log \delta)$. Nous utiliserons désormais deux fonctions : celle qui donne la première date de réception possible pour un départ après la date t, et celle qui donne la dernière date d'émission possible pour une arrivée avant la date t.

Définition 25 (Première date de réception possible $Q(e,t)$) *Soit e une arête de E, et t une date de $\mathbb{T}$. Soit DEP l'ensemble des dates d'émission possibles pour l'arête e. La première date de réception possible $Q(e,t)$ pour l'arête e pour un départ après la date t est définie par $Q(e,t) = (\min\ DEP \bigcap [t, +\infty]) + \zeta(e)$. Le calcul de $Q(e,t)$ se fait en $O(\log \delta)$. $Q(e,-\infty)$ est la première date de réception possible pour l'arête e. $Q(e,t) = +\infty$ signifie qu'il n'est pas possible d'émettre sur l'arête e après la date t.*

Définition 26 (Dernière date d'émission possible $Q^{-1}(e,t)$) *Soit e une arête de E, et t une date de $\mathbb{T}$. Soit DEP l'ensemble des dates d'émission possibles pour l'arête e. La dernière date d'émission possible $Q^{-1}(e,t)$ pour l'arête e pour une réception avant la date t est définie par $Q^{-1}(e,t) = (\min\ DEP \bigcap [t-\zeta(e), +\infty])$. Le calcul de $Q^{-1}(e,t)$ se fait en $O(\log \delta)$. $Q^{-1}(e,+\infty)$ est la dernière date d'émission possible pour l'arête e. $Q^{-1}(e,t) = -\infty$ signifie qu'il n'est pas possible d'émettre sur l'arête e avant la date $t-\zeta(e)$.*

Remarquons que la fonction Q^{-1} n'est pas l'inverse mathématique de la fonction Q. Cependant, nous avons pour toutes les dates d'émission possibles t pour l'arête e la propriété suivante : $Q^{-1}(e, Q(e,t)) = t$ et $Q(e, Q^{-1}(e, t+\zeta e)) = t + \zeta(e)$.

Chapitre 5

Trajets optimaux dans les graphes évolutifs

Sommaire

Dans ce chapitre, nous étudions les problèmes de routage dans les Graphes Évolutifs. Un problème de routage consiste à trouver un trajet d'un point de départ donné vers une destination donnée. En général, le trajet recherché doit minimiser certaines fonctions objectives (coût, délai ...). L'étude des problèmes de routage dans un cadre dynamique a commencé il y a environ quarante ans pour les réseaux de transports (voir par exemple [17, 34, 35, 42, 44, 83]). On peut également trouver des travaux récents sur les réseaux dynamiques, où les temps de traversée des arcs dépendent de la charge [32, 57, 59]. Lorsque les temps de traversée des arcs sont discrets, la méthode des graphes espace-temps de Ford et Fulkerson [34] peut être utilisée de manière efficace dans certains cas (voir [32, 57, 59]). Cependant, cette méthode n'est pas applicable dans le cas où les temps de traversée sont des nombres réels, ou lorsque les temps de traversée sont petits par rapport à la durée de vie du réseau. Enfin le cas particulier des problèmes de chemins au plus tôt ont été étudiés (et résolus) sous de nombreux modèles de réseaux dynamiques (voir notamment [17, 42]).

Ce chapitre est organisé de la manière suivante : La première section est consacrée aux définitions de distance dans les graphes évolutifs. Ensuite nous présenterons et analyserons le calcul d'un trajet au plus tôt (première date d'arrivée possible), puis le calcul d'un trajet le plus petit (plus petit nombre d'arcs), et pour terminer le calcul d'un trajet le plus rapide. Enfin, nous montrerons que lorsqu'il y a une fonction de coût arbitraire sur les arcs, le calcul d'un trajet de coût minimum est NP-difficile.

5.1 Distances sur les trajets, distances dans le graphe évolutif

Nous avons trois mesures de distance sur les trajets qui proviennent directement du modèle : le nombre d'arcs d'un chemin, la date d'arrivée d'un trajet, et la durée du trajet. La première mesure est un entier entre 1 et N, alors que les deux dernières sont dans le domaine temporel $\mathbb{T}$.

Soit $\mathcal{J} = (P, \mathcal{S}_\sigma)$ un trajet tel que $P = e_1, e_2, \ldots, e_k$ et $\mathcal{S}_\sigma = \sigma_1, \sigma_2, \ldots, \sigma_k$. Avec ces notations,
- Le nombre d'arcs du trajet $\mathcal{J}$ est noté $|\mathcal{J}| = k$.
- La date d'arrivée du trajet $\mathcal{J}$ est notée $a(\mathcal{J}) = Q(e_k, \sigma_k)$.
- La durée du trajet $\mathcal{J}$ est notée $t(\mathcal{J}) = a(\mathcal{J}) - \sigma_1$.

De même, il y a au moins trois différentes manières de définir la notion de "distance" dans un graphe évolutif :
- La **distance** proprement dite entre deux sommets x et y du graphe évolutif sera définie comme $d(x, y) = \min\{|\mathcal{J}|\}$ sur l'ensemble des trajets $\mathcal{J}$ entre x et y. De plus, on peut définir l'*excentricité* du sommet x comme la plus grande distance qui le sépare d'un sommet y du graphe : $\max_{y \in V}\{d(x, y)\}$. Un trajet qui minimise cette distance sera appelé trajet le plus petit entre x et y.
- La **date d'arrivée au plus tôt** du sommet x vers le sommet y, notée $Ead(x, y)$, est le minimum des dates d'arrivée sur les trajets allant de x vers y. S'il n'y a pas de trajet de x vers y, on a $Ead(x, y) = +\infty$. Un trajet qui minimise la date d'arrivée sera appelé trajet au plus tôt entre x et y.
- Le **délai** entre deux sommets x et y, noté $t(x, y)$ est le minimum des durées des trajets entre x et y. Un trajet qui minimise la durée entre x et y est appelé trajet le plus rapide entre x et y.

Pour résumer, $d(x, y)$ est le plus petit nombre de sauts requis pour aller de x à y ; $t(x, y)$ est le temps minimum requis pour aller de x vers y, et $Ead(x, y)$ est la date la plus proche à laquelle on peut atteindre y en partant de x. De plus, l'excentricité de x est le maximum de la distance de x à n'importe quel sommet du graphe évolutif $\mathcal{G}$.

5.2 Trajets au plus tôt

Nous allons montrer dans cette section comment calculer les trajets au plus tôt d'un sommet source s vers tous les autres sommets du graphe évolutif $\mathcal{G}$. Ce problème n'est pas nouveau, et a été largement étudié dans la littérature, notamment sous le nom de "plus court chemin dans les réseaux dépendant du temps" (voir notamment [17, 35, 44, 91]). Ce problème a été étudié par Bui-Xuan [9] dans le cadre des graphes évolutifs, ce qui a permis d'analyser précisément la complexité des algorithmes, en utilisant notamment la notion de dynamicité.

Nous rappelons que pour calculer un plus court chemin dans un graphe $G = (V, E)$, l'algorithme classique de Dijkstra [16] construit au fur et à mesure de son exécution un ensemble S de sommets pour lesquels le plus court chemin depuis la source a été calculé. La distance à s des autres sommets est estimée, et correspond à un chemin si elle est finie. On choisit le sommet x parmi $V - S$ avec la plus petite distance estimée pour l'ajouter à S avec le chemin correspondant. Enfin, les arcs émanant de x sont utilisées pour mettre à jour les distances et les chemins de s vers $V - S - \{x\}$. L'algorithme de Dijkstra maintient une file de priorité (*min heap priority queue*) sur les sommets

de $V - S$ pour récupérer le sommet de distance minimale de façon efficace.

L'algorithme de Dijkstra est basé sur une propriété essentielle sur les plus court chemins : les chemins préfixes des plus court chemins sont eux-mêmes des plus court chemins. Malheureusement, cette propriété n'est pas vraie pour tous les trajets au plus tôt dans les graphes évolutifs (voir par exemple la Figure 5.1. Par contre nous allons prouver que s'il existe un trajet entre deux sommets, il existe un chemin au plus tôt parmi ces deux sommets tel que tous les trajets préfixes sont eux-mêmes des trajets au plus tôt (Proposition 13).

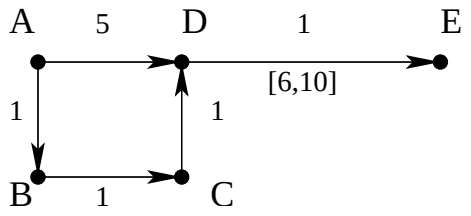


FIG. 5.1 – Exemple de graphe évolutif où les délais sont 1 ou 5. Les arcs sont tous présents durant l'intervalle $[0, 10]$ sauf l'arc DE présent durant l'intervalle $[6, 10]$. Le trajet $(ADE, 0, 6, 7)$ est un trajet au plus tôt de A vers E. Le trajet $(AD, 0, 5)$ n'est pas un trajet au plus tôt de A vers D.

Proposition 13 (Trajets *gloutons*) *Soient s et x deux sommets d'un graphe évolutif $\mathcal{G}$. S'il existe un trajet de s vers x, alors il existe un trajet au plus tôt de s vers x tel que tout ses trajets préfixes sont eux-mêmes des trajets au plus tôt. Nous appelons ce trajet un trajet* glouton.

Preuve
Soit $\mathcal{J} = (P = (e_1, \dots e_k), \mathcal{S}_\sigma = (\sigma_1, \dots \sigma_k))$ un trajet de s vers x. Si le nombre d'arcs de ce trajet est supérieur à N, alors il existe deux entiers i et j tels que e_i et e_j ont pour origine un même sommet y. Dans ce cas, $\mathcal{J}' = ((e_1, \dots e_{i-1}, e_j, \dots e_k), (\sigma_1, \dots \sigma_{i-1}, \sigma_j, \dots \sigma_k))$ est également un trajet de S vers x arrivant à la date $a(\mathcal{J})$. Aussi, on peut se restreindre à l'étude des trajets ayant moins de N arcs.

Considérons $\mathcal{J} = (P, \mathcal{S}_\sigma)$ et $\mathcal{J}' = (P', \mathcal{S}'_{\sigma'})$ deux trajets de s vers x. Nous classons $\mathcal{J}$ et $\mathcal{J}'$ par ordre lexicographique inversé sur les dates d'arrivée. Autrement dit, si on note $P = e_1, \dots e_k$, $P' = e'_1, \dots e'_{k'}$, $\mathcal{S}_\sigma = \sigma_1, \dots \sigma_k$ et $\mathcal{S}'_{\sigma'} = \sigma'_1, \dots \sigma'_{k'}$, on dira que $\mathcal{J} \leq_{Ead} \mathcal{J}'$ si et seulement si :

1. $a(\mathcal{J}) < a(\mathcal{J}')$ ou
2. il existe $i \in \{0, \dots \min(k, k')\}$ tel que $Q(e_{k-i}, \sigma_{k-i}) < Q(e'_{k'-i}, \sigma'_{k'-i})$ et tel que pour tout $j < i$, $Q(e_{k-j}, \sigma_{k-j}) = Q(e'_{k'-j}, \sigma'_{k'-j})$ ou
3. pour tout $j \in \{0, \dots \min(k, k')\}$, $Q(e_{k-j}, \sigma_{k-j}) = Q(e'_{k'-j}, \sigma'_{k'-j})$.

La relation $\leq_{Ead}$ ainsi définie est un préordre[1] total sur l'ensemble des trajets. De plus, les trajets *gloutons* correspondent exactement aux minimums pour cette relation. L'espace des trajets que nous considérons est borné (le nombre d'arcs est inférieur à N, les dates de $\mathcal{S}_\sigma$ sont positives) et fermé (les intervalles de présence sont fermés), ce qui implique que la relation $\leq_{Ead}$ admet un minimum, et donc qu'un trajet *glouton* existe entre s et x. □

[1] relation transitive et réflexive.

La proposition précédente a également pour conséquence qu'il existe des trajets *gloutons* ayant un seul arc, et que l'on peut obtenir les trajets *gloutons* de k arcs à partir de ceux de $k-1$ arcs. Si $\mathcal{J} = (P, \mathcal{S}_\sigma)$ est un trajet *glouton* de s vers x, et si y est le sommet tel que (y, x) est le dernier arc de P, nous avons par conséquent $Ead(s, x) = Q((y, x), Ead(s, x))$.

Nous allons maintenant présenter notre adaptation de l'algorithme de Dijkstra pour les trajets *gloutons* :

Algorithme 6 (Trajets *gloutons*)

Complexité: $O(M \log \delta + N \log N)$

Entrée: un graphe évolutif $\mathcal{G}$ et un sommet source s

Sortie: pour tout sommet $x \in V$, une date d'arrivée au plus tôt $t_{Ead}(x) \in \mathbb{T}$, et pour tout $x \neq s$ le dernier pas $(e(x), \sigma_x)$ dans le trajet de s vers x

Variables: une file de priorité F de sommets, la priorité étant fondée sur la date d'arrivée estimée t_{Ead}
Début

1. Initialiser $t_{Ead}(s) \leftarrow 0$; pour tout $x \neq s \in V$, $t_{Ead}(x) \leftarrow +\infty$. Initialiser F avec uniquement s à la racine.
2. Tant que $Q \neq \emptyset$ faire
   (a) Extraire x, le sommet à la racine de F (x est désormais fermé)
   (b) Pour chaque sommet y adjacent à x, faire
      i. soit $t = Q((x, y), t_{Ead}(x))$
      ii. si $t < t_{Ead}(y)$ alors
      mettre à jour $t_{Ead}(y) \leftarrow t$,
      mettre à jour $e(y) \leftarrow (x, y)$, $\sigma(y) \leftarrow Q^{-1}((x, y), t)$ et
      insérer x dans la file F s'il n'y était pas.
   (c) Mettre la file F à jour.

Fin

Le trajet au plus tôt vers un sommet x peut être retrouvé en récupérant les arcs e et les temps d'émission σ dans l'ordre inverse. A chaque pas de la boucle 2 un sommet est fermé, et ne sera jamais réinséré dans la file F. La boucle est donc répétée N fois au plus avant que l'algorithme ne s'arrête. La validité de l'algorithme est prouvée au Lemme 16.

Lemme 16 *Pour tous les sommets $x \in V$, la valeur de $t_{Ead}(x)$ est égale à $Ead(s, x)$ lorsque x est fermé.*

Preuve
La preuve se fait par récurrence sur le nombre de sommets fermés. Au départ, l'ensemble des sommets fermés contient uniquement la source s et $t_{Ead}(s) = 0 = Ead(s, s)$. Supposons, qu'à un certain point de son exécution, l'algorithme a correctement calculé l'ensemble des sommets fermés, et qu'un nouveau sommet x est sur le point d'être fermé. Cela veut dire que x a été inséré dans la file F, et par conséquent il existe un trajet de s vers x. Soit $\mathcal{J}$ un trajet *glouton* de s vers x. Ce trajet relie un sommet fermé (s) à un sommet non fermé (voir Figure 5.2). Soit y le premier sommet non fermé dans le trajet de s vers x. Comme le prédécesseur de y a été fermé, la valeur du

champ $t_{Ead}(y)$ a été calculée et est inférieure à $Ead(s,x)$. La seule possibilité est donc que $x = y$ et $t_{Ead}(x) = Ead(s,x)$. □

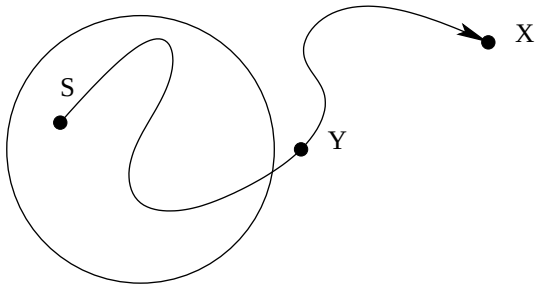


FIG. 5.2 – Trajet de s vers x. Le sommet y est le premier sommet ouvert rencontré.

Nous pouvons voir que cet algorithme est calqué sur celui de Dijkstra, à l'exception que la fonction Q est appelée à chaque fois qu'une arc est examiné. Le nombre d'opérations effectuées est donc au plus $O(M \log \delta + N \log N)$. Le théorème suivant est une conséquence des résultats précédents.

Théorème 20 *L'Algorithme 6 calcule correctement un arbre de trajets* gloutons *du sommet s vers tous les autres sommets en* $O(M \log \delta + N \log N))$ *opérations.*

5.3 Trajets les plus petits

Dans cette section, nous portons notre attention sur le nombre d'arcs des trajets, que nous voulons minimum. Nous présentons à nouveau un algorithme permettant de calculer tous les trajets les plus petits à partir d'un sommet source unique. La difficulté de ce problème réside dans les contraintes temporelles sur les trajets, qui implique que les préfixes d'un plus petit trajet ne sont pas nécessairement eux-mêmes des plus petits trajets. A titre d'exemple, étudions la Figure 5.3. Les intervalles indiqués correspondent aux intervalles de présence des arcs, et nous supposons que tous les délais sont égaux et valent 1. Un trajet de s vers h passera successivement par tous les sommets, de la gauche vers la droite, et contiendra 8 arcs. Par contre il existe un raccourci de s vers d utilisant l'unique arc de s vers d à la date 4.

Néanmoins, on peut remarquer que si (x,y) est le dernier arc d'un plus petit trajet $\mathcal{J}$ de s vers y, alors le trajet préfixe de s vers x est le plus petit parmi l'ensemble des trajets de s vers x arrivant avant la date $Q^{-1}((x,y), a(\mathcal{J}))$. De ce point de vue, la propriété des préfixes est respectée, c'est à dire que le préfixe d'un plus petit trajet sera un trajet le plus petit parmi les trajets arrivant avant une certaine date $t \in \mathbb{T}$. En utilisant cette propriété, nous construisons un arbre de trajets entre le sommet source s et des couples $(x,t) \in V \times \mathbb{T}$, tel que chaque sommet x apparaît au moins une fois dans l'arbre. Si nous reprenons l'exemple de la Figure 5.3, nous obtenons l'arbre des plus petits trajets de la Figure 5.4.

Pour pouvoir continuer, nous présentons l'algorithme 7 ci-dessous. Étant donné un tableau t_{mdd} de minorants sur les dates de départ (indexé sur les sommets de V), l'algorithme calcule un tableau

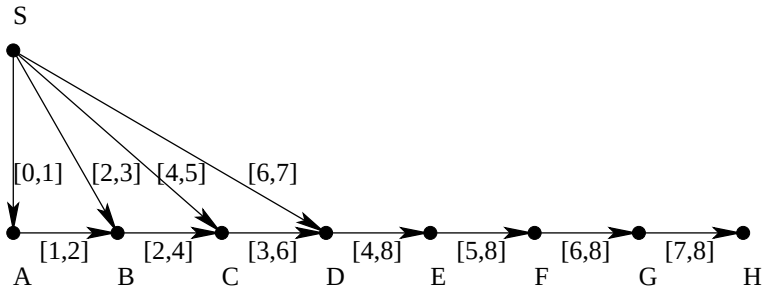


FIG. 5.3 – Le plus petit trajet de S vers H utilise 8 arcs alors qu'il existe un trajet d'un seul arc de S vers D.

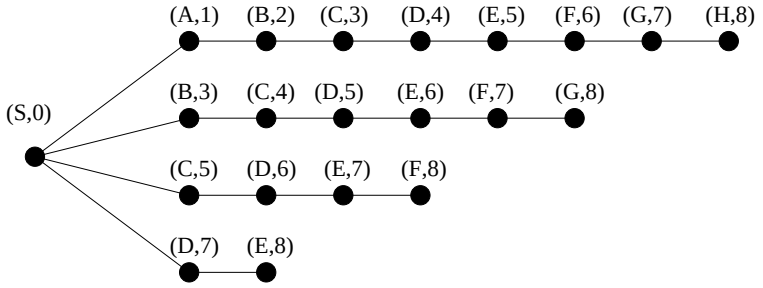


FIG. 5.4 – Arbre des plus petits trajets.

e_{min} d'arcs et un tableau σ_{min} de dates d'émission, tels que
- pour tout sommet y, il existe un sommet x tel que $e_{min}[y] = (x, y)$,
- $\sigma_{min}[y] \geq t_{mdd}[x]$,
- $\sigma_{min}[y]$ est une date d'émission possible pour l'arc $e_{min}[y]$,
- $\sigma_{min}[y] + \zeta(x, y)$ est minimum pour tous les choix $(e_{min}[y], \sigma_{min}[y])$ possibles.

Si aucun couple ne vérifie les conditions précédentes, le choix par défaut est $(nil, +\infty)$.

Algorithme 7 (Sélection des arcs et dates d'émission)

Complexité: $O(M \log \delta)$

Entrée: Un graphe évolutif $\mathcal{G}$, et un tableau $t_{mdd}[x] \in \mathbb{T}$ qui donne pour chaque sommet $x \in V$ un minorant pour les dates de départ depuis x.

Sortie: Deux tableaux $e_{min}[y] \in E$ et $\sigma_{min}[y] \in \mathbb{T}$ qui donne pour chaque sommet $y \in V$ un arc $e_{min}[y] = (x, y)$ et une date d'émission possible $\sigma_{min}[y]$ pour cet arc.

Variables: un tableau $t_a[y] \in \mathbb{T}$ qui donne la date de réception correspondant à l'arc $e_{min}[y]$ et à la date d'émission $\sigma_{min}[y]$.

Début

1. Pour tout sommet $y \in V$ initialiser $e_{min}[y] \leftarrow nil$; $\sigma_{min}[y] \leftarrow +\infty$ et $t_a[y] \leftarrow +\infty$.
2. Pour tout arc $(x, y) \in E$, faire :
   (a) soit $t = Q((x, y), t_{mdd}[x])$.
   (b) si $t < t_a[y]$ alors

i. mettre à jour $e_{min}[y] \leftarrow (x, y)$.
ii. mettre à jour $\sigma_{min}[y] \leftarrow t - \zeta(x, y)$.
iii. mettre à jour $t_a[y] \leftarrow t$.

Fin

A présent, on peut remarquer que si on a un plus petit trajet contenant k arcs, tous ces préfixes contiennent au plus $k-1$ arcs. L'algorithme suivant calcule toutes les dates d'arrivée possibles correspondant à des trajets de $k-1$ arcs, puis continue en considérant un arc supplémentaire. Cet algorithme se termine lorsque tous les sommets ont une date d'arrivée différente de $+\infty$ ou lorsque le nombre d'arcs des trajets dépasse N. On garde trace de la première fois qu'un sommet x a une date d'arrivée finie, et du plus petit trajet de s vers x. Le nombre d'itérations de notre algorithme sera donc l'excentricité du graphe évolutif $\mathcal{G}$, ou N si certains trajets n'existent pas.

Algorithme 8 (Trajets les plus petits)

Complexité: $O(M \times N \log \delta)$

Entrée: Un graphe évolutif $\mathcal{G}$, et un sommet source $s \in V$

Sortie: Pour chaque sommet x, le plus petit trajet $\mathcal{J}_{petit}(x)$ de s vers x.

Variables:

- Pour chaque sommet x, un trajet $\mathcal{J}(x)$ de trajets de s vers x, et un minorant sur les dates de départ possible $t_{mdd}[x]$.
- Pour chaque sommet y deux valeurs $e_{min}[y] \in E$ et $\sigma_{min}[y] \in \mathbb{T}$ provenant de l'Algorithme 7.
- le nombre d'arcs considérés $k \in \{0, 1 \ldots, N\}$.

Début

1. Initialiser $t_{mdd}[s] \leftarrow 0$, $\mathcal{J}(s) \leftarrow ()$ et définir $\mathcal{J}_{petit}(s) = ()$; pour tout $x \neq s$ $t_{mdd}[x] \leftarrow +\infty$ et $\mathcal{J}[s] \leftarrow ()$; $k \leftarrow 0$.
2. Tant qu'il existe un sommet $x \in V$ tel que $T_{mdd}[x] = +\infty$ et que $k < N$, faire :
   (a) $k \leftarrow k + 1$
   (b) exécuter l'Algorithme 7
   (c) Pour chaque sommet $y \in V$, faire :
   si $e_{min}[y] \neq nil$ alors
      i. soit $(x, y) = e_{min}[y]$
      ii. soit $(P, \mathcal{S}_\sigma) = \mathcal{J}(x)$.
      iii. mettre à jour $\mathcal{J}(y) \leftarrow ((P, (x, y)), (\mathcal{S}_\sigma, \sigma_{min}[y]))$.
      iv. si $t_{mdd}[y] = +\infty$ alors définir $\mathcal{J}_{petit}(y) = \mathcal{J}(y)$.
      v. mettre à jour $t_{mdd}[y] \leftarrow t_{min}[y] + \zeta(x, y)$.

Fin

Proposition 14 *L'Algorithme 8 calcule les trajets les plus petits d'un unique sommet s vers tous les autres sommets, si de tels trajets existent. Si tous ces trajets existent dans $\mathcal{G}$, alors la complexité de l'Algorithme 8 est $O(Md \log \delta)$ où d est l'excentricité de s. Dans le cas contraire, la complexité de l'algorithme est $O(MN \log \delta)$.*

Preuve
Nous allons montrer par récurrence que pour chaque k, la première date d'arrivée possible au sommet x en k sauts est bien $t_{lmdd}[x]$. Cela est immédiatement vrai pour $k = 0$. Comme nous l'avons vu précédemment à la section 5.2, un trajet qui donne la première date d'arrivée possible en k sauts peut être calculé d'après les préfixes possibles de $k-1$ sauts. Ceci nous montre que nous les trajets au plus tôt en k sauts sont bien calculés à chaque étape.

Pour chaque étape k, tous les arcs sont examinées, et le calcul de Q coûte $O(\log \delta$, ce qui fait que la complexité pour l'étape k est $O(M \log \delta)$. La complexité totale est donc $O(Md\ log\delta)$, si l'excentricité de s vaut d, ou $O(MN \log \delta)$ si certains trajets n'existent pas. □

5.4 Trajets les plus rapides

Nous nous intéressons dans cette section à la durée des trajets, et nous allons montrer comment calculer des trajets les plus rapides. Le problème des trajets les plus rapide est bien plus complexe que les deux problèmes précédents, car un trajet rapide peut commencer très tardivement, ou peut être très long. De plus, la durée des trajets préfixe n'a pas nécessairement de rapport avec la durée globale d'un trajet. En effet, un trajet préfixe de très petite durée peut impliquer par la suite un temps d'attente très long sur un sommet, annulant après coup le gain apparent en temps. En revanche, certains trajets seront à coup sûr inutiles car trop lents, et pourront être éliminés. Les trajets restant, dits *pertinents*, seront regroupés par classes de trajets comme nous le verrons par la suite. Nous utilisons un algorithme de trajet au plus tôt pour déterminer chaque classe de trajets, et nous allons montrer que le nombre de ces classes que nous devons examiner est borné par la taille de la donnée de $\mathcal{G}$, ce qui fait que la complexité de l'algorithme que nous proposons reste raisonnable.

Pour chaque trajet, la mesure de qualité (sa durée) est différente de sa longueur (nombre d'arcs), et de sa date d'arrivée. Nous ne nous intéressons pas à la longueur des trajets. Concernant les deux autres paramètres nous les associerons à chaque trajet sous la forme de deux variables, $a(\mathcal{J})$ pour la date d'arrivée et $d(\mathcal{J})$ pour la durée. Considérons deux trajets $\mathcal{J}_1$ et $\mathcal{J}_2$ de s vers x, de dates de départ t_1, t_2 et de dates d'arrivée t_{a1}, t_{a2}. Si $t_1 \leq t_2$ et $t_{a1} \geq t_{a2}$, le trajet $\mathcal{J}_1$ part plus tôt et arrive plus tard que le trajet $\mathcal{J}_2$. Le trajet $\mathcal{J}_2$ sera donc toujours meilleur que le trajet $\mathcal{J}_1$. Nous nous intéresserons désormais aux trajets pour lesquels il n'existe pas de trajet 'meilleurs'.

Pour toute date $t \in \mathbb{T}$, on appelle $\mathcal{G} \bigcap [t, +\infty[$ le graphe évolutif construit à partir de $\mathcal{G}$ en intersectant tous les intervalles de présence de $\mathcal{G}$ avec l'intervalle $[t, +\infty[$. Il existe une ou plusieurs dates t telles qu'un trajet le plus rapide dans $\mathcal{G}$ soit un trajet au plus tôt dans $\mathcal{G} \bigcap [t, +\infty[$. En conséquence, le trajet le plus rapide de $\mathcal{G}$ sera un minimum sur l'ensemble des trajets au plus tôt pour chaque $\mathcal{G} \bigcap [t, +\infty[$. Nous appelons $Ead_t(x, y)$ la date d'arrivée au plus tôt de x vers y dans le graphe évolutif $\mathcal{G} \bigcap [t, +\infty[$. La fonction $A : \mathbb{T} \to \mathbb{T}$ qui à $t \in \mathbb{T}$ associe $Ead_t(x, y)$ est nécessairement une fonction croissante sur $\mathbb{T}$.

Définition 27 (Trajets Pertinents) *Soit $\mathcal{G}$ un graphe évolutif, x et y deux sommets de $\mathcal{G}$, et t une date de $\mathbb{T}$. Soit $\mathcal{J}$ un trajet* glouton *de x vers y dans le graphe évolutif $\mathcal{G} \bigcap [t, +\infty[$. Si pour tout $\epsilon > 0$ $Ead_t(x, y) < Ead_{t+\epsilon}(x, y)$, alors $\mathcal{J}$ est appelé* trajet pertinent *de x vers y.*

Une autre observation que l'on peut faire est qu'un trajet peut engendrer toute une classe de trajets de même durée, qui s'obtiennent en décalant toutes les dates d'émissions. Nous illustrons ceci par l'exemple suivant : considérons deux arcs (x, y) et (y, z). Le délai de l'arc (x, y) est 3, et son intervalle de présence est $[1, 8]$. Le délai de l'arc (y, z) est 4 et son intervalle de présence est $[5, 13]$. Le trajet $((x, y), (y, z), 3, 6)$ entre x et z à une durée de 7. Tous les trajets $((x, y), (y, z), 2+\epsilon, 5+\epsilon)$ ont la même durée pour $\epsilon \in [0, 3]$. Cet exemple est présenté dans la Figure 5.5.

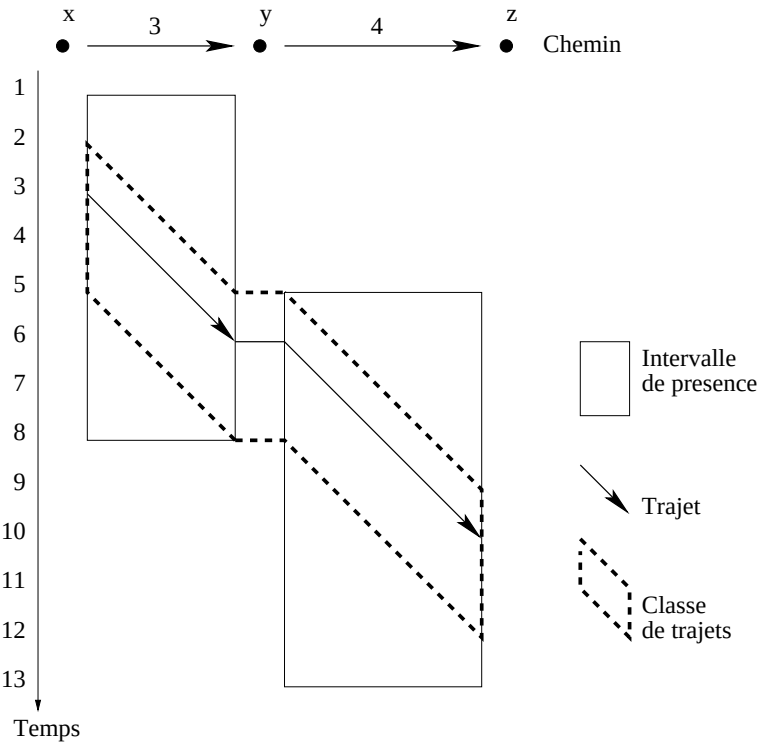


FIG. 5.5 – Un trajet et une classe de trajets similaires.

Définition 28 (Classe de trajets) *Soit $\mathcal{J} = (P, \mathcal{S}_\sigma)$ un trajet de $\mathcal{G}$ avec $P = e_1, e_2, \dots e_k$ et $\mathcal{S}_\sigma = \sigma_1, \sigma_2, \dots \sigma_k$. Soit $\Delta \in \mathbb{T}$ tel que pour tout i, tout $t \in [\sigma_i, \sigma_i + \Delta]$ est une date d'émission possible pour e_i. On note*

- *$\mathcal{S}_{\sigma+\epsilon}$ la suite de date d'émissions $\sigma_1 + \epsilon, \sigma_2 + \epsilon, \dots \sigma_k + \epsilon$*
- *$\mathcal{J}_\epsilon$ le trajet $(P, \mathcal{S}_{\sigma+\epsilon})$.*
- *$\mathcal{J}_{[0,\Delta]}$ l'ensemble de trajets $\{\mathcal{J}_\epsilon,\ \epsilon \in [0, \Delta]\}$*

Pour tout intervalle I inclus dans $[0, \Delta]$ on appellera classe de trajets $\mathcal{J}_I$ l'ensemble $\{\mathcal{J}_\epsilon,\ \epsilon \in I\}$.

Algorithme 9 (Trajets les plus rapides)

Complexité: $O(\mathcal{M} \times (M \log \delta + N \log N))$

Entrée: Un graphe évolutif $\mathcal{G}$, et un sommet source $s \in V$

Sortie: Pour chaque sommet x, le trajet le plus rapide $\mathcal{J}_{rapide}(x)$ de s vers x.

Variables:

- une date de départ t_d
- deux temps de décalage Δ, Δ_{min}
- pour chaque sommet x une date de départ du trajet le plus rapide $t_d(s,x)$
- pour chaque sommet x, le délai $t(s,x)$

Début

1. $t_d \leftarrow 0$.
2. pour tout x, initialiser $t(s,x) \leftarrow +\infty$.
3. tant que $t_d < +\infty$ faire
   (a) $\Delta \leftarrow +\infty$
   (b) calculer à l'aide de l'Algorithme 6 les dates d'arrivée au plus tôt $Ead_{t_d}(x)$ pour tous les sommets $x \in V$. Au cours de l'Algorithme 6, nous effectuons les calculs suivants lorsque le sommet $x \in V$ est sur le point d'être fermé.
      i. si $(Ead_{t_d}(x) - t_d) < t(s,x)$ alors
         A. $t(s,x) \leftarrow (Ead_{t_d}(x) - t_d)$
         B. $t_d(s,x) \leftarrow t_d$
      ii. si le trajet de s vers x ne comporte pas de temps d'attente, faire
         A. soit t la première date supérieure à $Ead_{t_d}(x)$ telle que un arc adjacent à x apparaît ou disparaît.
         B. si $\Delta \leftarrow \min(\Delta, (t - Ead_{t_d}(x)))$
   (c) $t_d \leftarrow t_d + \Delta$
4. pour tout sommet x calculer le trajet au plus tôt $\mathcal{J}(x)$ de s vers x partant à la date $t_d(s,x)$.

Fin

Proposition 15 *L'Algorithme 9 calcule les trajets les plus rapides de s vers les autres sommets de $\mathcal{G}$ en $O(\mathcal{M} \times (M \log \delta + N \log N))$ pas de calcul.*

Preuve
Soit t_d et Δ l'état des paramètres de l'Algorithme 9 à la fin de la boucle interne. Soit $0 < \epsilon < \Delta$ une durée de $\mathbb{T}$. Pour tout sommet $x \in V$, on a $Ead_{t_d}(x) \leq Ead_{t_d+\epsilon}(x) \leq Ead_{t_d+\Delta}(x)$. On peut remarquer que l'état du voisinage de x ne change pas entre $Ead_{t_d+\epsilon}(x)$ et $Ead_{t_d+\Delta}(x)$. Aussi, pour tout trajet au plus tôt de s à x partant à la date $t_d + \epsilon$ et empruntant un chemin P, il existe de s à x partant à la date $t_d + \Delta$ et empruntant le même chemin P. Ceci a pour conséquence que $Ead_{t_d+\Delta}(s,x) - t_d - \Delta \leq Ead_{t_d+\epsilon} - t_d - \epsilon$. Aussi, un trajet le plus rapide de s vers x sera bien calculé par l'Algorithme 9. De plus, le paramètre Δ correspond à un sommet x et une date t_d, tel qu'un trajet au plus tôt de s vers x et partant à la date $t_d + \Delta$ arrive en x alors qu'un changement intervient au sommet x (apparition ou disparition d'arc). Le nombre de ces évènements étant limité à $2\mathcal{M}$, la boucle principale est exécutée au plus $\mathcal{M}$ fois. La complexité de l'Algorithme 9 est en $O\mathcal{M} \times (M \log \delta + N \log N))$. □

5.5 Fonction arbitraire de coût

Dans cette section, nous supposons l'existence d'une fonction arbitraire de coût $c : E \rightarrow \mathbb{N}_+$ sur les arêtes. Le coût d'un trajet peut se définir simplement comme la somme des coûts des arêtes

empruntés par le trajet. Nous allons montrer que sous cette hypothèse, le problème du trajet de coût minimal est NP-difficile[2]. Par contre, ce problème devient simple à résoudre lorsque le délai est nul pour toutes les arêtes.

Théorème 21 (Fonction arbitraire de coût sur les trajets) *Soit $\mathcal{G}$ un graphe évolutif. Soit x et y deux sommets de $\mathcal{G}$. Soit $c : E \to \mathbb{N}_+$ une fonction de coût sur les arêtes du graphes sous-jacent de $\mathcal{G}$. Le problème de minimisation du coût total d'un trajet de x vers y est NP-difficile.*

Preuve
Étant donné un trajet de x vers y, il est facile de calculer sa validité en tant que trajet, et son coût, qui est la somme des coûts des arêtes. Le problème est donc NP. Nous allons maintenant faire une réduction du problème *Maximum Subset Sum*. Soit U un ensemble fini, $s : U \to \mathbb{N}_+$ une fonction de coût sur U, et s_{max} un entier naturel. Le problème du *Maximum Subset Sum* consiste à trouver un sous-ensemble U' de U telle que la somme

$$S = \sum_{u \in U'} s(u)$$

soit maximale sous la condition $S \leq s_{max}$ [66].

Nous construisons un graphe évolutif à partir de l'instance U, s, s_{max}. Numérotons les éléments de U : $u_1, u_2, \dots u_k$. Nous allons construire un graphe évolutif $\mathcal{G}$ de dynamicité 1, avec $3k + 2$ sommets et $4k + 1$ arêtes. Nous nommons les sommets de V $x_0, x_1, \dots x_k$, $a_1, a_2, \dots a_k$, $b_1, b_2, \dots b_k$ et pour finir, y. Les arêtes de E sont les arêtes

- $\forall i \in \{1, 2, \dots k\}$, (x_{i-1}, a_i) de délai 1 et de coût 1,
- $\forall i \in \{1, 2, \dots k\}$, (x_{i-1}, b_i) de délai 1 et de coût 1,
- $\forall i \in \{1, 2, \dots k\}$, (a_i, x_i) de délai 1 et de coût $1 + s(u_i)$,
- $\forall i \in \{1, 2, \dots k\}$, (b_i, x_i) de délai $1 + s(u_i)$ et de coût 1,
- et enfin, l'arête (x_n, y) de coût 1 et de délai 1.

Tous les sommets et arêtes ont un intervalle de présence égal à $[0, +\infty[$, excepté l'arête (x_n, y) qui est présente seulement durant l'intervalle $[0, s_{max} + 2k + 1]$. Le graphe construit est illustré par la Figure 5.6. Le problème posé est de trouver un trajet de x_0 à y de coût minimal.

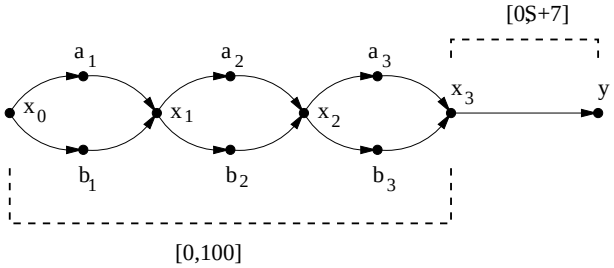


FIG. 5.6 – Graphe évolutif construit par réduction de problème SUMSET SUM pour 3 variables.

Soit $\mathcal{J}$ un trajet de x_0 vers y partant à la date $t = 0$ et n'attendant dans aucun sommet. Pour chaque entier $i \in \{1, 2, \dots k\}$, $\mathcal{J}$ passe soit par le sommet a_i soit par le sommet b_i. La date d'arrivée

[2]Ce problème peut également être vu comme une variation du problème du MINCOST QUICKEST FLOW [56].

du trajet $\mathcal{J}$ est égale à

$$a(\mathcal{J}) = \sum_{(b_i,x_i)\in\mathcal{J}} s(u_i) + 2k + 1$$

Le coût du trajet $\mathcal{J}$ est quant à lui égal à

$$\sum_{(a_i,x_i)\in\mathcal{J}} s(u_i) + 2k + 1 = 2k + 1 + \sum_{i=1}^{k} s(u_i) - \sum_{(b_i,x_i)\in\mathcal{J}} s(u_i).$$

De plus pour pouvoir utiliser l'arête (x_k, y), le trajet est soumis à la contrainte

$$\sum_{(b_i,x_i)\in\mathcal{J}} s(u_i) \leq s_{max}.$$

Minimiser le coût d'un trajet $\mathcal{J}$ de x vers y c'est donc maximiser la somme $S = \sum_{(b_i,x_i)\in\mathcal{J}} s(u_i)$ sous la contrainte $S \leq s_{max}$.

□

La difficulté du problème du trajet de coût minimum vient de la décorrélation entre la fonction de coût (coût arbitraire sur les arêtes) et de la contrainte temporelle (les dates d'émission sur un trajet doivent rester cohérentes avec les temps de traversée des arêtes). Lorsque l'on assouplit la contrainte temporelle en fixant tous les délais à zéro, le problème du trajet de coût minimum devient 'facile' : *l'hypothèse de discrétisation* est vérifiée, et il suffit de calculer un plus court chemin sur le graphe discrétisé. Le graphe discrétisé s'obtient à partir de la suite des sous-graphes $\mathcal{S}_G = G_1, G_2, \dots G_k$. Il se construit comme suit :

- $V_{discret} = \{1, 2, \dots k\} \times V$
- $E_{vertical} = \bigcup_{x\in V} \bigcup_{i\in\{1,2,\dots k-1\}} ((x, i), (x, i + 1))$
- $E_{horizontal} = \bigcup_{i\in\{1,2,\dots k\}} \bigcup_{(x,y)\in E_i} ((x, i), (y, i))$
- $E_{discret} = E_{vertical} \bigcup E_{horizontal}$
- $G_{discret} = (V_{discret}, E_{discret})$

L'ensemble d'arêtes $E_{vertical}$ correspond à la possibilité d'attendre dans les sommets. L'ensemble d'arêtes $E_{horizontal}$ correspond aux arêtes des graphes E_i. À chaque chemin de $G_{discret}$ correspond à un trajet de $\mathcal{G}$, et à chaque trajet de $\mathcal{G}$ correspond un chemin de $G_{discret}$. Pour trouver un trajet de coût minimum entre deux sommets x et y dans $\mathcal{G}$, il suffit donc de trouver un chemin de coût minimum entre $(x, 0)$ et (y, k) dans le graphe classique $G_{discret}$.

5.6 Conclusion

Dès que le temps devient un facteur, des mesures de distances fort différentes peuvent être pertinentes. Nous avons proposé des algorithmes de plus court trajet suivant trois principales mesures : trajets minimisant la date d'arrivée, trajets minimisant le temps de traversée du graphe, et enfin trajets minimisant le nombre de liens empruntés. Ces trois mesures étudiées ont un lien direct avec la topologie ou avec la contrainte temporelle. Nous avons montré que pour une mesure de coût arbitraire sur les arêtes, et donc sans rapport avec les contraintes liées aux trajets, trouver des trajets de coût minimum devient un problème NP-difficile.

Chapitre 6

Composantes connexes

Sommaire

6.1 Introduction

La recherche sur les réseaux dynamiques à d'abord été motivée par l'étude de problèmes provenant du monde des transports. Ainsi, les problèmes de trajets et de flots ont déjà été étudiés sous des modèles et des méthodes très divers. Cependant, l'étude des réseaux de télécommunications, et plus particulièrement des réseaux radio, a amené de nouveaux concepts et de nouveaux problèmes, et en particulier celui de la connexité qui fait l'objet de ce chapitre. En effet, dans le cadre de l'étude d'un réseau routier, il va sans dire que l'on suppose toujours l'existence d'une route entre le point de départ et le point d'arrivée d'une livraison. Cependant, dans le cadre des réseaux de télécommunications, il arrive fréquemment que l'existence même d'une liaison possible entre le point de départ et la destination d'un message soit un problème. Un problème de base des réseaux de télécommunication est de déterminer les ensembles de sommets capables de communiquer tous entre eux : il s'agit en théorie des graphes du problème des composantes connexes. Ce problème a été montré NP-difficile par Bhadra et Ferreira [5]. Nous verrons dans ce chapitre une démonstration un peu différente de ce résultat, et nous l'étendons au cas où le graphe sous-jacent est une *double grille*. Puis, nous résolvons le cas particulier des graphes évolutifs où le graphe sous-jacent (graphe formé par toutes les connexions possibles) est un arbre.

Adjacence, trajets, circuits, relation de connexité

L'objet de ce chapitre est d'étudier la relation de connexité sur les graphes évolutifs.

Dans un réseau de télécommunication, la question est de savoir si un message peut aller d'un point x vers un point y. Cela se traduit par l'existence d'un trajet de x vers y. Aussi s'il existe un trajet de x vers y, on dira que x est connecté à y, que l'on note xRy. La relation R est appelée relation de connexité sur le graphe évolutif $\mathcal{G}$. Cette relation R que nous venons de définir sur les sommets du graphe évolutif comporte bien sûr des similitudes avec la relation de connexité sur un graphe classique, mais également d'importantes disparités. Les contraintes de temps sur les trajets impliquent que la relation de connexité n'est pas transitive[1]. Ceci est illustré par l'exemple 6.1. Il faut bien noter que ce phénomène existe aussi lorsque les temps de traversée des arcs sont réduits à zéro. Ceci a plusieurs conséquences intéressantes. D'abord, le fait qu'il existe un trajet de x

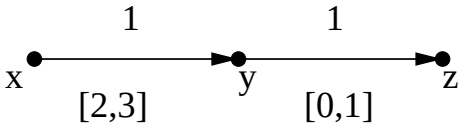


FIG. 6.1 – Deux arcs de délai 1. L'arc xy est présent durant l'intervalle $[2, 3]$, l'arc yz durant l'intervalle $[0, 1]$.

vers y et un trajet de y vers x ne signifie pas qu'il existe un trajet aller-retour entre x et y. En d'autres termes, le fait que l'on puisse envoyer un message de x vers y et un message de y vers x "simultanément", ne signifie pas qu'on puisse envoyer un message de x vers y, et une fois celui-ci arrivé, envoyer un accusé de réception de y vers x. Ensuite, s'il existe un trajet circulaire[2] de x vers x passant par les points y et z, cela ne signifie pas qu'il existe un trajet circulaire de y vers y ou de z vers z (voir figure 6.2).

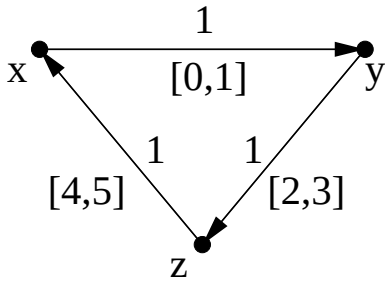


FIG. 6.2 – Trois arcs de délai 1. L'arc xy est présent durant l'intervalle $[0, 1]$, l'arc yz durant l'intervalle $[2, 3]$, et l'arc zx durant l'intervalle $[4, 5]$.

[1]Une relation $\overline{\prec}$ est transitive si $x \overline{\prec} y$ et $y \overline{\prec} z$ implique que $x \overline{\prec} z$. La relation de connexité dans les graphes usuels est transitive. La relation de connexité dans les graphes évolutifs n'est pas transitive.

[2]Trajet dont le sommet de départ est le même que celui de la destination.

6.2 Formalisation des composantes connexes

Nous allons maintenant formaliser ce qu'est une composante connexe dans un graphe évolutif. Un graphe G est dit (fortement) [3] connexe si pour tout couple de sommets x et y de G, il existe un chemin de x vers y. Nous adoptons une définition similaire pour les graphes évolutifs :

Définition 29 (fortement connexe) *Un graphe évolutif $\mathcal{G}$ est dit fortement connexe si pour toute paire de sommets x, $y \in V$, il existe un trajet de x vers y.*

Savoir si un graphe évolutif est fortement connexe est relativement aisé : il suffit de savoir s'il existe un trajet entre toute paire de sommets. En utilisant l'Algorithme 6 cela prend $O(N \times (M \log \delta + N \log N))$. Nous allons voir que définir les composantes connexes est moins aisé : dans un graphe classique, une composante (fortement) connexe peut se définir comme un ensemble maximal de sommets tel que le sous-graphe induit par ces sommets est lui-même (fortement) connexe [16]. Une composante (fortement) connexe peut aussi se définir comme un ensemble maximal de sommets tel qu'il existe un chemin entre toute paire de sommets de cet ensemble. Ces deux définitions diffèrent sur un point : dans la première définition, les chemins doivent passer uniquement par des sommets de la composante, alors que la deuxième définition autorise des chemins extérieurs à la composante. Cette distinction n'importe pas dans le cas des graphes classiques car les sommets d'un chemin entre deux points d'une même composante appartiennent nécessairement à cette même composante. Cependant, cette distinction s'impose dans le cas des graphes évolutifs. Nous définissons donc deux types de composantes connexes :

Définition 30 (composante connexe fermée) *Soit $S \subset V$ un sous-ensemble maximal de sommets du graphe évolutif $\mathcal{G}$ tel que le graphe évolutif engendré par S est fortement connexe. S est appelé composante connexe fermée de $\mathcal{G}$.*

Définition 31 (composante connexe ouverte) *Soit $S \subset V$ un sous-ensemble maximal de sommets du graphe évolutif $\mathcal{G}$ tel que pour tout couple (x, y) de sommets de S il existe un trajet de x vers y. S est une composante connexe ouverte de $\mathcal{G}$.*

6.3 Difficulté combinatoire du calcul des composantes connexes

Nous venons de voir que la relation R n'est pas transitive. A partir de cette observation, nous allons réduire le problème Clique au problème de trouver une composante connexe dans un graphe évolutif.

Théorème 22 (composante connexe dans les graphes évolutifs avec délai non nul) *Étant donné un graphe évolutif $\mathcal{G}$, le problème consistant à trouver une composante connexe ouverte ou fermée de taille k de $\mathcal{G}$ est NP-difficile.*

Preuve
Soit $G(V, E)$ un graphe orienté. Nous allons construire un graphe évolutif $\mathcal{G}$ dont le graphe sous-

[3] Connexe dans le cas non orienté, fortement connexe dans le cas orienté.

jacent est G et tel que la relation de connexité sur $\mathcal{G}$ correspond à la relation d'adjacence sur G. Nous définissons :

- pour tout $x \in V$, x est présent durant l'intervalle $[0, 1]$,
- pour tout $e \in E$, e est présent durant l'intervalle $[0, 1]$,
- pour tout $e \in E$, le délai de e est égal à 1.

La condition de délai sur les arcs de $\mathcal{G}$ implique qu'il n'y a pas de trajets contenant deux arcs ou plus dans $\mathcal{G}$. Aussi, il existe un trajet entre deux sommets x et y de V si et seulement si $(x, y) \in E$. De plus, toutes les composantes connexes ouvertes de $\mathcal{G}$ sont également fermées. On en conclut donc que toutes les cliques maximales de G sont des composantes connexes de $\mathcal{G}$, et que toutes les composantes connexes de $\mathcal{G}$ sont des cliques maximales de G. Le problème des composantes connexes est donc NP-difficile. □

Il faut bien noter que le délai n'est pas le seul facteur contrariant la transitivité de R. Nous allons montrer que le problème de trouver une composante connexe de taille k est également NP-difficile dans le cas où tous les délais sont nuls. Nous distinguons ici les composantes connexes ouvertes et fermées.

Théorème 23 (composante connexes ouvertes) *Le problème consistant à trouver une composante connexe ouverte de taille k dans un graphe évolutif est NP-complet, même lorsque les délais des arcs de $\mathcal{G}$ sont tous nuls.*

Preuve
Soit $G_{clique} = (V_{clique}, E_{clique})$ un graphe orienté. Nous construisons un graphe évolutif $\mathcal{G}$ à partir de G_{clique} comme suit :

- soit V l'ensemble de sommets défini par $V = V_{clique} \bigcup E_{clique}$
- soit E l'ensemble d'arcs tel que $xy \in E$ si et seulement si
  - y est un arc de E_{clique} reliant x à un autre sommet, ou si
  - x est un arc de E_{clique} reliant un sommet à y.
- tous les sommets de V sont présents durant l'intervalle $[0, 3]$
- les arcs de (x, y) de E sont présents durant
  - $[0, 1]$ si x est un sommet de V_{clique},
  - $[2, 3]$ si y est un sommet de V_{clique}.

Les sommets du graphe évolutif $\mathcal{G}$ ainsi défini se décomposent clairement en deux catégories, ceux appartenant à V_{clique} et ceux appartenant à E_{clique}. Il faut noter qu'un sommet de $\mathcal{G}$ appartenant à E_{clique} n'est en relation qu'avec deux sommets de $\mathcal{G}$ qui sont ses extrémités dans G_{clique} ; il ne peut faire partie que d'une composante connexe réduite à lui-même. Aussi, une composante connexe ouverte de $\mathcal{G}$ de taille au moins 2 correspond à une clique sur G_{clique}. On en conclut donc que toutes les cliques maximales de G sont des composantes connexes de $\mathcal{G}$, et que toutes les composantes connexes de $\mathcal{G}$ de taille supérieure à 2 sont des cliques maximales de G. Le problème des composantes connexes est donc NP-difficile. □

Théorème 24 (composante connexes fermées) *Le problème consistant à trouver une composante connexe fermée de taille k dans un graphe évolutif est NP-complet, même lorsque les délais des arcs de $\mathcal{G}$ sont tous nuls.*

Preuve
Soit $G_{clique} = (V_{clique}, E_{clique})$ un graphe orienté. Nous construisons un graphe évolutif $\mathcal{G}$ à partir

de G_{clique} comme suit (voir Figure 6.3) :
- soit V l'ensemble de sommets défini par $V = V_{clique} \bigcup E_{clique}$
- soit E l'ensemble d'arcs tel que $xy \in E$ si et seulement si
  - y est un arc de E_{clique} reliant x à un autre sommet, ou si
  - x est un arc de E_{clique} reliant un sommet à y, ou si
  - x et y sont des arcs de E_{clique}.
- tous les sommets de V sont présents durant l'intervalle $[0, 7]$
- les arcs de (x, y) de E sont présents durant
  - $[2, 3]$ si x est un sommet de V_{clique},
  - $[4, 5]$ si y est un sommet de V_{clique},
  - $[0, 1]$ et $[6, 7]$ si x et y sont des arcs de E_{clique}.

Les sommets de $\mathcal{G}$ appartenant également à E_{clique} sont tous reliés entre eux durant les intervalles $[0, 1]$ et $[6, 7]$. Comme par ailleurs tous les sommets de $\mathcal{G}$ appartenant également à V_{clique} sont reliés à un sommet représentant un arc de E_{clique} durant les intervalles $[2, 3]$ et $[4, 5]$, on en conclut que les sommets de E_{clique} sont en relation avec tous les sommets de $\mathcal{G}$: pour tout $x \in E_{clique}$, pour tout $y \in V$, xRy et yRx. Une composantes connexe fermée de $\mathcal{G}$ correspond donc à l'union d'une clique maximale sur G_{clique} et de l'ensemble E_{clique}. L'union d'une clique maximale de G_{clique} et de l'ensemble E_{clique} correspond à une composante connexe fermée de $\mathcal{G}$. Le problème des composantes connexes est donc NP-complet même lorsque les délais des arcs de $\mathcal{G}$ sont tous nuls. □

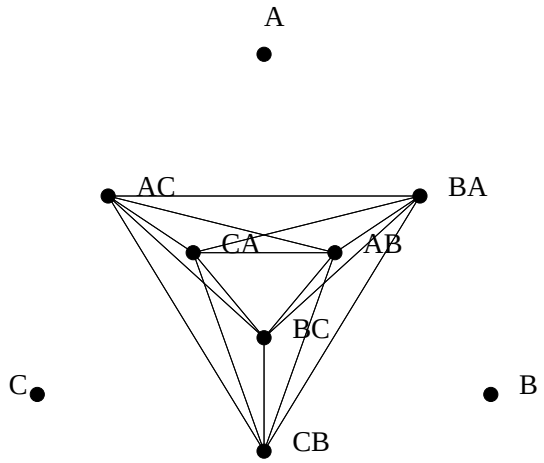
Notez que la difficulté du problème des composantes connexes est présente même lorsque la dynamicité du graphe évolutif est très faible. Les réductions que nous avons présentées ne comportent que 4 sous-graphes distincts. Nous allons voir à présent que la difficulté du problème subsiste également lorsque la topologie du graphe sous-jacent est très contrainte : nous allons étudier une classe de graphe sous-jacent particulière : la *double grille*.

Définition 32 (double grille (n, m)) *Soit V l'ensemble des paires $(2i, 2j)$, telles que $1 \leq i \leq n$ et $1 \leq j \leq m$. Deux sommets distincts $(i, j), (i', j') \in V$ sont reliés par une arête de E si $|i - i'| \leq 2$ et $|j - j'| \leq 2$. Le graphe $G = (V, E)$ est appelé une double grille (n, m).*

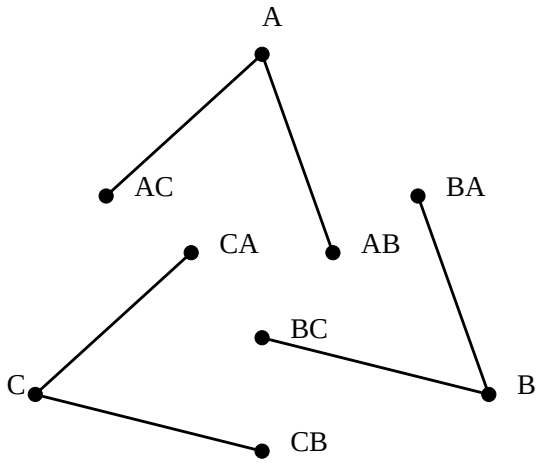
Nous pouvons voir un exemple de double grille $(5, 5)$ illustré à la Figure 6.4. Cette classe de graphe est un cas particuliers des *unit disc graphs*, utilisés pour modéliser des réseaux de stations radio. Un *unit disc graph* s'obtient en distribuant des noeuds sur un plan euclidien, chaque noeud représentant une station radio. Deux stations peuvent communiquer si leur distance est inférieure à la portée du signal radio, supposé uniforme. La Figure 6.5 représente le rayon d'émission d'une seule station et les communications possibles depuis cette station, lorsque l'ensemble des stations est réparti de façon régulière.

Théorème 25 (composante connexes fermées) *Le problème consistant à trouver une composante connexe fermée de taille k dans un graphe évolutif est NP-complet, même lorsque les délais des arcs de $\mathcal{G}$ sont tous nuls et que le graphe sous-jacent est une double grille.*

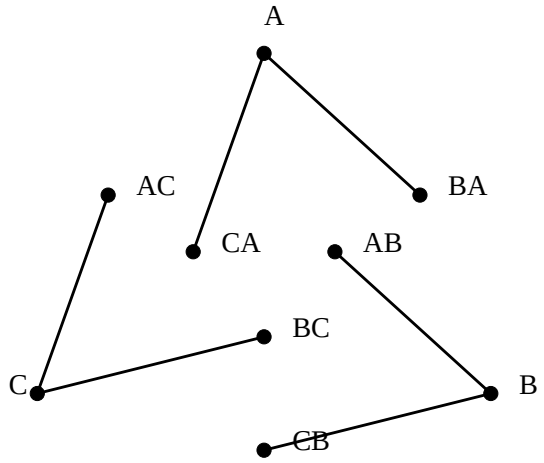
Preuve
Soit $G_{clique} = (V_{clique}, E_{clique})$ un graphe orienté. Nous construisons un graphe évolutif $\mathcal{G}$ à partir de G_{clique} comme suit (voir Figure 6.6) :
- soit $n = |V_{clique}| + (2 * |E_{clique}| - |E_{clique}|)$
- soit $G = (V, E)$ une double grille (n, n)



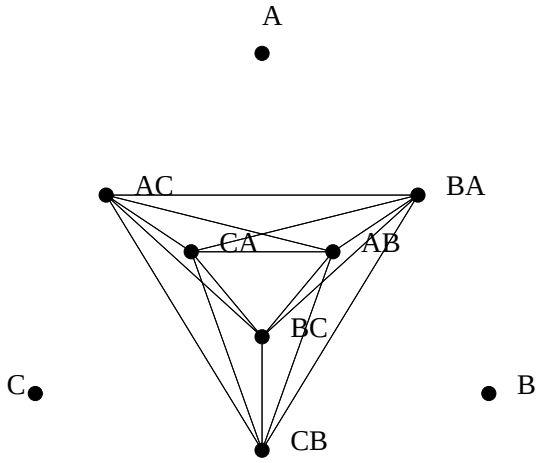
Configuration sur l'intervalle $I = [0, 1]$



Configuration sur l'intervalle $I = [2, 3]$



Configuration sur l'intervalle $I = [4, 5]$



Configuration sur l'intervalle $I = [6, 7]$

FIG. 6.3 – Graphe évolutif représentant le graphe complet à 3 sommets $\{A, B, C\}$, construit suivant la preuve du Théorème 24.

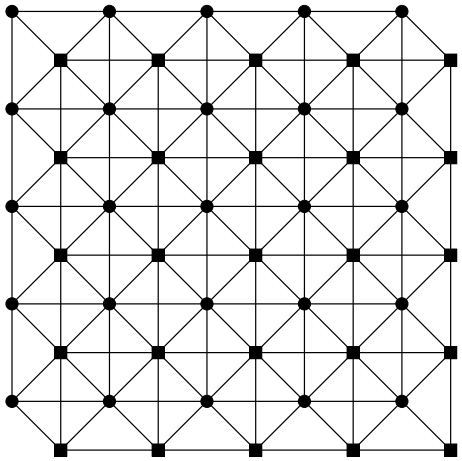


FIG. 6.4 – Double grille $(5, 5)$.

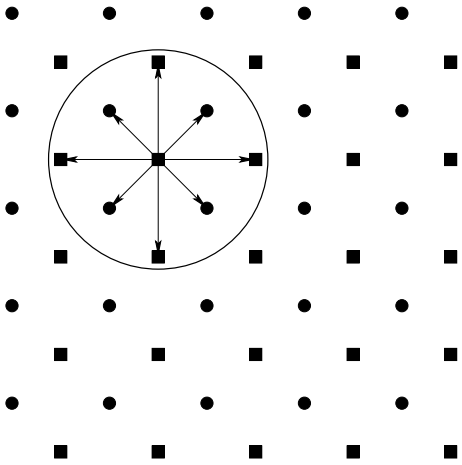


FIG. 6.5 – Les sommets situés à l'intérieur du rayon d'émission sont connectés à l'émetteur.

– tous les sommets de V sont présents durant l'intervalle $[0, 11]$ Ils peuvent cependant être actifs ou inactifs. Les arêtes de $\mathcal{G}$ seront présentes lorsque leurs deux extrémités sont des sommets actifs.

Les sommets de V_{clique} nommés $1, 2, \ldots$ sont représentés par les ensembles de sommets $V_1, V_2, \ldots$ de V :

– $k_0 = 0$, $l_0 = 0$.
– $k_i = k_{i-1} + 2\delta^+(i)$ ($\delta^+(i)$ est le degré sortant du sommet i de V_{clique}.
– $l_i = l_{i-1} + 2\delta^-(i)$ ($\delta^-(i)$ est le degré entrant du sommet i de V_{clique}.
– $V_i = \{(n-k_i, 0), (n-k_i+1, 0) \ldots (n-k_{i-1}-2, 0)\} \bigcup \{(0, n-l_i), (0, n-l_i+1) \ldots (0, n-l_{i-1}-2)\}$.

Chaque arc (A, B) de E_{clique} est représenté par une colonne C_{AB} et une ligne L_{AB} de la double grille. Une extrémité de la colonne fait partie de l'ensemble V_A, et une extrémité de la ligne fait partie de l'ensemble V_B.

– on suppose que A est le i-ième sommet, que B le j-ème sommet, que B est le j'-ième voisin sortant de A et que A est le i'-ième voisin entrant de B.
– $x_{AB} = k_{i-1} - 2j'$, $y_{AB} = l_{j-1} - 2i'$) et $v_{AB} = (x_{AB}, y_{AB} + 1)$
– $C_{AB} = \{(k_{i-1} - 2j', 2y) | 2y_{AB} \leq 2x \leq 2n\}$.
– $L_{AB} = \{(2x, l_{j-1} - 2i') | 2x_{AB} \leq 2x \leq 2n\}$.
– $x_{AA} = k_{i-1} - 3$, $y_{AA} = l_{i-1} - 3$
– $C_A = \{(k_{i-1} - 3, 2x) | 2yAA \leq 2y \leq 2n\}$.
– $L_A = \{(x, l_{i-1} - 3, 2x) | 2x_{AA} \leq 2x \leq 2n\}$.

Les sommets de la double grille sont activés de la manière suivante :

– durant l'intervalle $[0, 1]$, tous les sommets de V sont activés, excepté ceux appartenant à $\cup_{A \in V_{clique}} V_A$.
– durant l'intervalle $[2, 3]$, sont activés les sommets de $(\cup_{A \in V_{clique}} V_A) \bigcup (\cup_{A \in V_{clique}} C_A) \bigcup (\cup_{A \in V_{clique}} L_A)$
– durant l'intervalle $[4, 5]$ sont activés les sommets de $\cup_{AB \in E_{clique}} C_{AB}$.
– durant l'intervalle $[6, 7]$ sont activés les sommets de $(\cup_{AB \in E_{clique}} L_{AB}) \bigcup (\cup_{AB \in E_{clique}} \{v_{AB}\})$.
– durant l'intervalle $[8, 9]$, sont activés les sommets de $(\cup_{A \in V_{clique}} V_A) \bigcup (\cup_{A \in V_{clique}} C_A) \bigcup (\cup_{A \in V_{clique}} L_A)$.
– durant l'intervalle $[10, 11]$, tous les sommets de V sont activés, excepté ceux appartenant à $\cup_{A \in V_{clique}} V_A$.

Les sommets de $\mathcal{G}$ n'appartenant à aucun des V_A sont tous reliés entre eux durant les intervalles $[0, 1]$ et $[10, 11]$. Les sommets de V_A sont reliés entre eux (de façon séparée pour chaque $A \in V_{clique}$) aux intervalles $[2, 3]$ et $[8, 9]$. Les intervalles $[4, 5]$ et $[6, 7]$ simulent les connexions de G_{clique}. Une composantes connexe fermée de $\mathcal{G}$ correspond une clique maximale de G_{clique} et inversement. Le problème des composantes connexes est donc NP-complet même lorsque le graphe sous-jacent de $\mathcal{G}$ est une double grille. □

6.4 Calcul des composantes connexes sur les arbres

Nous allons à présent étudier le cas particulier des graphes évolutifs $\mathcal{G}$ dont le graphe sous-jacent est un arbre. Nous allons d'abord étudier le cas où le graphe sous-jacent est une chaîne, puis le cas où le graphe sous-jacent est une étoile, et enfin le cas plus général des arbres.

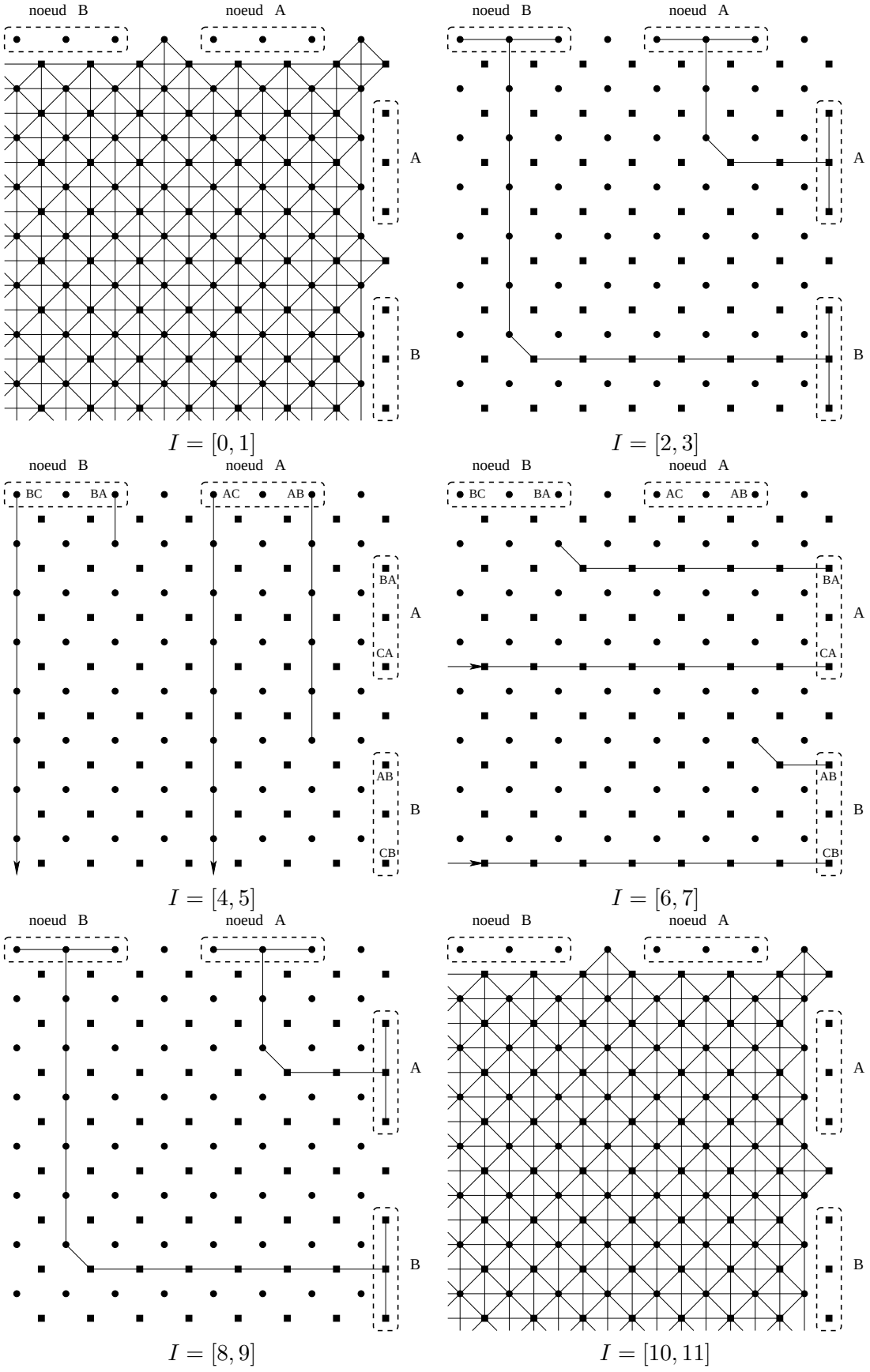


FIG. 6.6 – Graphe évolutif sur une double grille $(12, 12)$ représentant le graphe complet à 3 sommets $\{A, B, C\}$. Seul le premier quartier $(8, 8)$ de la double grille est représenté. La figure représente son évolution sur les intervalles $[0, 1]$, $[2, 3]$, $[4, 5]$, $[6, 7]$, $[8, 9]$ et $[10.11]$.

6.4.1 Cas particulier de la chaîne

Lorsque le graphe sous-jacent de $\mathcal{G}$ est une chaîne G, le calcul des composantes connexes est assez simple. En effet, à partir d'une extrémité de la chaîne x, on peut considérer un arbre des trajets au plus tôt depuis x : il s'agit d'une sous-chaîne de G contenant x. De façon similaire, l'ensemble des sommets pour lesquels il existe un trajet vers x est une sous-chaîne de G. On peut donc calculer facilement la plus grande composante contenant x (voir Figure 6.7). Pour calculer

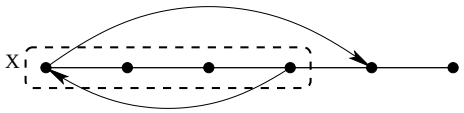


FIG. 6.7 – Composante connexe contenant l'extrémité x d'une chaîne. Les flèches représentent les trajets les plus longs possibles partant et arrivant à x.

une plus grande composante connexe, il suffit de calculer la plus grande composante contenant une extrémité de la chaîne, puis de recommencer à partir du sommet suivant, et ainsi de suite.

Algorithme 10 (Composante connexe maximale dans la chaîne)

Complexité: $O(N^2 \log N)$

Entrée: un graphe évolutif $\mathcal{G}$ dont le graphe sous-jacent G est une chaîne

Sortie: une plus grande composante connexe $\mathcal{C}$ de $\mathcal{G}$

Variables:

- une relation *père*, $fils$ sur les sommets de la chaîne
- pour tout $x \in V$ un ensemble $maxCC(x)$ qui contiendra la plus grande composante connexe fermée dans les descendants de x
- pour tout sommet $x \in V$, un ensemble $S(x)$ de composantes connexes fermées T annotées $(n(T), Ead(T), Ldd(T))$, où T est une composante connexe fermée, $n(T)$ est le cardinal de T, $Ead(T)$ son arrivée au plus tôt sur x, et $Ldd(T)$ est sa dernière date de départ possible depuis x.

Début

1. définir la relation *père*, $fils$ sur toute la chaîne
2. trouver le début y de la chaîne (y n'a pas de fils)
3. soit $T = \{y\}$, $n(T) = 1$, $EadT = -\infty$ et $LddT = +\infty$. $S(y) = \{T\}$
4. tant que y a un père, faire
   (a) soit x le père de y
   (b) soit $T = \{x\}$, $n(T) = 1$, $Ead(T) = -\infty$ et $Ldd(T) = +\infty$. $S(y) \leftarrow \{T\}$
   (c) pour tout T dans $S(x)$ faire
      i. soit $T' \leftarrow T \bigcup \{x\}$
      ii. $n(T') \leftarrow n(T) + 1$
      iii. $Ead(T') \leftarrow Q((y, x), Ead(T))$

iv. $Ldd(T') \leftarrow Q^{-}1((x,y).Ldd(T))$

v. si $Ead(T') \neq +\infty$ et $Ldd(T') \neq -\infty$ alors $S(x) \leftarrow S(x) \bigcup T'$

(d) calculer $maxCC(x)$ parmi $S(x)$ et $maxCC(y)$

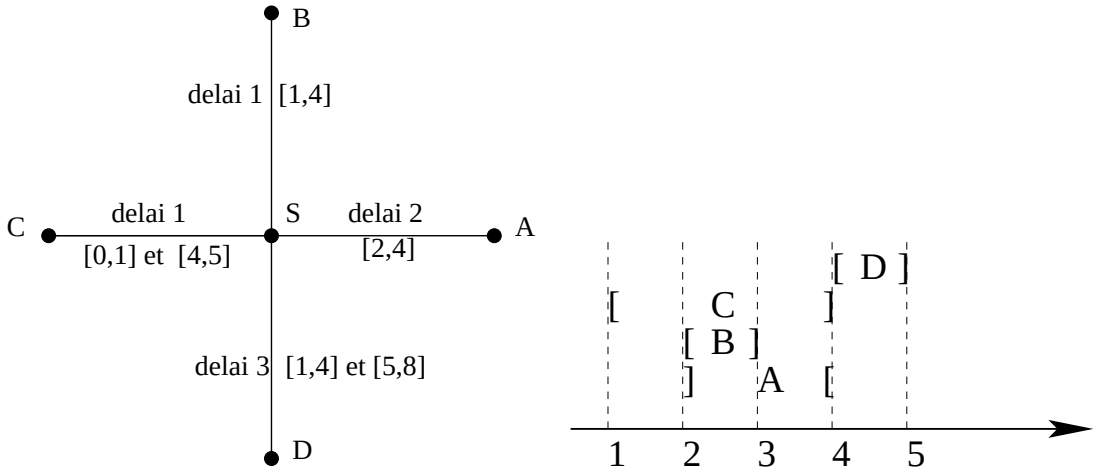
(e) $y \leftarrow x$

5. Résultat : $maxCC(y)$

Fin

6.4.2 Cas particulier de l'étoile

Nous étudions ici le cas où le graphe sous-jacent G est une étoile, de sommet central s. Dans ce cas, les composantes connexes maximales de plus de deux sommets contiennent nécessairement s. Chaque feuille x de l'étoile est reliée par une seule arête au sommet s. Pour simplifier la suite, nous ne considérons que les feuilles x de l'étoile telles que xRs et sRx. Nous notons d_x et a_x deux dates calculées de telle manière que : un trajet de x vers s arrive au plus tôt en s à la date a_x, et un trajet de s vers x peut partir au plus tard de s à la date d_x. Avec ces notations, pour toute feuilles x et y nous avons $xRy \Leftrightarrow a_x \leq d_y$. Ceci implique que pour toute composante connexe $\mathcal{C}$, il existe au plus une feuille z dans la composante telle que $d_z < a_z$. En effet, les autres feuilles x de la composante doivent vérifier $a_x \leq d_z$ et $d_x \geq a_z$, d'où $a_x \leq d_x$. Si on excepte les feuilles z telles que $d_z < a_z$, deux feuilles x et y vérifient xRy et yRx si et seulement si $[a_x, d_x] \cap [a_y, d_y] \neq \emptyset$. Maximiser la taille d'une composante connexe ne contenant s et ne contenant pas de feuilles z telles que $d_z < a_z$ revient à trouver une date t qui maximise le nombre d'intervalles $[d_x, a_x]$ tels que $t \in [d_x, a_x]$ (voir Figure 6.8).



(a) Graphe évolutif. (b) Intervalles de temps.

FIG. 6.8 – (a) Graphe évolutif à 5 sommets dont le graphe sous-jacent est une étoile. (b) Intervalles de temps durant lesquels le sommet s peut envoyer et recevoir des messages depuis et vers les feuilles A, B, C, et D. Les deux composantes connexes maximales sont $\{S, A, C\}$ et $\{S, B, C\}$.

Algorithme 11 (Composante connexe maximale dans l'étoile)

Complexité: $O(N \times (\log(N) + \log(\delta)))$

Entrée: un graphe évolutif $\mathcal{G}$ dont le graphe sous-jacent est une étoile

Sortie: une plus grande composante connexe $\mathcal{C}$ de $\mathcal{G}$

Variables:

- **constantes après l'initialisation**
  - le centre de l'étoile $s \in V$
  - trois ensembles de sommets $Feuilles, Inverse, Ignore \subset V$
  - deux dates $a_x, d_x \in \mathbb{T}$ pour chaque sommet $x \neq s$
  - une suite S d'évènements $(t$,sens$, x)$ avec $t \in \mathbb{T}$, sens$\in$ {arrivée,départ} et $x \in V$
- une date $t \in \mathbb{T}$
- un sommet $x \in V$
- sens $\in$ {arrivée,départ}
- un compteur $c \in \mathbb{N}$
- un ensemble de sommets $Z \subset V$
- **variables relatives à la composante connexe la plus grande**
  - une date $t_{max} \in \mathbb{T}$
  - un entier $c_{max} \in \mathbb{N}$
  - un ensemble de sommets $Z_{max} \subset V$

Début

1. initialisation :
   (a) trouver le centre de l'étoile s
   (b) $F \leftarrow \emptyset$, $Z \leftarrow \emptyset$, $S \leftarrow \emptyset$
   (c) pour chaque sommet $x \neq s$,
      i. calculer la première date d'arrivée possible a_x en s
      ii. calculer la dernière date de départ possible d_x depuis s
      iii. si $a_x = +\infty$ ou $d_x = -\infty$, alors $Ignore \leftarrow Ignore \bigcup \{x\}$
      iv. sinon, si $a_x \leq d_x$ alors $Feuilles \leftarrow Feuilles \bigcup \{x\}$
      v. sinon, $Inverse \leftarrow Inverse \bigcup \{x\}$
      vi. si $x \notin Ignore$, alors $S \leftarrow S : (a_x$,arrivée$, x)$ et $S \leftarrow S : (d_x$,départ$, x)$
   (d) trier la suite S par ordre croissant suivant la relation :
   - $t < t' \Rightarrow (t$,sens$, x) < (t'$,sens$', x')$
   - à date égale, $(t$,arrivée$, x) < (t$,départ$, y)$
   (e) $c \leftarrow 1$, $c_{max} \leftarrow 1$
2. pour $(t$,sens$, x)$ du début à la fin de S, faire
   (a) si $x \in Feuille$ et sens=départ alors
      i. $Z \leftarrow \emptyset$
      ii. $c \leftarrow c - 1$
   (b) si $x \in Feuille$ et sens=arrivée alors
      i. $Z \leftarrow \emptyset$
      ii. $c \leftarrow c + 1$
      iii. si $c > c_{max}$ alors $c_{max} \leftarrow c$, $t_{max} \leftarrow T$ et $Z_{max} \leftarrow \emptyset$

   (c) si $x \in Inverse$ et sens=départ alors $Z \leftarrow Z \bigcup \{x\}$
   (d) si $x \in Inverse$ et sens=arrivée et $x \in Z$ alors
      i. $c \leftarrow c + 1$
      ii. si $c > c_{max}$ alors $c_{max} \leftarrow c$, $t_max \leftarrow t$ et $Z_{max} \leftarrow \{x\}$
      iii. $c \leftarrow c - 1$
3. $\mathcal{C} \leftarrow \{s\}$
4. pour tout $x \in Feuille$ faire
   (a) si $a_x \leq t_{max}$ et $d_x \geq t_{max}$ alors $\mathcal{C} \leftarrow \mathcal{C} \bigcup \{x\}$
5. $\mathcal{C} \leftarrow \mathcal{C} \bigcup Z_{max}$

Fin
L'Algorithme 11 que nous proposons calcule les intervalles correspondant aux feuilles de l'étoile, puis calcule la plus grande intersection des intervalles en un seul balayage de l'ensemble $\mathbb{T}$. Sa complexité est $O(N \times (\log(N) + \log(\delta)))$.

6.4.3 Composantes connexes lorsque le graphe sous-jacent est un arbre.

Lorsque le graphe sous-jacent G est un arbre, les trajets élémentaires entre deux sommets x et y empruntent l'unique chemin élémentaire de x vers y dans l'arbre. Ceci a pour conséquence que les composantes connexes maximales de $\mathcal{G}$ sont fermées.

Proposition 16 *Soit $\mathcal{O}$ une composante connexe ouverte de $\mathcal{G}$. Le plus petit sous-arbre T de G contenant $\mathcal{O}$ est une composante connexe fermée.*

Preuve
Remarquons que les feuilles du sous-arbre T sont des sommets de $\mathcal{O}$. Soit x et y deux sommets quelconques de T. Il existe deux feuilles a et b de T et un chemin élémentaire de a vers b passant par x puis y. Comme a et b sont dans $\mathcal{O}$, il existe un trajet de a vers b dans $\mathcal{G}$, passant nécessairement par x puis y. Ainsi, aRb implique aRx, xRy et yRb. De même, bRa implique bRy, yRx et xRa. Nous savons donc que pour toute paire de sommets x,y de T les relations xRy et yRx. T forme par conséquent une composante connexe fermée de $\mathcal{G}$ (voir Figure 6.9). □

Algorithme 12 (Composante connexe maximale dans l'arbre)

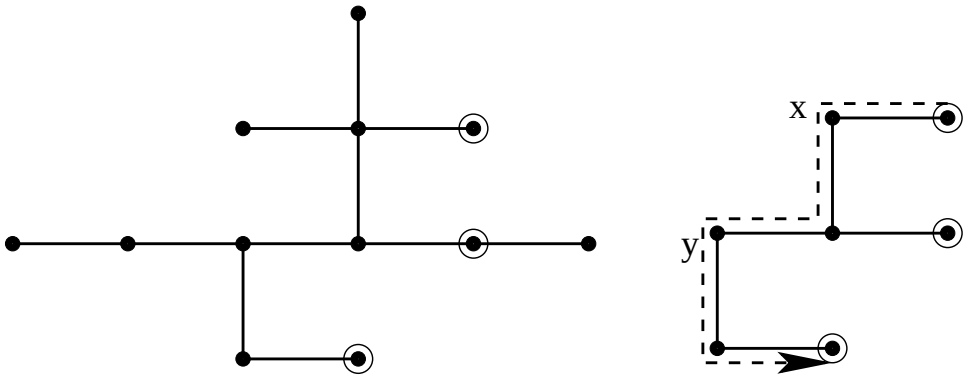
Complexité: $O(N^3 \log N log\delta)$

Entrée: un graphe évolutif $\mathcal{G}$ dont le graphe sous-jacent G est un arbre

Sortie: la composante connexe fermée maximale $maxCC$ de $\mathcal{G}$

Variables:

- une relation *père*, $fils$ sur les sommets de la chaîne
- pour tout sommet x, deux ensembles AD et DD de dates de $\mathbb{T}$
- pour tout sommet x, une matrice $M[ad, dd](x)$ avec $ad \in AD(x)$ et $dd \in DD(x)$ de composantes connexes fermées.



Graphe sous-jacent, 3 sommets dans $\mathcal{O}$. Arbre induit par $\mathcal{O}$, xRy.

FIG. 6.9 – Graphe évolutif dont le graphe sous-jacent est un arbre. Les 3 sommets entourés font partie d'une même composante connexe ouverte $\mathcal{O}$. Le sous-arbre induit par ces 3 sommets forme une composante connexe fermée.

Début

1. pour tout $x \in V$, initialiser
   (a) $AD(x) \leftarrow \{-\infty\}$
   (b) $DD(x) \leftarrow \{+\infty\}$
   (c) $M[-\infty, +\infty](x) \leftarrow \{x\}$
2. choisir un sommet quelconque comme racine de l'arbre
3. inférer la relation *père*, $fils$
4. exécuter l'Algorithme 13 sur la racine de l'arbre
5. $maxCC = \max_{x \in V, ad \in AD(x), dd \in DD(x)} M[ad, dd](x)$

Fin

Algorithme 13 (calcul récursif sur les noeuds de l'arbre)

Complexité: $O(N^2(\log\ N + \log\ \delta) \times nb(x))$ où $nb(x)$ est le nombre de fils de x

Entrée: un sommet x de l'arbre

Sortie: mise à jour des ensembles $AD(x), DD(x)$, et de la matrice $M(x)$

Variables:

- deux sommets y, z
- quatre dates $ad, ad', dd, dd' \in \mathbb{T}$
- deux composantes connexes fermées T et $T_{\max}$

Début

1. exécuter l'Algorithme 13 pour tout les fils y de x
2. si x n'a pas de fils, terminer.

3. pour tout fils y de x faire
   (a) pour chaque date ad dans $AD(y)$ faire
      i. $ad' \leftarrow Q((y,x), ad(T))$
      ii. $AD(x) \leftarrow AD(x) \bigcup \{ad'\}$
   (b) retirer $+\infty$ de $AD(x)$
   (c) pour chaque dd dans $DD(y)$ faire
      i. $dd' \leftarrow Q^{-}1((x,y), dd(T))$
      ii. $DD(x) \leftarrow DD(x) \bigcup \{dd'\}$
   (d) retirer $-\infty$ de $DD(x)$.
4. pour tout $ad \in AD(x)$ et $dd \in DD(x)$ tels que $ad \leq dd$ faire
   (a) $T \leftarrow \{x\}$
   (b) pour tout fils y de x faire
      i. soit $ad' = Q^{-1}((y,x), ad)$ et $dd' = Q((x,y), dd)$.
      ii. si $ad' \neq -\infty$ et $dd \neq +\infty$ alors
         A. soit ad_y la plus grande date de $AD(y)$ telle que $ad_y \leq ad$.
         B. soit dd_y la plus petite date de $DD(y)$ telle que $dd_y \geq dd$.
         C. $T \leftarrow T \bigcup M[ad_y, dd_y](y)$.
   (c) $M[ad, dd](x) \leftarrow T$.
5. pour tout $ad \in AD(x)$ et $dd \in DD(x)$ tels que $ad > dd$ faire
   (a) soit ad_x la plus grande date de $AD(x)$ telle que $ad_x \leq dd$.
   (b) soit dd_x la plus petite date de $DD(x)$ telle que $dd_x \geq ad$.
   (c) $T_{\max} \leftarrow M[ad_x, dd_x](x)$.
   (d) pour tout fils y de x, faire
      i. soit $ad' = Q^{-1}((y,x), ad)$ et $dd' = Q((x,y), dd)$
      ii. si $ad' \neq -\infty$ et $dd' \neq +\infty$, alors
         A. soit ad_y la plus grande date de $AD(y)$ telle que $ad_y \leq ad'$
         B. soit dd_y la plus petite date de $DD(y)$ telle que $dd_y \geq dd'$
         C. $T \leftarrow M[ad_x, dd_x](x) \bigcup [ad_y, dd_y](y)$
         D. $T_{\max} \leftarrow \max\{T, T_{\max}\}$.
   (e) $M[ad, dd](x) \leftarrow T_{max}$.

Fin

L'algorithme 13 calcule les ensembles $AD(x)$ (Dates d'arrivée importantes) et $DD(x)$ (Dates de départ importantes) pour pouvoir discrétiser l'ensemble $\mathbb{T}$. Si on prend uniquement on compte les trajets qui arrivent au sommet x après une certaine date $t_a \in \mathbb{T}$ alors la proposition suivante (Proposition 17) montre que l'on peut choisir la date t_a dans $AD(x)$, sans perte de généralité. Les ensembles $AD(x)$ et $DD(x)$ sont suffisamment petits pour être facilement calculés : ils ont la même taille que le sous-arbre enraciné en x. Étant donné un sommet x, une date de départ $t_d \in \mathbb{T}$ et une date d'arrivée $t_a \in \mathbb{T}$, on appelle $S(x, t_a, t_d)$ l'ensemble des sommets y du sous-arbre enraciné en x tel qu'il existe un trajet de x vers y partant à la date $t \geq t_d$ et tel qu'il existe un trajet de y vers x arrivant à la date $t' \leq t_a$.

Proposition 17 (sur les ensembles AD et DD) *Soit x un sommet de V, t_a et t_d deux dates de $\mathbb{T}$. Soit ad la plus grande date de $AD(x)$ telle que $ad \leq t_a$ et dd la plus petite date de $DD(x)$ telle que $dd \geq t_d$. Alors $S(x, t_a, t_d) = S(x, ad, dd)$.*

Preuve
Si il existe un trajet de x vers y partant à la date $t \geq dd$, alors $t \geq t_d$. De même, si $t' \leq ad$ alors $t' \leq t_a$.Aussi, on a $S(x, ad, dd) \subset S(x, t_a, t_d)$. Nous allons prouver l'inégalité opposée par récurrence. Si x est une feuille, alors $AD(x) = \{+\infty\}$ et $DD(x) = \{-\infty\}$ et sont calculés à l'étape 1 de l'Algorithme 12. Pour tout (t_a, t_d), $S(x, t_a, t_d) = \{x\} = S(x, ad, dd)$. Si x a des fils, alors $AD(x)$ et $DD(x)$ sont calculés à l'étape 3 de l'Algorithme 13. De plus, pour tout t_a, t_d, nous savons que

$$S(x, t_a, t_d) = \{x\} \bigcup_{y \ fils \ de \ x} S(y, Q^{-1}((y, x), t_a), Q((x, y), t_d))$$

D'après l'étape 3 de l'Algorithme 13 et si on suppose que la proposition est vérifiée pour tout fils y de x, $S(y, Q^{-1}((y, x), t_a), Q((x, y), t_d)) = S(y, Q^{-1}((y, x), ad), ((x, y), dd))$. Aussi, $S(x, t_a, t_d) = S(x, ad, dd)$. □

Les matrices $M(x)$ calculées par notre algorithme contiennent des composantes connexes fermées. Les deux propositions suivantes montreront que $M[ad, dd](x)$ contiennent une plus grande composante connexe

- contenant x,
- faisant partie du sous-arbre enraciné en x,
- telle que pour tout sommet $y \in M[ad, dd](x)$ il existe un trajet de x vers y partant après la date dd et un trajet de y vers x arrivant avant la date ad.

Le cas plus facile où $ad \leq dd$ est traité en premier.

Proposition 18 (sur $ad \leq dd$) *Soit x un sommet, ad une date de $AD(x)$ et dd une date de $DD(x)$. Si $ad \leq dd$, alors $S(x, ad, dd)$ est une composante connexe fermée et $S(x, ad, dd) = M[ad, dd](x)$.*

Preuve
Soient y et z deux sommets de $S(x, ad, dd)$. Il existe un trajet de y vers x arrivant à la date $t' \leq ad$ et un trajet de x vers z partant à la date $t \leq dd$. Comme $t \leq t'$, il existe un trajet de y vers z. De même, il existe un trajet se x vers y. Ceci prouve que $S(x, ad, dd)$ est bien une composante connexe fermée. Nous allons prouver par récurrence que $S(x, ad, dd) = M[ad, dd](x)$. Si x est une feuille, alors $S(x, \infty, +\infty) = \{x\} = M[-\infty, +\infty](x)$ (calculé à l'étape 1 de l'Algorithme 12). Si x a des fils, alors

$$S(x, ad, dd) = \{x\} \bigcup_{y \ fils \ de \ x} S(y, Q^{-1}((y, x), ad), Q((x, y), dd))$$

$M[ad, dd](x)$ est calculé de cette façon à l'étape 4 de l'Algorithme 13. □

Proposition 19 (sur $M(x)$) *Soit x un sommet du graphe évolutif, ad une date de $AD(x)$ et dd une date de $DD(x)$. Alors, $M(x)$ contient une plus grande composante connexe incluse dans $S(x, -\infty, +\infty)$.*

Preuve
Soit T la plus grande composante connexe incluse dans $S(x, -\infty, +\infty)$. Soit T_1, T_2, ... les sous-arbres maximum de $T - \{x\}$. D'après la Proposition 16 chaque sous-arbre T_i est une composante connexe fermée de $\mathcal{G}$. On appelle $Ead_i = \max_{y \in T_i}\{Ead(y, x)\}$ et $Ldd_i = \min_{y \in T_i}\{Ldd(x, y)\}$. D'après la Proposition 17, pour tout i, $Ead_i \in AD(x)$ et $ldd_i \in DD(x)$. Comme tous les trajets

d'un sommet de T_i vers un sommet de T_j ($j \neq i$) passent par x, on a pour tout $i \neq j$, $Ead_i \leq Ldd_j$. On en déduit qu'il existe au plus un entier i tel que $Ead_i > Ldd_i$.

(a) Si un tel entier existe, pour tout $j \neq i$ on a $Ead_j \leq Ldd_j$ et $[Ldd_i, Ead_i] \subset [Ead_j, Ldd_j]$. T est l'union des composantes connexes $M[Ldd_i, Ead_i](\mathrm{x})$ et T_i avec $T_i = M[Q((y,x), Ead_i), Q^{-1}((x,y), Ldd_i)$. T est calculé ainsi à l'étape 5 de l'Algorithme 13.

(b) Sinon, on a pour tout i $Ead_i \leq Ldd_i$ et $\bigcap_i [Ead_i, Ldd_i] \neq \emptyset$. La Proposition 18 affirme que la plus grande composante connexe T appartient bien à $M[ad, dd](x)$ pour un certain couple $(ad, dd) = (Ead_i, Ldd_i)$.

□

Théorème 26 *L'Algorithme 12 calcule une composante connexe maximale de $\mathcal{G}$.*

Preuve
Comme le graphe sous-jacent G est un arbre, une composante connexe maximale est un sous-arbre T. Soit x le noeud de T le plus proche de la racine de G : T appartient à $S(x, -\infty, +\infty)$. D'après la Proposition 19, $M(x)$ contient une composante connexe de même taille que T. L'algorithme 12 calcule bien une plus grande composante connexe. □

Théorème 27 *l'Algorithme 12 se termine en $O(N^3 \times (\log N + \log \delta))$ pas de calcul.*

Preuve
Établir la relation père et fils se fait en $O(N \log N)$ étapes. L'Algorithme 13 est appelé récursivement sur tous les sommets de G, de la racine vers les feuilles de G. On peut voir que les ensembles $AD(x)$ et $DD(x)$ ont la même cardinalité que le sous-arbre de G enraciné en x. Calculer tous ces ensembles coûte $O(N^2 \times \log \delta)$ pas de calcul. Ensuite, pour chaque sommet x, on peut voir que la matrice $M(x)$ est de taille $(N \times N)$. Son calcul prend $O(N^2 \times nb(x) \times (\log N + \log \delta))$, où $nb(x)$ est le nombre de fils de x. Le coût total du calcul des matrices est $O(N^3 \times (\log N + \log \delta))$ pas de calcul. Trouver la composante connexe maximum parmi ces matrices ne prend que $O(N^3)$ pas de calcul. □

6.5 Conclusion

Nous avons montré que la relation de connexité est suffisamment arbitraire dans les graphes évolutifs pour que calculer une composante connexe maximale revienne à résoudre le problème CLIQUE réputé NP-difficile, même lorsque la dynamicité du graphe est fixée à 11 et que le graphe sous-jacent est une double grille. D'autre part, lorsque la topologie d'un graphe évolutif est encore plus contrainte, comme dans le cas où l'ensemble des liens possibles forme un arbre, le problème devient tractable. La question que nous nous posons est de savoir quelle est l'influence des contraintes topologiques sur la relation de connexité. Nous nous demandons quelle est la classe de complexité du problème des composantes connexes maximales dans le cas où la largeur arborescente du graphe support est petite, et dans le cas où le graphe support est planaire.

Chapitre 7

Flots dans les graphes évolutifs

Sommaire

7.1 Introduction

Les problèmes de flot sur les graphes statiques ont été et sont toujours un sujet d'études depuis de longues années et représentent un domaine de l'optimisation combinatoire où les résultats positifs sont nombreux. Comme on peut s'y attendre, les problèmes de flots dynamiques sont significativement plus durs à traiter. Premièrement, toute une série de problèmes d'optimisation possibles n'ont pas de sens dans les réseaux statiques. Par exemple, on peut vouloir minimiser le temps nécessaire pour envoyer une certaine quantité de flot entre deux noeuds du réseau. Deuxièmement, les problèmes dynamiques sont en eux-mêmes plus difficiles à résoudre que les problèmes statiques. Plusieurs d'entre eux sont NP-difficiles. Les différences qu'ils présentent avec les problèmes statiques classiques ont rendu nécessaire l'élaboration de nouvelles techniques, bien que de nombreuses méthodes de résolution transforment in fine les problèmes dynamiques en problèmes statiques.

Dans ce chapitre, nous allons tout d'abord présenter la modélisation générale des flots dynamiques et leur modélisation dans le cadre des graphes évolutifs, suivi d'une récapitulation des principaux problèmes de flots dynamiques. Nous aborderons ensuite la notion de graphe espace-temps introduite par Ford et Fulkerson [34, 35]. Enfin nous étudierons le problème du flot maximum dans le cadre du modèle des graphes évolutifs. Il s'agit d'un problème qui n'a jamais été résolu lorsque des changements de topologie arbitraire interviennent sur un réseau. Ce problème est NP-difficile lorsque le réseau a une capacité de stockage faible sur les nœuds. Nous généraliserons un résultat de Ogier [77] en proposant une résolution polynomiale du problème lorsque l'ensemble des *dates remarquables*[1] est borné.

7.2 Modélisation du flot dans un contexte dynamique

D'un point de vue terminologique, certains auteurs (par exemple Fleischer [29]) avancent avec raison que le terme 'dynamique' (*'dynamic'*) devrait être uniquement utilisé lorsque les paramètres du réseau changent effectivement au cours du temps. Très souvent, dans un problème de flot dynamique, c'est la solution qui dépend du temps, alors que les paramètres du réseau sont constants. Dans ce cas, on préférera utiliser le terme de 'flot au cours du temps' (*flow over time*) ou bien 'flot dépendant du temps' (*time-dependent flow*). En revanche, il est toujours accepté qu'un problème est appelé dynamique si un ou plusieurs des paramètres du réseau est une fonction du temps (ce point de vue est défendu dans [85]). Cette notion recouvre tous les autres problèmes, depuis ceux où les paramètres changent constamment jusqu'à ceux ou un seul 'accident' peut intervenir.

Dans un flot statique, le flot *entrant* sur un arc est identique au flot qui *traverse* l'arc, car tout ce qui entre dans l'arc en sort immédiatement. Ce n'est plus le cas lorsque le flot est dynamique et qu'il passe du temps à traverser un arc. Une unité de flot entrée précédemment sur un arc peut donc très bien être encore en train de parcourir cet arc. Il y a donc une différence entre la charge d'un arc, c'est à dire la quantité de flot présente sur cet arc, et la quantité de flot entrant sur cet arc. C'est en général cette dernière quantité qui est au centre de notre intérêt. La charge et la quantité entrante sont toutes deux des fonctions du temps. Soit $F_a : \mathbb{T} \to \mathbb{R}_+$ et $f_a : \mathbb{T} \to \mathbb{R}_+$ la charge et la quantité de flot entrant sur un arc et soit $\zeta(a)$ le temps de traversée de cet arc. Ces deux fonctions sont bien sûr reliés, car la charge est l'intégrale du flot entrant, et le flot entrant la dérivée de la charge. Plus formellement, on a dans le cas continu et dans le cas discret :

$$F_a(t) = \int_0^{\zeta_a} f_a(t-\theta)d\theta \quad \text{et} \quad F_a(t) = \sum_{\theta=0}^{\zeta_a - 1} f_a(t-\theta)$$

Il s'agit bien sûr d'une convention de savoir si l'unité flot entrant dans l'arc a à $t - \zeta_a$ est déjà arrivé au temps t, ou s'il est sur le point d'arriver. Dans le cas discret, on considère que l'unité est déjà arrivée. Dans le cas continu, l'*unité* dF_a de flot considérée est infiniment petite ce qui fait que les deux conventions possibles sont strictement équivalentes. Pour illustrer ceci, une route à sens unique ayant 3 voies, et ayant un temps de traversée équivalent à 4 fois le temps d'entrée d'une voiture (ce qui fait autour de 8 et 2 secondes, respectivement), pourra contenir jusqu'à 12 voitures en transit dans des conditions normales.

[1] Définition 33 page 132.

Une des plus importantes mesures d'un réseau de flot dynamique est le débit (*throughput*), c'est à dire la quantité de flot qui peut être envoyé de la source vers le puits durant le temps imparti. Voici un exemple très simple : "combien de flot peut arriver au cours de l'intervalle $[0, 2\zeta_a)$ depuis le début de l'arc a jusqu'à son autre extrémité, alors que la capacité de a est u_a et que son temps de traversée est ζ_a ?" Remarquons que tout flot entrant dans l'arc après la date ζ_a n'a aucune chance d'arriver avant la date $2\zeta_a$. Par contre, les dates de départ situées dans $[0, delay_a)$ sont parfaitement possibles. Le flot maximum sera atteint un utilisant toute la capacité de a lorsque c'est possible. Le débit est également défini comme l'intégrale de la quantité de flot entrant dans a au cours du temps imparti. On a donc :

$$\int_0^{2\zeta_a} f_a(t)dt = \int_0^{\zeta_a} u_a dt + 0 = \zeta_a \times u_a$$

Si le temps est discret, et que nous remplaçons les intégrales par des sommes, nous obtenons également $\zeta_a \times u_a$ comme résultat.

Enfin, il existe une *transformation naturelle* du temps discret vers le temps continu. Si $f_a(t)$ est la quantité de flot en temps discret entrant dans l'arc a au temps t ($t = 0$ ou $t = 1$ ou $t = 2\ldots$), on peut définir la quantité de flot en temps continu $g_a(t)$ comme la fonction constante par morceaux, définie par $g_a(t) = f_a(\lfloor t \rfloor)$. Dans le cas où la capacité des arcs ne change pas au cours du temps, cette transformation conserve la validité des flots, et de plus, la quantité de flot ćhangées au cours des intervalles $[t, t + \theta)$, où t et θ sont entiers, est la même dans le cas discret avec f_a et dans le cas continu avec g_a. Fleischer et Tardos [25] ont montré que cette transformation préserve également l'optimalité des solutions dans de nombreux problèmes. Ceci montre que la version 'temps continu' de ces problèmes n'est pas plus difficile que la version 'temps discret', car des solutions pour la première version peuvent s'obtenir à l'aide de la deuxième.

7.2.1 Équations relatives aux flots dynamiques

Nous allons à présent définir les équations de conservation du flot dans un contexte dynamique. Si on suppose que les paramètres du réseau restent constants et que le temps est discret, les équations suivantes définissent l'ensemble des flots dynamiques possibles.

$$\sum_{t=0}^{T} \left(\sum_{a \in \delta^+(s)} f_a(t - \zeta_a) - \sum_{a \in \delta^-(s)} f_a(t) \right) = -v \tag{7.1}$$

$$\sum_{t=0}^{\theta} \left(\sum_{a \in \delta^+(x)} f_a(t - \zeta_a) - \sum_{a \in \delta^-(x)} f_a(t) \right) \geq 0 \quad \forall x \in V - \{s, d\}, \forall \theta \in \{0, 1, \ldots, T-1\} \tag{7.2}$$

$$\sum_{t=0}^{T} \left(\sum_{a \in \delta^+(x)} f_a(t - \zeta_a) - \sum_{a \in \delta^-(x)} f_a(t) \right) = 0 \quad \forall x \in V - \{s, d\} \tag{7.3}$$

$$\sum_{t=0}^{T} \left(\sum_{a \in \delta^+(s)} f_a(t - \zeta_a) - \sum_{a \in \delta^-(s)} f_a(t) \right) = v \tag{7.4}$$

$$0 \leq f_a(t) \leq u_a \qquad \forall a \in E, \forall t \in \{0, 1, \ldots, T\} \tag{7.5}$$

Lorsque le délai des arcs varient au cours du temps, il faut remplacer ζ_a par $\zeta_a(t)$ dans les équations que l'on vient de définir. Dans l'ensemble de la littérature, on fait l'hypothèse que les liens sont de type FIFO[2] ce qui se traduit par la condition $\forall t_1 \leq t_2, t_1 + \zeta_a(t_1) \leq t_2 + \zeta_a(t_2)$. La contrainte de capacité κ sur les arcs s'exprimera par l'équation $\forall\ a \in E,\ \forall\ t \in \mathbb{T},\ f_a(t) \leq \kappa_a(t)$. Une contrainte de stockage s sur les nœuds s'exprimera par l'équation

$$\forall x \in V - \{s,d\}, \forall \theta \in \{0,1,\ldots,T-1\}, \sum_{t=0}^{\theta} \left(\sum_{a \in \delta^+(x)} f_a(t-\zeta_a) - \sum_{a \in \delta^-(x)} f_a(t) \right) \leq s_x(\theta)$$

Dans le cas du temps continu, les sommes sur le temps sont remplacées par des intégrales. Par exemple, dans un modèle à temps continu avec des capacités variables sur les arcs, et des capacités de stockage variables, nous avons les équations suivantes :

$$\int_{t=0}^{T} \left(\sum_{a \in \delta^+(s)} f_a(t-\zeta_a) - \sum_{a \in \delta^-(s)} f_a(t) \right) dt = -v \tag{7.6}$$

$$0 \leq \int_{t=0}^{\theta} \left(\sum_{a \in \delta^+(x)} f_a(t-\zeta_a) - \sum_{a \in \delta^-(x)} f_a(t) \right) dt \leq s_x(\theta) \quad \forall x \in V - \{s,d\}, \quad \forall \theta \in [0,T) \tag{7.7}$$

$$\int_{t=0}^{T} \left(\sum_{a \in \delta^+(x)} f_a(t-\zeta_a) - \sum_{a \in \delta^-(x)} f_a(t) \right) dt = 0 \quad \forall x \in V - \{s,d\} \tag{7.8}$$

$$\int_{t=0}^{T} \left(\sum_{a \in \delta^+(s)} f_a(t-\zeta_a) - \sum_{a \in \delta^-(s)} f_a(t) \right) dt = v \tag{7.9}$$

$$0 \leq f_a(t) \leq u_a(t) \qquad \forall a \in E, \qquad \forall t \in [0,t) \tag{7.10}$$

Il faut remarquer que la taille de l'ensemble de ces équations n'est pas polynomial dans la taille des données. Lorsque le temps est discret, le nombre d'équations est supérieur à T, alors que la taille du temps est $\log T$. Lorsque le temps est continu, le nombre d'équations est tout simplement infini. Aussi, résoudre des problèmes de flots dynamiques simplement par programmation linéaire n'est pas une méthode polynomiale.

7.2.2 Modélisation du flot dans les graphes évolutifs

Dans le cadre du modèle des graphes évolutifs, nous avons besoin d'introduire la notion de capacité sur les arcs.

Notation 10 *Soit $\mathcal{G}$ un graphe évolutif et soit $G = (V,E)$ son graphe sous-jacent. Nous appelons $\kappa : E \to \mathbb{R}_+$ la fonction qui indique la capacité des arcs.*

Nous définissons la fonction de capacité comme une constante au cours du temps, de la même manière que nous avions défini la fonction de délai ζ. Cette capacité pourrait changer de manière

[2]First In First Out. Cela signifie que les messages sont émis et reçus dans le même ordre.

discrète au prix de la disparition et de l'apparition d'un arc. Il faut noter également que cette capacité s'apparente à une bande passante, c'est à dire qu'elle limite le débit du flot entrant dans chaque arc à tout instant.

À priori, nous pouvons tout à fait utiliser les équations de flot que nous avons vu précédemment. Le délai des arcs est constant, et la capacité des arcs est constante par morceaux :
- $\zeta_a(t) = \zeta(a)$ et
- $\kappa_a(t) = \kappa(a)$ si a est présent durant $[t, t + \zeta(a)]$, $\kappa_a(t) = 0$ sinon.

Un flot se définit alors naturellement comme une fonction du temps sur les arcs.

Cette approche fonctionnelle ne permet cependant pas de mesurer la complexité combinatoire de la donnée du flot, de la même manière qu'une approche fonctionnelle sur le délai ou la capacité des liens ne permet pas de mesurer la complexité de la donnée d'un réseau dynamique. Pour maîtriser la donnée du flot, il est nécessaire que la fonction de flot $f : (E \times \mathbb{T}) \to \mathbb{R}_+$ soit constante par morceaux, et que le nombre de morceaux soit polynomial dans la donnée de $\mathcal{G}$. Cette contrainte nécessaire sur le flot f pose malgré tout de sérieux problèmes : d'abord, la taille de la donnée de f n'est pas définitivement fixée, et il faudra toujours s'attacher à la contrôler. Ensuite, suivant l'objectif recherché, il n'est pas certain qu'une des solutions optimale soit de taille polynomiale en la donnée de $\mathcal{G}$.

Pour compléter notre modélisation, on peut s'attacher à ajouter une contrainte de stockage s sur les sommets de $\mathcal{G}$. Cette contrainte rend tout problème de flot NP-difficile.

Théorème 28 (Flot dans un graphe évolutif, avec contrainte de stockage) *Soit $\mathcal{G}$ un graphe évolutif muni d'une capacité unitaire. Soit s et d deux sommets de $\mathcal{G}$. Trouver un flot f de s vers d sur $\mathcal{G}$ qui vérifie les équations*

$$\forall x \in V - \{s, d\}, \forall \theta \in \mathbb{T}, \int_{t=0}^{\theta} \left(\sum_{a \in \delta^+(x)} f_a(t - \zeta_a) - \sum_{a \in \delta^-(x)} f_a(t) \right) = 0$$

est un problème NP-difficile.

Preuve
Nous allons faire une réduction du problème *Subset Sum*. Soit U un ensemble fini, $c : U \to \mathbb{N}_+$ une fonction de coût sur U, et c_{max} un entier naturel. Le problème du *Subset Sum* consiste à trouver un sous-ensemble U' de U telle que la somme

$$S = \sum_{u \in U'} c(u)$$

soit égale à c_{max} [66].

Nous construisons un graphe évolutif à partir de l'instance U, c, c_{max}. Numérotons les éléments de U : $u_1, u_2, \dots u_k$. Nous allons construire un graphe évolutif $\mathcal{G}$ de dynamicité 1, avec $3k + 2$ sommets et $4k + 1$ arcs. Nous nommons les sommets de V s, $x_0, x_1, \dots x_k$, $a_1, a_2, \dots a_k$, $b_1, b_2, \dots b_k$ et pour finir, d. Les arcs de E sont définis comme suit
- l'arc (s, x_0) de délai 1 et présent durant $[0, 1]$,
- $\forall i \in \{1, 2, \dots k\}$, (x_{i-1}, a_i) de délai 1 et présent durant $[0, +\infty[$,

– $\forall i \in \{1, 2, \dots k\}$, (x_{i-1}, b_i) de délai 1 et présent durant $[0, +\infty[$
– $\forall i \in \{1, 2, \dots k\}$, (a_i, x_i) de délai 1 et présent durant $[0, +\infty[$,
– $\forall i \in \{1, 2, \dots k\}$, (b_i, x_i) de délai $1 + c(u_i)$ et présent durant $[0, +\infty[$,
– et enfin, l'arc (x_n, y) de délai 1 et présent durant $[c_{max} + 2k + 1, c_{max} + 2k + 2]$

Le problème posé est de trouver un flot de valeur 1 entre s et d.

Soit $\mathcal{J}$ un trajet de x_0 vers y partant à la date $t = 0$ et n'attendant dans aucun sommet. Pour chaque entier $i \in \{1, 2, \dots k\}$, $\mathcal{J}$ passe soit par le sommet a_i soit par le sommet b_i. Pour pouvoir utiliser l'arc (x_k, y), le trajet est soumis à la contrainte

$$\sum_{(b_i, x_i) \in \mathcal{J}} c(u_i) = s_{max}.$$

Trouver un trajet sans temps d'attente entre s et d revient donc à trouver un sous-ensemble U' tel que

$$S = \sum_{u \in U'} c(u) = c_{max}.$$

□

Cette preuve est très proche de celle du Théorème 21 (page 101). Un problème similaire avait été étudié dans l'article *Shortest path with time constraints on movement and parking* [44]. Lorsque la capacité de stockage n'est pas nulle mais est très inférieure à la quantité de flot que l'on souhaite faire passer, le problème reste NP-difficile. Lorsqu'il n'y a pas de contrainte de stockage sur les nœuds, on peut espérer résoudre le problème du flot maximum de manière polynomiale. À la section 7.5, nous montrerons que sous certaines conditions, le problème de la coupe minimum peut être résolu en temps polynomial.

7.3 État de l'art sur les problèmes de flots dynamiques

La diversité des problèmes que l'on peut chercher à résoudre sur les réseaux dynamiques est bien sûr plus importante que sur les réseaux statiques, notamment parce que les réseaux statiques peuvent être vus comme un cas particulier des réseaux dynamiques. En plus des problèmes classiques de flot (comme le multiflot, les flots de coût minimum, ou bien les problèmes de livraison), il faut considérer des problèmes qui prennent le temps comme fonction objective. Nous allons donc présenter ici quelques uns des principaux problèmes de flots dynamiques. Pour une étude plus détaillée des travaux entrepris sur ces problèmes, nous référons le lecteur aux travaux de Kotnyek [60].

Flot maximum

Ford et Fulkerson [34, 35] ont été les premiers à formuler un modèle simple de flot dynamique. Dans leur modèle le temps est discret et n'apparaît que dans le temps de traversée de chaque arc (qui est constant pour un arc donné). Ce problème est régi par les équations (7.1-7.5). Ford et Fulkerson s'intéressaient au problème du flot dynamique maximum, et ont donné un algorithme polynomial pour le résoudre.

Flot le plus rapide

Le problème du flot le plus rapide (*quickest flow*), est de trouver un flot de valeur v qui se termine le plus tôt possible. Plus formellement, on cherche le T minimum tel qu'il existe un flot de valeur v de la source vers le puits durant l'intervalle $[0, T)$. Le problème du flot le plus rapide est en quelque sorte la réplique du problème du flot maximum, où le temps est fixé où la valeur doit être maximisée. Lorsque le temps est discret, le flot le plus rapide peut être trouvé en effectuant une recherche dichotomique sur le temps. Burkard, Dlaska et Klinz [11] ont proposé un algorithme plus efficace, fortement polynomial, qui est basé sur la méthode de Newton discrète. Lorsque le temps est continu, Fleischer et Tardos [25] ont prouvé que lorsque la valeur du flot est entière, et que les temps de traversée des arcs sont entiers, le temps nécessaire au flot le plus rapide set un nombre rationnel dont le dénominateur est borné par la taille d'une coupe minimum sur le réseau. Ce temps peut donc être également trouvé par recherche dichotomique.

Une autre distinction s'impose, suivant que l'on veut trouver T minimum tel que le flot s'effectue durant $[0, T)$ (*earliest flow*), ou que l'on veut trouver le plus petit intervalle (qui ne commence pas forcément à la date $t = 0$) durant lequel le flot peut s'effectuer (*fastest flow*). On peut être intéressé par l'une ou l'autre solution suivant que ce qui importe est la date d'arrivée, ou bien le temps mis à traverser le réseau. Autrement dit, s'il existe une obstruction temporaire du réseau, il peut être intéressant de retarder le départ jusqu'à ce que cette obstruction soit levée, ou bien accepter une exécution lente mais qui se terminera au plus tôt.

On peut encore donner une autre version de la rapidité, où ce qui importe est de minimiser le temps passé à attendre dans tous les noeuds du réseau. Ce problème à été traité dans le cas où les temps de traversée sont nuls, notamment dans les articles [40, 71, 77, 89].

Flot de coût minimum

Si les coûts d'utilisation des arcs sont arbitraires, on peut se demander quel est le flot de valeur v durant le temps $[0, T)$ qui coûtera le moins cher. Un cas particulier de ce problème est de demander un flot de valeur maximale, et dont le coût est minimum parmi l'ensemble des flots de valeur maximale, ou encore on peut fixer la valeur du flot, vouloir un flot le plus rapide, et minimiser le coût parmi ceux-ci. Klinz et Woeginger [56] ont prouvé que ces deux cas particuliers sont NP-difficiles. Fleischer et Skutella [33] ont présenté des résultats et des algorithmes donnant une solution approchée pour le problème du multiflot de coût minimum.

Flot maximum universel

Un flot qui maximise sa valeur v durant l'intervalle $[0, T)$ n'est pas nécessairement maximum sur une période plus courte.

Un flot qui maximise la valeur transmise de la source vers le puits pour tout intervalle $[0, t)$ avec $t \leq T$, est appelé flot maximum au plus tôt (*earliest maximum flow*). Il faut observer que la littérature n'est pas constante sur le sujet, car les flots au plus tôt sont souvent appelés universellement maximum (voir par exemple [30, 35, 38, 98]).Gale [38] à montré l'existence d'un tel flot lorsque le temps est discret, te lorsque les paramètres du réseau sont une fonction du temps. Le même résultat a été prouvé pour le temps continu par Philpott [83]. Minieka [69] et Wilkinson [98] ont donné des algorithmes pour trouver un flot maximum au plus tôt lorsque les paramètres du

réseau sont constants et lorsque le temps est discret. L'algorithme de Minieka appelle T fois l'algorithme de Ford et Fulkerson, alors que celui de Wilkinson construit une coupe minimum. Les deux algorithmes ont une complexité exponentielle. Hoppe et Tardos [45] en donnent une approximation en temps polynomial. Lorsque le temps est continu, Orda et Rom [80] donnent une construction pour calculer le flot, lorsque les paramètres du réseau sont des fonctions du temps constantes par morceaux.

Un problème presque identique est celui de trouver un flot maximum au plus tard (*latest departure flow*), qui maximise la valeur qui sort de la source pour tout intervalle $[t, T)$. Un flot qui est à la fois au plus tôt et au plus tard est parfois également appelé maximum universel (voir par exemple [25, 80]). Cela fait malheureusement confusion avec les flots au plus tôt, lorsqu'ils sont également appelés universellement maximum. Nous souscrivons au fait de garder la notation universellement maximum pour le seul cas où le flot est à la fois

Terminaux multiples

Supposons que certains des noeuds du réseau sont des terminaux. Un terminal est soit une source, soit un puits. Si on donne une priorité à ces terminaux, et si on veut obtenir un flot qui maximise la quantité échangée dans ces terminaux par ordre de priorité, il s'agit du problème du flot lexicographique maximum (*lexicographically maximum flow*).

Nous pouvons redéfinir le problème de livraison vu dans le cas statique. Le réseau contient certaines unités de flot réparties sur des terminaux, et qui doivent être transférées vers d'autres terminaux durant le temps imparti. Ce problème peut être réduit au problème du flot lexicographique maximum dans le cas où le temps est discret (Minieka [69], Hoppe et Tardos [46]) et dans le cas où le temps est continu (Fleischer et Tardos [25]). Il est à noter que contrairement au cas statique, le problème de livraison n'est pas équivalent au problème du flot maximum, et la raison en est que la capacité des arcs dans un réseau dynamique limite le débit et non la quantité totale de flot qui peut le traverser.

Le problème de livraison la plus rapide (*quickest transshipment*) impose de minimiser le temps mis pour la livraison. Le cas particulier où il n'y a qu'une seule destination, ou qu'un seul point de départ est appelé problème d'évacuation (*evacuation problem*). Ce problème se pose, comme son nom l'indique, pour l'évacuation des bâtiments publics, mais également pour récupérer les informations d'un réseau alors qu'il vient de subir une panne. Hoppe et Tardos [45] donnent un algorithme polynomial pour le problème d'évacuation avec un nombre limité de points de départ. Ces auteurs ont également donné un algorithme polynomial pour le problème de la livraison la plus rapide dans le cas discret, quelle que soit le nombre de terminaux [46]. Fleischer et Tardos [25] on généralisé ce dernier résultat au cas continu.

Dans le cas particulier où le graphe est biparti, et que tout les arcs connectent une source vers un puits, on parle du problème de transport (*transportation problem*). Une compilation extensive des résultats sur ce domaine à été écrite par Bookbinder et Sethi [7]. Il existe aussi une synthèse plus récente et plus succincte [65].

Enfin, le problème de livraison universellement rapide (*universally quickest transshipment*) est de trouver un flot qui minimise les unités qui restent à convoyer à chaque instant. Hajek et Ogier [40] donnent un algorithme polynomial lorsqu'il n'y a qu'une seule destination et que tout les temps de

traversée sont nuls. Fleischer [29] a exhibé un exemple avec deux sources et deux destinations pour lequel une livraison universellement rapide n'existe pas, et a donné un algorithme plus efficace que dans [40] pour le cas où il n'y a qu'une seule destination.

Temps de traversée nuls

Le modèle des flots dynamique avec temps de traversée nulle diffère uniquement du modèle statique par la possibilité de stocker des unités de flot dans les noeuds, ce qui permet d'augmenter la quantité totale de flot transmise. Les questions principales qui se sont posées dans ce modèle, concernent le temps nécessaire pour transmettre une certaine quantité de flot (voir par exemple [29, 30, 31, 40, 77]). Le premier algorithme polynomial pour le problème d'évacuation a été proposé par Hajek et Ogier [40], lorsque la capacité des arcs est constante au cours du temps. Ogier a généralisé le résultat lorsque les capacités sont des fonctions constantes par morceaux. Fleischer a amélioré les deux algorithmes [29, 30]

Paramètres variables au cours du temps

Pour rendre les choses plus complexes, il suffit de faire varier arbitrairement les délais ou les capacités au cours du temps, comme c'est le cas dans le monde réel des transports. Deux approches ont été utilisées : la première est de décréter que les capacités sont constantes mais que le délai varie (voir par exemple [58, 59]) ; à l'inverse, la deuxième méthode consiste à supposer que les délais sont constants et que c'est la capacité qui varie (voir [1, 2, 43, 70, 83]). Dans le cas où le délai et les capacités varient, Orda et Rom [80] ont prouvé quelques résultats théoriques sur les flots dynamiques où le temps est continu.

Lorsque le temps de traversée changent, ils sont généralement une fonction de la charge des liens. Köhler et Skutella [59] avancent que cette relation entre délai et charge modélisent bien le trafic routier sur des routes courtes, et argumentent que les routes longues peuvent se concevoir comme une série de routes courtes. Sur ce modèle, ils ces auteurs ont prouvé que le problème du flot le plus rapide est fortement NP-difficile, et qu'il n'existe donc pas d'$(1+\epsilon)$-approximation en temps polynomial (si $P \neq NP$), mais proposent néanmoins une $(2+\epsilon)$-approximation.

Maintenant, on considère des délais constants, mais des capacités (sur les arcs et de stockage) variables au cours du temps. Dans le cas continu, de nombreux articles [1, 2, 80, 83] prouvent l'existence de solutions et proposent des algorithmes dont ils prouvent la convergence. Dans le cas discret, Halpern [43] propose également un algorithme. Cependant, toutes les méthodes proposée sont inefficaces en pratique et ne peuvent traiter des problèmes concernant plus que quelques noeuds. Il faut noter que dans le cas le plus général de variation des paramètres, ceux-ci sont donnés par une fonction mathématique Lebesgue mesurable, fournie par un Oracle. Le problème de la complexité même de la donnée des paramètres n'est pas abordé, mais conduit implicitement à une variété incontrôlable de configurations possibles.

Enfin, certains articles [30, 43, 70, 77] utilisent des capacités continues par morceaux. Malheureusement, des algorithmes efficaces ne sont proposés que dans le cas où les délais sont nuls [30].

Complexité des problèmes abordés

Nous récapitulons brièvement la complexité combinatoire des problèmes que nous venons de

présenter.

Des algorithmes polynomiaux existent pour le flot maximum, le flot le plus rapide et la livraison la plus rapide dans le cas où les paramètres sont constants au cours du temps, que ce dernier soit discret ou continu. Le problème du flot de coût minimum est NP-difficile, mais un schéma d'approximation polynomial a été proposé [33]. On ne connaît pas la difficulté du problème du flot au plus tôt, ni de celui du flot maximum universel : aucun algorithme polynomial et aucune preuve démontrant sa difficulté n'ont été trouvés. Le problème du multiflot fractionnaire a été prouvé NP-difficile [41] contrairement à ce qui existe pour les multiflots statiques.

7.4 Notion de graphe espace-temps

Nous abordons ici la notion de graphe *espace-temps* introduite par Ford et Fulkerson [34, 35] en même temps que la notion de flot dynamique. Nous définissons d'abord ce qu'est un graphe espace-temps, puis nous présentons des méthodes génériques pour la résolution de problèmes de flots dynamiques, issues de cette notion. Enfin, nous présentons la relation entre le modèle des graphes espace-temps et le modèle des graphes évolutifs.

7.4.1 Définition du graphe espace-temps

La version *espace-temps* d'un réseau dynamique est un réseau statique qui contient une copie des nœuds et une copie des liens pour chaque date de $\mathbb{T}$. L'origine et la destination des liens sont décalées dans le temps pour refléter le temps de traversée. De façon formelle, si un réseau dynamique est représenté par un graphe $G = (V, E)$ muni d'une fonction $\zeta : (E \times \mathbb{T}) \rightarrow \mathbb{T}$ représentant le délai des arcs au cours du temps et d'une fonction $\kappa : (E \times \mathbb{T}) \rightarrow \mathbb{R}_+$ représentant la capacité des arcs au cours du temps, la version espace-temps consistera en un graphe $G_{exp} = (V_{exp}, E_{exp})$ muni d'une fonction $\kappa_{exp} : E_{exp} \rightarrow \mathbb{R}_+$. Les relations entre G, ζ, κ, G_{exp} et κ_{exp} sont les suivantes :

- $V_{exp} = V \times \mathbb{T}$ est l'ensemble des couples (x, t) avec $x \in V$ et $t \in \mathbb{T}$.
- E_{exp} est l'ensemble des paires $((x, t_1), (y, t_2))$ telles que $(x, y) \in E$ et $t_2 = t_1 + \zeta(t_1)$
- κ_{exp} associe à $((x, t_1), (y, t_2))$ la valeur $\kappa((x, y), t_1)$

On peut voir qu'un flot dynamique dans G muni des fonctions ζ et κ sera équivalent à un flot dans le graphe G_{exp} muni de la capacité κ_{exp}. Tous les problèmes concernant les flots dynamiques peuvent donc être ramenés par ce biais à des problèmes statiques classiques. Cependant, la taille de G_{exp} n'est pas polynomiale dans la taille des données car G_{exp} contient $|\mathbb{T}|$ copies du réseau. Aucun algorithme polynomial ne peut donc prétendre utiliser G_{exp} de manière explicite. Par contre, G_{exp} peut être utilisé pour prouver l'optimalité des solutions proposées.

7.4.2 Techniques dérivées

Lorsque les paramètres du réseau sont constants au cours du temps, on peut définir un flot sur le graphe espace temps G_{exp} à partir de la *répétition* d'un flot sur G en utilisant la formulation arc-chemin. Chaque chemin $P^i = (x_0, x_1)(x_1, x_2)...(x_{k-1}, x_k)$ sur G peut être utilisé pour définir un chemin $P^i(t) = ((x_0, t), (x_1, t_1 + t))((x_1, t_1 + t), (x_2, t_2 + t))...((x_{k-1}, t + t_{k-1}), (x_k, t + t_k)$ sur

G_{exp} avec $t_1 = \zeta(x_0, x_1)$, $t_2 = t_1 + \zeta(x_1, x_2)$ ... et $t_k = t_{k-1} + \zeta(x_{k-1}, x_k)$. Étant donné un flot f sur G, il suffit de répéter chaque chemin P^i de ce flot $(P^i(t))$ pour $t \in \{0, 1, 2...\}$ pour obtenir un flot répété f_{exp} sur G_{exp}. Ford et Fulkerson [34, 35] ont montré d'une part que si f satisfait les contraintes de capacité sur G, alors f_{exp} satisfait les contraintes de capacité sur G_{exp}, et d'autre part que l'ensemble des flots répétés sur G_{exp} contenait un flot maximum. Anderson et Philpott [2] ont montré un résultat similaire lorsque le temps est continu. Par contre, cette technique ne peut pas toujours être employée, même dans le cas où tous les paramètres du réseau sont statiques : pour le problème du flot de coût minimum, Klinz et Woeginger [56] ont montré que l'ensemble des flots répétés dans le temps ne contenait pas les solutions optimales.

Comme c'est la taille du graphe espace-temps qui pose problème, certains auteurs ont cherché à réduire la taille de ce graphe en discrétisant le temps de façcon suffisamment grossière. Cette approche mènent naturellement à des résultats d'autant moins précis que le pas des discrétisation est grand. Philpott et Cradock [84] ont proposé dans ce sens un algorithme adaptatif qui choisit le pas de discrétisation en fonction de la précision recherchée. Fleischer et Skutella [32, 33] montrent que l'on peut obtenir une $(1+\epsilon)$ approximation aux problèmes de flot le plus rapide et de livraison la plus rapide en utilisant une graphe espace-temps de taille polynomiale.

7.4.3 Graphe espace-temps construit à partir d'un graphe évolutif.

Soit $\mathcal{G}$ un graphe évolutif dont le graphe sous-jacent est $G = (V, E)$. Soit $\kappa : E \to \mathbb{R}_+$ une fonction de capacité sur $\mathcal{G}$. Nous supposons en outre que $\mathcal{G}$ dispose d'une capacité de stockage illimitée sur ses sommets. Nous allons construire un graphe espace-temps $G_{exp} = (V_{exp}, E_{exp})$ muni d'une capacité κ_{exp} équivalent à $\mathcal{G}$ muni de la capacité κ. L'ensemble des sommets V_{exp} est égal à $(V \times \mathbb{T})$. L'ensemble des arcs E_{exp} est égal l'union

- de l'ensemble des couples $((x, t_1), (x, t_2))$ tels que x in V et $t_1 < t_2$, qui représente le stockage sur les nœuds de V,
- et de l'ensemble des couples $((x, t_1), (y, t_2))$ tels que l'arc (x, y) est présent durant l'intervalle $[t_1, t_2]$ et tel que $t_1 + \zeta(x, y) = t_2$, qui représente les liens de $\mathcal{G}$.

La fonction de capacité $\kappa_{exp} : E_{exp} \to \overline{\mathbb{R}_+}$ se construit de la manière suivante :

- pour tout $((x, t_1), (x, t_2)) \in E_{exp}$, $\kappa_{exp}((x, t_1), (x, t_2)) = +\infty$
- pour tout $((x, t_1), (y, t_2)) \in E_{exp}$ tel que $x \neq y$, $\kappa_{exp}((x, t_1), (y, t_2)) = \kappa(x, y)$.

Le graphe espace-temps G_{exp} est la version développée dans le temps du graphe évolutif $\mathcal{G}$. Un flot entre deux sommets s et d de $\mathcal{G}$ sera équivalent à un flot entre les deux sommets $(s, 0)$ et $(d, +\infty)$ de G_{exp}.

7.5 Coupes minimum et flot maximum

Dans cette section, nous étudions le problème du flot maximum : Étant donné un graphe évolutif $\mathcal{G}$ muni d'une capacité κ, et étant donnés deux sommets s et d de $\mathcal{G}$, quel est la valeur d'un flot maximum entre s et d ? Peut-on exprimer ce flot maximum de valeur polynomiale ?

Les algorithmes de flot classiques sont basés sur le théorème de coupe minimum de Menger [67]. La notion de graphe espace-temps permet d'utiliser cette notion pour les flots dynamiques. Dans

le cadre des flots dynamiques, les nœuds du réseau se trouvent d'un côté ou de l'autre d'une coupe en fonction du temps. Lorsque les paramètres du réseau sont constants, une coupe minimum peut s'exprimer de manière polynomiale. Ce résultat a été montré par Ford et Fulkerson [35] lorsque le temps est discret, et par Philpott [1, 83] lorsque le temps est continu. La notion généralisée de coupe dynamique [1, 2, 83] a été utilisée pour montrer que la valeur du flot maximum était égale à la valeur de la coupe minimum. Ogier [77] a utilisé les coupes minimum pour trouver un algorithme polynomial au problème du flot maximum lorsque les paramètres du réseau sont constants par morceaux et lorsque les délais sont nuls. Nous allons généraliser ce dernier résultat sur les graphes évolutifs en montrant que lorsque l'ensemble des dates remarquables $\mathbb{T}_{rem}$ (Définition 33 page 132) d'un graphe évolutif $\mathcal{G}$ est polynomial dans la taille de $\mathcal{G}$. C'est notamment le cas lorsque l'ensemble les délais des arcs $\{\zeta(a), a \in A\}$ est de taille bornée.

7.5.1 Coupes minimum sur un graphe évolutif

Soit $\mathcal{G}$ un graphe évolutif muni d'une capacité κ. Comme nous l'avons défini à la section précédente, nous construisons un graphe espace-temps G_{exp} et une capacité κ_{exp} tel que G_{exp} muni de κ_{exp} est équivalent à $\mathcal{G}$ muni de κ.

L'hypothèse de stockage illimitée entraîne la présence d'arcs de capacité infinie sur G_{exp} : les arcs de la forme $((x,t_1),(x,t_2))$ avec $(t_1 < t_2)$. Si une coupe sur G_{exp} contient un seul de ces arcs, sa valeur est $+\infty$. On en déduit qu'une coupe de valeur finie entre $(s,0)$ et $(t,+\infty)$ ne contient aucun arc de la forme $((x,t_1),(x,t_2))$ avec $(t_1 < t_2)$. Ceci veut dire que pour tout sommet $x \in V$, il existe une date $t(x) \in \mathbb{T}$ telle que le sommet (x,t) du graphe espace-temps est d'un coté de la coupe si $t \geq t(x)$, et de l'autre côté si $t < t(x)$. La donnée d'une coupe de valeur finie peut donc s'écrire sous la forme d'un N-uplet, $C = (t(x), x \in V)$. Un arc $((x,y),t_1)$ de G_{exp} traverse la coupe si l'arc (x,y) de $\mathcal{G}$ si $[t, t+\zeta(x,y)]$ est contenu dans un intervalle de présence $[a_{xy}, b_{xy}]$ de l'arc (x,y), si $t \geq t(x)$ et si $t + \zeta(x,y) < t(y)$. La valeur $\kappa(C)$ d'une coupe ainsi définie est égale à la somme

$$\kappa(C) = \sum v([a_{xy}, b_{xy}]) \tag{7.11}$$

$$v([a_{xy}, b_{xy}]) = \kappa(x,y) \times \max(0, \min(t(y, b_{xy})) - \max(t(x), a_{xy}) - \zeta(x,y). \tag{7.12}$$

Si l'on recherche la valeur d'une coupe minimum entre deux sommets s et d du graphe évolutif $\mathcal{G}$, il suffit d'ajouter les égalités $t(s) = 0$ et $t(d) = +\infty$. On cherche à minimiser $\kappa(C)$ sous toutes ces contraintes. Malheureusement, la fonction $\kappa(C)$ n'est pas convexe : on ne peut pas trouver son minimum de manière polynomiale par programmation linéaire. Cette fonction est néanmoins linéaire par morceaux, et nous allons pouvoir utiliser cette propriété.

7.5.2 Résolution partielle du problème de flot maximum

Nous allons étudier à présent les points de non linéarité de la fonction $\kappa(C)$. Certains de ces points de non linéarité concernent des dates remarquables.

Définition 33 (Ensemble $\mathbb{T}_{rem}$ des dates remarquables) *Soit $\mathcal{G}$ un graphe évolutif. On appelle ensemble des dates remarquables $\mathbb{T}_{rem}$ de $\mathcal{G}$ l'ensemble des dates t telles que :*

- $t = 0$, $t = +\infty$, *ou bien*
- t *est la date d'apparition ou de disparition d'un arc* (x, y), *ou bien*
- *il existe une suite de sommets distinctsx*$_0, x_1, ..x_k$ *et une suite de dates* $t_0, t_1, t_2, ..t_k$ *de* $\mathbb{T}$ *avec*
  - *pour tout* $1 \leq i \leq k$, *un des arcs* (x_{i-1}, x_i), $(x_i, x_{i-1}$ *est présent dans l'intervalle* $[\min(t_{i-1}, t_i), \max(t_{i-1}, t_i)]$
  - $t_0 = t$
  - *soit* $t_k = 0$, *soit* $t_k + \infty$, *soit* t_k *est une date d'apparition ou de disparition d'un arc adjacent à* x_k.

Lemme 17 *Soit* $\mathcal{G}$ *un graphe évolutif muni d'une capacité* κ, *soit* s *et* d *deux sommets de* V. *Il existe une coupe minimum* $C = (t(x), x \in V)$ *entre* s *et* d *telle que* C *ne contient que des dates remarquables de* $\mathcal{G}$.

Preuve
Soit $C = (t(x), x \in V)$ une coupe minimum entre s et d. On a $t(s) = 0$ et $t(d) = +\infty$. Notons X l'ensemble des sommets x tels que $t(x)$ n'est pas une date remarquable de $\mathcal{G}$. Pour tout $\alpha \in \mathbb{R}$ nous considérons la coupe $C(\alpha) = (t(x) + \alpha, x \in X, t(y), y \notin X)$. On note g la fonction $\alpha \to \kappa(C(\alpha))$. g est linéaire par morceaux et admet un minimum pour $\alpha = 0$. Si g est linéaire en 0, alors elle également constante au voisinage de 0. On choisit un plus grand α tel que $g(\alpha) = g(0)$ et tel que g soit non linéaire en α. Ce point de non linéarité correspond à un point de non linéarité pour un des éléments $v([a_{xy}, b_{xy}])$ (équation 7.12). Sur ce point de non linéarité, il existe donc deux sommets $x \in X$ et $y \in V$ tels que

- l'arc (x, y) apparaît ou disparaît à la date $t(x) + \alpha$, ou bien
- l'arc (y, x) apparaît ou disparaît à la date $t(x) + \alpha$, ou bien
- $t(y)$ est une date remarquable et $t(x) + \alpha + \zeta(x, y) = t(y)$, ou bien
- $t(y)$ est une date remarquable et $t(y) + \zeta(y, x) = t(x) + \alpha$.

Dans tous les cas précédents, $t(x) + \alpha$ est une date remarquable. $C(\alpha)$ est une coupe minimum entre s et d et contient une date remarquable de plus que C. Par induction, on en déduit qu'il existe une coupe minimum entre s et d ne contenant que des dates remarquables. □

Théorème 29 (Flot maximum dans les graphes évolutifs) *Soit* $\mathcal{G}$ *un graphe évolutif muni d'une capacité* κ, *soit* s *et* d *deux sommets de* $\mathcal{G}$. *La valeur du flot maximum entre* s *et* d *peut être calculée de façon polynomiale en* $(N + |\mathbb{T}_{rem}|)$.

Preuve
On construit le graphe des dates remarquables G_{rem} à partir du graphe espace-temps G_{exp} en fusionnant toutes les paires de sommets $(x, t_1), (x, t_2)$ telles qu'il n'existe pas de date remarquable t avec $t_1 \leq t < t_2$. Ce nouveau graphe $G_{rem} = (V_{rem}, E_{rem})$ a pour ensemble de sommets $(V \times \mathbb{T}_{rem})$. Il admet un flot maximum entre $(s, 0)$ et $(d, +\infty)$ de valeur au moins aussi grande qu'un flot maximum dans G_{exp}, car il a été obtenu par la fusion de sommets de G_{exp}. De plus, d'après le Lemme 17, il existe une coupe minimum dans $\mathcal{G}$ qui correspond exactement à une coupe dans le graphe G_{rem}. On peut donc en conclure que la valeur du flot maximum entre $(s, 0)$ et $(d + \infty)$ est la même dans le graphe G_{rem} que dans $\mathcal{G}$.

□

7.6 Conclusion

Le problème du flot, largement étudié dans les réseaux statiques et dynamiques, se révèle être particulièrement difficile lorsque l'on permet des changements arbitraires de topologie au cours du temps, et n'a pu être résolu que lorsque les temps de traversée des liens sont nuls [30, 77]. Dans le cadre du modèle des graphes évolutifs, on ne sait pas quelle serait la taille de la donnée d'un flot maximum. Par contre, nous avons montré que le problème dual de la coupe minimum trouvait une expression simple : une coupe minimum se présente comme un N-uplet de dates de $\mathbb{T}$. Nous avons également généralisé le cas des délais nuls : nous pouvons calculer en temps polynomial la valeur d'un flot maximum lorsque l'ensemble des *dates remarquables* $\mathbb{T}_{rem}$ du réseau est borné. Cette condition est vérifiée lorsque les délais sont tous égaux, mais aussi lorsque l'ensemble des sommes de N délais est petit. Elle également vérifiée lorsque le réseau oblige les messages à attendre à chaque nœud, comme dans le cas des réseaux radio.

Conclusion et Perspectives

Nous avons étudié dans cette thèse des problèmes de routage et de connexité dans les réseaux de télécommunications optiques, et dans les réseaux dynamiques. Pour les réseaux optiques, nous avons démontré une borne sur le nombre d'arêtes minimum des graphes 2-connexes, et nous avons donné des algorithmes polynomiaux pour résoudre des problèmes de routage de chemin disjoints dans les graphes symétriques. Lorsque le nombre de requêtes est borné. Nous avons également résolu le problème du multiflot entier symétrique à deux commodités. Pour les réseaux dynamiques, nous avons contribué à développer un modèle combinatoire pour représenter l'évolution temporelle du réseau. Sur ce modèle, nous avons proposé des algorithmes polynomiaux pour résoudre différents problèmes de routage et d'arbres couvrant. Nous avons également proposé des algorithmes polynomiaux pour résoudre dans certains cas particuliers les problèmes de composantes connexes maximales et des problèmes de flot.

Récapitulation des résultats

- **connexité :** nous avons déterminé le coût en arêtes des graphes 2-connexes et des graphes 2-arête-connexes de diamètre fixé, prouvant ainsi une conjecture de Bolobás.
- **graphes symétriques(chemins) :** le problème des chemins disjoints est NP-complet pour une instance quelconque de communication. Nous avons démontré que lorsque le nombre de requêtes est borné, il existe un algorithme polynomial pour résoudre ce problème. La preuve de cette démonstration repose en grande partie sur l'étude des "mineurs" effectuée par Robertson et Seymour, ce qui implique malheureusement que nous n'avons pas d'algorithmes réellement efficaces pour résoudre ce problème à cause de la grande complexité des algorithmes de Robertson et Seymour. Nous ne savons pas si cette difficulté est inhérente au problème lui-même, ou provient de la méthode employée.
- **graphes symétriques(multiflots) :** Nous avons montré que la présence d'un seul flot est suffisante pour rompre la symétrie du problème des multiflots entiers sur les graphes symétriques. Ceci fait que les problème de multiflots sur les réseaux orientés symétriques sont presque équivalents aux problèmes de multiflots dans les réseaux orientés "à un flot près". Aussi nous avons montré que le problème des multiflots entiers est NP-difficile pour trois paires de requêtes. Nous avons ensuite étudié une instance particulière du problème ou les requêtes sont elles-mêmes symétriques. Ceci peut se justifier d'un point de vue applicatif par la résolution de communications type pair-à-pair ou type cablées. D'un point de vue théorique, ce problème est entièrement symétrique. Dans ce cas, nous avons proposé dans le cas de deux commodités un algorithme exact et rapide (correspondant à 6 calculs de flots simples). Il

est remarquable qu'il s'agit d'un des rares cas particuliers de multiflot entier ou il existe un algorithme polynomial pour deux commodités. Nous avons aussi montré, sans surprise, que le problème est NP-difficile pour un nombre indéterminé de requêtes. Les questions suivantes sont restées ouvertes : (1) Existe-t-il un algorithme polynomial pour la résolution du problème du multiflot entier pour deux requêtes ? (2) A partir de combien de requêtes (trois ou plus) le problème des multiflots symétriques ne peu plus être résolu en temps polynomial ?

- **graphes évolutifs :**
  - La notion de trajet, qui modélise une communication ayant une dimension temporelle a été formalisée par. Nous avons proposé des algorithmes polynomiaux pour calculer des trajets optimaux pour minimiser la date d'arrivée, le temps de trajet, ou le nombre de liens utilisés. Nous avons également montré qu'en présence d'une fonction arbitraire de coût sur les liens, trouver un trajet de coût minimum est un problème NP-difficile.
  - Du point de vu structurel, il a été montré par Bhadra et Ferreira [5] que le problèmes de composantes connexes maximales était NP-difficile. Zvi Lotker a montré que le problème est également NP-difficile dans le cas où l'ensemble des liens forme une double grille (obtenue en ajoutant des liens entre des sommets à distance 2 sur une grille classique). Nous avons proposé un algorithme polynomial pour résoudre le problème dans le cas où l'ensemble des liens forment un arbre.
  - Du point de vue applicatif, nous avons montré que pour les réseaux radio, calculer une arbre couvrant de coût énergétique minimal est NP-difficile, contrairement au cas statique. Ce résultat implique entre autres que les heuristiques actuelles de communication sur les réseaux radio, qui sont fondées sur le calcul d'arbre couvrant ne peuvent pas s'appliquer directement au cas dynamique. Nous avons cependant fourni un algorithme permettant de minimiser la consommation d'énergie du noeud le plus vulnérable.
  - Enfin, nous avons travaillé sur le problème du flot maximum. Nous avons formalisé la correspondance flot-max coupe-min dans le cadre des graphes évolutifs, qui repose sur la modélisation espace temps de Ford et Fulkerson. Le problème dual de coupe minimum s'exprime de manière très simple dans les graphes évolutifs, et peut être résolu dans certains cas particuliers. Nous ne connaissons pas la classe de complexité du problème dans le cas général.

Perspectives

Les problèmes de routage et de multiflot sur les graphes symétriques n'ont pas été entièrement résolus. Il serait intéressant de disposer d'algorithmes efficaces pour le problème des chemins disjoints, lorsque le nombre de requêtes est faible. Il s'agit d'un sujet ardu si on le compare au travaux similaire dans le cadre des graphes non orientés. Pour les multiflots symétriques, il faudrait trouver s'il existe des algorithmes polynomiaux lorsque le nombre de commodités est supérieur ou égal à trois.

Le modèle des graphes évolutifs présente un cadre prometteur pour étudier de nombreux problèmes d'optimisation pour les réseaux dynamiques. Du point de vue théorique, il serait intéressant de résoudre le problème du flot maximum, qui a commencé à été résolu il y a plus de 50 ans pour les réseaux entièrement statiques, et qui a connu peu de progrès depuis Un classe particulière de réseaux dynamiques, les réseaux de mobiles, font l'objet aujourd'hui d'une recherche active, et il serait intéressant d'intégrer certaines de leur caractéristiques et de leurs problématiques aux graphes évolutifs. Le caractère distribué des ces réseaux serait particulièrement intéressant à étudier dans le cadre dynamique.

Dans le cadre de l'analyse compétitive, il serait intéressant de développer des expérimentations en créant des graphes évolutifs à partir des traces (délai, bande passantes) de réseaux dynamiques et de comparer les algorithmes exacts au algorithmes utilisés couramment par les protocoles de communication.

Bibliographie

[1] E. J. Anderson, P. Nash, and A. B. Philpott. A class of continuous network flow problems. *Mathematics of Operations Research*, pages 501–504, 1982. Voir aussi l'Erratum dans *Math.Oper.Res.*,8(3) :478,1983.

[2] E. J. Anderson and A. Philpott. Optimisation of flows in networks over time. In F. P. Kelly, editor, *Probability, statistics and optimisation*, pages 369–382. Wiley Ser.Probab.Math.Statist., 1994.

[3] J. E. Aronson. A survey of dynamic network flows. *Annals of Operations Research*, (20) :1–66, 1989.

[4] J. Bermond, J. Bond, M. Paoli, and C. Peyrat. Graphs and interconnection networks : Diameter and vulnerability. *London Mathematical Society Lectures Notes, Cambridge University*, pages 1–30, 1983.

[5] S. Bhadra and A. Ferreira. Computing multicast trees in dynamic networks using evolving graphs. Technical Report 4531, INRIA, aôut 2003.

[6] B. Bollobás. A problem of the theory of communication networks. *Acta Math. Acad. Sci. Hungar*, 19 :75–80, 1968.

[7] J. H. Bookbinder and S. P. Sethi. The dynamic transportation problem : A survey. *Naval Research Logistic Quarterly*, 27(1) :65–87, 1980.

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

[9] 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.

[10] B. Bui-Xuan, A. Ferreira, and A. Jarry. Evolving graphs and least cost journeys in dynamic networks. In *Proceedings of WiOpt'03 – Modeling and Optimization in Mobile, Ad-Hoc and Wireless Networks*, pages 141–150, Sophia Antipolis, Mar. 2003. INRIA Press.

[11] R. E. Burkard, K. Dlaska, and B. Klinz. The quickest flow problem. *ZOR Methods and Models of Operations Research*, 37(1) :31–58, 1993.

[12] L. Cacetta. Extremal graphs of diameter 4. *Journal of Combinatoric theory*, 21 :104–115, 1976.

[13] P. Chanas. *Réseaux ATM : Conception et optimisation*. PhD thesis, Université de Grenoble, Juin 1998. France télécom CNET.

[14] S. Choplin. *Dimensionnement de réseaux virtuels de communications*. PhD thesis, Université de Nice Sophia Antipolis, Nov. 2002.

[15] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. Wiley-Interscience, 1998.

[16] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Agorithms*. The MIT Press, 1990.

[17] S. Dreyfus. An appraisal of some shortest-path algorithms. *Operations Research*, 17 :269–271, 1969.

[18] T. Eilam, S. Moran, and S. Zaks. Approximation algorithms for survivable optical networks. *DISC*, 2000.

[19] E. Ekici, I. F. Akyildiz, and M. D. Bender. Datagram routing algorithm for leo satellite networks. In *IEEE Infocom*, pages 500–508, 2000.

[20] G. Ellinas, A. Hailemariam, and T. Stern. Protection cycles in mesh wdm networks. *IEEE Journal on Selected Areas in Communication*, 18, October 2000.

[21] H. Enomoto and Y. Usami. Minimum number of edges in graphs with given diameter and connectivity. private commmunication, 1983.

[22] D. Eppstein. Subgraph isomorphism in planar graphs and related problems. *Journal of Algorithms and Applications*, 3(3) :1–27, 1999.

[23] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM J. Comput.*, 5(4) :691–703, Décembre 1976.

[24] A. Faragò and V. Syrotiuk. A unified framework for routing protocol. In *Proceedings ACM Mobicom 01*, pages 53–60, 2001.

[25] L. Feisher and E. Tardos. Efficient continuous-time dynamic network flow algorithms. *Operations Research Letters*, 23 :71–80, 1998.

[26] A. Ferreira. On models and algorithms for dynamic communication networks : The case for evolving graphs. In *4e rencontres francophone sur les aspects algorithmiques des télécommunications*, Mèze, France, mai 2002. INRIA.

[27] A. Ferreira, J. Galtier, and P. Penna. Topological design, routing and handover in satellite networks. In I. Stojmenovic, editor, *Handbook of Wireless Networks and Mobile Computing*, pages 473–493. John Wiley and sons, Feb. 2002.

[28] A. Ferreira and A. Jarry. Complexity of minimum spanning tree in evolving graphs and the minimum-energy broadcast routing problem. In *Proceedings of WiOpt'04 – Modeling and Optimization in Mobile, Ad-Hoc and Wireless Networks*, Cambridge, Mar. 2004.

[29] L. Fleisher. Faster algorithms for the quickest transshipment problem. *SIAM Journal on Optimisation*, 12(1) :18–35, 2001.

[30] L. Fleisher. Universally maximum flow with piecewise-constant capacities. *Networks*, 38(3) :115–125, 2001.

[31] L. Fleisher and J. B. Orlin. Optimal rounding of instantaneous fractional flows over time. *SIAM Journal on Discrete Mathematics*, 13(2) :145–153, 2000.

[32] L. Fleisher and M. Skutella. The quickest multicommodity flow problem. In W. Cook and A. Shulz, editors, *Integer Programming and Combinatorial Optimization*, volume 2337 of *Lecture Notes in Computer Science*, pages 36–53. Springer, Berlin, 2002.

[33] L. Fleisher and M. Skutella. Minimum cost flows over time without intermediate storage. In *Proceedings of the 35th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, january 2003.

[34] L. R. Ford and D. R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, 6 :419–433, 1958.

[35] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, New Jersey, 1962.

[36] S. Fortune, J. Hopcroft, and J. Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, (10) :111–121, 1980.

[37] A. Frank. *Handbook of Combinatorics*, volume 1, chapter Connectivity and Network Flows. 1995.

[38] D. Gale. Transient flows in networks. *Michigan Mathematical Journal*, 6 :59–63, 1959.

[39] M. D. Grigoriadis and L. G. Khachiyan. Approximate minimum-cost multicommodity flows in $\tilde{o}(\epsilon^{-2}knm)$ time. Technical report, DIMACS, Mai 1995. Cooperative project of Rutgers University, Princeton University, AT&T Bell Laboratories and Bellcore.

[40] B. Hajek and R. G. Ogier. Optimal dynamic routing in communication networks with continuous traffic. *Networks*, 14 :457–487, 1984.

[41] A. Hall, S. Hippler, and M. Skutella. Multicommodity flows over time : Efficient algorithms and compexity. In J. C. M. B. et al., editor, *Automata, Languages and Programming*, Lecture Notes in Computer Science, pages 397–409. Berlin.

[42] J. Halpern. Shortest route with time dependent length of edges and limited delay possibilities in nodes. *Zeitschrift für Operations Research*, 21 :117–124, 1977.

[43] J. Halpern. A generalized dynamic flow problem. *Networks*, 9 :133–167, 1979.

[44] J. Halpern and I. Priess. Shortest path with time constraints on movement and parking. *Networks*, 4 :241–253, 1974.

[45] B. Hoppe and E. Tardos. Polynomial time algorithms for some evacuation problems. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 433–441, 1994.

[46] B. Hoppe and E. Tardos. The quickest transshipment problem. *Mathematics of Operations Research*, 25(1) :36–62, 2000.

[47] T. Ibaraki and S. Poljak. Weak three-linking in eulerian digraphs. *SIAM journal on Discrete Mathematics*, 4 :84–98, 1991.

[48] M. Iri. On an extension of the maximum-flow minimum-cut theorem to multicommodity flows. *Journal of the Operations Research Society Japan*, 13 :129–135, 1970.

[49] A. Jarry. Integral symmetric 2-commodity flows. In *Proceedings of STACS'04*, Montpelliers, Mar. 2004.

[50] A. Jarry and A. Laugier. Graphes 2-connexes à diamètre donné. In U. d'Avignon et des Pays de Vaucluse, editor, *ROADEF 2003*, number 5 in Proceedings in Informatics, pages 102–104, Avignon, France, June 2003.

[51] A. Jarry and Z. Lotker. Connectivity in evloving graphs with geometric properties. In *Proceedings of DIALM-POMC'04*, 2004.

[52] A. Jarry and S. Pérennes. Disjoint Paths in Symmetric Digraphs. In *International Colloquium on Structural Information and Communi cation Complexity – SIROCCO*, pages 211–222, Andros, Greece, June 2002. Carleton.

[53] R. Karp. On the complexity of combinatorial problems. *Networks*, (5) :45–68, 1975.

[54] L. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244 :1093–1096, 1979.

[55] P. Klein, S. Rao, A. Agrawal, and R. Ravi. An approximate max-flow min-cut relation for undirected multicommodity flow, with applications. *Combinatorica*, 2(15) :187–202, 1995.

[56] B. Klinz and G. J. Woeginger. Minimum cost dynamic flows : the series-paralle case. In *Integer Programming and Combinatorial Otimisation*, volume 920 of *Lecture Notes in Computer Science*, pages 329–343. Springer, Berlin, 1995.

[57] E. Köhler, K. Langkau, and M. Skutella. Time-expanded graphs for flow-dependent transit times. In *proc. ESA'02*, 2002.

[58] E. Köhler, K. Langkau, and M. Skutella. Time-expanded graphs with flow-dependent transit times. In R. H. Möhring and R. Raman, editors, *Algorithms - ESA'02*, volume 2461 of *Lecture Notes in Computer Science*, pages 599–611, Berlin, 2002. Springer.

[59] E. Köhler and M. Skutella. Flows over time with load-dependent transit times. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'02)*.

[60] B. Kotnyek. An annotated overview of dynamic network flows. Technical Report 4936, INRIA, septembre 2003.

[61] A. S. Lapaugh and R. L. Rivest. The subgraph homeomorphism problem. In *Proceedings of the 10th Annual ACM Sumposium on Theory of Computation*, pages 40–50, San Diego, 1978.

[62] A. Laugier. Communication personnelle.

[63] A. Laugier, F. Boyer, and O. Goldschmidt. Two-connected graphs with given diameter. private communication, 1997.

[64] T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with application to approximation algorithms, 1988.

[65] S. Lovetski and I. I. Melamed. Dynamic flows in networks. *Automation and Remote Control*, 48 :1417–1434, 1987. Traduit de Avtomatika i Telemekhanika, No.11, pp 7-29, 1987.

[66] G. Lueker. Two np-complete problems in nonnegative integer programming. Technical Report 178, Computer Science Laboratory, Princeton University, Princeton, New Jersey, 1975.

[67] K. Menger. Zur allgemeinen kurventheorie. *Fundam. Math.*, 10 :96–115, 1927.

[68] M. Middendorf and F. Pfeiffer. On the complexity of the disjoint paths problem. *Combinatorica*, 13(1) :97–107, 1993.

[69] E. Minieka. Maximal, lexicographic, and dynamic network flows. *Operations Research*, 21 :517–527, 1973.

[70] E. Minieka. Dynamic network flows with arc changes. *Networks*, 4 :255–265, 1974.

[71] F. H. Moss and A. Segall. An optimal control approach to dynamic routing in networks. *IEEE Transactions on Automatic Control*, 27(2) :329–339, 1982.

[72] U. Murty. On some extremal graphs. *Acta.Math.Acad.Sci.Hungar*, 19 :69–74, 1968.

[73] U. Murty. *Proof techniques in graph theory*, chapter Extremal nonseparable graphs of diameter 2. Academic press, 1969.

[74] U. Murty and K. Vijayan. On accessiblility in graphs. *Sankhya (A)*, 26 :221–234, 1964.

[75] H. Nagamochi and T. Ibaraki. Multicommodity flows in certain planar directed networks. *Discrete Applied Mathematics*, 27 :125–145, 1990.

[76] T. Nishizeki, J. Vygen, and X. Zhou. The edge-disjoint paths problem is np-complete for series-parallel graphs. *Discrete Applied Mathematics*, 115(1-3) :177–186, 2001.

[77] R. G. Ogier. Minimum-delay routing in continuous-time dynamic networks with piecewise constant capacities. *Networks*, 18 :303–318, 1988.

[78] H. Okamura and P. Seymour. Multicommodity flows in planar graphs. *Journal of Combinatorial Theory*, B(31) :75–81, 1981.

[79] K. Onaga and O. Kakusho. On feasibility conditions of multicommodity flows in networks. *IEEE Transactions on Circuit Theory*, 18 :425–429, 1971.

[80] A. Orda and R. Rom. On continuous network flows. *Operations Research Letters*, 17 :27–36, 1995.

[81] J. Orlin. Maximum-throughput dynamic network flows. *Mathematical Programming*, 27(2) :214–231, 1983.

[82] J. Orlin. Minimum convex cost dynamic network flows. *Mathematics of Operations Research*, 9(2) :190–207, 1984.

[83] A. B. Philpott. Continuous-time flows in networks. *Mathematics of Operations Research*, 1990.

[84] A. B. Philpott and M. Craddok. An adaptative discretization algorithm for a class of continuous network programs. *Networks*, 26 :1–11, 1995.

[85] W. B. Powell, P. Jaillet, and A. Odoni. Stochastic and dynamic networks and routing. In M. O. B. et al., editor, *Handbook in Operations Research and Management Science - Network Routings*, volume 8, chapter 3. Elsevier Science, 1995.

[86] N. Robertson and P. Seymour. Graph minors xiii. the disjoint paths problem. *Journal of Combinatorial Theory*, B(63) :65–110, 1995.

[87] C. Scheideler. Models and techniques for communication in dynamic networks. In H. Alt and A. Ferreira, editors, *Proceedings of the 19th International Symposium on Theoretical Aspects of Computer Science*, volume 2285, pages 27–49. Springer-Verlag, Mar. 2002.

[88] M. Sridharan and A. Somani. Revenue maximization in survivable wdm networks. *OPTICOM*, 2000.

[89] G. I. Stassinopoulos and P. Konstantopoulos. Optimal congestion control in single destination networks. *IEEE Transactions on Communications*, 33(8) :792–800, 1985.

[90] I. Stojmenovic, editor. *Handbook of Wireless Networks and Mobile Computing*. John Wiley and sons, Feb. 2002.

[91] L. Viennot. Routage entre robots dont les déplacements sont connus – un exemple de graphe dynamique. Réunion TAROT, ENST, Paris, Nov. 2001.

[92] J. Vygen. *Disjoint Paths in Directed Graphs*. PhD thesis, University of Bonn, 1992. en allemand.

[93] J. Vygen. Disjoint paths. Technical Report 94816, Research Institute for Discrete Mathematics, University of Bonn, February 1994. updated September 1998.

[94] J. Vygen. Np-completeness of some edge-disjoint paths problems. *Discrete Applied Mathematics*, 61 :83–90, 1995.

[95] D. Wagner and K. Weihe. A linear-time algorithm for edge-disjoint paths in planar graphs. *Combinatorica*, 1(15) :135–150, 1995.

[96] M. Werner and G. Maral. Traffic flows and dynamic routing in leo intersatellite link networks. In *Proceedings 5th International Mobile Satellite Conference (IMSC '97)*, Pasadena, California, USA, June 1997.

[97] M. Werner and F. Wauquiez. Capacity dimensioning of ISL networks in broadband LEO satellite systems. In *Sixth International Mobile Satellite Conference : IMSC 99*, pages 334–341, Ottawa, Canada, June 1999.

[98] W. L. Wilkinson. An algorithm for universal maximal dynamic flows in a network. *Operations Research*, 19 :1602–1612, 1971.

Annexe A

Annexe

A.1 Notations Standard sur les graphes

Nous rappelons ici des notations standard de théorie des graphes que nous utilisons dans cette thèse.

Un graphe G est en fait une paire (V, E) composée de deux ensembles V et E. Lorsque l'on introduit un nouveau graphe, on utilise en général la formule "soit $G = (V, E)$ ...". L'ensemble V s'appelle l'ensemble des sommets du graphe. Lorsque le graphe représente un réseau de télécommunication, V représente l'ensemble des nœuds du réseau. Au cours de cette thèse nous n'utilisons que des graphes où l'ensemble de sommets est fini. Le nombre de sommets du graphe est souvent notée $n = |V|$ ou $N = |V|$. Ce nombre est parfois appelé *ordre* de G. L'ensemble E s'appelle l'ensemble des arêtes du graphe, ou bien l'ensemble des arcs du graphe. Une arête est une paire de sommets (x, y) appartenant à V. Un arc est un couple de sommets (x, y) appartenant à V. Ce qui distingue un arc d'une arête est le fait qu'un arc est orienté, c'est à dire que $(x, y) \neq (y, x)$, alors qu'une arête n'est pas orientée : $(x, y) = (y, x)$. Lorsque E est un ensemble d'arêtes, on dit que le graphe G est non orienté. Lorsque E est un ensemble d'arcs, on dit que G est orienté. Lorsque G représente un réseau de télécommunication, l'ensemble E représente l'ensemble des liens du réseau (fibres optiques, voies de chemin de fer, connexions radio). Le nombre d'arêtes (ou d'arcs) du graphe est souvent noté $m = |E|$ ou $M = |E|$.

Définition 34 (chemin) *Soit $x_0, x_1...x_k$ une séquence de sommets de V, et $e_1, e_2, ..e_k$ une séquence d'arêtes (ou d'arcs) de E tels que pour tout $i \in \{1, ..k\}$ l'arête (ou l'arc) e_i est égal à (x_{i-1}, x_i). La séquence d'arêtes (ou d'arcs) $\mu = e_1, e_2, ..e_k$ est appelée* chemin de longueur k reliant x à y *ou encore* chaîne de longueur k reliant x_0 à x_k. *On note V_μ l'ensemble des sommets intérieurs $\{x_1, ..x_{k-1}\}$ et E_μ l'ensemble des arêtes (ou des arcs) $\{e_1, ..e_k\}$.*

Définition 35 (distance) *Pour toute paire de sommets x, y de V, on appelle distance de x à y, notée $D(x, y)$, la longueur du plus court chemin reliant x à y. S'il n'existe pas de chemin reliant x à y, on a $D(x, y) = +\infty$. Pour tout sommet $x \in V$, et pour tout ensemble de sommets S inclus dans V, on appelle distance de x à S, notée $D(x, S)$, le minimum des distances de $D(x, y)$ sur les sommets y de S : $D(x, S) = \min_{y \in S} D(x, y)$. Si S est l'ensemble vide, on a $D(x, S) = +\infty$.*

Définition 36 (excentricité) *Soit S un sous ensemble de sommets non vide de V. On appelle excentricité de G par rapport à S, notée $\alpha(G,S)$, la plus grande distance $D(x,S)$ sur tous les sommets x de V : $\alpha(G,S) = \max_{x \in V} D(x,S)$. Par abus de langage, si H est un sous-graphe de G ayant S pour ensemble de sommets, on notera $\alpha(G,H) = \alpha(G,S)$.*

Définition 37 (chemins disjoints)

- *Deux chemins sont dits **sommet-disjoints** s'ils n'ont aucun sommet en commun, excepté éventuellement leurs extrémités.*
- *Dans un graphe non-orienté, deux chemins sont dits **arête-disjoints** s'ils n'ont aucune arête en commun.*
- *Dans un graphe orienté, deux chemins sont dits **arc-disjoints** s'ils n'ont aucun arc en commun.*

Définition 38 (symétrie) *Un graphe orienté $G = (V,E)$ est dit symétrique si pour tout arc $(x,y) \in E$ il existe un arc $(y,x) \in E$. Une fonction f de E vers $\mathbb{N}$ est dite symétrique si pour tout arc $(x,y) \in E$, $f(x,y) = f(y,x)$.*

Définition 39 (voisinage) *Soit $G = (V,A)$ un graphe orienté. Pour tout $x \in V$, on appelle $\Gamma^+(x)$ l'ensemble des sommets $y \in V$ tels que $(x,y) \in A$, et $\Gamma^-(x)$ l'ensemble des sommets $y \in V$ tels que $(y,x) \in A$. Nous appelons voisinage de x l'ensemble $\Gamma(x) = \Gamma^+(x) \cup \Gamma^-(x)$. Si G est symétrique, $\Gamma(x) = \Gamma^+(x) = \Gamma^-(x)$.*